CAMPUS
DE EXCELENCIA
INTERNACIONAL

Graduado en Ingeniería Informática

Universidad Politécnica de Madrid

Escuela Técnica Superior de
Ingenieros Informáticos

TRABAJO FIN DE GRADO

Aplicación web para la generación automática de relaciones de ejercicios científicos

Autor: Uriel Javier Aizensztain Goltz
Director: Dr. Emilio Serrano Fernández

MADRID, JUNIO 2017

RESUMEN

El proyecto aquí presentado tiene como propósito la realización de una aplicación web sencilla orientada al mundo académico, que facilite el acceso a ejercicios aleatorios y a sus soluciones en el caso de que se desee. Para conseguir este objetivo se pretende realizar una aplicación usable, que permita al usuario ahorrar tiempo.

Este proyecto surge a raíz de que cada vez existen repositorios más grandes de ejercicios, lo que conlleva tener que dedicar más tiempo a la búsqueda de los ejercicios deseados. Además, al automatizar la creación de ejercicios aleatorios, se permite obtener varios conjuntos de ejercicios diferentes sin tener que hacerlo manualmente.

Para realizar el trabajo se ha tenido que investigar las contribuciones de otras personas en la misma área. Y se ha tenido que investigar como conectar Latex con el Back-end. Para desarrollar este último se ha utilizado Node. La parte del Front-end se ha desarrollado con HTML5 y CSS3, además del framework Bootstrap.

ABSTRACT

The purpose of the present project is the realization of a simple web page oriented to the academic field, that intends to facilitate the access to random exercises and their solutions, when desired. To achieve this goal the intention is to make a usable application, which allows the user to save time.

This project arises as a result of the constant creation of vast collections of excercises, which leads to dedicating an excessive amount of time searching for the desired excercises. In addition, the fact of automating the creation of random excercises allows the obtainment of various groups of different excercises without having to do it manually.

The elaboration of this ending degree project implied the investigation of the contributions that other people have provided in this field. As well as a research concerning how to connect Latex with the back-end. Nodejs was used to develop this. The front-end was elaborated with HTML5 and CSS3, along with the Bootstrap framework.

Índice de contenido

Índice de figuras

1 INTRODUCCIÓN

Dentro del ámbito didáctico, un procedimiento muy utilizado para ayudar en el aprendizaje de nuevos conceptos es la realización de ejercicios. Esta técnica permite a los estudiantes interiorizar más fácilmente la teoría, por ello es que se utilizan tantos ejercicios en las clases. Estos recursos son obtenidos generalmente a partir de libros, Internet y de exámenes anteriores

Generalmente tenemos a disposición muchos de estos recursos, por lo que encontrar el que se necesita puede acarrear demasiado tiempo. La tarea de búsqueda de los ejercicios que se adecuen a las necesidades de los profesores y alumnos suele requerir de mucho tiempo que podría mejor emplearse realizando otras tareas como en estudiar o corregir. Para ello el método de búsqueda tiene que ser eficaz y eficiente.

Como se suele tener un gran repositorio de ejercicios, resulta difícil encontrar lo que se desea, es por lo tanto necesario disponer de un método eficiente para poder conseguir los ejercicios sobre un tema determinado. En ocasiones los profesores buscan diferentes conjuntos de ejercicios, sobre un mismo tema, para poder distribuirlos entre los alumnos, y así evitar que se copien, o para que cada grupo o alumno tenga un trabajo diferente. Pero hacer todo esto de forma manual puede resultar muy tedioso.

Por todo ello se ha decidido crear una aplicación web que tenga la capacidad de resolver estos problemas. Características de la aplicación:

1. Ser escalable: Tiene que poder servir y almacenar todos los archivos que se quieran, sin tener un límite de ejercicios.
2. Estar organizado: Para su correcto funcionamiento los temas tienen que tener una carpeta con los ejercicios, las soluciones y los metadatos de estos.
3. Ser Web: Para que se pueda utilizar desde cualquier lugar, ya sea desde un ordenador o un móvil.

Se ha planteado resolver el problema anteriormente mediante una aplicación web debido a que así el usuario no necesita tener instalado ningún programa en su ordenador, cualquier persona que esté autorizada podrá acceder a la aplicación en cualquier momento, y para que el usuario tenga todos los ejercicios en la nube, sin que deba preocuparse por el almacenamiento.

Este servicio está basado en un servidor que contiene ejercicios en formato Latex, y cuando recibe una petición, la cual contiene algunas opciones, selecciona ejercicios de forma aleatoria y los devuelve en un único fichero en formato PDF. Esta aplicación tiene como fin automatizar la tarea de encontrar y generar ejercicios, de una forma usable.

1.1 OBJETIVOS

El objetivo final del trabajo es la realización de una aplicación web para la generación automática de relaciones de ejercicios científicos. Esto se puede desglosar en los siguientes objetivos:

1) Análisis de requisitos y diseño de prototipo
2) Estudio del arte y aportación del trabajo: Investigar que trabajos se han realizados en el área del "e-learning" relacionados con la generación automática de ejercicios.
3) Clasificación de material digitalizable y familiarización con Latex: Estudiar los diferentes paquetes de Latex existentes que pueden ayudar a finalizar el proyecto.
4) Análisis de riesgo, investigación de técnicas de compilación online de Latex, y elección de sistema.
5) Creación de una arquitectura cliente-servidor: capa de negocios, capa de presentación y capa de datos.
6) Codificación de la aplicación: realizar el código de la parte del servidor, y el de la parte del cliente.
7) Pruebas: Comprobar el correcto funcionamiento de la aplicación.
8) Despliegue de aplicación: Importar la aplicación a un servidor de producción, el cual será usado por los usuarios.
9) Escritura de memoria y documentación del software.
10) Preparación de la presentación

1.2 ESTRUCTURA

El documento está estructurado de la siguiente manera:

1. Introducción: descripción de los objetivos del proyecto y los problemas a resolver.
2. Estado del arte: investigación de proyectos anteriores sobre la misma área.
3. Desarrollo: implementación de la solución propuesta.
4. Despliegue: arranque del servidor remoto.
5. Pruebas: comprobación del correcto funcionamiento de la aplicación.
6. Conclusiones: consideraciones finales.
7. Futuros trabajos: propuestas para la mejora del proyecto.

2 ESTADO DEL ARTE

La informática ha contribuido de manera esencial a la automatización de tareas, siendo su principal objetivo mejorar la eficiencia de éstas. Una de las áreas en las que se han hecho avances en cuanto a la automatización de sus tareas es en el de e-learning. Es habitual que los profesores tengan que crear ejercicios para sus alumnos, ésta es una de las tareas que se está intentado automatizar.

En el campo de la generación automática de ejercicios hay varios trabajos realizados. En su mayoría, los proyectos se centran en la creación de ejercicios aleatorios de algún tipo en concreto, como múltiple opción, rellenar huecos, resolver problemas matemáticos, etc.

Una de las soluciones fue presentada por Papasalouros [1]. En su trabajo se explica un método para conseguir ejercicios MCQ (Multiple choice questions) de manera automática. El proyecto se basa en: la existencia de una base de conocimiento expresada en un lenguaje de representación de conocimiento, en el uso de relaciones semánticas entre los elementos de la base de conocimiento y en la aplicación de técnicas para la generación de frases.

Otros trabajos también investigan la creación de los ejercicios del tipo múltiple opción, pero de forma diferente. En el trabajo de Kojiri, T. et al. se presenta un modelo en el que se utilizan técnicas de machine learning para generar los problemas [2]. En este caso los ejercicios son del tipo rellenar campos vacíos de una oración según las múltiples opciones disponibles. Para conseguir generar estos ejercicios de forma aleatoria el sistema necesita como entrada un texto, del cual extrae las oraciones apropiadas para las preguntas utilizando técnicas de “Preference Learning”. A partir de estas frases se estima que parte debe de estar vacía utilizando “Conditional Random Field”. Y una vez determinado el campo que se debe de rellenar, se generan soluciones incorrectas basadas en patrones estadísticos de preguntas ya existentes. Los autores tras generar el sistema crearon un servicio web, que permite introducir un texto para que el servicio pueda crear las preguntas correspondientes.

En el trabajo presentado por Grun y Zeileis se explica el funcionamiento de un framework para la generación automática de exámenes estadísticos [3]. Con esta herramienta se pueden generar varios exámenes diferentes, pero de la misma temática. Para utilizar la herramienta hace falta tener una colección de ejercicios y un archivo maestro que se encarga de unirlos. Éstos están definidos en archivos Sweave, los cuales contienen el código de R para la generación de datos, y los comandos de Latex para especificar la descripción de los problemas y las soluciones. Sweave es un componente del lenguaje R que permite integrar código en documentos Latex [4].

Otros trabajos han creado un sistema entero, que tiene en cuenta la forma de generar los ejercicios, la capa de presentación, etc. Uno de estos es el realizado por Wang et al. [5]. En él se presenta un algoritmo para la generación automática de preguntas lógicas, que solo son posibles de resolver con una solución. Los algoritmos que generan dichos ejercicios utilizan técnicas de inteligencia artificial como las redes semánticas. El sistema está pensado para ser usado en dispositivos móviles, por lo que los algoritmos tienen un tamaño pequeño. El equipo realizó una interfaz gráfica en Android. No se especifica mucho sobre la interfaz, pero se puede apreciar que está todo en escrito en chino y que tiene una estética poco cuidada.

Una de las iniciativas más interesantes es la explicada en el trabajo de Soler, J. et al. [6]. En él se presenta un sistema para la evaluación continua de los estudiantes. Dicha evaluación se realiza a través de un libro de ejercicios personal (único por cada alumno), que es generado al inicio del curso escolar. Los ejercicios son iguales para todos, lo que cambia son los datos de los problemas. El sistema se puede utilizar desde la web, la cual a pesar de no ser responsive tiene algunas características muy interesantes. Los ejercicios se resuelven desde la misma web, y ésta informa si la solución es la correcta o no. Además, permite que los profesores puedan estar al tanto de los ejercicios que realizan sus alumnos, y también conocer en cuales tienen más dificultades.

