Trabajo de Fin de Grado

Grado en Ingeniería Informática

Automatización de los procesos de evaluación de código

Automation of code evaluation processes

Omar Adolfo Álvarez Hernández

La Laguna, 5 de septiembre de 2017

D. **Israel López Plata**, con N.I.F. 42.193.800-W personal docente e investigador de la Universidad de La Laguna, como tutor.

D. **Christopher Expósito Izquierdo**, con N.I.F. 78.851.649-J personal docente e investigador de la Universidad de La Laguna, como cotutor.

C E R T I F I C A (N)

Que la presente memoria titulada:

"Automatización de los procesos de evaluación de código"

Ha sido realizada bajo su dirección por D. **Omar Adolfo Álvarez Hernández**, con N.I.F. 45.371.06-B.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 5 de septiembre de 2017.

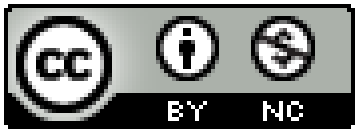
Agradecimientos

Agradecer a mis padres, a mi hermano, a mi abuela y al resto de mi familia por siempre ayudarme en todo y aportarme confianza y ánimo para lograr mis objetivos. Además, agradecerles mucho su paciencia y generosidad ya que sin ellos esto no hubiera sido posible.

Agradecer a mis amigos y a mi novia por siempre animarme y ayudarme, ya que de esta forma han posibilitado siempre que tenga la fuerza suficiente para alcanzar mis metas.

Y por último agradecer a mi tutor y cotutores por darme la oportunidad de hacer un trabajo de fin de grado tan interesante a la vez que tan educativo ya que me ha permitido conocer una tecnología súper útil hoy en día.

Licencia



Resumen

El objetivo de este proyecto es automatizar los procesos de evaluación de las prácticas de programación dentro del ámbito educativo, creando para este propósito una plataforma web que permita tanto alumnos como a profesores poder realizar la ejecución y corrección de las prácticas dentro de dicha plataforma.

Por un lado, los profesores se verían beneficiados ya que se les facilitaría la corrección de las prácticas abstrayéndose de la evaluación individual de los alumnos ya que esa tarea sería propia de este proyecto, y por el otro lado, los alumnos contarían con una herramienta que les permitiera conocer el grado de completitud que poseen sus códigos fuentes y un conocimiento más claro de los objetivos que se esperan conseguir de la realización de dicha práctica.

Dentro de la plataforma web se ha buscado conseguir englobar todas las herramientas necesarias para llevar a cabo las labores de evaluación de código. Además, se busca abstraer en gran medida la participación de los profesores y alumnos, de tal manera que sólo se deben encargar de subir el contenido necesario para llevar a cabo una sesión de evaluación, dejando en manos de la plataforma todo el trabajo que lleva dicha labor. Y, por último, se ha querido añadir otros módulos útiles como la generación de informes que ayuden a conocer las dificultades de una determinada práctica de programación y la capacidad de detectar plagios entre códigos fuentes de diferentes alumnos, ya que esta plataforma pertenece al sector educativo y es uno de los problemas que se busca eliminar.

Palabras clave: Spring, evaluación de código, pruebas, plataforma web, agilizar procesos evaluativos, programación.

Abstract

The objective of this project is to automate the evaluative processes of the programming practices within the educational field, creating a web platform that allows both students and teachers to execute and mark the aforementioned practices.

On one side, teachers would benefit from this as the marking process would be made easier by abstracting the practices from the individual evaluation of students, this task being typical of the project itself. On the other side, students would count with a feature that would allow them to know the grade of completeness that their source codes possess, and a clearer consciousness of the objectives to be met after completion of practice.

Within the web platform, aiming at covering all the necessary features to accomplish the evaluation of code tasks has been key. Moreover, it has been attempted to abstract as much of the students and teachers' participation so that they only need to be in charge of uploading the necessary content to perform an evaluation session, leaving at the platform's hands all the work that this session comprises. Finally, we have wanted to implement other useful modules, including report generation (helping users know the difficulty of a given programming practice), or the ability to detect plagiarism between different students' source codes, which is one of the problems that can be found in an educational platform, therefore being one of the targeted issues.

Keywords: Spring, code evaluation, tests, web platform, streamline evaluative processes, programming.

Índice General

Capítulo 1. Introducción

1.1. Programación en el ámbito educativo

La importancia de la programación ha sido crucial en el avance de la civilización gracias al aporte de nuevas tecnologías que han ayudado y facilitado el manejo de la información y la ejecución y sistematización de tareas complejas como cotidianas. Por ello, se ha convertido en una herramienta fundamental para diversos ámbitos y carreras profesionales que se han visto beneficiados por el avance y los aportes de nuevas tecnologías, ejemplo de ello, en el ámbito sanitario, aplicaciones en el diagnóstico por imagen, la telemedicina, los sistemas de gestión hospitalaria y el registro clínico electrónico nos permiten ver que la programación ha sido de gran ayuda.

Apoyando lo dicho, vivimos en la era de la tecnología y podemos observar que los conocimientos en la informática son los más demandados a día de hoy, propiciando de esta manera que sea necesario que estos nuevos conocimientos y utilidades se hayan empezado a impartir en los últimos años como asignaturas universitarias en carreras como Matemáticas o Ingeniería electrónica entre otras.

1.2. Sistema de evaluación en prácticas de programación

El sistema de corrección de prácticas, a través de mi experiencia en la universidad (existen otros tipos de correcciones de prácticas, en los que no se va a entrar en detalles, puesto que únicamente me centraré en mi experiencia personal) que se sigue a día de hoy en las asignaturas de programación, ha consistido que en las clases de teoría se expliquen conocimientos de un lenguaje de programación o algoritmos con propósitos determinados, y luego plasmar en las clases de prácticas a través de una tarea determinada la adquisición de dichos conocimientos. De esta forma, el alumno validaba la calidad de su código mediante un caso práctico propuesto por el profesor, y a partir de ello, si era capaz o no de superar dicho caso, obtenía una cierta nota al respecto.

En este proceso evaluativo se aprecia con la experiencia tanto por el profesorado como por el alumno que hay diversas carencias:

- Realización de prácticas cuyo funcionamiento no era el esperado, ya que el alumno no cuenta con una información precisa y detallada de lo que se espera conseguir en una determinada práctica.
- Falta de pruebas para comprobar con exactitud si un código está cumpliendo los propósitos de la práctica.

- Falta de tiempo para que, si se cuenta con un número grande de casos prácticos el profesor, pueda ver que el código de un alumno supera cada uno de ellos.
- Inconsistencia de código que no aprecia casos muy concretos.
- Puesta la misma calificación a dos alumnos, a pesar de que uno de ellos tenga un mejor código, ya que sólo se ha comprobado la calidad de dicho código con un número reducido de casos prácticos.

1.3. Importancia de la evaluación de software

Muchas de las carencias nombradas anteriormente, tienen relación con la falta de la evaluación de software, ya que es una pieza fundamental dentro de la programación. El hecho de que sea una pieza tan importante es porque está muy ligado con la calidad del software, lo que significa que un código dado cumpla con todas las funcionalidades para poder realizar una determinada tarea.

Es muy importante que nuestro código cumpla con una cierta calidad mínima como para poder cumplir con unas necesidades dadas, y por ello es necesario que la calidad del código este presente a la hora de entregar una práctica, porque esto significaría que el código proporcionado por un alumno, cumple al menos con las necesidades especificadas en el informe de una práctica y que le permita, ya que es el objetivo, superar la misma.

Por ello, es necesario que además de que el alumno evalué su propio código mediante pruebas unitarias, el profesor también sea capaz de especificar con exactitud los objetivos y metas que se pretenden conseguir. De esta forma, se pretende aunar ambas cosas con el objetivo de que, mediante las pruebas dadas por un profesor, el alumno pueda conocer si está cumpliendo con los objetivos además de satisfacer las necesidades que se pretenden conseguir con el código realizado.

1.4. Antecedentes y estado actual

En estos momentos se encuentra una carencia de herramientas, que proporcionen la funcionalidad que estamos buscando con este proyecto. Podríamos llegar a pensar que lo que se busca con este proyecto ya lo conseguimos a través de pruebas unitarias, pero lo que buscamos es ir más allá siendo el profesorado el que detalle dichas pruebas y con ello abarcar todas las funcionalidades que se buscan conseguir con el desarrollo de un código para una determinada práctica. Y ya no es únicamente eso, sino encontrar esa herramienta que no encontramos en el mercado o de forma gratuita, que nos permita a través de unos pequeños pasos reducir notablemente los tiempos de corrección de prácticas si se llevasen a cabo mediante un gran número de pruebas. Hay algunas herramientas que evalúan código, pero no con el objetivo que se pretende alcanzar con este proyecto, el cual es dar posibilidad a poder realizar una corrección de código sobre un programa escrito en cualquier lenguaje de programación. Ejemplo de esas herramientas son:

- Clang [1] es una herramienta similar a lo que se busca conseguir, es decir, reducir el tiempo en las correcciones de prácticas y permitir al profesorado hacer un análisis más exhaustivo con un mayor número de ejercicios. La desventaja es que no es una plataforma web que permite un acceso más cómodo y sencillo por parte del alumnado y del profesorado, además está pensado principalmente para los lenguajes de C y C++, y no como en este proyecto, donde se quiere proporcionar libertad para cualquier lenguaje.
- CAP (Corrector Automático de Programas) [2] es una herramienta muy similar a Clang con la diferencia de que está pensada para el lenguaje de programación Java.

De esta manera, podemos observar que, existen algunas herramientas, pero que sólo están disponibles para un lenguaje de programación determinado, por lo que se aprecia la carencia de una herramienta donde no haya ningún tipo de inconveniente por trabajar con cualquier lenguaje de programación. A pesar de ello, si existe una librería que permite trabajar con cualquier lenguaje de programación, siendo esta la librería de los tutores de este proyecto, la cual se ha integrado en este proyecto y que se explicará con mayor cantidad de detalle más adelante. El hecho de utilizar esta herramienta y ver que se trata de una gran opción es el hecho de que permite la evaluación de código escrito en diferentes lenguajes de programación, y que mediante la aportación de los recursos necesarios para que la librería funcione, nos aporta resultados bien detallados, que se pueden plasmar de manera sencilla en una plataforma web.

Por ello, el deseo que se busca conseguir es además de poder contar una herramienta que satisfaga lo comentado anteriormente, poder aumentar su funcionalidad y potencia integrándolo dentro de una plataforma web, acompañándola de esta manera de otros módulos necesarios para aumentar la calidad y el proceso de corrección de prácticas como son detención de plagios, capacidad de subir ficheros a un repositorio o servidor y perfiles propios en un entorno. Teniendo de esta forma, una aplicación muy completa, además de no estar presente o escasear en el mercado a día de hoy.

