ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

IMPLEMENTACIÓN DE UN MÓDULO DE CALIFICACIÓN AUTOMÁTICA PARA UNA PLATAFORMA MOOC

PROYECTO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y REDES DE INFORMACIÓN

DARWIN MARCELO POVEDA OYOS
darwin.poveda@epn.edu.ec

DIRECTOR: ING. GABRIEL ROBERTO LÓPEZ FONSECA, MSc.
gabriel.lopez@epn.edu.ec

CODIRECTOR: ING. JORGE EDUARDO CARVAJAL RODRÍGUEZ, MSc.
jorge.carvajal@epn.edu.ec

Quito, Febrero 2018

DECLARACIÓN

Yo, Darwin Marcelo Poveda Oyos, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedo mis derechos de propiedad intelectual correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

_____

Darwin Marcelo Poveda Oyos

CERTIFICACIÓN

Certificamos que el presente trabajo fue desarrollado por Darwin Marcelo Poveda Oyos bajo nuestra supervisión.

_____

Ing. Gabriel López, MSc.

DIRECTOR DEL PROYECTO

_____

Ing. Jorge Carvajal, MSc.

CO-DIRECTOR DEL PROYECTO

AGRADECIMIENTO

Agradezco a Dios, por brindarme la fuerza y sabiduría necesaria para culminar con éxito esta etapa de mi vida.

A mis padres, por su apoyo incondicional durante los buenos y malos momentos de mi vida.

A mis hermanas, Cynthia y Pamela por su cariño, compresión y motivación para cumplir mis metas.

A mi director del proyecto de titulación, Msc. Gabriel López Ing., por la guía durante el desarrollo del presente proyecto.

A Alejandra, por apoyarme y cuidarme con tanto amor cada día.

DEDICATORIA

A mis padres, Pedro y Emérita, por haberme forjado como la persona que soy en la actualidad; les debo muchos de mis logros entre los que incluye este. También agradezco a todas las personas que de una manera u otra manera me ayudaron durante toda mi carrera universitaria.

CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE CÓDIGOS

RESUMEN

En este proyecto, se realizó la integración de un Calificador que soporta diferentes métricas de calificación con la plataforma Edx. El mismo que sirve de apoyo al proceso de aprendizaje de la asignatura de Programación Orientada a Objetos de la carrera de Electrónica y Redes de Información de la Facultad de Ingeniería Eléctrica y Electrónica de la Escuela Politécnica Nacional. El documento está divido en 5 capítulos.

En el capítulo I se presentan trabajos afines y estudios previos en torno a soluciones empleadas en la enseñanza de programación a partir del año 2010 hasta la actualidad. Además, se presenta las características de los MOOC analizados, para seleccionar una. Finalmente, se describe las tecnologías encontradas y utilizadas en el desarrollo del presente proyecto.

En el capítulo II se describe la arquitectura propuesta en [1], el escenario de prueba usado para verificar su funcionamiento y los correctivos necesarios para el acoplamiento con el proyecto planteado.

El capítulo III se divide en dos secciones: análisis de requerimientos y diseño. En la sección de análisis de requerimientos, se determina los requisitos que la solución debe satisfacer. En la sección de diseño, se describe el funcionamiento completo del sistema, y se describen los diferentes componentes a través de algunos artefactos diseñados.

El capítulo IV se divide en dos secciones: implementación y evaluación. En la sección de implementación, se presenta la implementación del módulo de calificación automática y todos los artefactos diseñados. En la sección de evaluación se presenta las pruebas realizadas que permitieron verificar la funcionalidad de todo el sistema.

En el capítulo V se presenta las conclusiones y recomendaciones conseguidas en el desarrollo del presente proyecto.

PRESENTACIÓN

La calificación de deberes de Programación Orientada a Objetos de forma manual implica que en ocasiones la retroalimentación llegue de forma tardía. Además, generalmente como principal criterio de calificación se usa únicamente la funcionalidad, lo cual deja otros criterios de evaluación a un lado. El objetivo del presente proyecto es implementar una plataforma que soporte diferentes procesos de calificación automática con retroalimentación instantánea.

En el presente proyecto se integró el Calificador propuesto en [1], que soporta diferentes métricas de calificación con la MOOC Edx. Para esto se hizo una revisión de las plataformas MOOC usadas actualmente para analizar sus características. Luego, se seleccionó a Edx como plataforma base para el presente proyecto, debido a que tiene licencia GNU/GPL, arquitectura flexible y un componente llamado *External Grader*. Para la integración se requiere analizar la arquitectura del Calificador, de cada uno de los componentes, de cada una de las capas y de cada elemento de software. Se usó el IDE Eclipse y el lenguaje de programación Java para comprobar la funcionalidad del Calificador y se realizaron los correctivos necesarios. Usando la metodología de desarrollo de software *Extreme Programming* (XP) se diseñó los artefactos de software necesarios para la creación del módulo de integración del Calificador con Edx. A continuación, se desplegó y configuró el MOOC Edx en un servidor y se codificaron los artefactos de software diseñados. Finalmente, se desplegó este módulo diseñado en la plataforma Edx y se realizó un conjunto de pruebas para verificar su correcto funcionamiento y realizar correcciones.

La actual investigación integró con éxito el Calificador propuesto en [1] con Edx. El mismo que permite la calificación automática con retroalimentación inmediata considerando los submódulos de calificación, la administración de estos y configuración del proceso.

CAPÍTULO 1: MARCO TEÓRICO

En este capítulo se reportarán trabajos afines y estudios previos en torno a soluciones empleadas en la enseñanza de programación a partir del año 2010 hasta la actualidad. Además, se reportarán las características de los MOOC estudiados, haciendo énfasis en el seleccionado.

Así como también, se describirán las tecnologías encontradas y utilizadas en el desarrollo del presente proyecto.

1.1 INTRODUCCIÓN

El aprendizaje de POO (Programación Orientada a Objetos) es un requisito dentro de algunas ingenierías técnicas. La dificultad que tiene el aprendizaje de la misma se ve manifestado en un alto índice de fallos de los estudiantes en las materias encaminadas a la enseñanza de programación [2]. Por lo tanto, este proceso de enseñanza-aprendizaje se torna relevante para los educadores.

Algunas investigaciones han asegurado que el aumento en la cantidad de ejercicios que un estudiante efectúa, y una retroalimentación temprana, es esencial en su proceso de aprendizaje [3]. El aumento de la cantidad de ejercicios y un gran número de estudiantes en ingeniería hacen poco viable un proceso de calificación de forma manual. Por tal motivo, varios proyectos se han encaminado a la creación o uso de herramientas de calificación automática con retroalimentación temprana en el proceso de enseñanza-aprendizaje [4] [5] [6] .

1.2 ESTADO DEL ARTE

La colaboración en el proceso enseñanza-aprendizaje de la asignatura de Programación, tomando en cuenta su importancia en las carreras de ingeniería, continúa siendo la motivación de varios trabajos. Algunas herramientas, despuntando aquellas encaminadas a la calificación automática de ejercicios, han sido y están siendo manipuladas en distintas instituciones élite en el mundo, por ejemplo en: *California State University, Edinburgh Napier University* [7], *MIT (Massachusetts Institute of Technology)* [8] y *Harvard University* [9] .

Hay una gran cantidad de herramientas y plataformas creadas, es por esto que varios autores han elaborado una revisión de herramientas y su progreso en el tiempo.

En [10] se revisan varias herramientas de calificación automática de deberes de programación, tales como: *CourseMaker, Marmoset, WebCat, Grading Tool by Magdeburg University, JavaBrat, AutoLEP, Petcha* y *Virtual Programming Lab*. Se divide en las herramientas maduras y las creadas recientemente. El análisis comparativo entre los dos tipos de herramientas muestra mejoras en este campo de investigación, que incluyen seguridad, más apoyo lingüístico, detección de plagio, etc. Esta revisión incluye la definición y descripción de las características clave, como: Idiomas, tecnología utilizada, infraestructura, etc. Además, se analiza la falta de métricas de calificación para las evaluaciones de deberes de programación, lo cual se identifica como una brecha importante en las herramientas revisadas.