Todos los trabajos encontrados se centran principalmente en la forma de generar los ejercicios, pero no se preocupan por la manera en la que los usuarios puedan interactuar con el sistema. En la actualidad es importante tener en consideración este aspecto, ya que puede llegar a ser decisivo para que un sistema triunfe o no. Cada vez es más común utilizar los dispositivos móviles para todo, de modo que considerar desde qué dispositivo los usuarios accederán a la herramienta, es un dato que debe ser tenido en cuenta al crear la aplicación. El sistema que se describirá en este trabajo estará basado en un repositorio de archivos Latex. Este es un sistema de preparación de documentos. Los archivos estarán en este formato debido a que se pueden generar documentos de alta calidad profesional. Esto es muy notorio si se utilizan fórmulas o ecuaciones. Debido a su naturaleza, es muy común encontrarse archivos en este formato en el mundo académico. Y es por esto que el repositorio está basado en este tipo de documentos [7, 8]. Para utilizar estos documentos, se ha tenido que investigar como se suele realizar la conexión entre Latex y la web. Normalmente, los archivos en Latex una vez creados, hay que compilarlos, y si no tiene errores, se genera un archivo en formato DVI o PDF. Este proceso se puede hacer de forma local o remota.

En la web hay editores y compiladores online de Latex. Las páginas deben de tener algún método para compilar los documentos requeridos. En general, los usuarios envían sus archivos al servidor que están utilizando, para que éste lo compile y se lo devuelva en formato PDF. Para compilar un archivo de formato Latex lo más normal es realizar llamadas al sistema, lo que puede llegar a bloquear el sistema operativo. Para intentar eludir esto se ha intentado implementar librerías que contuvieran Latex, lo cual

hubiera sido útil para crear servidores que pudieran alojar muchas peticiones de compilación a la vez. Algunos de los proyectos que intentaron implementar Latex en otros lenguajes, como en Java, fueron Extex [9] y javaTex [10]. Pero finalmente no se terminaron de desarrollar.

Por ello, finalmente los servidores que están pensados para la compilación de archivos Latex tienen que realizar llamadas al sistema. Esto se puede observar en la implementación de CLSI (Common LaTeX Service Interface) [11] la cual es la forma más común de conectar Latex con la web. Esto es un web API para compilar documentos Latex en la nube. Una de las páginas que utiliza esto es ShareLatex [12]. Esta es una de las páginas más famosas para editar y compilar Latex en la nube, y son los actuales responsables del proyecto CLSI en GitHub.

Por otro lado, hay un proyecto en GitHub que ha implementado texlive (una de las distribuciones de Latex más famosas) en JavaScript [13]. Para conseguir esto se han utilizado varios "traductores" de lenguaje de programación, que parten del lenguaje original en el que fue implementado Latex para llegar a JavaScript. Está herramienta es muy útil, ya que sirve para compilar archivos Latex desde el navegador, sin necesidad de un servidor para la fase de compilación. El servidor sólo debe enviar los archivos necesarios a la página, para que ésta puede compilarlos, así se ahorra mucho tiempo de procesamiento en el servidor.

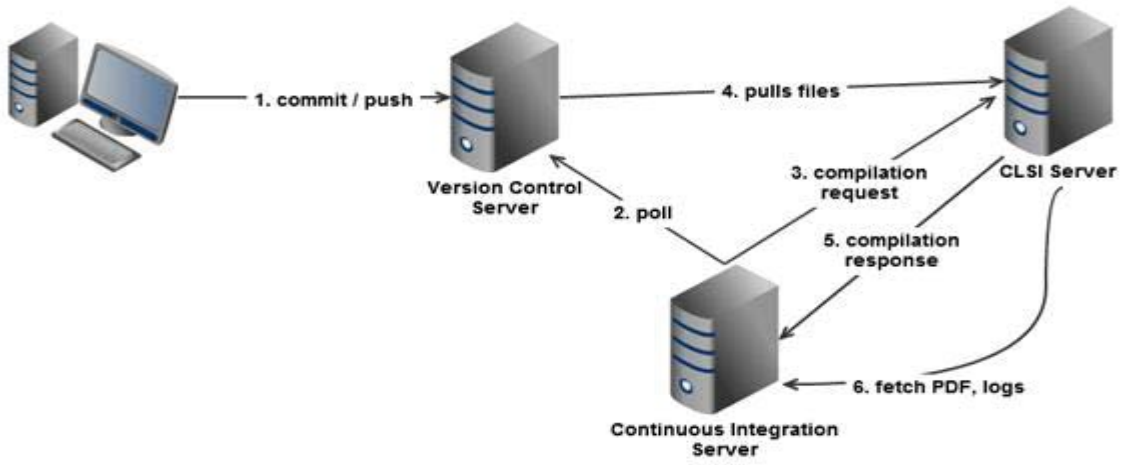


Figura 1. Estructura básica de un servicio web que utiliza CLSI

Tras analizar las diferentes vías por las cuales se puede compilar un fichero Latex a través de una página web, se ha decido crear un servidor que realice llamadas al sistema. Esta decisión se basa en que el proyecto que implementa Latex en JavaScript tiene algunas limitaciones, y en que utilizar un servidor CLSI conllevaría más recursos para obtener el mismo resultado.

3 DESARROLLO

A la hora de desarrollar la aplicación hay que tener en cuenta varias cosas. En primer lugar, se tiene que tener claro lo que se pretende conseguir con el proyecto, luego los requisitos que se tienen que cumplir para poder realizar la tarea de forma adecuada y por último se tiene que decidir que tecnologías se van a utilizar. Hoy en día existen muchos lenguajes de programación y cada uno tiene sus ventajas y desventajas, por lo tanto, es preciso analizarlos para determinar cuál es el más apropiado para alcanzar nuestro propósito. Además, teniendo en cuenta que este trabajo consiste en una aplicación web, hay que ver que tecnologías nos facilitarán dicha tarea, tanto en el lado del back-end como en el del front-end. Una vez realizada la aplicación se debe desplegar. Para ello se necesitará un servidor, o sea que requerirá de una plataforma para poder ponerla en funcionamiento. Después de todo el proceso descrito, habrá que realizar las pruebas, para comprobar que todo funciona correctamente.

3.1 REQUISITOS

Para definir los requisitos del trabajo se debe de tener en cuenta cual es la finalidad del mismo. En este caso la finalidad es crear una web a través de la cual se puedan descargar una recopilación de ejercicios de diferentes temas, con sus soluciones, y con meta-información si se desea. Además, que se pueda acceder a esta aplicación web desde distintos dispositivos móviles (ya que en estos tiempos es cada vez más habitual conectarse a Internet desde cualquier lugar)

También se tendrán en cuenta otros requisitos que puedan surgir a la hora de intentar hacer la aplicación lo más usable y escalable posible.

Existen cuatro tipos de requisitos. Los de usuario, los del sistema, los funcionales y los no funcionales. Los funcionales describen lo que el sistema debe de hacer. Los no funcionales son aquellos que imponen restricciones a los funcionales, un ejemplo seria el tiempo de respuesta del servidor. Los requisitos definidos en la lista siguiente son funcionales.

1. Poder seleccionar diferentes temas para la petición de descarga.
2. Poder decidir cuantos ejercicios se desean de cada tema.
3. Seleccionar ejercicios de forma aleatoria.

4. Tener la opción de elegir si se desean las soluciones, y en dicho caso, que salga la ruta o un identificador de la solución.
5. Tener la opción de elegir si se desea meta-información de los ejercicios.
6. Poder utilizar cualquier archivo látex como ejercicio y solución.

En los puntos siguientes se va a describir como se ha ido desarrollando la aplicación para poder cumplir todos estos requisitos de la mejor manera posible. Hay que tener en cuenta que a veces hay limitaciones, que están fuera de nuestro alcance, lo que hace que en ocasiones no se puedan conseguir los requisitos de la manera que se deseaba.

3.2 ARQUITECTURA Y CASOS DE USO

Tras definir los diferentes requisitos que tiene la aplicación, se puede desarrollar un diagrama de casos de uso [14]. Estos diagramas se utilizan para tener una vista general de los casos de usos que existen en el sistema, es decir, de los diferentes escenarios en los que el sistema interactúa con personas o sistemas externos [15]. Definir los casos de usos es muy importante, pues la implementación del sistema debe conseguir que los casos de usos se ejecuten correctamente y porque a la hora de probar la aplicación se debe comprobar que todos los casos de uso que existen funcionan como se desea, es por lo tanto una buena práctica tenerlos documentados.

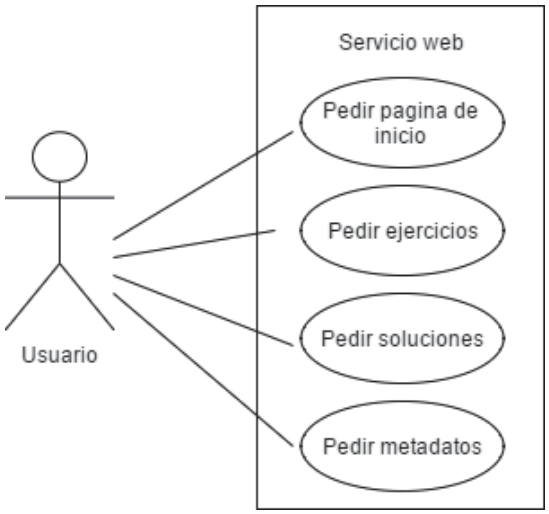


Figura 2. Diagrama de casos de uso

Los casos de usos son una descripción de como el cliente opera con el sistema. En este caso, la figura que aparece a continuación representa los cuatro casos de usos posibles que tiene la aplicación.

Estos diagramas se representan mediante un actor, el cual representa a una persona, organización o cualquier componente externo, y un caso de uso, que representa las acciones que realizan los actores.

Antes de empezar a desarrollar el código, y después de definir los casos de uso, es necesario tener una estructura de como deberá funcionar la aplicación, ya que de esta forma es más fácil ir realizando el proyecto. En primer término, se tienen que considerar los elementos que se relacionan durante el proceso de descarga de una petición. Esta es una arquitectura de alto nivel, en la cual no se tiene en cuenta cómo será luego la organización de los ficheros, de los códigos, de los ejercicios, etc. En este caso es bastante simple, ya que solo existen dos elementos durante la interacción, el dispositivo del usuario, y el servidor. Este es el que maneja las peticiones de los usuarios, busca los archivos deseados, los compila, y manda como respuesta a la petición un archivo PDF que contiene todos los ejercicios.