1.5. Objetivos generales y específicos

El objetivo general de este proyecto es la creación de una plataforma web con un modelo cliente-servidor que permita la automatización de los procesos de evaluación de los códigos de una determinada práctica de programación. Como objetivos específicos tenemos:

- Gestionar los roles de administrador, profesor y estudiante: dentro del sistema educativo tenemos profesores y alumnos, por ello, es necesario tenerlos presentes en nuestra plataforma junto con un rol que sea capaz de administrar y de gestionar las tareas más técnicas como es el rol de administrador.

- Posibilidad de añadir nuevas herramientas de compilación: es necesario la posibilidad de la petición de nuevos compiladores para distintos lenguajes ya que de

esta manera no hacemos que la herramienta quede limitada a un rango definido de lenguajes de programación.

- Subir test como códigos fuentes: permitir tanto a los profesores subir los archivos con los casos prácticos, como a los alumnos subir el código fuente de una determinada práctica.

- Creación de test a través de la herramienta: el profesorado puede diseñar los casos prácticos directamente desde la plataforma y evitar el proceso de subir el fichero.

- Listado de notas: tanto alumnos como profesores pueden consultar las notas obtenidas de la asignatura.

- Gráficas de evolución y estadísticas: los profesores pueden conocer la evolución y el número de aprobados de una práctica, de esta manera, podrán ver que prácticas tienen mayor dificultad o en qué momento a lo largo de la asignatura, los alumnos empiezan abandonarla, entre otras utilidades.

- Comprobación de códigos plagiados: dentro del proceso de evaluación, la plataforma detecta que alumnos tienen códigos muy similares, y, por lo tanto, evaluar sus prácticas como suspendidas ya que han cometido trampas dentro de la realización de una práctica.

- Utilizar el framework Spring: la idea del proyecto es desarrollar la plataforma en este framework de Java, dado que se trata de un framework potente y de uso empresarial, adquiriendo además conocimientos muy útiles para el futuro.

1.6. Motivación del proyecto

La motivación del proyecto nace por las carencias nombradas dentro de los procesos de evaluación de prácticas de programación y la necesidad de una herramienta de un uso más cómodo mediante el acceso web y que permita estar disponible para diferentes lenguajes de programación, además de aportar un detallado análisis de la corrección de una práctica, por ello se ha diseñado una plataforma que beneficie a los dos roles más importantes en una asignatura relacionada con la programación, como son el profesor y el alumno, dotando a cada uno de ellos con una herramienta que les beneficie en el proceso tan complejo que lleva evaluar y realizar una práctica respectivamente.

Capítulo 2. Librería de evaluación de código

Ahora nos dispondremos a comentar el funcionamiento que sigue la librería de evaluación de código con la que partimos, que ha sido desarrollada por los tutores de este proyecto.

2.1. Funcionamiento

El funcionamiento de la librería consiste en que mediante un conjunto de pruebas (casos prácticos) y uno o varios códigos fuentes, la librería sea capaz de compilar dichos códigos, obtener una salida de ellos y compararla con la salida esperada de una determinada prueba, obteniendo como resultado "true" en el caso de que se haya superado la prueba o "false" en el caso contrario. Esta librería será integrada dentro de la parte de servidor del proyecto y nos permitirá ejecutar de manera muy rápida la corrección de una clase de prácticas en unos pocos segundos.

Esta librería se emplea básicamente por el hecho de trasladar el trabajo de los tutores de este proyecto a un entorno más accesible, siendo dicho entorno la plataforma web donde estará integrado como ya hemos dicho. Para su uso, el proyecto obtendrá la salida de la librería mediante línea de comandos, se encargará de obtener los datos necesarios, entre los que estarán como más importantes si el resultado que se ha obtenido ha sido true o false, y plasmarlo en la web.

2.2. Conjunto de pruebas

Las pruebas con las que trabaja la librería tienen que tener un determinado formato para que la librería las valide como una prueba válida, ya que en el caso contrario y como veremos más adelante en la ejecución de la misma, el test será dado como inválido y no se hará ningún tipo de comprobación con él. Los campos necesarios son:

- name: nombre del test.

- input: son los argumentos que le pasaremos a cada uno de los códigos fuentes con el objetivo que dichos parámetros tengan alguna finalidad dentro del código para obtener la salida esperada.

- output: es la salida que se espera obtener con los argumentos que le hayamos pasado, o sencillamente, en el caso de no haberle pasado parámetros, la salida que debe devolvernos un determinado código.

Otro posible campo el cual no es necesario es “dependencies”, y nos permite definir con que otros archivos de pruebas guarda relación, ya sea porque el caso práctico es similar o porque prueba una funcionalidad determina, el archivo de prueba al que hemos añadido este campo.

2.3. Compiladores

La librería a la hora de su ejecución también hace uso de un fichero JSON en el cual se establece los lenguajes de programación que se van a soportar junto el comando que permite para ese lenguaje de programación determinado, compilar un código fuente y convertirlo a ejecutable.

El formato de ese fichero es el siguiente:

```
{
    "java": "javac CODE",
    "c": "gcc CODE -o EXECUTABLE",
}
```

Ilustración 1. Archivo compilers.json

De esta forma podemos ver lo que hemos hablado con anterioridad, a la izquierda el lenguaje de programación en cuestión, y a la derecha el comando de ejecución.

2.4. Ejecución de la librería

Una vez descritos los recursos necesarios para la ejecución de la librería, a expensas del código fuente que puede tratarse de cualquier programa cuyo lenguaje debe estar soportado dentro del archivo JSON, vamos a describir en partes toda la salida que obtenemos de la librería. Para esta prueba, contaremos con cuatro casos prácticos más dos códigos fuentes.

El comando de ejecución de la librería es el siguiente:

```
java -jar code-analysis-1.0-SNAPSHOT-jar-with-dependencies.jar -test tests -sc sourcecodes -cps compilers.json -ver true
```

Ilustración 2. Comando de ejecución

Como podemos observar en el comando, debemos cumplimentar una serie de requisitos:

- Nombre de la librería: code-analysis-1.0-SNAPSHOT-jar-with-dependecies.jar.
- Después del -test debemos indicar la carpeta de pruebas o el test que vamos a probar.
- Después de -sc debemos indicar la carpeta donde se encuentran los códigos que vamos a testear o simplemente un código en particular.
- Después de -cps indicaremos el archivo JSON donde se encuentran los compiladores que soporta la librería.

Una vez, explicado el comando de ejecución, vamos a describir por partes, la salida que obtenemos:

1.- Se indica el compilador que se usa para poder compilar los archivos que se encuentran en la carpeta de códigos fuentes.

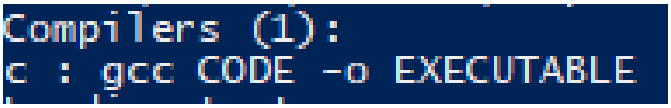


Ilustración 3. Compiladores utilizados para la ejecución

2.- Se muestran las pruebas que se encuentran dentro de la carpeta donde se encuentre el conjunto de pruebas.

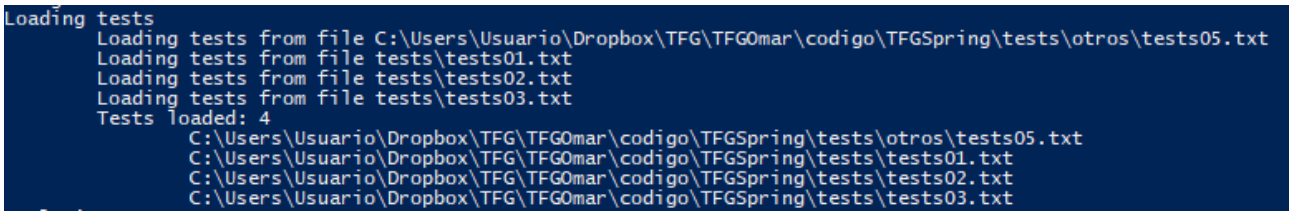


Ilustración 4. Cargando las pruebas

3.- Como mencionamos antes, en este paso descarta aquellas pruebas que no sean consideradas como pruebas válidas, y si carga para su uso, las que si hayan sido consideradas como pruebas válidas.

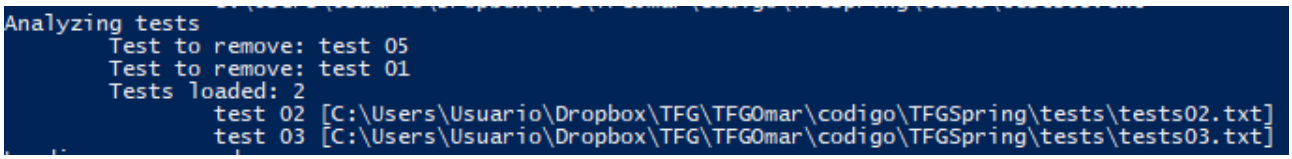


Ilustración 5. Analizando las pruebas

4.- Ahora es el turno de los códigos fuentes, cargando de esta manera aquellos códigos que ha detectado de la carpeta de códigos fuentes que hayamos determinado.

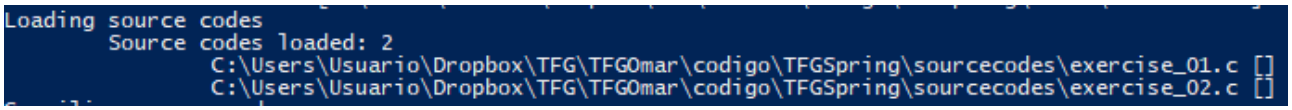


Ilustración 6. Cargando códigos fuentes

5.- Los códigos que haya cargado, serán compilados y se indica si se han compilado correctamente y de esta manera, determinando que están listos para poder pasar las pruebas.

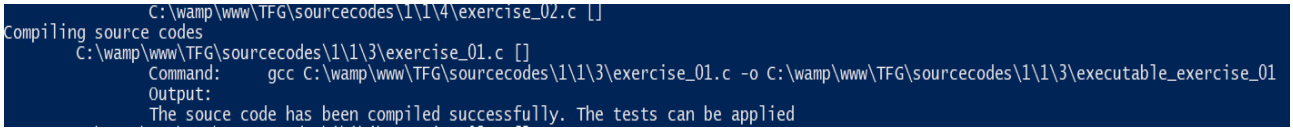


Ilustración 7. Compilando códigos

6.- Y como último, la parte de la salida que es aprovechada dentro de la plataforma para poder obtener las notas de los alumnos, y la que de verdad realiza el procedimiento de corrección de una práctica en unos pocos segundos.