En [11] se establece una tipología de las herramientas de calificación automática. Se determinó los siguientes tipos: herramientas de calificación automática, herramientas multimedia, sistemas inteligentes de tutoría y herramientas de aprendizaje visual. Se muestra una discusión para determinar los parámetros relevantes que se deberían discurrir para comparar y seleccionar una herramienta. Esta revisión busca ayudar a profesores e investigadores que están motivados en realizar nuevas investigaciones para optimizar el proceso de enseñanza-aprendizaje.

Para conseguir un mejor rendimiento en el proceso de enseñanza-aprendizaje de programación no basta el uso de las herramientas de calificación automática, sino que hay nuevas herramientas y plataformas *e-learning* creadas para el aprendizaje. Por lo tanto, en el presente proyecto, se realizó una búsqueda de los trabajos relacionados a partir del año 2010 hasta la actualidad y se muestra su resultado en el ANEXO 1. Dichos trabajos fueron clasificados según ciertos parámetros, incluyendo los siguientes: número de referencias, si soporta varios lenguajes de programación, si se reporta una arquitectura, si son de código abierto, entre otros, con el objetivo de obtener los más relevantes y poder determinar adecuadamente el contexto.

1.3 E-LEARNING

El *E-learning* o aprendizaje en red, es el desarrollo del proceso de educación a distancia, basado en el uso de Internet y de las tecnologías de la información y las telecomunicaciones [12]. El mismo que posibilita un aprendizaje flexible, interactivo y accesible a cualquier estudiante, siendo este último el actor principal dentro del proceso de aprendizaje.

En [13] se analiza la evolución del concepto de *e-learning* en 5 generaciones. La primera generación radica en la idea de red en sí, por ejemplo, herramientas como el correo electrónico que permitían la comunicación. La segunda generación, que tiene lugar a principios de la década de 1990, hace uso de los juegos para uso educativo. La tercera generación nace con las plataformas de aprendizaje o LMS (*Learning Management System*) [14], como aplicaciones que permiten la creación y gestión de contenidos en la plataforma, los cuales son vistos en una página Web. En la cuarta generación se realiza progresos no afines a los LMS, en esta generación radica lo que se ha denominado genéricamente como Web 2.0[1] [15], y que en el área del *e-Learning* se denominó *eLearning* 2.0. La quinta generación se caracteriza por el *marketing* de servicios web 2.0, la mejora considerable de los sistemas de videoconferencia, el fortalecimiento del mercado de los gestores de contenidos y plataformas de *e-Learning*, la computación en la nube o *cloud computing* y los contenidos de manera gratuita. Finalmente, la sexta generación viene de la mano de los Cursos *Online* Masivos en Abierto o MOOC (*Massive Open Online Course*). Los MOOCs son abiertos y gratuitos a un auditorio muy extenso, y puede ser masivo.

1.4 PLATAFORMAS MOOC

Un MOOC es una manera de conectar, colaborar y aprender en el internet que tiene cierta semejanza con una clase. Este es un curso en línea, que ofrece material de aprendizaje para ser modificado, reusado o distribuido de manera gratuita con personas alrededor del mundo [16].

[1] **Web 2.0:** es un término moderno que se refiere a las páginas *World Wide Web*(WWW) y permite a los usuarios interactuar y colaborar con cada uno de los diálogos de social media. Por ejemplo: redes sociales, *blogs, wikis* y otros.

Dentro de las características más notables de un MOOC se pueden señalar las siguientes:

- Permite acceso remoto a través de la Web.
- Implementa control de acceso y gestión de usuarios.
- Cuenta con una arquitectura cliente/servidor.
- Presenta la información de sus páginas siguiendo estándares del protocolo HTTP (*Hyper Text Transfer Protocol*) [17], HTML (*HyperText Markup Language*) [18] y XML (*eXtensible Markup Language*) [19] .
- Soporta bases de datos para el registro de la información de usuario.
- Admite la creación de cursos.
- Cuenta con servicios de comunicación.
- Permite la generación de informes.
- Posibilita la gestión de actividades y material educativo.
- Dispone de una interfaz gráfica donde se integran los diferentes elementos que constituyen los cursos [20].
- Posee materiales tales como: conjunto de vídeos o lecturas.
- Soporta actividades para evaluar de diferentes maneras como: evaluación entre pares, autoevaluación y evaluación automática [21].
- Una tarea tiene fechas de inicio y fin.

Hoy en día, hay muchas plataformas existentes, como Moodle, Canvas, Sakai BlackBoard, Coursera, Udacity, o Edx. Muchos estudiantes y universidades están utilizando esas plataformas. Muchas universidades como Harvard y el MIT, además empresas como Google y Microsoft proporcionaron apoyo, que impulsó a los MOOCs a ganar importancia en el contexto de la educación superior [16].

Existen varias formas de clasificar los cursos MOOC. La más popular distingue dos tipos de MOOC: los MOOC comerciales o de código cerrado (xMOOC) y los MOOC conectivistas o de código abierto (cMOOC) [21].

1.4.1 PLATAFORMAS MOOC DE CÓDIGO CERRADO (XMOOC)

Los cursos xMOOC se han hecho más conocidos y se ofrecen a través de plataformas comerciales, por ejemplo: Coursera, Blackboard y Udacity.

1.4.1.1 Coursera

Coursera es una plataforma MOOC. Este es un sitio web educativo con fines de lucro y tiene un gran número de socios universitarios de alrededor del mundo [22]. A continuación, se lista las principales características de la plataforma:

- Proporciona una aplicación móvil para que los estudiantes accedan al curso en su teléfono.
- Las universidades pagan a Coursera una cuota para recibir cursos.
- No es posible que las universidades modifiquen la plataforma Coursera para agregar nuevas herramientas [23].

1.4.1.2 Udacity

Udacity es una plataforma MOOC que ofrecen cursos gratuitos y pagados en Ciencias de la Computación. Sus cursos son asincrónicos, es decir, que los estudiantes pueden comenzar en cualquier momento que deseen [24]. Una característica única sobre Udacity es que sus conferencias de vídeo incluyen muchos cuestionarios que aparecen en pocos minutos y el contenido de la prueba está relacionado con el contenido del video. Esta característica ayuda a comprobar si los estudiantes siguen la conferencia de video. Udacity no está trabajando con muchas universidades. Una característica similar a Coursera, es que no es posible que terceros modifiquen su plataforma para agregar nuevas herramientas [25].

1.4.2 PLATAFORMAS MOOC DE CÓDIGO ABIERTO (CMOOC)

Los cursos cMOOC surgieron con la idea de código abierto, gratuito y conocimiento compartido. La motivación para la creación de estos cursos fue: la creatividad, la autonomía, la creación de conocimiento por parte de los estudiantes, y el aprendizaje social y colaborativo.

1.4.2.1 Canvas

Canvas es una plataforma de código abierto, que fue desarrollado por una empresa de tecnología educativa con sede en *Salt Lake City*. La infraestructura también desarrolló la Red Canvas, siendo una plataforma MOOC [26]. A continuación, se listan características importantes de la plataforma:

- Permite grabar audio y video dentro de la plataforma.

- Proporciona una aplicación móvil (en IOS y Android) para que los estudiantes y profesores accedan al curso en su teléfono.
- Su infraestructura es abierta, adaptable, confiable y con tecnologías para subir a la nube [27].
- Canvas no tiene una API (*Application Programming Interface*)[2] para añadir tipos de preguntas, para crear un nuevo tipo de pregunta personalizada, se debe usar el estándar de *Learning Tools Interoperability* (LTI) [29].
- Para crear un nuevo tipo de ejercicio en esta plataforma, se tiene que desarrollarlo como una aplicación independiente que admita el estándar LTI y luego integrarlo en Canvas.