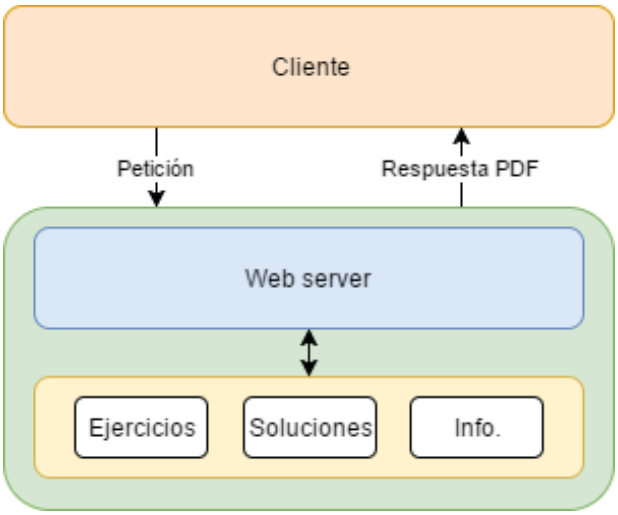


Figura 3. Arquitectura de alto nivel

Como se puede ver en la Figura 3, los ficheros están organizados en tres carpetas. Una contiene los ejercicios, otra las soluciones, y la ultima la meta-información. Estas tres carpetas deben existir en cada tema, para poder obtener ejercicios de ellos. Este punto se va a ser desarrollado más adelante.

3.3 REPOSITORIO DE ARCHIVOS LATEX

Como ya se explicó en la sección del “ESTADO DEL ARTE”, los ejercicios que el servicio otorgara están en formato Latex. Como la finalidad del servicio es enviar un solo PDF que contenga todos los ejercicios deseados es necesario una herramienta para unir todos los archivos deseados (los de

ejercicios, soluciones e información). Aquí nos encontramos con un problema, ¿Cómo unificamos estos ficheros?

En primer lugar, Latex tiene dos comandos "\input{path}" e "\include{path}", los cuales sirven para dicho propósito, pero no son válidos para este proyecto ya que el segundo comando introduce un "\clearpage", lo que hace que después de incluir un archivo se salte a la siguiente página, por lo que el fichero final termina teniendo muchas páginas, y esto no es lo deseado. El problema del primer comando consiste en que a los archivos que se quieran unificar no se les pueden poner preámbulos (la parte del archivo Latex que va antes de "\begin{document}"), que es la zona en que se definen los comandos, se importan paquetes, etc [16]. Esto se debe a que el comando lo que hará será una copia del contenido del fichero auxiliar en el principal. En consecuencia, se tendrán dos preámbulos, uno dentro de otro, y esto Latex no lo acepta.

Para resolver estos problemas se ha utilizado el paquete Standalone [17], el cual provee la funcionalidad de poder unir los preámbulos de varios ficheros en un sólo. La unión de los preámbulos queda presente en un archivo con formato STA. Luego, en el fichero principal se puede utilizar la opción de "subpreamble", para que tenga en cuenta los preámbulos del fichero generado anteriormente.

Gracias a esta característica, se podrán unir varios archivos auxiliares que tengan preámbulos diferentes en un mismo fichero, lo que da una gran libertad a la hora de crear el repositorio de ejercicios, y además los usuarios encargados de rellenar el repositorio no se tendrán que preocupar por los estilos.

Es importante aclarar que, para que el funcionamiento sea correcto, en el supuesto de que se utilice un paquete que contenga varias opciones de configuración, tanto en el fichero del ejercicio, como en el de su solución, se deben de activar las mismas opciones.

Es recomendable que el usuario que desea subir dichos ficheros, compruebe desde su ordenador personal si se unifican bien, y si Latex no da errores de compilación. Hay que tener en cuenta que la característica de unir preámbulos no es perfecta, aunque sí es funcional, y que tampoco hay muchos más paquetes que tengan una funcionalidad parecida.

Otro paquete que se ha utilizado es el de Exam [18], el cual sirve para definir entornos, es decir, se puede dar a un texto el formato de pregunta, y a otro texto el formato de respuesta. Esto es útil porque el paquete enumera los ejercicios, da una tipografía o unas características tipográficas a cada formato, y

permite decidir si se quiere mostrar las respuestas o no. Para que esta opción funcione es importante definir correctamente que archivos son las soluciones.

```
\documentclass{exam}
\printanswers
\usepackage[subpreambles=true]{standalone}
\title{Ejercicios}

\begin{document}
  \begin{questions}
  \question\input{ejercicio1.tex}
  \begin{solutionorbox}[2in]
  \input{solucion1.tex}
  \end{solutionorbox}
  \smallskip
  \end{questions}
\end{document}
```

Figura 4. Ejemplo de documento Latex

Con estos dos paquetes se puede crear un fichero principal parecido al de la Figura 4. A partir de esta estructura básica, se le puede ir insertando más preguntas y respuestas de ficheros externos, los cuales pueden tener sus preámbulos propios. En este caso las soluciones tienen definido un estilo que crea un borde negro alrededor de las soluciones.

Una vez definida como debe de funcionar la web, y sabiendo que paquetes de Latex serán necesarios para que el servicio termine realizando lo que se desea en este proyecto, es hora de desarrollar el código. En primer lugar, hay que decidir que lenguajes de programación usar. Para ello se han investigado diferentes aspectos que están relacionados con la práctica. Realizar este estudio es bastante importante ya que nos puede ahorra tiempo saber que lenguaje nos puede ayudar a realizar el proyecto de la manera más correcta posible.

3.4 ANALISIS DE RIESGO

La aplicación está pensada para almacenar archivos en formato Latex, y que, de acuerdo a la petición de los usuarios, se seleccionen algunos de esos archivos, se compile, y se convierta a formato PDF, de modo que es necesario tener en cuenta como conectar el código con Latex.

La idea principal era intentar encontrar alguna librería que compilara dichos archivos, sin la necesidad de realizar llamadas al sistema, y así evitar un consumo extra para el sistema operativo. Como se comentó en el ESTADO DEL ARTE, han existido proyectos que lo han intentado, pero finalmente no se han terminado. Y como al final se ha decidido realizar llamadas al sistema, el lenguaje que se utilice finalmente no será definido por este tema.

Otro de los puntos que se han tenido en cuenta a la hora de seleccionar el lenguaje de programación, era poder intentar escoger el que tuviera los frameworks que más facilitaran el trabajo de crear un servidor. Al existir un montón de frameworks y una gran variedad de lenguajes de programación, se ha acotado la investigación entre dos lenguajes y sus correspondientes frameworks especializados en la creación de servidores web.

Actualmente dos de los lenguajes más utilizados para crear la parte del servidor de una aplicación web son Java y JavaScript. El primero tiene una plataforma llamada Java Enterprise Edition, la cual está construida sobre Java Standard Edition. Java SE es una especificación que describe una plataforma Java, y proporciona una base para la compilación y despliegue de aplicaciones. Java EE añade numerosas herramientas, además de los conceptos de la capa de aplicación y la capa de presentación. Otra de las funcionalidades que trae JEE son los Servlet. Estos son clases de java que se utilizan en el lado servidor para manejar las peticiones HTTP que recibe, y para generar las respuestas HTTP. Y una tecnología que se combina con los Servlet es JavaServer Pages (JSP). Esta tecnología ayuda a los desarrolladores a crear páginas webs dinámicas. [19]

Cada lenguaje tiene su framework especializado en la tarea de realizar servidores web. En el caso de Java se suele usar Spring [20], y en el caso de JavaScript se utiliza Node.js [21]. En los foros y blogs de tecnología que hay en internet existen grandes discusiones sobre cual es mejor, ya que los dos a grandes rasgos sirven para lo mismo, pero lo cierto es que cada uno tiene sus ventajas y desventajas.

Finalmente se ha decidido programar el Back-End en JavaScript con Node.js. Este es un intérprete JavaScript del lado del servidor, que concretamente ejecuta el motor v8 de JS. Este motor se utiliza en el navegador Chrome, y fue desarrollado por Google [22]. Está basado en la programación orientada a eventos. En este paradigma se espera a que ocurra algo, para reaccionar a ello. Por ejemplo, el servidor puede tener una función que se active cuando se conecta alguien o podría tener una función que esté esperando a que se termine de leer un archivo para luego realizar algo con esa lectura.

Uno de los puntos fuertes de Node es su escabilidad. Grandes empresas como Netflix, Linkedin, Paypal y muchas otras han desarrollado sus nuevas aplicaciones utilizando Node por este motivo [23]. Como se explica en un artículo de IBM [22], en los servidores basados en Java, con cada conexión se genera un nuevo hilo de ejecución, el cual consume 2MB de memoria RAM; de modo que, si se tiene en un sistema con 8GB, el máximo número de conexiones simultáneas son 4000. Para resolver esto, Node con cada conexión nueva dispara un evento en el servidor. Estas conexiones pasan al “event

pool", desde el cual se realiza un seguimiento de las operaciones asíncronas, y al finalizar se ejecuta un callback, una función [24].

Otra de las ventajas de realizar el proyecto en Node, es que está hecho en JavaScript, el cual es un lenguaje bastante fácil de aprender. Además, si este proyecto se realizará mediante el framework Spring, seguramente se debería de desarrollar un controlador, un modelo, un servicio… [25] Con Node no hacen falta todas estas clases, por lo que se reducen bastante las líneas de código.

Teniendo en cuenta que en esta aplicación se tienen que realizar varias operaciones de entrada/salida, Node es ventajoso porque realiza todas estas operaciones bloqueantes de forma asíncrona. Envía estas tareas al "event loop", y ya cuando se terminan de ejecutar, el servidor reacciona según el callback del evento. Está característica hace que Node no se bloquee al trabajar con archivos, y en este proyecto se necesita leer y escribir muchos archivos.

Este framework es un proyecto open source. Los desarrolladores han creado miles de librerías, llamadas módulos. Todos estos módulos están disponible a través del gestor de paquetes más grandes de JavaScript, llamado NPM. Esto demuestra el apoyo de la comunidad que tiene Node.

3.5 BACK-END

En la parte del servidor, como se ha mencionado en el apartado anterior, se usa Node.js. Esta parte es la encargada de procesar los datos de entrada que recibe el Front-end, es decir, es la capa de presentación. Aquí se establecen todas las reglas que debe de seguir la aplicación final.

3.5.1 ORGANIZACIÓN

Para poder comprender mejor el funcionamiento de la aplicación, primero se explicará que ficheros lo conforman, que contiene cada uno, y como están organizados. La figura 5 es una representación gráfica de dicha organización.