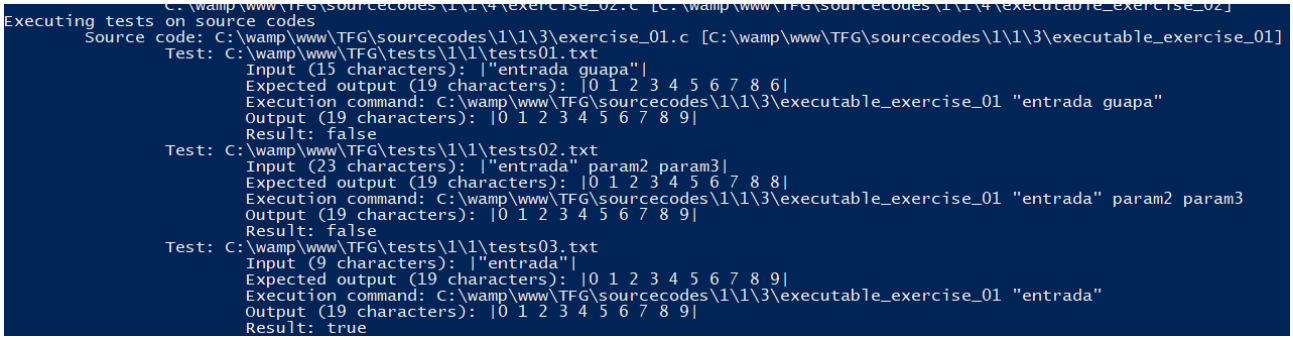


Ilustración 8. Pasando pruebas sobre los códigos

Dentro de este paso podemos ver varios elementos a tener en cuenta:

- Ruta del código fuente donde se va a ejecutar todas las pruebas válidas.
- Ruta de la prueba que se va a ejecutar en ese preciso momento
- Parámetros de entrada que utiliza el código fuente.
- Salida esperada por parte de la prueba
- Comando de ejecución que se lleva a cabo.
- Salida del programa
- Resultado de la prueba, donde si la salida esperada coincide con la salida del programa devuelve “true”, y en caso contrario, devuelve “false”.

El procedimiento para el resto de pruebas y códigos es el mismo, es decir, aparece el resultado de la ilustración 9 para cada prueba sobre un mismo código, procediendo del mismo modo al siguiente código, al que se le pasa todas las pruebas de nuevo. El funcionamiento de la librería es el que hemos explicado en estos pasos, y del cual la plataforma hace uso, obteniendo un número determinado de true o false para un concreto código, el cual pertenece a un cierto alumno, consiguiendo de esta manera una nota para dicho alumno en una determinada práctica.

Capítulo 3. Herramientas y tecnologías

Ahora pasamos a describir las herramientas usadas para llevar a cabo el desarrollo de este proyecto, las cuáles, han sido escogidas debido a la propia experiencia con ellas, lo que implica que al haber trabajado con ellas se reducen los tiempos de desarrollo y tiempo en aprender su utilización.

3.1. Framework principal: Spring

La idea de este proyecto es desarrollarlo principalmente con Spring, ya que uno de los objetivos del mismo era aprender nociones básicas de este potente framework tan importante en el ámbito profesional, además de que este framework aporta: simplificar el desarrollo de aplicaciones Java gracias a que ayuda a tener un código más limpio y reutilizable de modo que nuestra aplicación sea altamente modular y con bajo acoplamiento, administra los componentes de nuestra aplicación y los conecta entre sí lo que ayuda en proyectos muy grandes y mediante el uso de anotaciones que parecen insignificantes podemos administrar y gestionar una aplicación de manera muy sencilla.

Spring [3] es un framework de Java basado en el patrón de Inversión de control, el cual permite la configuración de los componentes de la aplicación y la administración del ciclo de vida de los objetos Java. Además, permite la Programación Orientada a Aspectos de tal forma que no ensucia la lógica de negocio de un proyecto.

Spring se puede gestionar tanto mediante archivos de configuración XML, que fue la primera opción que se tomó al principio del proyecto, como por anotaciones que fue la opción elegida como definitiva para desarrollar el mismo. Este framework es tan complejo que cuenta con numerosos módulos destacando para este proyecto como módulos utilizados:

- Spring Core: como módulo principal de este framework

- Spring Data: módulo que nos facilita la persistencia de datos en las bases de datos.

- Spring Boot: módulo que nos permite mediante una serie de anotaciones en un archivo Java poder lanzar nuestro proyecto en un servidor Tomcat embebido aparte de ayudarnos a abstraernos de la configuración de archivos XML necesarios para que nuestro proyecto pueda funcionar.

- Spring MVC: módulo que permite desarrollar una arquitectura Modelo Vista Controlador, de tal forma que, nos permite conectar las vistas de nuestro proyecto a los controladores del servidor que son capaces de gestionar las llamadas propias del front-end, llamar a los servicios que ejecutan las funcionalidades y crear los modelos con los accesos a los datos. Podemos destacar algunas características como: clara separación de roles declarados cada uno en sus propios objetos de java, potente y sencilla configuración, …

Ilustración 9. Framework Spring

3.2. Java Server Pages

Es la tecnología para renderizar plantillas que se ha decido utilizar en este proyecto y nos ayuda a poder crear archivos HTML donde dentro del propio HTML se puede manejar de manera muy sencilla la lógica de negocio y el acceso a datos. Dentro de la propia tecnología JSP, se ha usado una librería llamada JSTL (*JSP Standard Tag Library)* que nos permite realizar condicionales e iteraciones necesarias dentro del propio código HTML para poder mostrar múltiples datos.

3.3. Maven

Maven [4] es una herramienta que nos permite la gestión y creación de proyectos Java y como características, podemos destacar la fácil gestión de librerías externas y una estructura de proyecto bien definida y estructurada. Tiene un modelo de configuración basado en XML y utiliza un POM (*Project Object Model*) para poder describir el proyecto, las dependencias y el orden de construcción de los elementos.

Ilustración 10. Maven

3.4. MySQL

La base de datos que se ha decidido utilizar ha sido MySQL [5] debido que se ha querido usar como servidor WAMP [6] y además de ello, es la base de datos *open source* más popular del mundo. Es muy utilizado en aplicaciones web y muy rápida, y como en las aplicaciones web hay baja concurrencia en modificación de datos y es intensivo en la lectura de los mismos, la hace una base de datos genial para aplicaciones web.

Ilustración 11. Base de datos MySQL

3.5. WAMP

Es una herramienta para Windows que cuenta con un servidor apache más la interfaz gráfica de MySQL que permite gestionar de manera sencilla y rápida nuestra base de datos. Además, WAMP, nos crea una carpeta que se comentará más adelante donde se va a guardar los ficheros de nuestro proyecto.

Ilustración 12. Servidor WAMP

3.6. Hibernate

Hibernate [7] es una herramienta de mapeo objeto-racional para Java que facilita el mapeo de los elementos de la base de datos con el modelo de negocio de nuestro proyecto. Nos permite mediante una serie de anotaciones, definir como

queremos que sean los atributos de una determinada tabla para que luego podamos definir objetos que se inserten en la base de datos con las especificaciones definidas.

Además, mediante un sencillo archivo de propiedades se establece la conexión con la base de datos MySQL definiendo a que base de datos debe conectarse, para luego de forma automática crear todas las tablas que hayamos definido en nuestro proyecto como entidades.

3.7. Flot.js

Flot.js [8] es una librería de JavaScript que nos permite la creación de múltiples y diferentes tipos de gráficas para poder representar la parte estadística del proyecto.

La herramienta fue elegida aparte de que tiene un diseño más que aceptable, porque podemos enviar de una manera rápida los objetos de Java que queríamos representar y obtener de forma rápida unas gráficas bien detalladas. Además, cuenta con los dos tipos de gráficas que queremos usar en nuestro proyecto: gráfica circular y gráfica de líneas.

3.8. google-dif-match-patch

Librería de JavaScript que nos permite la comparación de dos textos, de tal forma que, aportando dos cadenas, la librería es capaz de devolver las diferencias entre ambos. Esta herramienta fue integrada dentro del módulo de plagios del proyecto.

3.9. Herramientas básicas de front-end

Aparte de las herramientas básicas como HTML, CSS y JavaScript se ha decidido utilizar jQuery debido que en ocasiones era beneficioso aprovecharse de las ventajas de esta librería, ya que extienden las propias de JavaScript y Ajax debido que en ciertos casos era necesario mostrar mensajes de éxito o fracaso sin el hecho de refrescar la página, utilidad que aporta esta librería.

Capítulo 4. Arquitectura

En este capítulo describiremos la arquitectura del proyecto que hemos decidido utilizar.

4.1. Análisis de la arquitectura

El análisis para elegir la arquitectura del proyecto ha sido mínimo debido que la idea para el desarrollo de la arquitectura cliente-servidor era usar la que nos aporta el módulo de Spring MVC, consiguiendo de esta manera contar con una arquitectura que siga el famoso patrón de Modelo Vista Controlador que permite desarrollar aplicaciones web flexibles y bien acopladas.

4.2. Arquitectura interna del proyecto de Spring

Está dividida en cuatro capas o niveles:

- Cliente: desarrollado con JSP y contiene todas las vistas de front-end de la plataforma.

  Dentro del servidor:

- Controladores: encargados de renderizar las vistas, de crear el modelo con los datos que se van a representar en las páginas HTML y de llamar a los servicios.
- Servicios: encargados de ejecutar las funcionalidades del sistema.
- Repositorios: encargados de hacer las llamadas a las bases de datos para obtener los datos necesarios.

Todas estas capas estarían manejadas por Spring MVC que se encarga del ciclo de vida e intercomunicación de todas.

4.3. Ciclo de funcionamiento dentro de Spring

En este subapartado, hemos querido mostrar mediante un diagrama como sería el ciclo de funcionamiento dentro de nuestro proyecto de Spring:

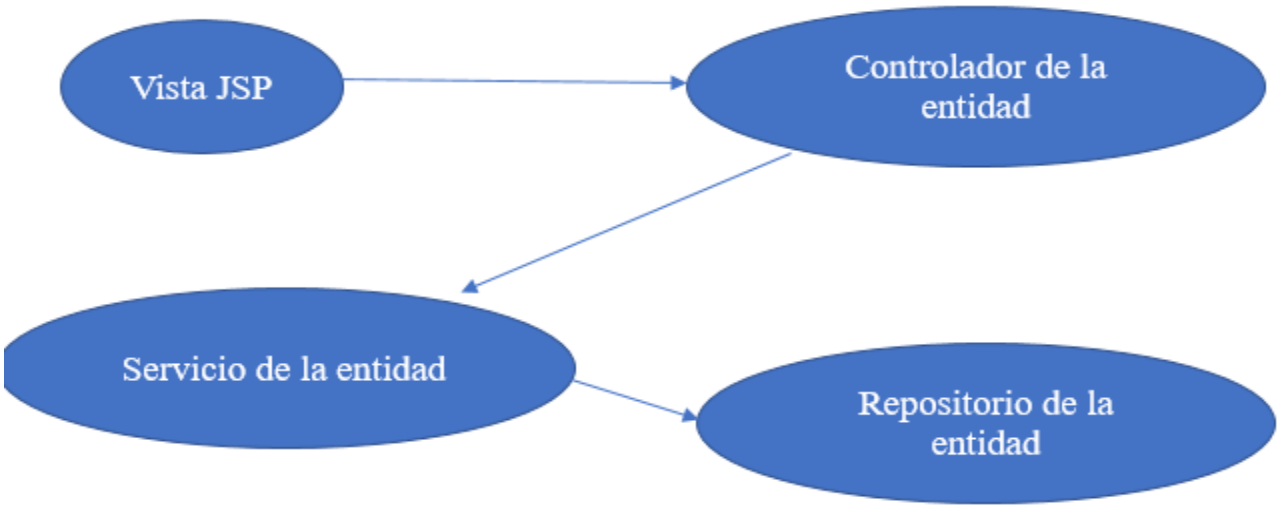


Ilustración 13. Ciclo de funcionamiento

Los pasos que se hacen en el ciclo de funcionamiento son:

1. Se hace una petición a la base de datos mediante una clase anotada como repositorio.
2. Las clases anotadas como servicios hacen la llamada a los repositorios para poder obtener la petición realizada a la base de datos, y poder de esta forma, trabajar con dichos objetos para cumplir una cierta funcionalidad.
3. Las clases anotadas como controladores son las encargadas de atender a las llamadas hechas desde el cliente web, y mediante la llamada a los servicios, guardar los datos en el modelo, siendo el modelo el que se visualice en las páginas HTML.