1.4.2.2 Edx [30]

Edx es una iniciativa de acceso gratuito, entorno abierto creada por las Universidades de Harvard y el MIT, compuesto por decenas de instituciones en el mundo en un grupo llamado *xConsortium* [31]. Edx, es la segunda gran plataforma para MOOCs, la cual ha ratificado la importancia de la creación de cursos masivos y cambiado la educación abierta. Edx fue fundada en mayo de 2012 como se observa en la Figura1.1

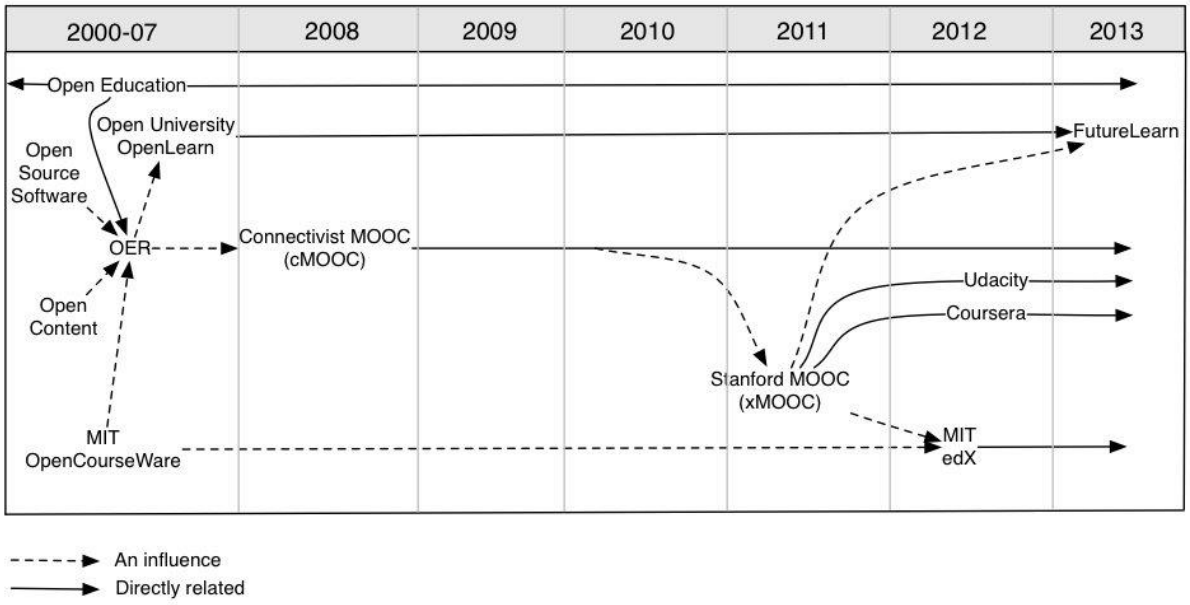


Figura 1.1 Evolución de las plataformas MOOCs y la Educación Abierta [32]

[2] **API:** es un mecanismo para la comunicación entre componentes de software. Puede intercambiar mensajes o datos en formato XML o JSON [28].

Las ideas y tendencias clave que se muestran en la Figura 1.1 incluyen:

- Se muestra cuando las plataformas MOOC se han creado y como han crecido en el tiempo.
- El desarrollo de cursos en línea es un modelo de evolución con el mercado, la mejora a sí mismos ha sido importante para ofrecer una gama más amplia de soluciones a los estudiantes [33].
- MIT lanzó *OpenCourseWare* en el año 2002 y la OU (*Open University*) creó *OpenLearn* en el año 2006, lo que representa un desarrollo continuo del movimiento de educación abierta.
- Las grandes instituciones motivadas por el desarrollo de las MOOCs, han creado nuevas plataformas de aprendizaje abierto, por ejemplo: desde el 2012, el MIT lanzó Edx y OU lanzó *Futurelearn* [32].

1.4.2.3 Moodle [34]

Moodle es una plataforma de aprendizaje que proporciona a administradores, profesores y estudiantes ambientes de aprendizaje personalizados, mediante un sistema integrado seguro, único y robusto. Dicha plataforma es de libre distribución que se constituye en una aplicación web que posibilita la creación de cursos online. Tiene integrado herramientas como: calendarios, foros, wikis, correo electrónico, etc., para apoyar el proceso de enseñanza-aprendizaje.

1.5 SELECCIÓN DE LA PLATAFORMA MOOC

Para usar el Calificador, se seleccionó como plataforma base Edx (versión Ficus.3), debido a que posee las siguientes características [12]:

- Arquitectura flexible, esto quiere decir que la arquitectura de la plataforma permite la comunicación con otros componentes, por ejemplo, un Calificador Automático.
- Licencia GNU/GPL (*General Public License* de *GNU's Not Unix*) [35], entonces permite modificar y mejorar su diseño para necesidades específicas.
- Fácil acceso a documentación y descarga de código fuente.

- Proporciona un componente llamado *External Grader*, que recibe respuestas del estudiante a una determinada tarea. Esta es enviada al Calificador para su procesamiento y recibe una retroalimentación inmediata.
- Posee una estructura de curso con temarios. Por ejemplo, si un estudiante en Moodle se desconecta sin haber terminado una tarea, cuando vuelva a conectarse la próxima vez no podrá volver a donde estaba y tienen que buscar la tarea dentro de la carpeta correspondiente para continuar resolviéndola. En cambio, si un alumno que está realizando un ejercicio en la plataforma Edx y cierra su sesión, cuando vuelva a conectarse a la plataforma se abrirá en el ejercicio donde estaba.

Por estas características mencionadas y tomando en cuenta que el objetivo es la Integración del Calificador con una plataforma MOOC, una de las mejores opciones es Edx.

1.6 DESCRIPCIÓN GENERAL DE EDX [30]

La plataforma Edx es un sistema CMS (*Content Management System*) de código abierto y gratuito desarrollado por la empresa Edx. Se utiliza para albergar MOOCs, cursos pequeños y módulos de capacitación. Esta plataforma tiene incluido:

- *Studio*
- LMS
- Módulo XBlock, el cual implementa los diferentes tipos de problemas
- ORA2 XBlock, el cual permite un problema del tipo ORA (*Open Response Assessment*)
- Foro de Discusión

El *Studio* es la herramienta que se utiliza Edx para crear los cursos, es decir la estructura del curso y añadir contenido como: problemas, videos y otros recursos para los estudiantes.

El LMS es la herramienta que los estudiantes utilizan para acceder a los contenidos del curso, como: videos, libros de texto, problemas, foro de discusión y comprobar su progreso en el curso.

El módulo XBlock es una arquitectura para la construcción de componentes de cursos, es decir, los problemas que se ven en el contenido del curso. Un XBlock es un fragmento de aplicación web que se puede agrupar en una página más grande. En lugar de construir una página web completa, un desarrollador de XBlock puede construir un pequeño componente que se centra en un tipo de ejercicio en particular.

1.6.1 CARACTERÍSTICAS PRINCIPALES

Edx es un sistema muy potente, y acorde a [16] es la plataforma MOOC más usada en la actualidad por las funcionalidades que ofrece. A continuación, se listan las principales características de esta plataforma:

- Dispone de una interfaz de usuario fácil de usar e intuitiva.
- Es altamente personalizable.
- Permite asignar permisos a los usuarios del sistema en base a el uso de roles.
- Su funcionalidad puede ser extendida a través de XBlocks.
- Fácil integración con recursos externos.
- Posibilita la colaboración.
- Flexibilidad para añadir nuevas funcionalidades gracias a su diseño modular.
- Proporciona opciones para el monitoreo de actividades realizadas en la plataforma.
- Soporta la configuración de multilenguaje.
- Licencia GNU/GPL, entonces se puede modificar y mejorar su diseño para nuestras necesidades.
- Fácil acceso a la documentación ayuda y descarga de su código fuente.