En la raíz de la carpeta contenedora de la aplicación está el fichero "index.html", que es la página principal de la aplicación web, le sigue el archivo "app.js", que es el encargado de contener la capa de negocio. Y por último está el fichero "package.json" que contiene información sobre el proyecto, como puede ser su nombre y su versión entre otras cosas; lo más importante de este fichero es que contiene los nombres de los módulos que necesita la aplicación. Esto es bastante útil, ya que en el momento que se

exporte la aplicación al servidor de producción no hará falta instalar cada módulo independientemente, con un comando a través de la terminal se podrán instalar todos los módulos a la vez.

Además de estos archivos existen dos carpetas. La primera se llama “views” y contiene todos los ficheros estáticos, es decir los archivos de formato CSS y los de formato JS. El primero es una hoja de estilos, que define y crea la presentación del documento index.html, o de cualquier archivo en formato HTML que se desee. El segundo permite crear efectos atractivos y dinámicos en las páginas web, además de usarse para realizar la conexión con los servidores

La segunda carpeta se llama “Lógica” (esta carpeta se podría llamar de forma diferente, si se modificaran pocas líneas de código). En ella tienen que estar todas las carpetas de los diferentes temas sobre los que se quiera ofrecer ejercicios.

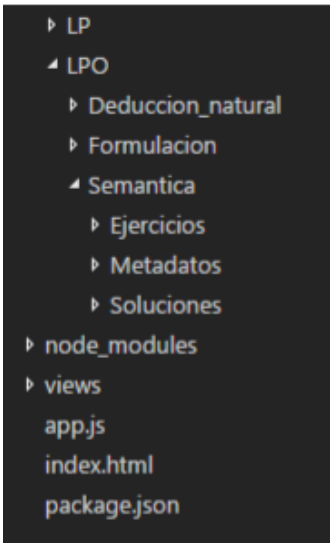


Figura 5 Jerarquía de carpetas

Hay que recalcar que las carpetas de los temas deben contener 3 carpetas: Ejercicios, Soluciones y Metadatos, o al menos contener otro tema en su interior. En el supuesto de que el nombre de la carpeta conste de varias palabras, estas tendrán que ir separadas por una barra baja “_”, y no por un espacio.

Por otro lado, los ficheros de los ejercicios tienen que tener como nombre “ejercicio” más un número identificativo al final. Con los otros ficheros se tiene que realizar lo mismo, pero con el nombre de solución o meta-datos. Los tres ficheros tienen que tener el mismo identificador final. En un principio hay que introducir estos nombres a manos, pero como se explica más adelante, se podría automatizar este proceso.

3.5.2 DIRECCIONAMIENTO

La página está pensada para realizar peticiones de descargas y para acceder a ella. Con el objetivo de prestar un servicio: que los usuarios puedan acceder a unos recursos determinados.

Para ello se han definido dos rutas de acceso. Las rutas definen cómo responde una aplicación a una solicitud de cliente a una determinada URI y un método se solicitud HTTP especifico. En este caso es necesaria para que el navegador web se conecte con el servidor [26]. Para crearla se ha utilizado el módulo llamado "express", que es un framework con características para crear aplicaciones web y móviles.

Con el módulo principalmente se describen las rutas de los recursos, para que el servidor sepa cómo tiene que tratar las peticiones. Para definir una ruta, basta con describir el recurso al que se desea acceder y el método http necesario. En el ejemplo siguiente se ha definido una ruta por la cual si una aplicación realiza una petición GET a la uri "…/usuarios", este tendrá como respuesta una lista de todos los usuarios que almacene el servidor.

```
const express = require("express");

router.get('/usuarios', function(req, res){
 [...]
})
```

Figura 6. Ruta del recurso "index.html"

En las tablas siguientes se describe como acceder a los diferentes recursos del servicio, y sus parámetros.

URI	http://www.url.com/
Método	GET
Cuerpo de la petición	---
Devuelve	200 (OK) + POX(servicio/index +xml)

Tabla 1. Descripción del recurso índex

URI	http://www.url.com/descargar	
Método	GET	
Cuerpo de la petición	Tema =	Nombre de los temas
	Numero =	Cantidad de ejercicios
	Soluciones =	Indicador de interés por las soluciones
	Metadatos =	Indicador de interés por los metadatos
Devuelve	200 (OK) + POX(servicio/index +xml)	

Tabla 2. Descripción del recurso descargar

3.5.3 CREACIÓN DINAMICA DEL DOCUMENTO HTML

Si un usuario accede al primer recurso, o sea a la página de inicio, ésta le presentará un formulario con checkboxs para realizar la petición de descarga. Para que el usuario reciba la página de inicio, el usuario debe de realizar una petición GET al recurso "…\". El servidor reconocerá esta petición como válida, ya que está definida dentro de la capa de negocio. Una vez controlada la petición se ejecuta el código de la figura 7.

```
const express = require("express");
const cheerio = require('cheerio')
[...]
router.get('/', function(req, res){
        const $ = cheerio.load(fs.readFileSync('index.html'));
        html = changeIndex($,__dirname+"/Logica");
        res.writeHead(200, { 'Content-Type': 'text/html' });
        res.end(html.html(), 'utf-8');
})
```

Figura 7. Ruta del recurso "index.html"

Una de las ventajas que tiene el servicio es que cada vez que se crea una carpeta para un tema nuevo, este aparece directamente en la página web, sin necesidad de tocar su código HTML, ni de reiniciar el servidor. Esto se consigue porque la página principal, más concretamente el formulario, se crea dinámicamente, es decir, cada vez que alguien intenta acceder a la página se crea un HTML y se envía.

Pero la página no se crea desde cero, se tiene un esqueleto básico, el cual está en "index.html" (se explicará más adelante como funciona). A partir de esta plantilla se van agregando los elementos que se necesitan, o sea, los títulos de los temas, o los checkboxs de los subtemas, que se utilizan para marcar los ejercicios que se quieran pedir.

Para añadir estos elementos se utiliza el módulo de "cheerio". Este es una implementación del core de Jquery diseñada específicamente para servidores, que consiste en el modelo DOM. Este modelo es una interfaz de programación de aplicaciones (API) para documentos HTML y XML [27]. Define el modo en el que se accede y de manipula un documento. Este módulo es por ello muy utilizado actualmente para hacer Web Scrapping (técnica utilizada para extraer información de sitios web).

Para poder editar el HTML, primero hay que leerlo, y a continuación cargarlo con cheerio. Luego se hace una llamada a la función "changeIndex", que es la encargada de editar el documento, y es la que consigue que el formulario se cree dinámicamente. A ella se le debe de pasar el fichero a modificar, que en nuestro caso es el "index.html", que contiene el esqueleto de la página, y la dirección del sistema en donde se encuentran los temas, que en el ejemplo de la figura 4 sería la carpeta "Lógica" que está en la misma ruta que la aplicación.

```
var changeIndex = function($, carpeta){
        var folders = getFoldersIn(carpeta);
        if(onlyFolders(carpeta)){ // Caso 1
                [...]
        } else if(folders.length>=1) { // Caso 2
                [...]
        } else { // Caso 3
                return$;
        }
    return$;
}
```

Figura 8. Función changeIndex

Una vez que le hemos pasado los argumentos a la función, ésta revisa el contenido de la carpeta, de forma recursiva. Existen tres situaciones posibles. En la primera, la carpeta pasada como argumento contiene en su interior una carpeta llamada “Ejercicios”, otra “Soluciones” y una tercera llamada “Metadatos”, por lo tanto, el algoritmo se habría encontrado con un subtema. En la segunda situación, que la carpeta que se está estudiando no está vacía, pero tampoco es un subtema. Y la última situación posible es que la carpeta está vacía, o que no tiene las 3 carpetas necesarias para ser un subtema.

En el caso de estar en la primera situación, se introduce un nuevo elemento al formulario de la página, un checkbox, para ello se realiza la inserción de la etiqueta <input> con el parámetro “type=checkbox”. El checkbox sirve para seleccionar el tema, si se desea descargar ejercicios sobre él. Para configurar correctamente el botón, la función escribirá en variables el número de ejercicios disponible que hay del subtema, para así definir un límite máximo de ejercicios que se pueden pedir de dicho tema. Para ello se realiza una llamada a la función “numeroArchivos”, que como su nombre indica, devuelve el número de archivos en una carpeta.

También se configurará los parámetros “id” y “value”, que serán necesarios para que el archivo JS pueda identificar que temas son los requeridos por el usuario, y así hacer la petición al servidor. Pondrá en estas variables el nombre del subtema y del tema, es decir, de la carpeta padre. Estos valores son muy útiles ya que el navegador en el caso de que se introduzca un número mayor al máximo disponible, mostrara un mensaje de error, del cual se hablara en la sección de FRONT-END.

```
var max = numeroArchivos(carpeta+'/Ejercicios');
var n = carpeta.lastIndexOf('/');
var str = carpeta.substring(n + 1);
var parent = carpeta.substring(0, carpeta.lastIndexOf('/'));
var n2 = parent.lastIndexOf('/');
var raiz = "/carpetaContendorDelProyecto/";
var parent = parent.slice(parent.indexOf(raiz) + Y.length);
parent = parent.split('/').join('-');
var path = parent +"-"+ str;
str = str.split('_').join(' ');

$('div.temas'+parent).append('<div class="seleccion_hijo"><input type="checkbox" class="input_control"
name="tema" value="'+path+'">'+ '<span class="temas">'+str+'</span><input type="number" class="textbox"
name="numero" id="'+path+'" size="2" maxlength="2"'+ 'value="1" min="1" max="'+max+'"tabindex="1"
style="display:none"/><br></div>');
```

Figura 9. Caso de ser una subcarpeta

En el caso de estar en una carpeta que contiene temas, que es la segunda situación, pueden darse dos casos. El primero que la ruta que se está analizando sea la raíz de los temas. En este caso se crearía una división en el HTML para poder indicar en el formulario de que asignatura son los ejercicios. Esto se realiza insertando la etiqueta en el HTML. En el segundo caso se estaría contemplando que la carpeta no es la raíz, pero tampoco es un subtema. También se crearía una división para indicar en el formulario que los ejercicios tratan sobre ese tema.