4.4. Sistema de directorios en el servidor

Es necesario explicar qué tipo de estructura se siguió para almacenar los ficheros, tanto códigos fuentes como pruebas, dentro del servidor WAMP.

WAMP cuenta con una carpeta dentro de nuestro servidor donde hemos colocado una carpeta raíz para nuestro proyecto, y a su vez dentro de ella, una carpeta para códigos fuentes y otra para las pruebas. De tal forma, que, si guardamos un test dentro del servidor, éste se guarda dentro de la carpeta tests como: {idAsignatura}/{idSesion}/test. Y para el caso de un código fuente: {idAsignatura}/{idSesion}/{idAlumno}/código-fuente.

De esta forma conseguimos una fácil ejecución de la librería sobre los directorios, porque atacamos directamente sobre la sesión o práctica de una respectiva asignatura.

4.5. Tablas de la base de datos

Las tablas o entidades que se han querido implementar para este proyecto han sido:

- Usuarios: almacena todos los usuarios del sistema. Los campos que contiene son id, correo, nombre y apellidos, contraseña, teléfono, rol y nombre de usuario.
- Asignaturas: almacena las asignaturas del sistema. Los campos que contiene son: id, número de sesión, nombre, id del usuario y contraseña.
- Compiladores: almacena los compiladores del sistema. Los campos que contiene son: id; nombre, estado (estado de la petición del compilador), id del usuario y nombre del comando de compilación.
- Ficheros: almacena los ficheros del sistema. Los campos que contiene son: id, fecha de subida, nombre, número de sesión, id de la asignatura e id del usuario.
- Matriculaciones: almacena las matriculaciones de los estudiantes a las asignaturas. Los campos que contiene son: id, id de la asignatura e id del usuario.
- Notas: contiene las notas de los estudiantes. Los campos que contiene son: id, nota, sesión, tipo de nota (nota de práctica o nota final), id de la asignatura e id del usuario.
- Plagios: contiene los plagios que se producen en una sesión. Los campos que contiene son: id, nombre del plagio, sesión e id de la asignatura.
- Sesiones (Prácticas de una asignatura): contiene la información de la ejecución de la librería para una determinada práctica. Los campos que contiene son: id, entrada (parámetros de ejecución), privacidad (si se trata de una práctica de carácter oficial o de prueba para el alumno), resultado, salida esperada, salida del programa, sesión, nombre del código, nombre del test, id de la asignatura e id del usuario.

De esta forma, y para representarlo mejor usaremos una imagen donde veremos las relaciones entre las diferentes entidades:

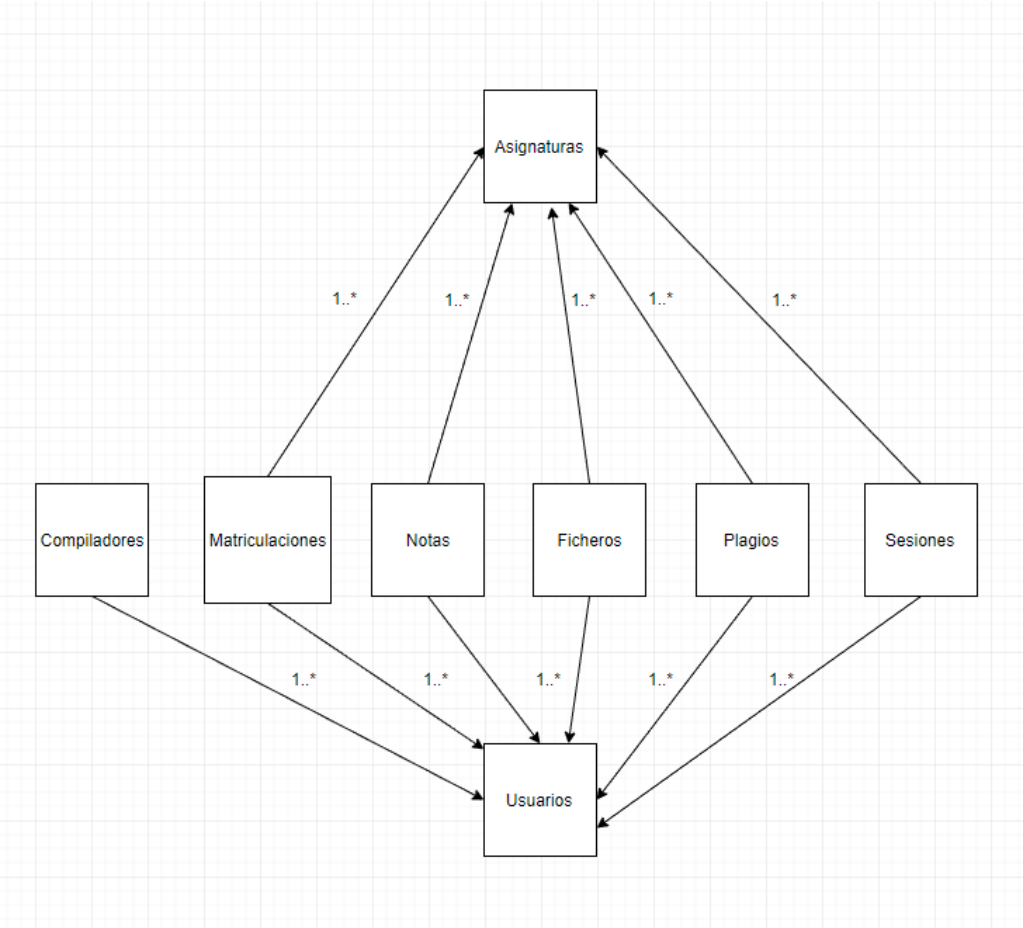


Ilustración 14. Modelo E/R

Capítulo 5. Funcionamiento

El funcionamiento de la plataforma lo describiremos en tres partes, que corresponden con los tres diferentes roles con los que se puede acceder a la aplicación, siendo estos: administrador, profesor y estudiante. El nombre que se ha decidido poner a la plataforma es: CheckCode.

5.1. Pantalla de inicio

En la página principal del proyecto podemos conocer de primera mano que se trata de una aplicación destinada a la universidad aparte de dar una pequeña bienvenida explicando el objetivo principal para el cual ha sido diseñada.

Además, es necesario explicar que la primera vez que entremos en la aplicación sólo podemos acceder mediante un usuario admin que ha sido insertado mediante un script, ya que es necesario poder contar con al menos un primer usuario que sea el que como vayamos a ver, el encargado de crear a los demás usuarios, ya que dota del rol de administrador.

En el panel que nos aparece, debemos introducir nuestro correo y contraseña, y la plataforma se encargará de redirigirnos según nuestro rol a una determinada pantalla.

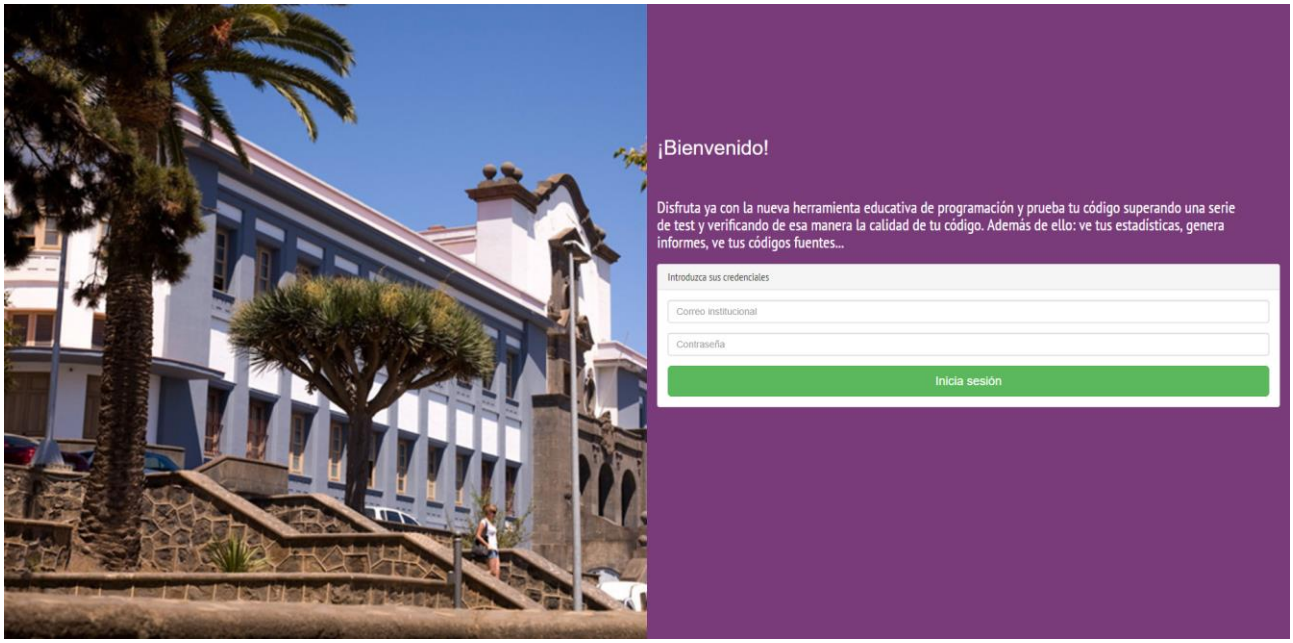


Ilustración 15. Página de inicio

5.2. Administrador

Cuando entramos con el rol de administrador dentro de la aplicación, podemos apreciar en el menú las dos funcionalidades generales que tiene, aparte de la opción de cerrar sesión.