1.6.2 ARQUITECTURA [36]

La plataforma Edx es una aplicación web para crear, entregar y analizar cursos en línea a gran escala.

Hay muchos componentes principales en la arquitectura Edx, tales como: herramientas externas y clientes, el código base, aplicaciones ejecutas independientemente y los sistemas de persistencia. El componente principal es código base como se puede ver en la Figura 1.2, que contiene las aplicaciones de administración de aprendizaje y de creación de cursos (LMS y *Studio*,

respectivamente). Los componentes se comunican utilizando interfaces de programación de aplicaciones o APIs estables y documentadas.

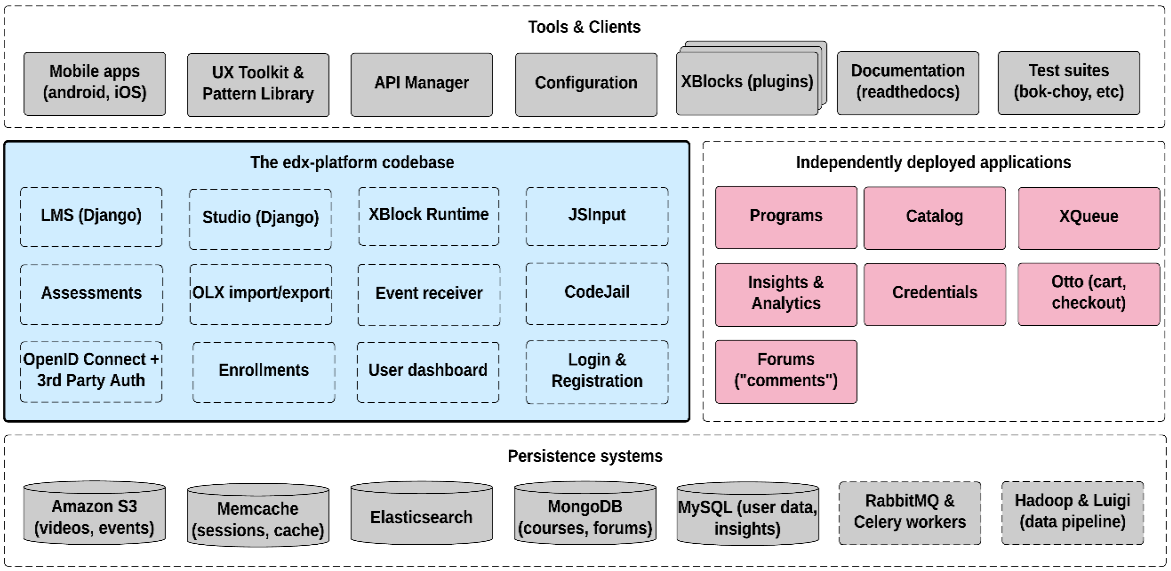


Figura 1.2 Arquitectura de la plataforma Edx [36].

Teniendo en cuenta que Edx es una aplicación web, también contiene tecnologías web como HTML, CSS (*Cascading Style Sheets*) [37], *JavaScript*[3] [38] y también plantillas Mako[4] [39]. Casi todo el código en el servidor Edx está escrito en Python [40], con Django[5] [41] como la estructura de aplicaciones web.

Edx utiliza dos bases de datos: MySQL (*My Structured Query Language*) [16] y MongoDB, como se observa en la Figura 1.2. MySQL es una base de datos relacional compuesta de tablas, mientras que MongoDB almacena las colecciones de documentos BSON (*Binary JSON-JavaScript Object Notation*). La información relacionada con el curso se guarda en MongoDB. Los datos relacionados con los estudiantes se almacenan en MySQL.

1.6.3 EDX FRENTE A OTRAS PLATAFORMAS MOOC

En la Tabla 1.1 se muestra un cuadro comparativo de las características de Edx frente a otros MOOCs como: Coursera, Udacity, Canvas, Moodle.

[3] **JavaScript (JS):** es un lenguaje de programación interpretado de HTML y Web, el cual es orientado a objetos.
[4] **Mako:** es una biblioteca de plantillas escrita en Python. Proporciona una sintaxis familiar que no es de XML que se compila en módulos Python para obtener el máximo rendimiento.
[5] **Django:** es un *framework* para aplicaciones web de alto nivel escrito en Python, el cual fomenta un desarrollo rápido y un diseño limpio y pragmático.

Características	**Coursera**[22]	**Udacity**[25]	**Canvas**[27]	**Moodle**[34]	**Edx**[30]
Código Abierto	X	X	✔	✔	✔
Sistema Operativo	Linux, Mac OS, Windows	Linux, Mac OS, Windows	Linux, Mac OS, Windows	Linux, Mac OS, Windows	Linux, Mac OS, Windows
Max. Tamaño de aula	Ilimitado	Ilimitado	2.000	10.000	300.000
Colaboración	✔	✔	✔	✔	✔
Reportes/Informes	✔	✔	✔	✔	✔
Multi-idioma	✔	✔	✔	✔	✔
Calendario	✔	✔	✔	✔	✔
Admisión de diferentes formatos de contenidos	✔	✔	✔	✔	✔
Autenticación	✔	✔	✔	✔	✔
Calificación	✔	✔	✔	✔	✔
Administración de contenido	✔	✔	✔	✔	✔
Actualizaciones con soporte	✔	✔	✔	✔	✔
Acceso móvil	✔	✔	✔	✔	✔
Personalización	X	X	✔	✔	✔

Tabla 1.1 Comparación de Plataformas MOOC

1.6.4 COMPONENTE EXTERNAL GRADER DE EDX[42]

En Edx, se proporciona muchos tipos de ejercicios, para que los profesores usen en sus cursos [43]. Existen dos grupos para escoger los problemas el momento de crear una actividad: los Problemas Comunes en Blanco y los Problemas Avanzados en Blanco como se puede observar en la Figura 1.3. En el primer grupo se puede escoger un *Checkbox Problem, Dropdown Problem, Multiple Choice Problem*, entre otros; en el segundo grupo se puede escoger un *Custom JavaScript Display and Grading, External Grader, Math Expression Input Problems, Circuit Schematic Builder Problem*, entre otros.

External Grader, es un componente muy útil para los cursos de programación de software en los que se pide a los estudiantes que envíen su código solución a una tarea, por tal motivo, se lo usó en el presente proyecto.

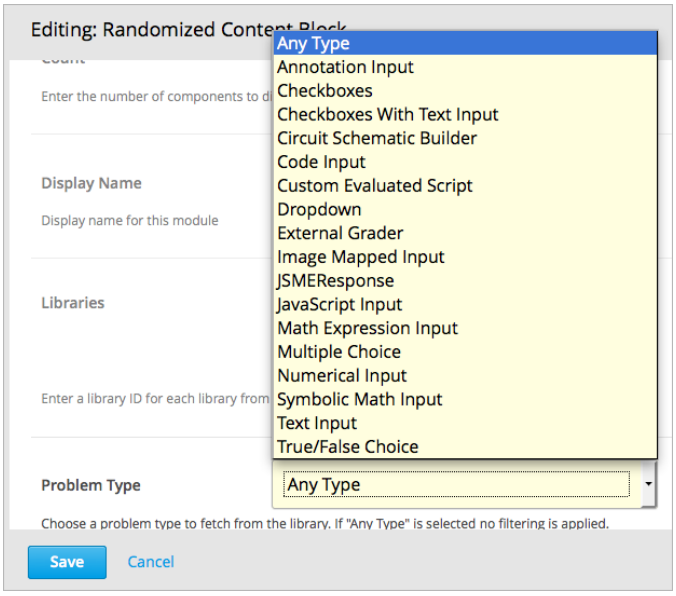


Figura 1.3 Tipos de problemas en Edx.