En cualquiera de los casos luego se haría una llamada recursiva para analizar todas las carpetas que tienen en su interior. Gracias a esto, se pueden anidar tantas carpetas de temas como se quieran. Siempre y cuando la última de la jerarquía tengas las 3 carpetas mencionadas anteriormente. Y finalmente, si se analizara una carpeta vacía, simplemente se retornaría.

```
var n = carpeta.lastIndexOf('/');
var str = carpeta.substring(n + 1);
var raiz = "/carpetaContendorDelProyecto/";
var ruta = carpeta.slice(carpeta.indexOf(raiz) + Y.length);
ruta = ruta.split('/').join('-');
if(carpeta === __dirname+"/Logica"){
      $('div.seleccion').append('<div class="temas'+str+'"><span
class="temasGen">'+str+'</div>')
} else {
            var parent = carpeta.substring(0, carpeta.lastIndexOf('/'));
            var raiz = "/carpetaContendorDelProyecto/";
            parent = parent.slice(parent.indexOf(raiz) + Y.length);
            parent = parent.split('/').join('-');
            $('div.temas'+parent).append('<div class="temas'+str+'"><span
                        class="temasGen">'+str+'</div>')
}
for(var i = 0; i < folders.length; i++){
            $=changeIndex($,folders[i]);
}
```

Figura 10. Caso de ser la carpeta raíz o un tema

En el momento que la función cahngeIndex termina de ejecutarse, se tiene el HTML final que se le envía al usuario, como se acaba de explicar, este archivo contiene un formulario para seleccionar los temas que interesa descargar. En el caso que se quisiera crear más carpetas, el código seguiría funcionando.

3.5.4 CREACÓN DEL PDF

Por otro lado, si el usuario intentara acceder al otro recurso, es decir al PDF de ejercicios, se ejecuta la ruta siguiente.

```
router.get('/descargar', function(req, res) {
        if(typeof req.query.tema !== 'undefined'){
                var download = Peticion(req, res);
        }
});
```

Figura 11. Ruta del recurso "descarga"

Al entrar en esta parte del código, lo primero que hace la aplicación es comprobar que se ha pedido al menos un tema. En principio, este requisito no puede fallar, pues el código HTML está hecho para obligar al usuario a pedir al menos un ejercicio.

En el caso de que el usuario solicite ejercicios, se llama a la función "petición", esta función recibe como argumentos la petición del cliente, y la respuesta del servidor, que habrá que rellenar. En ella es donde se genera el PDF final.

Al no existir ningún compilador de Latex escrito en un lenguaje moderno, ni existir servidores públicos que implementen "Common LaTeX Service" se ha decidido hacer llamadas al sistema. Una opción podría haber sido desplegar dicho servidor, ya que el código está en Github, pero realmente ésta no hubiera aportada nada, ya que hubieran sido más recursos necesarios (un servidor para Latex y otro para alojar la aplicación en sí) y, además, porque el servidor de Latex hace llamadas al sistema.

Con Node.js se pueden hacer llamadas al sistema, pero se ha decidido utilizar el modulo "node-latex", porque se le puede pasar una cadena de caracteres (es decir, el contenido de un fichero Latex), un fichero Latex, o un array de cadena de caracteres. El módulo es capaz de generar un PDF, que es el formato que nos interesa, pero también puede generar un DVI. Se puede decidir con que programa compilarlo, algunos ejemplos son Latex o pdflatex entre otros. El módulo se encarga de crear carpetas temporales, que en cuanto se han enviado los ficheros, se borran. En nuestro caso le pasaremos un array de Strings, y le pediremos que nos genere un PDF.

En el array iremos introduciendo los diferentes comandos que necesite Latex para generar el archivo (figura 9). Primero debemos de tener en una variable la ruta en la que está la carpeta raíz de los ejercicios. Y por otro lado se tiene que crear un array vacío, que será el que vayamos rellenando.

A la hora de ir introduciendo los comandos hace falta poner una doble barra "\\" en vez de una, que es lo habitual en Latex. Esto mismo ocurre si intentamos compilar un archivo Latex desde la terminal, pasándole como argumento un string.

Primero se introducen comandos que configuran el fichero final. Después se declara que el documento es de tipo exam (como se ha comentado anteriormente, esto servirá para poder dar formato a las preguntas y a las respuestas). A continuación, se comprueba si el usuario ha decidido que quiere las respuestas de los exámenes. En tal caso se utilizará el comando "printanswer". En el caso de que no se activara dicho comando, el fichero final tendría unos huecos vacíos para responder a las preguntas.

Por otro lado, hace falta el paquete de "standalone" para poder juntar los preámbulos de los distintos ficheros. Se necesita activar la opción de "subpreable", para que el fichero principal (que se está construyendo) use los preámbulos de los otros ficheros. También se utiliza el paquete "babel", que es capaz de manejar diferentes reglas tipográficas, como puede ser un acento. Y por último se importa el paquete "geometry" que sirve para dar formato a la página, es decir, para definir opciones del estilo de la página como pueden ser los márgenes [28].

```
function Peticion(req, res){
const pathBase = "/ruta/Logica/"
var comando = [];

comando.push("\\documentclass{exam}");
// Si se ha pedido que el pdf tenga las soluciones.
if(typeof req.query.soluciones !== 'undefined') {
        comando.push("\\printanswers");
}
comando.push("\\usepackage[subpreambles=true]{standalone}");
comando.push("\\usepackage[spanish]{babel}");
comando.push("\\usepackage{geometry}");
comando.push("\\geometry{tmargin=2.5cm,bmargin=2.5cm,lmargin=2.5cm,rmargin=2.5cm}");
```

Figura 12. Configuración del preámbulo del archivo final

Una vez que se ha definido el preámbulo principal, hay que definir un título, utilizar el comando "\begin{document} para que Latex sepa que a partir de este momento aparecerá el cuerpo del documento, y por último iniciar el entorno de preguntas. Esto es necesario para que la clase exam empiece a tener en cuenta los comandos propios de su clase.

```
comando.push("\\title{Ejercicios}");
comando.push("\\begin{document}");
comando.push("\\maketitle");
comando.push("\\begin{flushleft}");
comando.push("\\begin{questions}");
```

Figura 13. Continuación de la configuración del preámbulo

Después de haber introducido en el array los comandos principales que sirven para crear un documento Latex, con las características deseadas, hay que coger el argumento “req” (que es un JSON) que consiste en la información de la petición del usuario. A partir de este parámetro se coge la variable “tema”, que se haya en “req.query”. En esta variable estarán los nombres de los temas que el usuario haya escogido desde el cliente web.

En el caso de que solo se haya escogido un tema, hay que guardar en una variable el nombre del tema que el usuario haya seleccionado. Además, gracias al parámetro “req.query.numero” se puede saber cuántos ejercicios se quiere del tema seleccionado.

También se deben de guardar en 3 arrays, todos los ficheros que hay en la carpeta Ejercicios, en la carpeta Soluciones y en la carpeta Metadatos. Esto es necesario porque se usará para la generación automática de ejercicios.

```
var tema = req.query.tema.split('-').join('/');
var num = parseInt(req.query.numero);
var files = getFiles(pathBase + tema +"/Ejercicios");
var files2 = getFiles(pathBase + tema +"/Soluciones");
var files3 = getFiles(pathBase + tema +"/Metadatos");
```

Figura 14. Extracción de los archivos disponibles

Una vez establecidas estas variables es momento de realizar un bucle. El bucle se realizará tantas veces como la cantidad de archivos que el usuario haya deseado. Dentro de él se escogerá un número aleatorio, acotado entre 1 y la longitud de los arrays anteriormente mencionados, para poder así, seleccionar un índice de ese array. Este índice es el que permite coger el fichero correspondiente y quitarlo del array, para que en la próxima vuelta del bucle no sea tenido en cuenta y así se evita ejercicios repetidos en el archivo final.

```
var aux = Math.floor((Math.random() * files.length));
var rand = files.splice(aux, 1);
var rand2 = files2.splice(aux, 1);
var rand3 = files3.splice(aux, 1);
var ejercicioLatex = pathBase + tema + "/Ejercicios/"+ rand;
var solucionLatex = pathBase + tema + "/Soluciones/"+ rand2;
var metaLatex = pathBase + tema + "/Metadatos/"+ rand3;
```

Figura 15. Selección aleatoria de ejercicios

Teniendo seleccionados los ficheros que se introducirán en el archivo final, se los tiene que importar. Pero previamente hay que especificar si el fichero es una pregunta o una solución. En el caso de que sea una pregunta se especifica utilizando el comando "\question", y si es una respuesta utilizando el comando "\begin{solutionorbox}". Esto se hace para que cada sección tenga su estilo. Por ejemplo: las soluciones salen con un borde negro.

Diferenciar lo que es una pregunta o una solución es imprescindible para que cuando el usuario quiera ver las soluciones, Latex las muestre.

```
comando.push("\\question\\input{"+ejercicioLatex+"}")
comando.push("\\begin{solutionorbox}[2in]");
comando.push("\\input{"+solucionLatex+"}");
comando.push("\\end{solutionorbox}");
comando.push("\\smallskip");
```

Figura 16. Definición de preguntas y soluciones

En el caso de que las soluciones tuvieran que aparecer, también se vería la ruta en donde se encuentra dicha respuesta, por si hubiera alguna errata y se tuviera que corregir. Sin esta forma de identificar el fichero, sería un problema encontrar la solución dentro del repositorio de ejercicios. Y en el caso de que el usuario quisiera meta-información del ejercicio, habría que ver el parámetro "req.query.metadatos".

```
if(typeof req.query.metadatos !== 'undefined'){
comando.push("\\textit{\\underline{Metadatos:}\\input{"+metaLatex+"}}");
}

if(typeof req.query.soluciones !== 'undefined'){
 comando.push("\\underline{Ruta:}"+tema + "/Soluciones/"+rand2);
}
```

Figura 17. Opción de mostrar los metadatos

En el supuesto de que el usuario hubiera elegido varios temas, habría que realizar el mismo proceso recién explicado, solo dentro de un bucle, pero que se repetirá tantas veces como temas se quieran.

```
for(var temaSeleccionado in req.query.tema){
        [...]
}
```

Figura 18. Bucle recorriendo los temas

Finalmente, el array que contiene los comando de Latex, y que se ha ido rellenando según lo descrito anteriormente, se termina de crear. Necesitamos para ello dos comando, el primero para que la clase exam sepa que se ha terminado el entorno de preguntas, y el segundo para terminar el documento.

```
comando.push("\\end{questions}");
comando.push("\\end{flushleft}");
comando.push("\\end{document}");
const output = fs.createWriteStream('/ruta/output'+req.log.id+'.pdf');
const pdf = latex(comando).pipe(output);
```

Figura 19. Finalización del archivo Latex y su compilación

Una vez terminado el array de Latex, es hora de compilarlo y convertirlo a PDF. Para ello se hace una llamada al módulo “Latex” (explicado anteriormente), al cual se le pasa el array, y una variable que contiene la ruta en la que se debe de crear el fichero. Esta ruta es diferente para cada petición que reciba la aplicación, así si hay varias peticiones a la vez, se sabe que fichero hay que mandar a cada petición. Para diferenciar estas peticiones, la aplicación les otorga un identificador único. Esto se realiza antes de entrar a cualquiera de las rutas disponibles en la API. Para ello se declara una variable global, a la cual se le va añadiendo 1 cada vez que haya una petición. El identificador se introduce en el “log” de la petición.

```
var logIdIterator = 0;
[...]
app.all('*', function(req, res, next) {
req.log = {
        id: ++logIdIterator
}
        return next();
});
```

Figura 20. Asignación de identificador único a la petición

Node.js utiliza un sistema de eventos. Mientras que una función está haciendo algo, se puede ir ejecutando otras cosas del código, y cuando la función que se estaba ejecutando termina, manda un evento. Existen varios tipos de eventos, algunos son “finish” o “error”. Entonces, la función que está asociada al evento que se desea se activa, y se ejecuta. Esto es lo que se realiza en la figura 17.