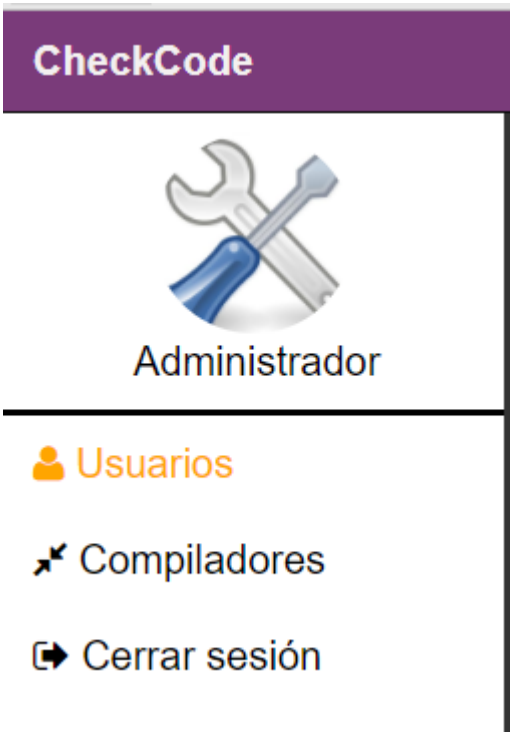
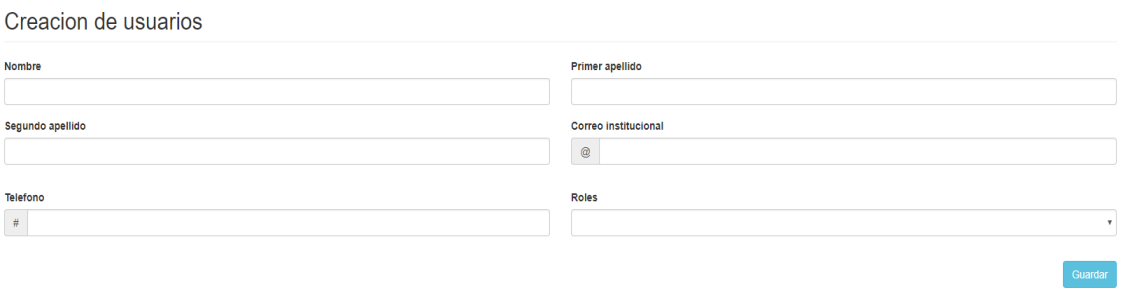


Ilustración 16. Menú del administrador

5.2.1 Usuarios

Dentro de la opción de usuarios, tenemos las primeras dos funcionalidades:

- Creación de usuarios: para esta funcionalidad se dispone de un pequeño formulario donde debemos rellenar el nombre, primer y segundo apellido, correo institucional, teléfono y roles donde podemos elegir entre los tres que ya hemos comentado. Una vez rellenado dichos campos, debemos apretar el botón de guardar.



La plataforma aparte de crearnos un usuario con los datos que hayamos rellenado, asigna un nombre de usuario basándose en la primera letra del nombre, las tres primeras letras del primer apellido y las tres primeras letras del segundo apellido. Por otro lado, nos crea una contraseña al azar mezclando números y letras. Una vez el usuario se haya dado de alta, se envía un correo diciéndole la contraseña que debe usar para poder entrar en la aplicación.

Ilustración 17. Correo con la contraseña

- Listado de usuarios: muestra todos los usuarios que hemos creado.

También podemos ver dentro de las opciones, la posibilidad de dar de baja aquellos usuarios que sean profesores y alumnos. En el caso de tener el rol de administrador, el usuario no puede ser dado de baja.

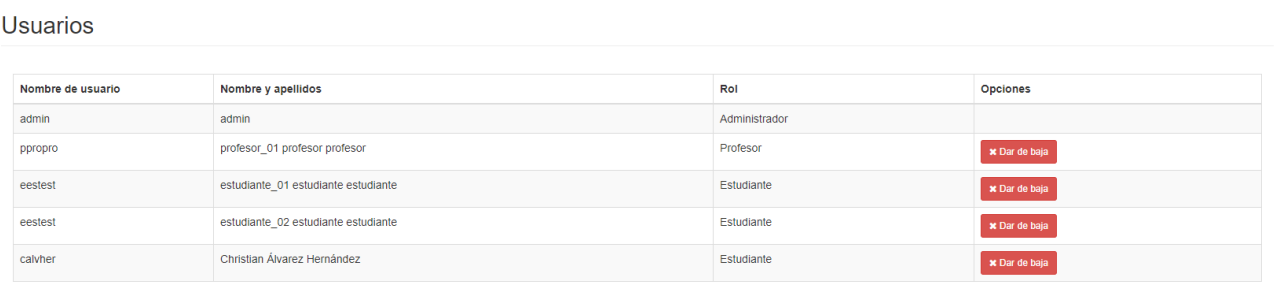


Ilustración 18. Listado de usuarios

5.2.2 Compiladores

Dentro de la opción de compiladores, tenemos dos funcionalidades:

- Compiladores existentes: encontramos todos los compiladores posibles con los que trabaja nuestra librería, es decir, como ya hemos comentado con anterioridad, los compiladores que se encuentra dentro del archivo JSON que debemos pasar en la ejecución de la librería. Hay que decir, que hay dos compiladores que se insertan a partir de un script para al menos soportar los casos de lenguajes de código más generales como son Java y C. En la siguiente imagen podemos ver el lenguaje y el nombre del comando que se usa para poder compilar dicho lenguaje.

Ilustración 19. Listado de compiladores

- Peticiones de compiladores: los administradores pueden atender las peticiones de añadir nuevos compiladores de lenguajes a partir de esta funcionalidad. Se muestra un listado de peticiones, donde se indica el nombre de usuario y el lenguaje que quiere añadir.

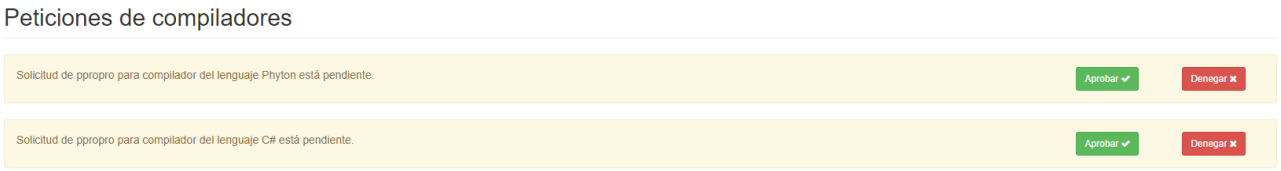


Ilustración 20. Petición de compiladores

A partir de ello, el administrador puede denegar dicha solicitud, y en el caso contrario, puede añadir el nuevo compilador asociándole el nombre del comando respectivo que aparece a través de un modal.

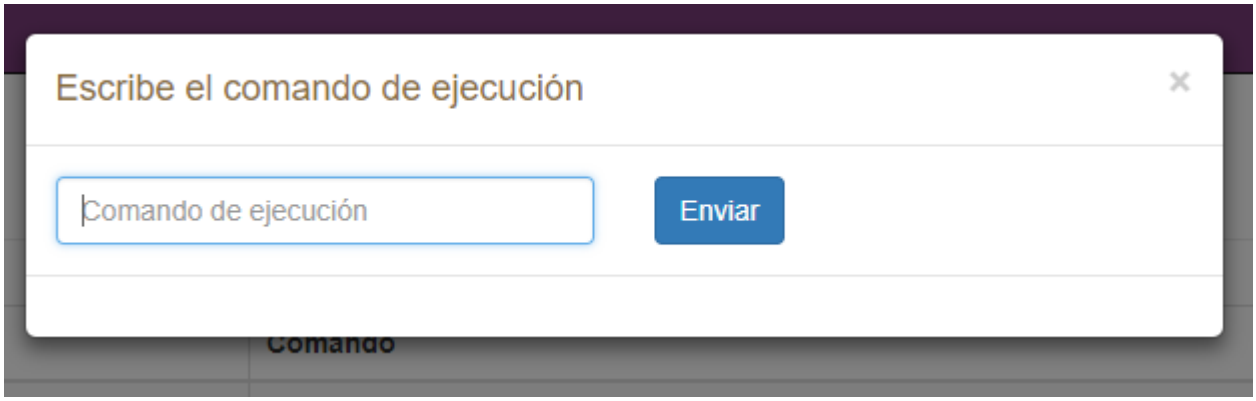


Ilustración 21. Añadir comando de ejecución

Una vez enviemos el comando de ejecución, el listado de compiladores tendrá una nueva entrada con el compilador que hemos añadido.

5.3. Profesor

Cuando entremos con el rol de profesor dentro de la aplicación, podemos apreciar en el menú las dos funcionalidades generales que tiene, aparte de la opción de cerrar sesión.

Ilustración 22. Menú del profesor

5.3.1 Tablón

Dentro del tablón podemos observar dos funcionalidades:

- Datos personales: el profesor en cuestión puede observar todos sus datos personales, además de dos pequeñas estadísticas que le aportan al profesor conocer el número de test que ha subido junto al número de asignaturas que tiene a su cargo.

Ilustración 23. Datos personales del profesor

- Creación de peticiones de compiladores: como ya pudimos observar antes, los administradores reciben peticiones de compiladores, y es desde esta parte de la aplicación donde el profesorado puede crear dichas peticiones. El profesor debe enviar el lenguaje de programación para el que quiere que haya un compilador disponible. Una vez el administrador acepte o rechace la petición, el profesor puede ver el estado de dicha petición.

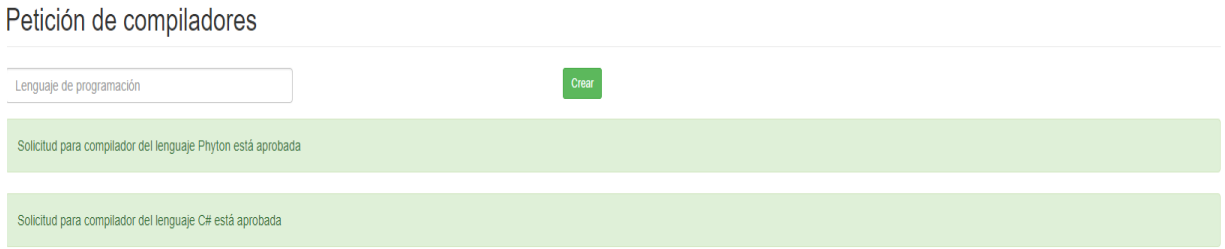


Ilustración 24. Enviar petición de compilador

5.3.2 Asignaturas

Dentro del apartado de asignaturas el profesor puede ya utilizar todo el potencial de la aplicación, y es por ello, que esta opción del menú es bastante extensa ya que permite bastantes funcionalidades.