1.6.4.1 Prueba de Concepto de External Grader

En la Figura 1.4, se define un escenario, el cual requiere que los alumnos resuelvan un deber de POO. Se ha creado un Servidor de Calificación para que el *External Grader* pueda ejecutar su proceso de calificación. Cuando un estudiante introduce su código de respuesta para el deber y selecciona Enviar, el código se envía al Servidor de Calificación para su prueba. Si el código pasa todas las pruebas, el calificador devuelve la retroalimentación que indica que la solución es correcta.

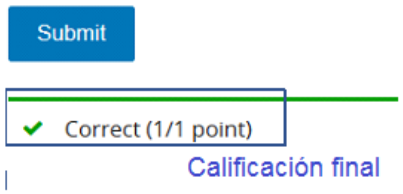


Figura 1.4 Resultado del *External Grader*.

1.6.5 EXTERNAL GRADER Y XQUEUE [43]

Edx posee un servicio llamado *XQueue*, el cual permite comunicar al *External Grader* con un Servidor de Calificación. *XQueue* intercambia información a través de objetos JSON (*JavaScript Object Notation*)[6], un ejemplo se muestra en el Código 1.1. Este objeto contiene los siguientes objetos o llaves [42]:

[6] **JSON**: es un formato para el intercambio de datos y está basado en el lenguaje de programación JavaScript [44].

- *Xqueue_header*, este objeto contiene información para que *XQueue* pueda asociar los resultados con él envió o *submission* correspondiente.
- *Xqueue_files*, este objeto contiene una lista de archivos que fueron enviados por el estudiante. Esta lista está estructurada de tal manera que el nombre del archivo es la clave y la ubicación del archivo es el valor.
- *Xqueue_body,* este objeto contiene a tres objetos *student_info*, *student_response* y *grader_payload*.
- *Student_info*, es un objeto que contiene la siguiente información sobre el estudiante en relación con el envío o *submission*:
    - *Anonymous_student_id*, es un *string* que contiene un identificador anónimo del estudiante.
    - *Submission_time*, es un *string* que contiene una marca de tiempo asociado al momento del envío. Esta marca de tiempo posee el siguiente formato: (aaaammddhhmmss).
    - *Random_seed*, es un valor entero que contiene la semilla que fue utilizada para inicializar el script aleatorio (criptografía) y puede estar adjunto al problema.
- *Student_response*, es un *string* que contiene el código de respuesta enviado por el estudiante.
- *Grader_payload*, es un *string* opcional que se puede usar para enviar información adicional al Calificador.

```
{
  "xqueue_header": {
    "submission_id": 12,
    "submission_key": "280587728458c29e1e66ae0c54a806f4"
  }
  "xqueue_files": {
    "helloworld.c": "http://download.location.com/helloworld.c"
  }
  "xqueue_body":
  "{
    "student_info": {
      "anonymous_student_id": "106ecd878f4148a5cabb6bbb0979b730",
      "submission_time": "20160324104521",
      "random_seed": 334
    },
    "student_response": "def double(x):\n return 2*x\n",
    "grader_payload": "problem_2"
   }"
}
```

Código 1.1 Estructura del objeto JSON [42].

XQueue recoge las respuestas enviadas del estudiante, a partir de ese último crea los objetos JSON. Estos objetos permanecen en *XQueue* hasta que el Calificador los procese o los elimine a cada uno de los envíos de la cola para su calificación. El Calificador examina *XQueue* a través de una interfaz RESTful en un intervalo de tiempo regular. Cuando el Calificador recupera un envío, ejecuta las pruebas en él, luego envía la respuesta a *XQueue* a través de la interfaz RESTful. *XQueue* entrega la respuesta al LMS de Edx.

Los siguientes pasos describen el proceso completo de *External Grader* al evaluar una tarea, como se muestra en la Figura 1.5:

- El estudiante edita su código solución para una tarea en un cuadro de texto de *External Grader*. Luego selecciona Enviar y espera su retroalimentación.
- El Calificador toma el código de *XQueue*.
- En el Calificador se ejecutan las pruebas que ha definido el profesor.
- El Calificador devuelve la calificación final y los comentarios del envío a *XQueue*.
- *XQueue* entrega los resultados al LMS de Edx y son almacenados en la base de datos.
- El estudiante observa la calificación final y la retroalimentación de la tarea.

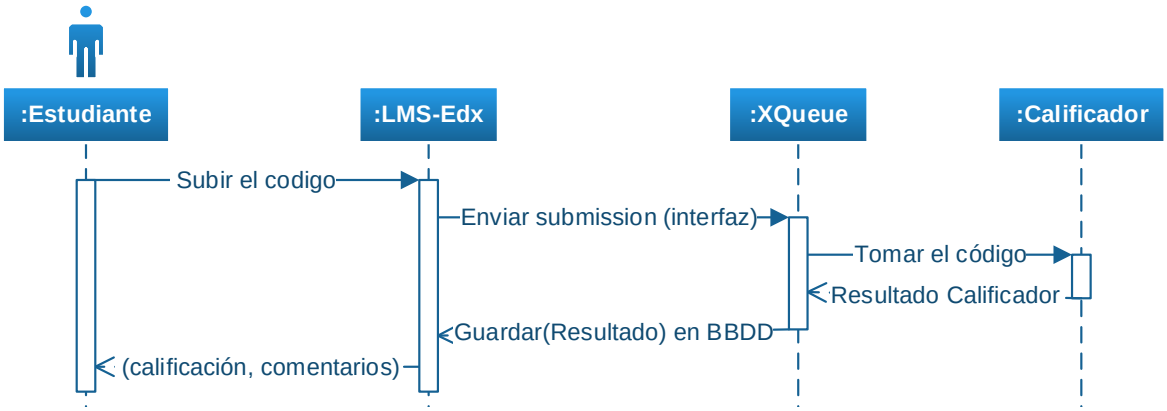


Figura 1.5. Diagrama de proceso al evaluar una tarea.

1.7 APLICACIONES WEB [45]

Las Aplicaciones Web son aquellas herramientas que se ejecutan en Internet, es decir, que la información usada es procesada y guardada en la Web. Están afines

con el almacenamiento en la nube o *cloud storage*, ya que en grandes servidores de Internet es guardada de forma permanente toda la información.

Se describirá brevemente las tecnologías para aplicaciones web, para ilustrar el funcionamiento de Edx.

1.7.1 ARQUITECTURA DE APLICACIONES WEB

La arquitectura de una aplicación web está basada en el modelo cliente-servidor y se compone de los siguientes elementos:

Cliente: es un programa que realiza peticiones al servidor sobre ciertos recursos o servicios; este programa usualmente es un navegador web.

Servidor: está compuesto por un servidor Web, cuya principal función es la de recibir y contestar las peticiones de los clientes. Este proporciona recursos o servicios.

Protocolos: El protocolo por medio del cual se comunican o realizan las peticiones entre el cliente y el servidor web es el protocolo sobre TCP/IP (*Transmission Control Protocol/Internet Protocol*) (puerto 80) o el protocolo HTTPS (*Hyper Text Transfer Protocol Secure*) sobre TCP/IP con SSL (*Secure Sockets Layer*) o TLS(*Transport Layer Security*) (puerto 443), el cual se encarga de definir el formato de los mensajes y la manera en la cual estos serán transmitidos.

A continuación, se detallan las 3 capas de una Aplicación Web.