Cuando se termina el proceso de creación del archivo PDF se ejecuta el método que tiene como función enviar el archivo. Para poder enviar un archivo desde el servidor, es necesario configurar la cabecera de la respuesta http. Para ello primero se debe especificar el tamaño del archivo, utilizando la función “stat” que da información sobre él y luego se debe de especificar el tipo de archivo que es, y el nombre del archivo que se va a adjuntar.

```
pdf.on('finish', function (err) {
if(err){
        console.log("Error");
} else {
        fs.stat('/ruta/'+req.log.id+'.pdf', function (err, stats) {
        if (err){
                console.log(err);
        } else {
                res.setHeader('Content-Length', stats.size);
                res.setHeader('Content-Type', 'application/pdf');
                res.setHeader('Content-Disposition', 'attachment;
filename=Archivo.pdf');
                var send = fs.createReadStream('/ruta/'+req.log.id+'.pdf').pipe(res);
  // Continua en la siguiente figura
```

Figura 21. Configuración de la cabecera de la respuesta y su envío

Una vez se tiene la cabecera de la respuesta configurada, y también el fichero PDF final, es hora de enviárselo al usuario.

Para el envío también existe un método que está a la espera de su terminación. Éste lo que realiza es la eliminación del fichero generado, para no tener acumulados todos los pdfs que se van generando. Esta era una de las causas por las que era interesante tener un identificador para cada petición.

```
send.on('finish' ,function(err){
                    if(!err){
                            fs.unlink('/ruta'+req.log.id+'.pdf', function(err){
                                    if(err) console.log(err)
                            })
                    }
                    else console.log(err);
                    });
            }
            });
            }
});
```

Figura 22. Eliminación de los ficheros generados

La parte del Back-end tiene algunas funciones auxiliares, las cuales no se han mencionado porque pueden ser más triviales.

3.6 FRONT-END

A la hora de realizar en Front-end de una aplicación hay que tener en cuenta la estética, la funcionalidad y la usabilidad. Si alguna no se consigue de manera correcta, puede ocasionar una mala experiencia para el usuario, y al final el trabajo que hay detrás no servirá para nada. Es por ello que hay que tener muy en cuenta la parte que ve el usuario.

Hoy en día, en cuanto a la estética se refiere, todas las aplicaciones intentan tener una interfaz limpia y minimista, para que no entorpezca la usabilidad de éstas. Pero el punto que más hay que tener en cuenta es la usabilidad. La gran mayoría de proyectos solo se centran en cómo crear un sistema, un método… en cómo resolver un problema, pero no le dedican tanto tiempo a que ese trabajo sea usable.

La gente cada vez se conecta más desde los móviles. De hecho, existen más móviles que personas en el mundo. Un gran número de personas prefieren tener un móvil inteligente que un ordenador. Es por eso que el sector de las aplicaciones webs orientadas a móviles se encuentra en auge. Si una empresa quiere sacar su aplicación para iOS y para Android tiene que tener dos equipos de desarrollo, lo que

supone un gran coste. Pero si se decide crear una aplicación híbrida, es decir, web, solo hay que desarrollar una, lo que implica abaratar los costes. Obviamente, cada sistema tiene sus ventajas y desventajas, por ello es importante hacer un análisis previo antes de crear la App.

En lo que se refiere al trabajo, con una aplicación Web resulta suficiente. Se ha decidido crear una página muy simple, en la que se explica en que consiste el proyecto, un poco de información sobre ella, y un formulario que sirva para realizar las peticiones. Con una sola vista se puede abarcar todo el proyecto. Para crear la página de la figura 23 se han utilizado las tecnologías de HTML5, CSS3 y JavaScript. La primera de estas tecnologías nombradas es un lenguaje de marcado de hipertextos, es decir, un lenguaje de etiquetas. Es la base de cualquier contenido web. HTML es el lenguaje que describe la estructura y el contenido de una página web. Además, separa el contenido de la presentación. Para describir estos documentos se utilizan diferentes tipos de elementos que son definidos mediante etiquetas. Estas etiquetas van entre los símbolos "< … >". La última versión es la 5, HTML5. Esta versión trae nuevos elementos y atributos. [29]

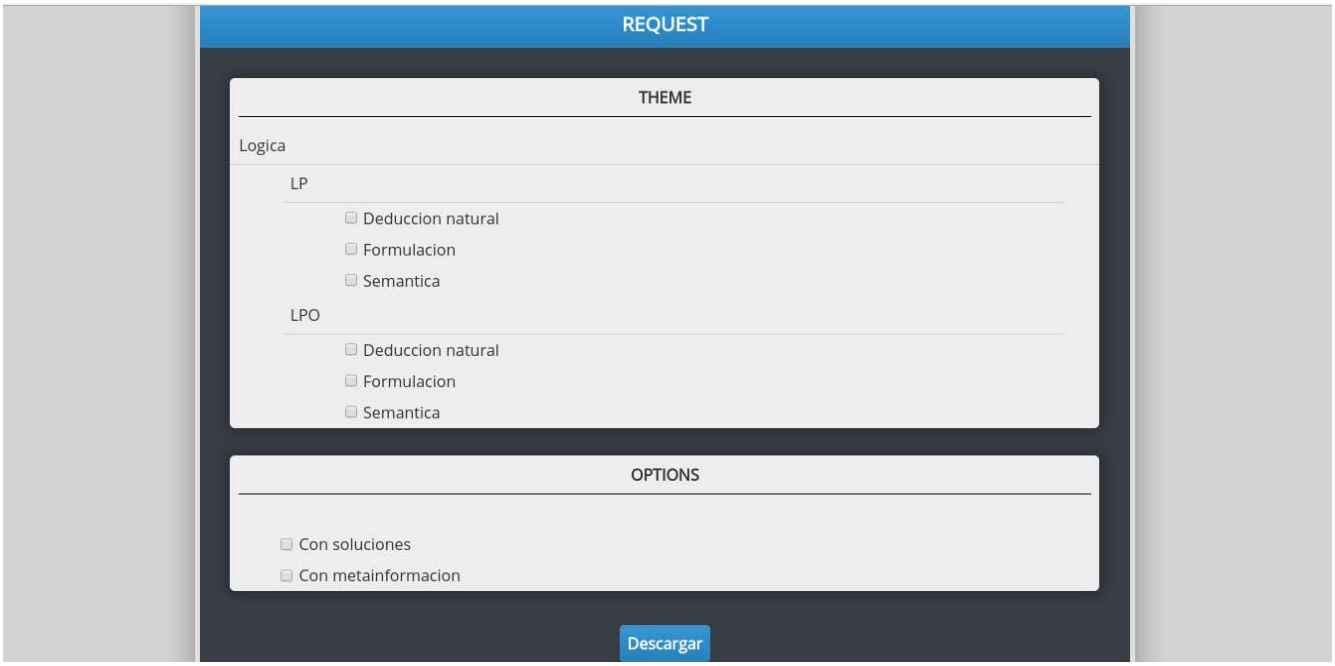


Figura 23. Formulario visto desde un ordenador

La segunda es la tecnología CSS3, también llamada Hojas de Estilo en Cascada, es un mecanismo que describe cómo se va a mostrar un documento en la pantalla. Esta forma de descripción de estilos ofrece a los desarrolladores el control total sobre el estilo y formato de sus documentos. Esta es la tecnología clave para realizar una web usable.

CSS funciona a base de reglas, es decir, declaraciones sobre el estilo de uno o más elementos. Los documentos CSS están compuestos por una o más de esas reglas. Las reglas constan de dos partes: un selector y una zona de atributos. Está zona está compuesta por propiedades y el valor que se les asigne [30].

Un ejemplo serio “p {color: blue}”. En este caso se estaría seleccionando todas las etiquetas de tipo “p” y se les estaría aplicando el color azul al texto de dicha etiqueta.

Y la tercera tecnología utilizada es JavaScript. Esta es un lenguaje de programación interpretado, con el que se realizan mejoras en la interfaz de usuario y páginas web dinámicas. El uso más común que se le da a JavaScript es la interactuación con el DOM. [31]

En el proyecto se utiliza para tener un contenido interactivo. Si desea descargar ejercicios de algún tema, se debe seleccionar su checkboxs asociado. Al presionar sobre el checkbox aparece un campo de texto vacío a su lado que sirve para introducir el número de ejercicios que se desea pedir.

También se utiliza para realizar validaciones de los valores de entrada de los formularios, para así asegurar que los datos que se envían al servidor son aceptables. En la web descrita en este trabajo se verifica mediante JavaScript si se ha seleccionado un tema antes de realizarse la petición.

Además de estas 3 tecnologías, también se ha utilizado un framework llamado Bootstrap [32, 33]. Éste es un conjunto de herramientas pensado para el diseño de sitios y aplicaciones web. Ofrece plantillas de diseño tipográfico, diseño de formulario, botones, menús y otros elementos. Utilizando este framework se consigue crear páginas webs en menos tiempos, con una buena estética, y que además sean responsive.

Este último punto es clave dentro de cualquier desarrollo web actual, ya que como se ha comentado con anterioridad, la gente se conecta cada vez más desde los móviles a internet para realizar cualquier tarea. En la figura 23 se puede apreciar cómo queda la página siendo utilizada desde un móvil.