Cuando entremos dentro de asignaturas vamos a encontrarnos:

- Crear nueva asignatura: consta de un pequeño formulario donde debemos rellenar el nombre de la asignatura, el número de sesiones y una contraseña con la que los estudiantes pueden matricularse. Una vez rellenado dichos

campos, podemos crear una nueva asignatura. Además, hay que comentar que cuando creamos una nueva asignatura, se crea directamente en el servidor dentro de la carpeta tests, una carpeta para dicha asignatura, y dentro de ella tantas carpetas como sesiones hayamos indicado.

Ilustración 25. Crear nueva asignatura

- Listado de asignaturas: podemos observar todas las asignaturas que hemos creado mediante una tabla. Además, cada asignatura consta de diferentes opciones (alumnos, sesiones, notas y dar de baja). Las opciones disponibles para cada asignatura se comentan en los siguientes apartados (la opción de dar de baja no se describe por el hecho de quedar claro su funcionalidad).

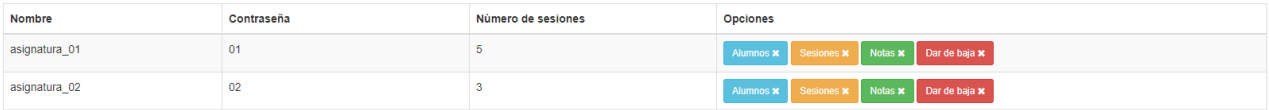


Ilustración 26. Listado de asignaturas

5.3.2.1 Alumnos

Si accedemos al botón “Alumnos”, avanzamos a una página donde podemos ver un listado de todos los alumnos matriculados en una asignatura, con la posibilidad mediante el botón de desmatricular, poder darlos de baja en la asignatura.

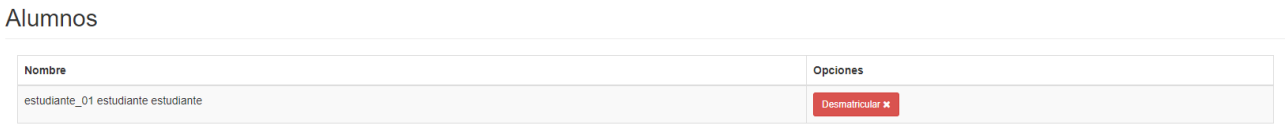


Ilustración 27. Listado de alumnos

En el caso de que no haya ningún alumno matriculado, la plataforma nos mostrará un mensaje diciendo que no hay alumnos matriculados.

5.3.2.2 Sesiones

Si accedemos al botón “Sesiones”, avanzamos a una página donde podemos ver:

- Estadísticas de la asignatura: si apretamos sobre el botón de ver estadísticas podemos observar un gráfico circular con el número de aprobados y suspendidos para la asignatura en cuestión.

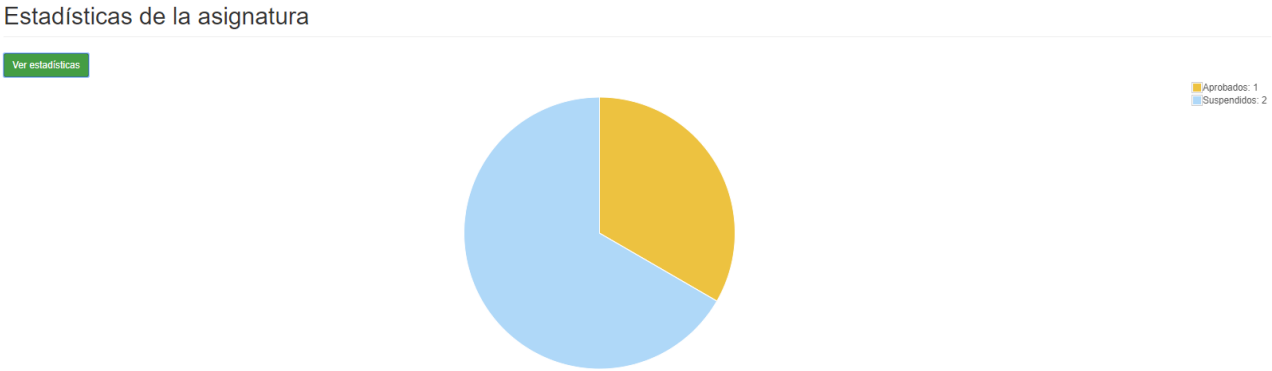


Ilustración 28. Estadísticas de la asignatura

- Listado de sesiones: podemos ver todas las sesiones con las que cuenta una asignatura. Además, nos permite una serie de opciones para cada una de las sesiones:

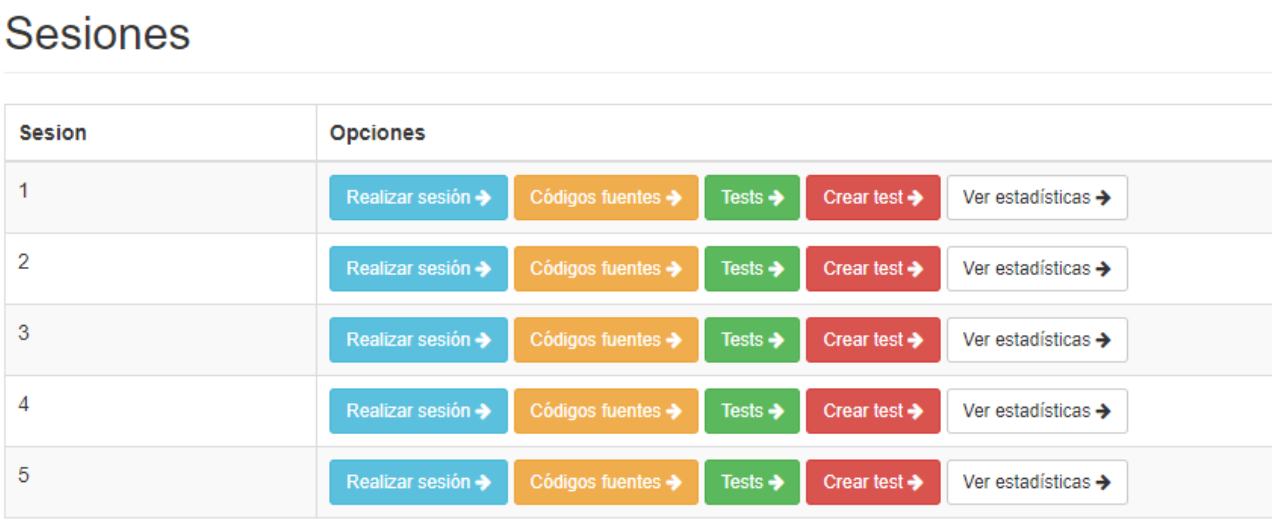


Ilustración 29. Sesiones de una asignatura

1. Ver estadísticas de una sesión: muestra en un diagrama de tarta el número de aprobados y suspendidos de la sesión.

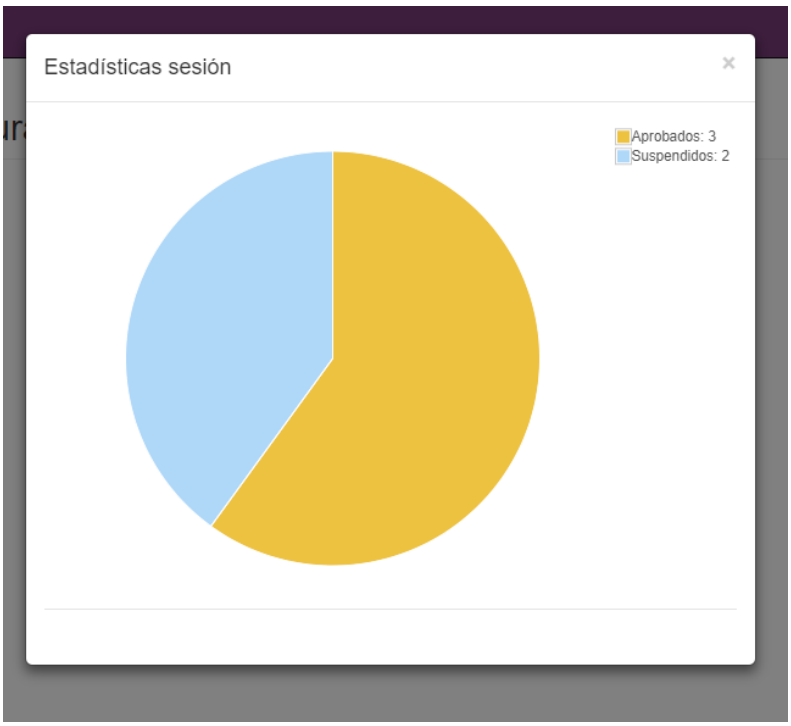


Ilustración 30. Estadísticas de la sesión

2. Tests: cuando accedemos a esta opción, avanzamos a una página donde tenemos dos funcionalidades, siendo la primera la de poder subir tests al servidor, donde elegiremos los test que queramos subir ya sea de forma individual o mediante un comprimido que la plataforma se encargará de descomprimir, para luego guardar en la carpeta de la sesión correspondiente.

Ilustración 31. Subir tests

La segunda funcionalidad es la de poder ver mediante un listado los tests subidos a esa sesión, con el añadido de poder borrar cualquier fichero que hayamos subido.

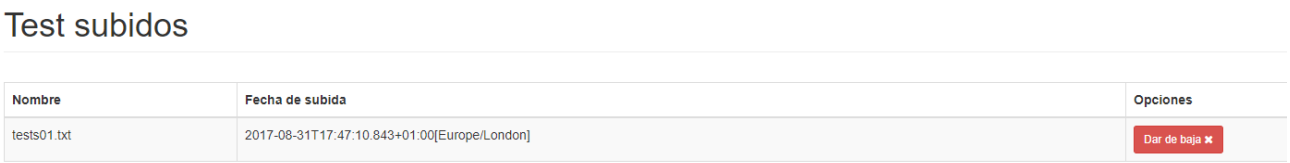


Ilustración 32. Listado de tests

3. Códigos fuentes: cuando accedemos a esta opción, avanzamos a una página donde tenemos el listado de los códigos fuentes subidos de todos los alumnos matriculados a la plataforma junto con el nombre de usuario, el nombre del fichero, la fecha y las opciones tanto de descargar como de borrar el fichero.

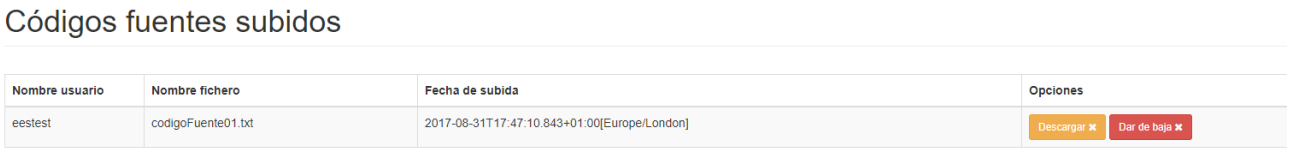


Ilustración 33. Listado de códigos fuentes

4. Realizar la sesión: en este paso es donde ejecutamos la librería de evaluación de código del profesorado, obteniendo de esta manera toda la salida que nos aporta la librería, con ello podemos obtener un listado de los resultados obtenidos y de la creación de notas para esa sesión en cuestión. Además, podemos seleccionar en el formulario, si queremos, ejecutar la librería pasando el módulo de plagios o no, que tendría como resultado en el caso de que lo seleccionemos y haya plagios, una actualización de las notas, es decir, valorando las notas de los alumnos para esta sesión con un 0, aunque si podemos seguir viendo los resultados obtenidos de todas formas. También es necesario comentar que las sesiones se van actualizando si ejecutamos de nuevo la librería al igual que con las notas, con la diferencia de que se mantiene siempre la nota más alta conseguida.