1.7.1.1 Capa del navegador

En el navegador web se ejecutan las aplicaciones del lado del cliente. Entre sus principales funciones están: recoger información del usuario y enviar al servidor, recibir los resultados del servidor, mostrar los resultados al cliente. Aquí se pueden encontrar arquitecturas o lenguajes que no son exclusivamente lenguajes de programación, entre ellas se pueden mencionar:

- HTML
- JavaScript
- CSS

1.7.1.2 Capa del Servidor

Esta capa es la encargada de recibir datos de la capa navegador, interactuar con la capa de persistencia para realizar operaciones y enviar los resultados a la capa navegador. Para el desarrollo de aplicaciones web en el servidor existen numerosos lenguajes de programación que pueden ser utilizados, algunos de ellos son:

- PHP (*Hypertext Preprocessor*)
- *Java Servlets*
- JSP (*Java Server Pages*)
- ASP (*Active Server Pages*)
- Perl (*Practical Extracting and Reporting Language*)
- Ruby

1.7.1.3 Capa de Persistencia

Las funciones de esta capa son: almacenar, recuperar, mantener y asegurar la integridad de los datos. La mayoría de las aplicaciones web, necesitan almacenar información haciendo uso de bases de datos. Ejemplos de software para crear bases de datos y manipularlas son:

- MySQL
- Oracle
- PostgreSQL
- MongoDB

1.7.2 SERVICIOS WEB

El consorcio W3C(*World Wide Web Consortium)* [7] [46] define los Servicios Web como "sistemas de software diseñados para soportar una interacción interoperable maquina a máquina sobre una red. Este tiene una interfaz descrita en un formato procesable por una máquina (exclusivamente WSDL[8])" [47]. Los Servicios Web suelen ser APIs Web que pueden ser accedidas dentro de una red (principalmente Internet) y son ejecutados en el sistema que los aloja. Entonces, un servicio Web se puede definir de una manera más sencilla como una tecnología que utiliza un conjunto de estándares y protocolos para el intercambio datos entre aplicaciones,

[7] **W3C:** es una comunidad internacional donde las organizaciones trabajan para desarrollar estándares Web.
[8] WSDL (*Web Services Description Language*).

como: SOAP (*Simple Object Access Protocol*), WSDL y UDDI (*Universal Description, Discovery and Integration*). Estos pueden ser desarrollados sobre varios lenguajes de programación para ser implementados sobre muchos tipos de redes de computadores [48].

A continuación, se listan algunos estándares:

- *Web Services Protocol Stack:* es el conjunto de servicios y protocolos de los servicios web y son usados para definir, localizar, implementar y permitir que un servicio web interactúe con otro.
- XML: es uno de los formatos más utilizados para los datos que se vayan a intercambiar sobre la Web.
- WSDL: es un tipo de documento XML, el cual posee los requisitos funcionales necesarios para establecer una comunicación con los servicios web.
- SOAP: es un protocolo de capa aplicación que permite el intercambio de información basada en XML.
- UDDI: es un protocolo para describir servicios, negocios e integrar servicios de negocios. Permite comprobar qué servicios web están disponibles.
- REST (*Representational State Transfer*): es una arquitectura para el diseño de redes y que haciendo uso del protocolo HTTP, proporciona una API que utiliza cada uno de sus métodos (*POST, GET, DELETE, PUT*, y otros) para poder realizar diferentes operaciones entre la aplicación que ofrece el servicio web y el cliente.

1.7.2.1 REST [49]

Restful representa una arquitectura de interfaces de comunicación apoyado en un protocolo de cliente/servidor sin estado HTTP, con operaciones determinadas en el que los recursos están descritos específicamente por URI (*Uniform Resource Identifier*)[9].

Los cuatro principios fundamentales de diseño de REST son:

- Utiliza los métodos HTTP

[9] **URI:** es una cadena de caracteres que vincula los recursos de una red de manera única.

- No mantiene estado
- Expone URIs en forma de directorios
- Transfiere XML, objetos JSON, o ambos

Esta arquitectura hace que los desarrolladores usen los métodos HTTP explícitamente de modo que resulte coherente con la definición del protocolo. Este principio de diseño establece una agrupación uno a uno entre las operaciones de CRUD (*Create, Read, Update and Delete)* y los métodos HTTP. Los métodos utilizados son:

- *POST*, crea un recurso en el servidor
- *GET*, obtiene un recurso
- *PUT*, cambia el estado de un recurso o actualizarlo
- *DELETE*, elimina un recurso

En la Figura 1.6 se presenta un ejemplo de una petición al servicio web de GitHub. En el contenido de la respuesta no se incluye hipervínculos a otras acciones.

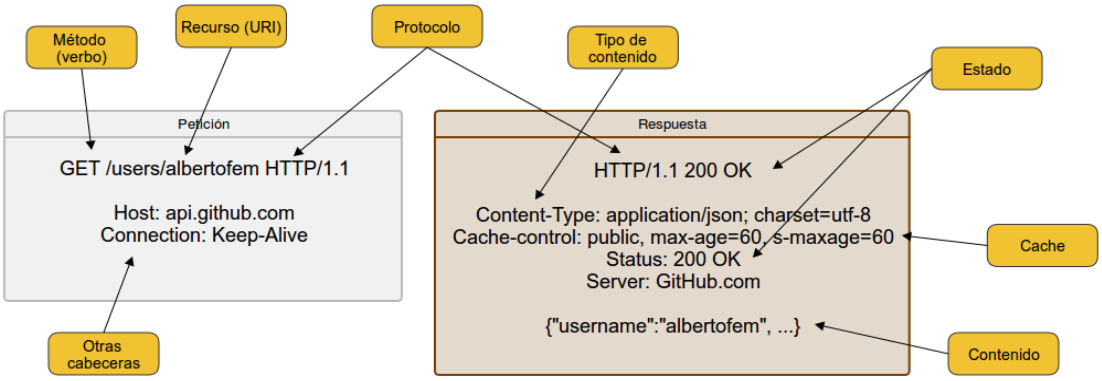


Figura 1.6 Ejemplo de un Servicio Web Rest [50].

1.8 METODOLOGÍA DE DESARROLLO SELECCIONADA

1.8.1 METODOLOGÍA PROGRAMACIÓN EXTREMA O XP (EXTREME PROGRAMMING)

XP [51] es una metodología ágil de desarrollo de software, la cual potencia las relaciones interpersonales, promoviendo el trabajo en grupo, preocupándose por el aprendizaje de los desarrolladores, y brindando un buen clima de trabajo[52]. Esta metodología se basa en retroalimentación continua entre el cliente y el equipo de

desarrollo, permitiendo cambios continuos en el transcurso del proyecto. La metodología XP posee un grupo importante de prácticas y reglas [53], como se puede ver en la Figura 1.7.