Para conseguir que la página se visualice según el tamaño de la pantalla, en el documento HTML se ha incluido una etiqueta <meta>, la cual no es visible para el usuario, pero gracias a él, el navegador es capaz de tener más información sobre la página. Dentro de esta etiqueta se establece el parámetro “content” con valor “width=device-width”. Con esto se consigue saber el ancho de la pantalla.

Por otro lado, dentro de la hoja de estilo, se ha usado la regla “@media”. Esta regla es usada para definir diferentes reglas de estilo para cada dispositivo. En este caso interesa crear un estilo para los

dispositivos móviles. Una de las características que tienen todos los móviles inteligentes es su gran densidad de pixeles por pulgadas. El estándar hoy en día es tener una pantalla full hd (1920x1080) en tan solo 5 pulgadas.

Figura 24. Formulario visto desde un smartphone

Esta característica se puede utilizar para que CSS identifique si se encuentra delante de un móvil. Con la regla "@media" se puede establecer una condición llamada "min-resolution", para dar un estilo según la densidad de píxeles del dispositivo. En el proyecto se ha establecido como un mínimo de densidad de píxeles 150. Esto quiere decir, que, si la densidad de pixeles de un dispositivo son 150, se aplicarán las reglas que aparecen a continuación.

Además de las validaciones que se realizan mediante JavaScript, gracias a unas etiquetas HTML se hacen otras validaciones. Se definen valores máximos y mínimos que establecen el rango de ejercicios que se pueden pedir de cada tema. Esto se hace para que el usuario no pueda cometer fallos y para que al servidor le llegue la información que necesita. De todas formas, el Back-end comprueba el número de ejercicios que se han pedido. En las siguientes figuras se pueden observar los mensajes que muestra el navegador en caso se introducir cantidades invalidas.

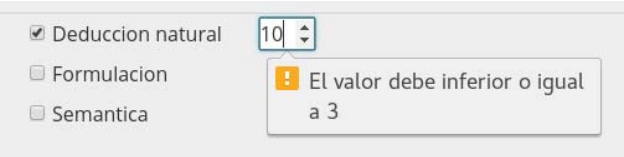


Figura 25. Mensaje de error número 1

Figura 26. Mensaje de error número 2

En la siguiente figura se muestra el mensaje de error que se le indica al usuario si no ha seleccionado ningún tema, pero ha hecho una petición, ya sea teniendo algunas de las opciones activadas o no. Los archivos que contienen el fichero CSS y JS están en la carpeta views en el lado servidor.

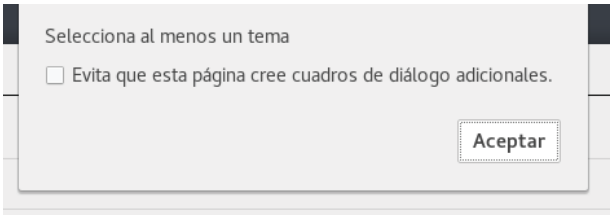


Figura 27. Mensaje de de error número 3

4 DESPLIEGUE

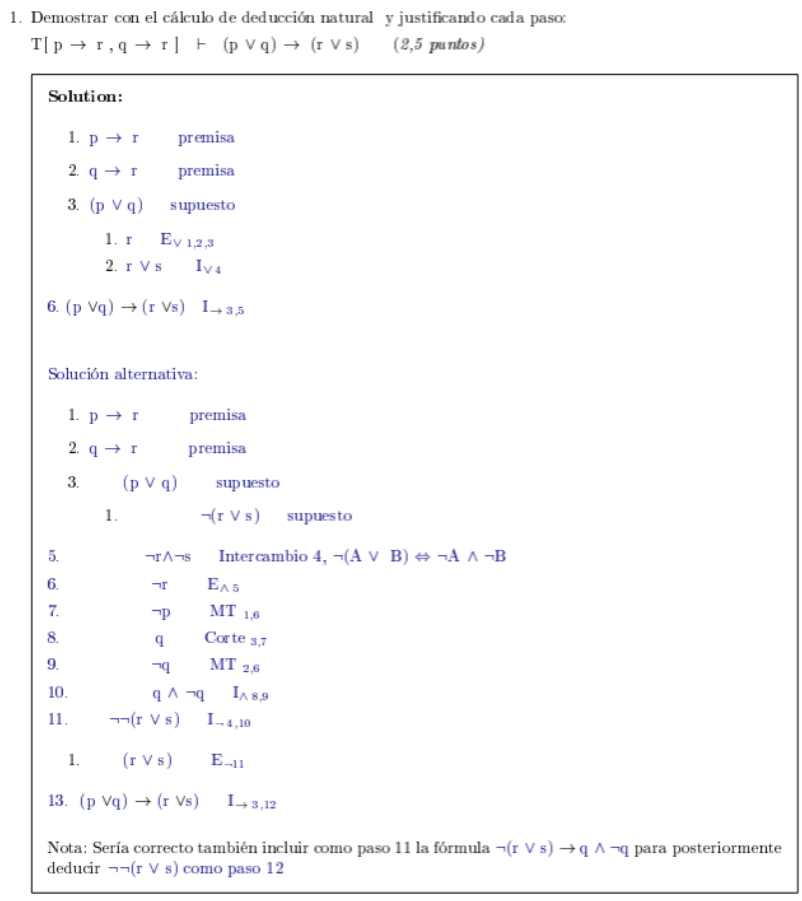
Una vez realizado el código del servicio, es momento de probarlo, pero antes se ha querido desplegar el servicio en un entorno real. Esto quiere decir que el servicio sea ha subido a un servidor, y es accesible desde cualquier lugar del planeta. Para poder desplegar una aplicación web es necesario un host que hospede nuestro servicio. Hoy en día existen una gran cantidad de empresas que ofrecen estos servicios. Entre los servicios más famosos están Amazon, Azure de Microsoft o Heroku. Como esta aplicación necesita realizar llamadas al sistema, y más concretamente a Latex, es necesario tener una máquina virtual, ya que se tendrá que instalar algún software como Latex. Finalmente se ha desplegado la aplicación en una instancia de Amazon EC2 [34], que como ellos mismos describen, es un servicio web que proporciona capacidad informática en la nube, es decir, que otorgan espacio para hacer funcionar nuestras aplicaciones. Para el servicio descrito en este trabajo se ha escogido una instancia con Ubuntu Server 16,04 como sistema operativo. En cuanto a hardware tiene un disco solido con capacidad de 8Gb, 1Gb de memoria ram, y un procesador de la familia Xeon a 2,5GHz. Una máquina con estas características es suficiente para esta aplicación, ya que de momento no tiene muchos ejercicios ni muchas peticiones de usuarios. Si en algún momento se necesitaran más recursos, se puede configurar desde un panel que tiene Amazon. Por otro lado, durante el proceso de configuración de la instancia se consigue una clave que será necesaria para conectarse de forma remota por ssh desde nuestro ordenador. Gracias a esto podremos subir archivos a la instancia, configurarla, o ver los informes del servidor. Una vez subido los archivos, simplemente hay que instalar todos los módulos de Node necesarios, y configurar las tablas ip de la instancia, para que redirija las peticiones del puerto 80 al deseado. Una vez configurado esto, se debe de abrir una terminal y ejecutar “node <<nombre de la aplicación>>”.

5 PRUEBAS

Una vez desplegada la aplicación, ya se la puede probar en su fase final. Para ello han hecho falta algunos ejercicios de muestra. Estos ejercicios son de exámenes de la asignatura de Lógica. En un principio estaban en formato DOCX, por lo que se han tenido que pasar a Latex.

Para automatizar dentro de lo posible esta tarea se ha utilizado un plugin de LibreOffice [35]. La función de este módulo es convertir archivos en formato .doc y. odt entre otros, a Latex. No lo realiza de forma perfecta, pero sí que agiliza la tarea.

Las pruebas que se han realizado son las que se describen en el apartado de "ARQUITECTURA Y CASOS DE USO". Pero también se ha comprobado que la página web mostrara los diferentes mensajes de errores descritos anteriormente en los momentos adecuados.



1. Demostrar con el cálculo de deducción natural y justificando cada paso:
   $T[\, p \to r\,,\, q \to r\,] \vdash (p \vee q) \to (r \vee s)$ *(2,5 puntos)*

Solution:

1. $p \to r$ premisa
2. $q \to r$ premisa
3. $(p \vee q)$ supuesto
   1. r $E_{\vee\, 1,2,3}$
   2. $r \vee s$ $I_{\vee\, 4}$

6. $(p \vee q) \to (r \vee s)$ $I_{\to\, 3,5}$

Solución alternativa:

1. $p \to r$ premisa
2. $q \to r$ premisa
3. $(p \vee q)$ supuesto
   1. $\neg(r \vee s)$ supuesto

5. $\neg r \wedge \neg s$ Intercambio 4, $\neg(A \vee B) \Leftrightarrow \neg A \wedge \neg B$
6. $\neg r$ $E_{\wedge\, 5}$
7. $\neg p$ MT $_{1,6}$
8. q Corte $_{3,7}$
9. $\neg q$ MT $_{2,6}$
10. $q \wedge \neg q$ $I_{\wedge\, 8,9}$
11. $\neg\neg(r \vee s)$ $I_{\neg\, 4,10}$
    1. $(r \vee s)$ $E_{\neg 11}$

13. $(p \vee q) \to (r \vee s)$ $I_{\to\, 3,12}$

Nota: Sería correcto también incluir como paso 11 la fórmula $\neg(r \vee s) \to q \wedge \neg q$ para posteriormente deducir $\neg\neg(r \vee s)$ como paso 12

Figura 28. Ejemplo de documento final

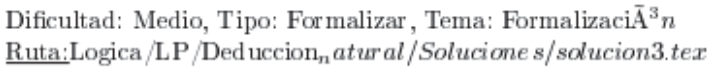


Figura 29. Ejemplo de documento final

En la figura 28 y 29 se puede observar el resultado de realizar una prueba en la que se ha pedido un ejercicio, con la solución y su meta-información. Los ejercicios salen enumerados y las respuestas salen con un marco como se comentó en la sección de REPOSITORIO DE ARCHIVOS LATEX. Se imprimió también la ruta donde se encuentra la solución, por si hiciera falta realizar una corrección, y como se ha realizado una prueba en la que se pedía información extra, esta también sale en el archivo final (figura 29).