   El procedimiento que se utiliza para llevar a cabo la corrección de una práctica consiste en obtener de la salida de la librería la parte seis que se describe en el apartado 2.4, de tal manera que, todas las ejecuciones código fuente-test se guarden en la base de datos. Además, mientras se va guardando cada una de las pruebas, se calcula la nota del alumno para esa práctica. Una vez finalizada la ejecución, se calcula la nota final de la asignatura para cada uno de los alumnos

   De esta manera, obtenemos una vez finalizada la llamada al servicio web encargado de esta funcionalidad los resultados para esa sesión junto a las notas actualizadas. Tenemos que tener en cuenta, que si a la hora de corregir la práctica, activamos el detector de plagios, aquellos alumnos que se hayan copiado tendrán una nota de 0 para esa práctica.

   Aquí podemos ver el formulario para ejecutar la librería:

Ilustración 34. Ejecutar corrección

Y por otro lado podemos ver el listado de los resultados de una sesión:

Resultados

Código fuente	Usuario	Test	Entrada	Salida esperada	Salida del programa	Resultado
exercise_01.c	eestest	tests01.txt	|"entrada"|	|0 1 2 3 4 5 6 7 8 6|	|0 1 2 3 4 5 6 7 8 9|	
exercise_01.c	eestest	tests02.txt	|"entrada" param2 param3|	|0 1 2 3 4 5 6 7 8 8|	|0 1 2 3 4 5 6 7 8 9|	
exercise_01.c	eestest	tests03.txt	|"entrada"|	|0 1 2 3 4 5 6 7 8 9|	|0 1 2 3 4 5 6 7 8 9|	
exercise_02.c	eestest	tests01.txt	|"entrada"|	|0 1 2 3 4 5 6 7 8 6|	|0 1 2 3 4 5 6 7 8 9|	
exercise_02.c	eestest	tests02.txt	|"entrada" param2 param3|	|0 1 2 3 4 5 6 7 8 8|	|0 1 2 3 4 5 6 7 8 9|	
exercise_02.c	eestest	tests03.txt	|"entrada"|	|0 1 2 3 4 5 6 7 8 9|	|0 1 2 3 4 5 6 7 8 9|	

Ilustración 35. Listado de resultados de la corrección

5. Crear test: si accedemos a este apartado, avanzamos a una página donde podemos crear nuestros propios test desde la página. Lo único que debemos hacer es escribir nuestro propio test y mandarlo a la plataforma, de esta forma la plataforma comprueba si se trata de un test válido y lo sube a la sesión correspondiente.

   El procedimiento que se utiliza para llevar a cabo la corrección de una práctica consiste en obtener de la salida de la librería la parte tres que se describe en el apartado 2.4, de tal manera que, el test que queremos validar y convertimos a fichero mediante la llamada al servicio web que se encarga de esta funcionalidad, ver si la librería lo acepta o no como un test válido, en el caso de que si lo acepte, mostraremos un mensaje de éxito, y en el caso contrario, un mensaje diciendo que el test tiene un formato inválido.

Ilustración 36. Crear tests

5.3.2.3 Notas

Si accedemos al apartado de notas podemos ver un listado de las notas por cada alumno para cada una de las sesiones de la asignatura junto a su nota final.

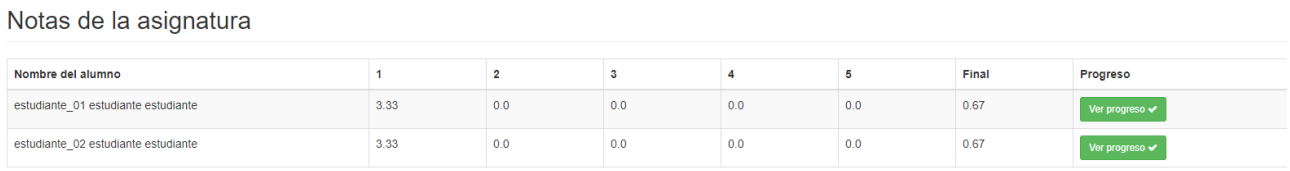


Ilustración 37. Listado de notas

Además, si apretamos el botón ver progreso podemos ver una gráfica de líneas donde se puede conocer el progreso del alumno a lo largo de la asignatura.

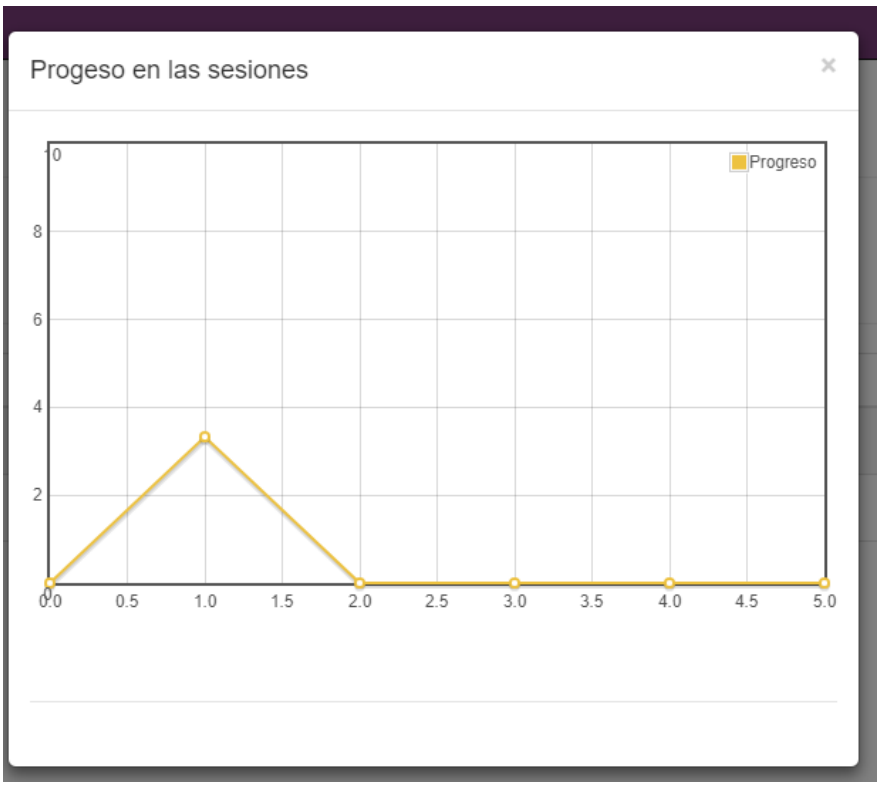


Ilustración 38. Evolución de un alumno

Y, por último, en el caso de que se hayan producido plagios, nos aparece un mensaje diciendo los alumnos que se han copiado y en qué sesión lo han hecho.

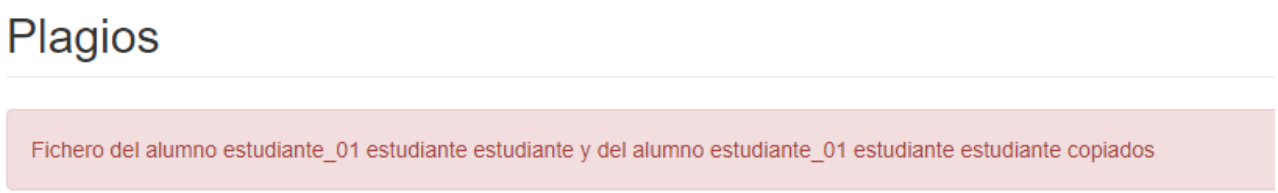


Ilustración 39. Mensaje de plagio

5.4. Estudiante

Cuando entramos con el rol de estudiante o alumno dentro de la aplicación, podemos apreciar en el menú las dos funcionalidades generales que tiene, aparte de la opción de cerrar sesión.

Ilustración 40. Menú del estudiante

5.4.1 Tablón

Dentro del tablón podemos observar dos funcionalidades:

- Datos personales: el estudiante en cuestión puede observar todos sus datos personales, además de dos pequeñas estadísticas que le aportan al estudiante conocer el número de códigos fuentes que ha subido junto al número de asignaturas matriculadas.

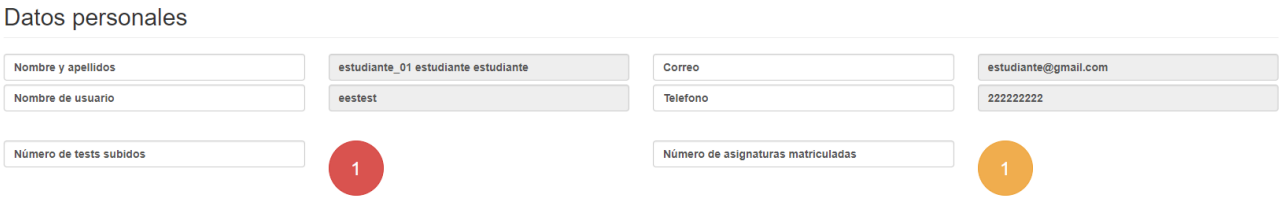


Ilustración 41. Datos personales del alumno

- Notificaciones: tendrá un aparatado de notificaciones donde le aparecerá que sesiones ya han sido ejecutadas, y de esta forma, notificándole que su nota ya está publicada.

5.4.2 Matriculaciones

Cuando entremos dentro de matriculaciones vamos a encontrarnos:

- Matricularse: mediante un formulario donde nos aparecen todas las asignaturas a las que nos podamos matricular, escogeremos una de ellas y mediante la contraseña podremos matricularnos o no en dicha asignatura.
- Lista de matriculaciones: podremos ver en un listado todas las asignaturas a las que estamos matriculados.