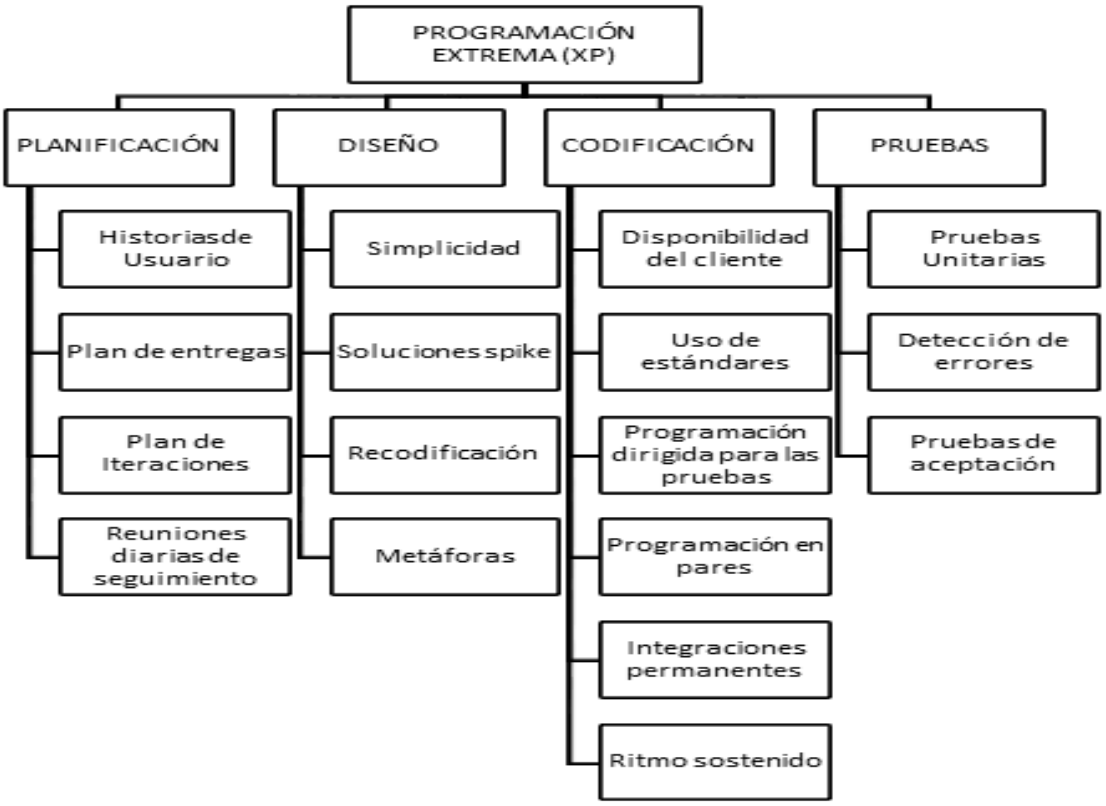


Figura 1.7 Reglas y Prácticas de la Metodología XP [53].

1.8.1.1 Planificación

- Historias de Usuario

  Las historias de usuario son usadas como herramienta usada para especificar los requerimientos y necesidades del software. Cada historia de usuario debe ser muy clara y delimitada para que los programadores logren estimar el tiempo que se demorará su implementación. Estas se obtienen después de una entrevista con el cliente.

- Plan de entregas

  El plan de entregas permite establecer y ordenar las historias de usuario que serán agrupadas para conformar un entregable. Todos los actores del proyecto (cliente, desarrolladores, gerentes, etc.) se reúnen para determinar dicho plan de entregas.

- Plan de Iteraciones

  Las historias de usuario que pertenecen a un entregable son realizadas y probadas en un ciclo de iteración, según con un orden preestablecido. Adicionalmente, para cada historia de usuario se establecen pruebas de aceptación.

- Reuniones diarias de seguimiento

  Las reuniones diarias tienen como objetivo mantener la comunicación entre todo el equipo, compartir problemas y soluciones encontradas en el desarrollo del proyecto.

1.8.1.2 Diseño

- Simplicidad

  Un diseño simple es más rápido de implementar que uno complejo. Por lo tanto, XP propone un diseño más simple, el cual funcione.

- Soluciones *spike*

  Un *spike* o pequeño programa de prueba, es usado cuando es difícil estimar el tiempo para implementar una historia de usuario o aparecen problemas técnicos. Este programa prueba o evalúa diferentes soluciones a un problema, y son desechados luego de su evaluación.

- Recodificación

  La recodificación se basa en escribir de nuevo parte del código de un programa, sin cambiar su funcionalidad, con el objetivo de hacerlo conciso, simple y entendible.

- Metáforas

  XP sugiere utilizar el concepto de “metáfora” para explicar de una forma simple el propósito del proyecto, así como guiar la arquitectura y estructura del mismo. El uso de nombres claros o específicos, que no requieran de mucha explicación, se traduce en ahorro de tiempo.

1.8.1.3 Codificación

- Disponibilidad del cliente

  En esta metodología es fundamental el involucramiento del cliente, para que pueda desarrollarse un proyecto. El cliente no solamente debe participar como ayuda a los desarrolladores, sino siendo parte del equipo.

- Uso de estándares

  XP promueve el uso estándares para la programación, para que sea fácil de entender por todo el equipo y facilite la recodificación.

- Programación dirigida para las pruebas

  Se refiere a las pruebas unitarias realizadas por los desarrolladores. Se deben definir estas pruebas al comienzo del proyecto.

- Programación en pares

  XP propone que un mismo computador, trabajen juntos dos programadores. Al parecer esta técnica aumenta los costos en recursos humanos, pero al trabajar en pares se merman los errores y se alcanzan mejores diseños, subsanando la inversión en horas.

- Integraciones permanentes

  La metodología XP, promueve publicar lo antes posible las nuevas versiones, y si no son las últimas que estén sin errores. Todos los desarrolladores necesitan trabajar siempre sobre la última versión.

- Propiedad colectiva del código

  En XP, todo el equipo de trabajo puede apoyar con nuevas ideas a cualquier parte del proyecto. Así como, cualquier pareja de programadores puede modificar el código que sea necesario para corregir problemas, agregar métodos o recodificar.

- Ritmo sostenido

  La metodología XP, promueve que debe llevarse un ritmo sostenido de trabajo. Se debe planificar el trabajo para mantener un ritmo constante y

razonable, sin sobrecargar al equipo. El trabajo extra desmotiva al grupo y eso afecta a la calidad del producto.

1.8.1.4 Pruebas

- Pruebas Unitarias

  Las pruebas unitarias son muy importantes dentro de esta metodología, debido a que todos los módulos deben pasar las pruebas unitarias antes de ser publicados.

- Detección y corrección de errores

  Cuando se halla un error, este debe ser corregido inmediatamente. Luego, se debe generar nuevas pruebas para comprobar que el error haya sido resuelto.

- Pruebas de aceptación

  Las pruebas de aceptación son establecidas según las historias de usuarios, en cada ciclo de iteración. Solo si todas las pruebas de aceptación son correctas, se puede considerar que una historia de usuario ha sido terminada.

CAPÍTULO 2: ANÁLISIS DE LA ARQUITECTURA DE CALIFICACIÓN

En el presente capítulo se describirá la arquitectura de Calificación propuesta en [1], los correctivos necesarios para el acoplamiento con el proyecto planteado y el escenario de prueba usado para verificar su funcionamiento.

2.1 INTRODUCCIÓN

La arquitectura del Calificador propuesta en [1] está basada en el uso del servicio de orquestación [54]; este último tiene algunas características importantes:

- Control completo de los procesos, llamadas a servicios y provee todo el proceso de calificación.
- Usa un archivo de configuración XML para definir el proceso de calificación.
- Usa llamadas y respuestas a los submódulos del proceso de calificación.

A continuación, se muestra la arquitectura propuesta, el diseño de clases y diagrama de procesos de la arquitectura propuesta con una breve explicación.

2.2 ARQUITECTURA DE CALIFICACIÓN PROPUESTA

En la Figura 2.1 se muestra la arquitectura del Calificador propuesta en [1], basada en capas. Las dos capas superiores son fijas y sus tres capas inferiores son completamente dinámicas.

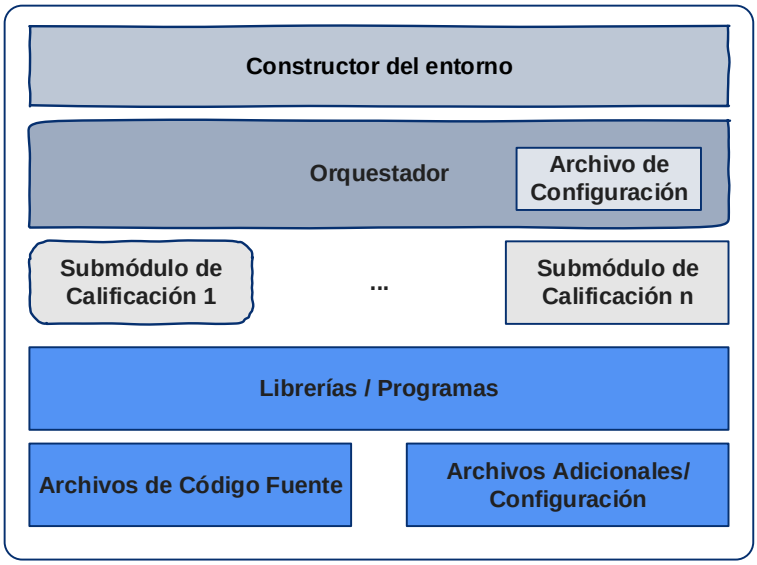


Figura 2.1 Arquitectura propuesta para el proceso de calificación [1].