La aplicación es funcional desde cualquier lugar del mundo, y se pueden descargar ejercicios, con sus soluciones si se desea. A pesar de esto, durante la fase de desarrollo se han ido encontrando varios bugs que se han ido solucionando.

Una gran cantidad de errores eran causado en gran parte por el paquete de Latex “Standalone”, que era el necesario para unir archivos. Cuando se utiliza este paquete, y se compila el fichero Latex que lo contiene, “Standalone” genera un archivo en formato “. sta”, el cual tiene los preámbulos de los archivos que se están importando al documento principal. Pero no se generaba el documento PDF correctamente. Era necesario compilar el archivo principal de nuevo, para que se tuviera en cuenta el archivo de formato “. sta” mencionado anteriormente.

Para arreglar esto se ha tenido que modificar el módulo de Latex [36]. En él se ha buscado la zona en la que se compilaban los archivos. Una vez identificada esa zona, se han añadido un par de líneas de código, para que se compile dos veces. Se podría modificar el módulo para que uno de los parámetros que se le pase indique la cantidad de veces que se quisiera compilar el archivo. Esto sería útil porque el paquete standalone no es el único motivo por el cual a veces es necesario compilar los archivos dos veces.

6 CONCLUSIONES

Vivimos en la era de la tecnología. Todos los días aparecen nuevos artículos científicos en los que se presentan ideas que pueden ayudar al ser humano en algún ámbito. En cuanto al área de la informática, se están creando diariamente sistemas y servicios que tienen como fin automatizar tareas, o al menos hacer que las tareas sean más fáciles.

Uno de los problemas es que en general se centran en describir cómo se puede conseguir dicha mejora, con que método, con qué modelo, con qué servicio, pero no se preocupan en como el usuario va a interactuar con él. Este es un punto muy importante porque en el supuesto de que sea difícil de interactuar con él, se dejará de usar, y todo el trabajo que hay detrás también.

En cuanto al área del "estudio electrónico" existen varios proyectos para generar ejercicios de forma automática, como se ha visto en el capítulo de "ESTADO DEL ARTE". Pero si no se crean APIs o software con una interfaz usable estos trabajos no llegarán al público. Es por eso que en cualquier proyecto de software es importante desarrollar ya sea una interfaz para éste, o un interfaz de programación de aplicaciones.

En este caso se ha creado una aplicación que facilita la descarga de ejercicios aleatorios, creando una aplicación web, que también funciona en móvil. Este proyecto se podría enriquecer bastante si se complementara con algún proyecto que generara ejercicios de forma automática en formato Latex. Este proyecto puede resultar de bastante utilidad para profesores y alumnos, pues es una herramienta de fácil acceso que les permite automatizar una tarea de búsqueda que suele ocupar gran parte de su tiempo,

7 FUTUROS TRABAJOS

Para futuros trabajos, a la aplicación se le podría poner un sistema de autenticación, para permitir usar el servicio sólo a las personas autorizadas. Además, podría implementarse un sistema de roles, en el cual existan dos tipos, los profesores y los alumnos. Y así según quien que se conectara a la web, tendría un formulario diferente, o bien podría al profesor aparecerle más opciones.

Una de estas opciones sería la posibilidad de subir ejercicios a la plataforma. Para ello se tendría que crear un formulario en el cual se le obligue al profesor a adjuntar 3 archivos. Uno del ejercicio, otro de la solución, y el de los metadatos. También, se le pediría que introduzca a que tema se corresponden los ejercicios. O para facilitar más el trabajo, solo tendrían que adjuntar el fichero del ejercicio y la solución, ya que se les podría presentar un formulario con campos específicos como la dificultad del ejercicio, y a raíz de los valores de entrada del formulario se crearía un archivo de metadatos. Así el servicio podría asegurar que todos los ejercicios tienen la misma información.

El nombre de los archivos no sería muy importante, ya que luego el servidor los cambiaría de acuerdo al tipo de archivo (ejercicio, solución o metadato) y les incluiría al final de este un número identificador igual. Esto evitaría que los usuarios se tengan que preocupar de ciertos detalles, y además impediría que la aplicación tuviera demasiados fallos.

8 Bibliografía

[1] A. Papasalouros, «Automatic generation of multiple choice questions from domain ontologies.,» 2008.

[2] T. Kojiri, T. Goto, T. Watanabe, T. Iwata y T. Yamada, «Automatic Generation System of Multiple-Choice Cloze,» 2010.

[3] A. Z. Bettina Grun, «Automatic Generation of Exams in R».

[4] Wikipedia, «Sweave,» [En línea]. Available: https://es.wikipedia.org/wiki/Sweave. [Último acceso: 2 6 2017].

[5] K. Wang, T. Li, J. Han y Y. Lei, «Algorithms for Automatic Generation of Logical Questions on Mobile Devices,» 2012.

[6] J. Ripoll, J. Soler, J. Poch, E. Barrabés y D. Juher, «A tool for the continuous assessment and improvement of the student´s skills in a mathematics course,» 2002.

[7] «LaTeX, a typesetting system,» [En línea]. Available: https://www.latex-project.org/. [Último acceso: 2 6 2017].

[8] «Guía rápida de LaTeX,» [En línea]. Available: http://nokyotsu.com/latex/guia.html. [Último acceso: 2 6 2017].

[9] G. Neugbau, «User's Guide of ExTeX,» [En línea]. Available: http://www.extex.org/documentation/man/extex-users.pdf.

[10] «javaTex, a implementation of TeX in Java,» [En línea]. Available: https://www.ctan.org/pkg/javatex. [Último acceso: 1 6 2017].

[11] «Common LaTeX Service Interface,» [En línea]. Available: https://github.com/sharelatex/clsi-sharelatex. [Último acceso: 1 6 2017].

[12] «ShareLatex, "El editor de LaTeX fácil de usar,» [En línea]. Available: https://es.sharelatex.com/. [Último acceso: 1 6 2017].

[13] «texlive.js, a port of TexLive 2016 to JavaScript,» [En línea]. Available: https://github.com/manuels/texlive.js/. [Último acceso: 1 6 2017].

[14] Wikipedia, «Diagramas de caso de usos,» [En línea]. Available: https://es.wikipedia.org/wiki/Diagrama_de_casos_de_uso. [Último acceso: 2 6 2017].

[15] Microsoft, «Diagramas de casos de uso de UML: Instrucciones,» [En línea]. Available: https://msdn.microsoft.com/es-es/library/dd409432.aspx. [Último acceso: 1 6 2017].

[16] «Manual de LaTeX,» [En línea]. Available: https://es.wikibooks.org/wiki/Manual_de_LaTeX/Texto_completo#Paquetes. [Último acceso: 1 6 2017].

[17] «Standalone,» [En línea]. Available: https://www.ctan.org/pkg/standalone. [Último acceso: 2 6 2017].

[18] «Exam,» [En línea]. Available: https://www.ctan.org/pkg/exam. [Último acceso: 1 6 2017].

[19] Oracle, «Servlets y JavaServer Pages,» [En línea]. Available: https://docs.oracle.com/cd/E17904_01/web.1111/e13712/basics.htm#WBAPP124. [Último acceso: 7 6 2017].

[20] «Spring, framework de java,» [En línea]. Available: https://spring.io. [Último acceso: 1 2 6].

[21] «Nodejs, entorno de ejecución de js,» [En línea]. Available: https://nodejs.org/es/. [Último acceso: 1 6 2017].

[22] IBM, «¿Simplemente qué es Node.js?,» [En línea]. Available: https://www.ibm.com/developerworks/ssa/opensource/library/os-nodejs/index.html. [Último acceso: 1 6 2017].

[23] Quora, «What companies are using Node.js in production?,» [En línea]. Available: https://www.quora.com/What-companies-are-using-Node-js-in-production. [Último acceso: 2 6 2017].

[24] P. Roberts, «What is the event loop?,» [En línea]. Available: https://www.youtube.com/watch?v=8aGhZQkoFbQ&t=179s. [Último acceso: 2 6 2017].

[25] «Web MVC framework,» [En línea]. Available: https://docs.spring.io/spring/docs/current/spring-framework-reference/html/mvc.html. [Último acceso: 2 6 2017].

[26] Quora, «What is an API?,» [En línea]. Available: https://www.quora.com/What-is-an-API-4. [Último acceso: 2 6 2017].

[27] J. Robie, «¿Qué es el Modelo de Objetos del Documento?,» [En línea]. Available: http://html.conclase.net/w3c/dom1-es/introduction.html. [Último acceso: 2 6 2017].

[28] «geometry – Flexible and complete interface to document dimensions,» [En línea]. Available: https://www.ctan.org/pkg/geometry. [Último acceso: 1 6 2017].

[29] Mozilla, «Introducción al HTML,» [En línea]. Available: https://developer.mozilla.org/es/docs/Web/Guide/HTML/Introduction_alhtml. [Último acceso: 2 6 2017].

[30] w3c, «Guía Breve de CSS,» [En línea]. Available: http://www.w3c.es/Divulgacion/GuiasBreves/HojasEstilo.

[31] Wikipedia, «JavaScript,» [En línea]. Available: https://es.wikipedia.org/wiki/JavaScript. [Último acceso: 2 6 2017].

[32] Wikipedia, «Bootstrap (framework),» [En línea]. Available: https://es.wikipedia.org/wiki/Bootstrap_(framework). [Último acceso: 2 6 2017].

[33] «Bootstrap, framework de html, css y js,» [En línea]. Available: http://getbootstrap.com/. [Último acceso: 2 6 2017].

[34] «Amazon Web Service EC2,» [En línea]. Available: https://aws.amazon.com/es/ec2. [Último acceso: 8 6 2017].

[35] «Writer2LaTeX,» [En línea]. Available: https://extensions.openoffice.org/en/project/writer2latex. [Último acceso: 2 6 2017].

[36] mikolalysenko, «node.js wrapper for LaTeX,» [En línea]. Available: https://github.com/mikolalysenko/node-latex. [Último acceso: 1 6 2017].

Este documento esta firmado por

Firmante	CN=tfgm.fi.upm.es, OU=CCFI, O=Facultad de Informatica - UPM, C=ES
Fecha/Hora	Thu Jun 08 19:05:15 CEST 2017
Emisor del Certificado	EMAILADDRESS=camanager@fi.upm.es, CN=CA Facultad de Informatica, O=Facultad de Informatica - UPM, C=ES
Numero de Serie	630
Metodo	urn:adobe.com:Adobe.PPKLite:adbe.pkcs7.sha1 (Adobe Signature)