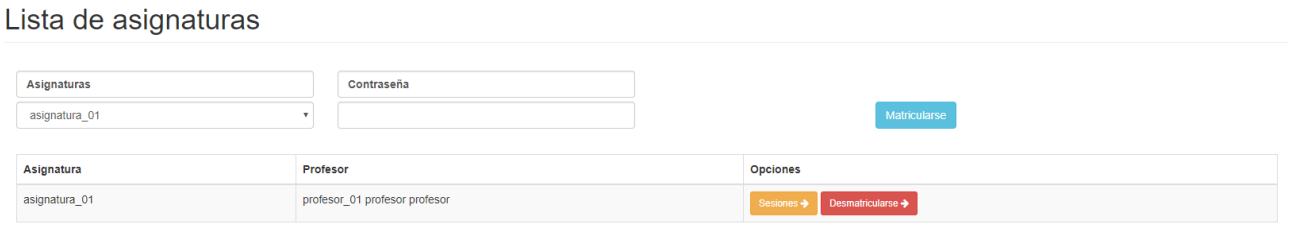


Ilustración 42. Listado de asignaturas matriculadas

Además de ello, tendremos una serie de opciones:

1. Desmatricularse: podemos darnos de baja de una asignatura.
2. Sesiones: accederemos a una pantalla donde por un lado tendremos las notas de las sesiones más la nota final de la asignatura y por el otro lado tendremos un listado de las sesiones con una serie de opciones.

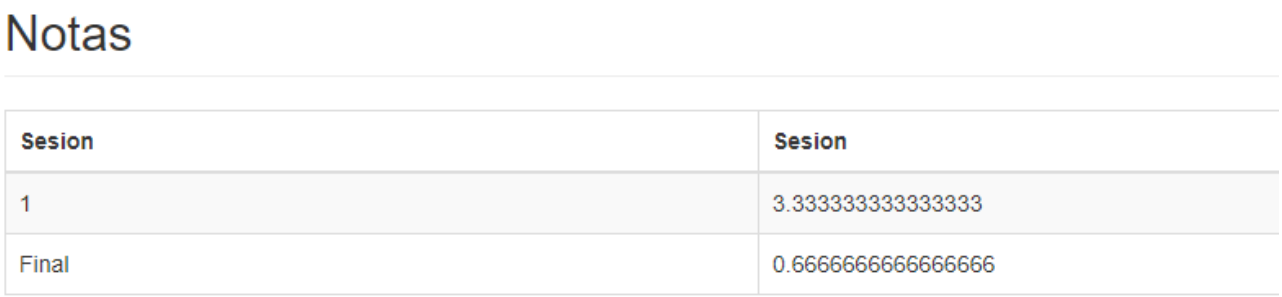


Ilustración 43. Listado de notas

Y en el listado tenemos una serie de opciones muy similares a las pantallas del rol de profesor:

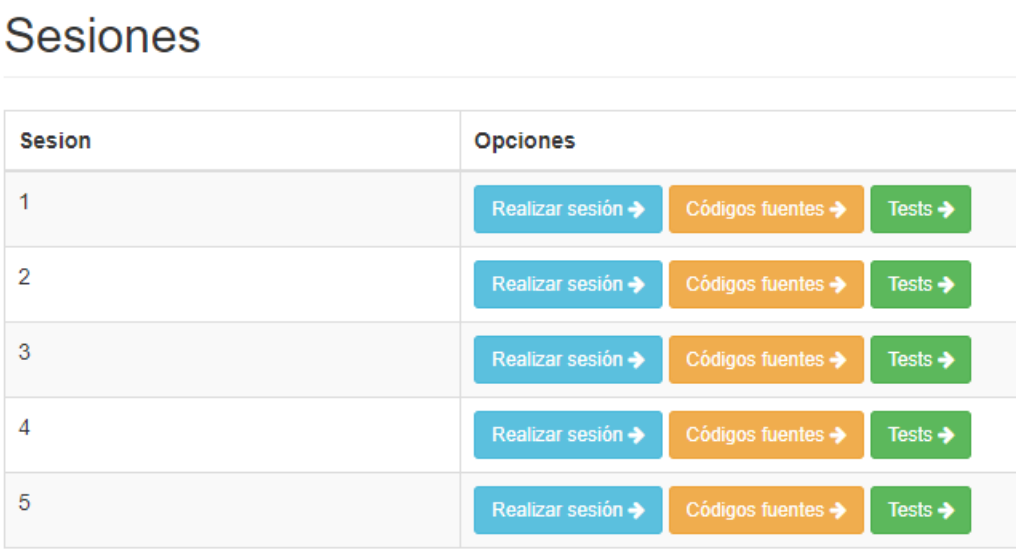


Ilustración 44. Listado de sesiones

1. El alumno podrá realizar una sesión y poder ver en que test falla o acierta, igual que el profesor, pero con la diferencia de que cuando se ejecute la sesión sólo se guardará una nota representativa para dicha ejecución a modo de prueba, y no de manera oficial
2. El alumno podrá ver los códigos fuentes que ha subido, además de la posibilidad de subir dichos códigos.
3. El alumno podrá ver los tests subidos a una determinada sesión, con la posibilidad de descargárselos.

Capítulo 6. Conclusiones y líneas futuras

CheckCode es una plataforma que presenta todas las herramientas necesarias para llevar a cabo los procesos de evaluación y corrección de prácticas de programación de manera automatizada, y consiguiendo de esta forma, un ahorro de tiempo grande en comparación a realizar dicha corrección de la forma tradicional que se realiza en una sesión de prácticas normal. Aportando de esta forma, la posibilidad de contar con una herramienta de ayuda para profesores y alumnos, que se ven beneficiados tanto en tiempo como en calidad a la hora evaluar una práctica determinada.

Las conclusiones de este trabajo son muy positivas ya que me han permitido utilizar un framework muy potente dentro del ámbito empresarial que permite multitud de posibilidades, creando con ello una plataforma de sencillo uso y de rápido acceso a cualquier posibilidad dentro de la evaluación de código, y después de haber realizado todo el trabajo, haber adquirido una gran cantidad de conocimientos respecto a esta gran herramienta que me servirá de gran ayuda en el futuro.

El propio desarrollo de la aplicación me hizo ver que hubiera sido una herramienta de gran ayuda a lo largo de mi camino por la carrera, pero mucho más para los profesores, que les facilitaría en gran medida las labores de corrección de prácticas. En cualquier caso, sería una herramienta que si fuese implantada en la universidad supondría un gran avance, y, sobre todo, una mejora sustancial de ahorro de tiempo y aumento de la calidad de la evaluación de las prácticas que es lo que todos estamos buscando.

La verdad que estoy bastante contento del trabajo que he realizado, pero sé que es necesario mejorar el proyecto si se quiere usar dentro de la universidad, pero como primera versión establece bastante bien las bases que se buscan conseguir y el camino a donde se tiene que apuntar para conseguir poder contar con una poderosa herramienta de la cual podamos hacer uso.

Como líneas futuras, se pretendería:

- Hacer uso de frameworks de front-end como Angular, el cual es un framework para aplicaciones web de TypeScript de código abierto basado en el Modelo Vista Controlador donde se destaca que es multiplataforma, su velocidad y rendimiento, productividad y una historia completa de desarrollo, que permitan manejar de manera más modular y escalable un proyecto que adquiriría mayores dimensiones.
- Añadir mayor seguridad como por ejemplo con la ayuda de un módulo de Spring como es Spring Security.

- Implantarlo en la universidad para aprovechar su gran utilidad y que entre todos podamos mejorar esta herramienta.
- Añadir métodos de creación de usuarios masivos.
- Mejorar la herramienta de detención de plagios para aumentar su precisión o contar con alguna herramienta que permita realizar el uso que se busca.
- Profesionalizar la herramienta añadiendo sistemas de integración continua (como Jenkins que es un sistema de integración continua open source escrito en Java que soporta herramientas de control de versiones y se trata de un sistema corriendo en un servidor) o de mejora de la calidad de código (como Sonar que se trata de una herramienta que analiza y detecta errores de sintaxis en el código y propone soluciones para poder erradicarlos).
- Aportar una mayor retroalimentación al alumno en la ejecución de las pruebas.
- Mejorar la accesibilidad y usabilidad del proyecto.
- Mejorar el editor de test para poder diseñar de mejor modo los mismos, al igual que poder permitir que los estudiantes puedan desarrollar su código desde la plataforma.

Capítulo 6. Summary and conclusions

CheckCode is a platform that presents all the necessary tools to execute the evaluative processes and marking of programming practices in an automated manner, thus achieving a big time saving in comparison to the time taken to do so traditionally, such as in an ordinary practice session. Therefore, contributing to the possibility of counting with a helping feature for teachers and students, who benefit both when it comes to time or the quality of marking a given practice.

The conclusions on this project are very positive since they have allowed me to use a very powerful framework within the business sector, which allows multiple possibilities, thus creating a platform of easy use and quick access to any possibility within the evaluation of code. In addition, after having completed the project, I have acquired a vast amount of knowledge with respect to this big tool, which will surely be of help in the future.

The development itself of this application made me realise that it would have been a helpful feature for me in the course of my career, but even more for professors, to facilitate the practice marking. In any case, it would be a feature that, if implemented at university, would represent a great improvement and, above all, a substantial time-saving improvement and an enhancement of quality of the marking assessment, which is what we all are aiming at.

The truth is that I'm quite satisfied with all the work I've done, yet I am aware of the continuous necessity of improving the project in case we want it to be used within university, but, as a first version, it establishes fairly well the bases which were required and the way to have at our disposal a powerful tool of which we can make use.

In the future, we would like to:

- Make use of front-end frameworks such as Angular, which is a framework for open source code TypeScript web applications based on the Vista Control Model where we highlight that it is a multiplatform, its speed and performance, productivity and a complete development history, which allow us to manipulate a project that would get bigger dimensions in a modular and scalable manner.
- Add greater security, like for example, with the help of a Spring module, such as Spring Security.
- Set it up at university to take advantage of its big usability, and allowing everyone to contribute to the further development of this tool.
- Add creation methods for massive users.

- Improve the detection of plagiarism feature to increase its precision or count with a feature that allows the execution of the searched use.
- Give professional status to the tool adding continuous integration systems (such as Jenkins, which is an open source continuous integration system written in Java that supports version control tools, being a system running in a server) or quality of code improvement systems (such as Sonar, which is a tool that analyses and detects syntax errors in the code and proposes solutions in order to eradicate them).
- Contribute to a better feedback towards students when it comes to executing tests.
- Improve accessibility and usability of project.
- Improve the test editor to be able to design these in a better manner, as well as to allow students to develop their code from the platform itself.

Capítulo 7. Presupuesto

Dentro de este capítulo, describiremos el supuesto presupuesto que nos ha costado la realización de este proyecto mediante la siguiente tabla:

Descripción	Cantidad	Precio	Cantidad x Precio
Horas de trabajo	300	15 euros	4500 EUROS
Sistema informático	1	1200 euros	1200 EUROS
Total			5700 EUROS

Capítulo 8. Bibliografía

[1] Clang. https://clang.llvm.org/

[2] CAP. https://users.dsic.upv.es/grupos/grps/downloadPaper.php?paperId=178

[3] Spring. https://spring.io.

[4] Maven. https://maven.apache.org/.

[5] MySQL. https://www.mysql.com/.

[6] WAMP. http://www.wampserver.com/en/.

[7] Hibernate. http://hibernate.org/.

[8] Flot.js. http://www.flotcharts.org/.