Esta arquitectura soporta varios criterios o métricas, que son considerados submódulos de calificación, los cuales son independientes entre sí. Esta independencia ayuda a proporcionar modularidad, extensibilidad y flexibilidad.

Dentro de cada capa existen componentes y cada uno de ellos tiene una función específica. La función de cada una de las componentes se describe a continuación:

- **Constructor del entorno,** esta capa está compuesta por dos clases, evaluación y ejecución. El archivo de ejecución se genera dinámicamente. Se utilizan para mantener la compatibilidad con la plataforma. Este nivel puede ser implementado para copiar o mover archivos, codificar o decodificar información y otros procesos necesarios por los submódulos de calificación. Luego de configurar el entorno, se llama al orquestador.
- **Orquestador,** esta capa es la encargada de controlar todo el proceso de calificación. Su primera labor es cargar un archivo de configuración XML donde se indique cómo realizar el proceso de calificación. Dicho archivo debe incluir información sobre los envíos o *submissions* y la lista de submódulos de calificación a usarse, con sus parámetros. Basado en la lista, llama a cada programa asociado al submódulo de calificación.

  Después de ejecutar los submódulos de calificación se obtiene el resultado de cada submódulo; el orquestador procesará todos estos resultados para calcular la calificación final y unir todos los comentarios. Finalmente, la retroalimentación (calificación final y comentarios) se enviará de vuelta a la plataforma.

- **Archivo de configuración del envío (submissionConf.xml),** Este archivo contiene la lista de archivos y parámetros que debe utilizar cada programa asociado a un submódulo dentro del proceso de calificación. La descripción completa de este archivo se mostrará en el Código 2.1.
- **Sub-módulo de calificación,** esta capa contiene a todos los Submódulos de Calificación, los cuales se asocian a una métrica o criterio de calificación. Estos submódulos tienen un programa asociado (para evaluar el código fuente) que se ejecutará en el proceso de calificación. El número de

submódulos no tiene límite y su orden es arbitrario, brindando extensibilidad y flexibilidad respectivamente.

- **Bibliotecas y programas,** esta capa se refiere a programas o paquetes externos requeridos por los submódulos de calificación.
- **Archivos fuente,** esta capa contiene los códigos o archivos enviados por los estudiantes para ser calificados.
- **Archivos adicionales o de configuración,** este componente contiene los archivos definidos por el profesor o requeridos por los submódulos de calificación, como casos de prueba, reglas de calificación, archivos ejecutables, etc. Por ejemplo, en Java, se necesitan herramientas como JUnit [55] o CheckStyle [56].

2.3 PROCESO DE CALIFICACIÓN

El proceso de calificación comienza cuando todos los archivos requeridos están dentro del Servidor de Calificación. En ese momento comienza un proceso ordenado. La Figura 2.2 muestra todo el proceso de calificación. Esta figura es muy similar a la arquitectura mostrada en la Figura 2.1, pero es útil para comprender mejor las responsabilidades de los diferentes elementos de la arquitectura propuesta en el proceso de calificación.

El *script* de evaluación y el archivo de ejecución están dentro de la Arquitectura del Calificador. El archivo de ejecución debe ser cuidadosamente creado para comunicar al Calificador con alguna plataforma. El orquestador llama a cada submódulo de calificación asociado a un programa, recibe las respuestas de cada uno de ellos, calcula la calificación final y une los comentarios para ser enviados como retroalimentación. Finalmente, el Calificador recibe los comentarios y la calificación final, estos son enviados como una respuesta HTTP (Post de RESTful) y se destruye el *sandbox*.

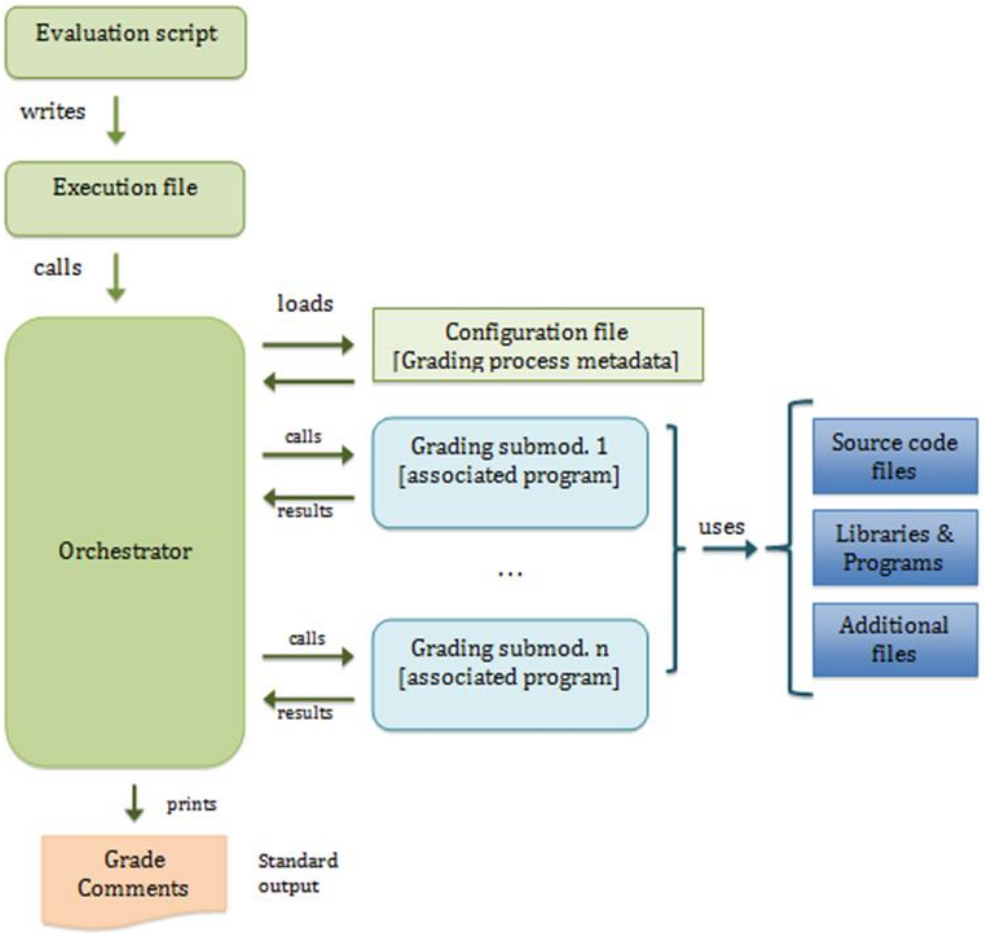


Figura 2.2 Interacción de elementos dentro del proceso de Calificación [1].

Una ventaja de la arquitectura propuesta es que los archivos de evaluación y ejecución no son esenciales para el buen funcionamiento de toda la arquitectura. Esto significa que esta arquitectura puede ser utilizada por otros sistemas simplemente implementando una interfaz, que configura el entorno y llama al orquestador.

2.4 DIAGRAMA DE CLASES DEL CALIFICADOR

Usando la Programación Orientada a Objetos se ha implementado todos los programas y su implementación fue hecha usando Java como lenguaje de programación. Entonces, se muestra el diagrama de clases UML (*Unified Modeling Language*) [57] en la Figura 2.3.

La clase *SubmissionConf* incluye información sobre todo él envío o *submission* que será utilizado por el orquestador para iniciar el proceso de calificación. La clase *GradingSubmoduleConf* representa la información que se utilizará para cada

submódulo de clasificación. Una revisión de la estructura del archivo de configuración XML será útil para entender este par de clases.

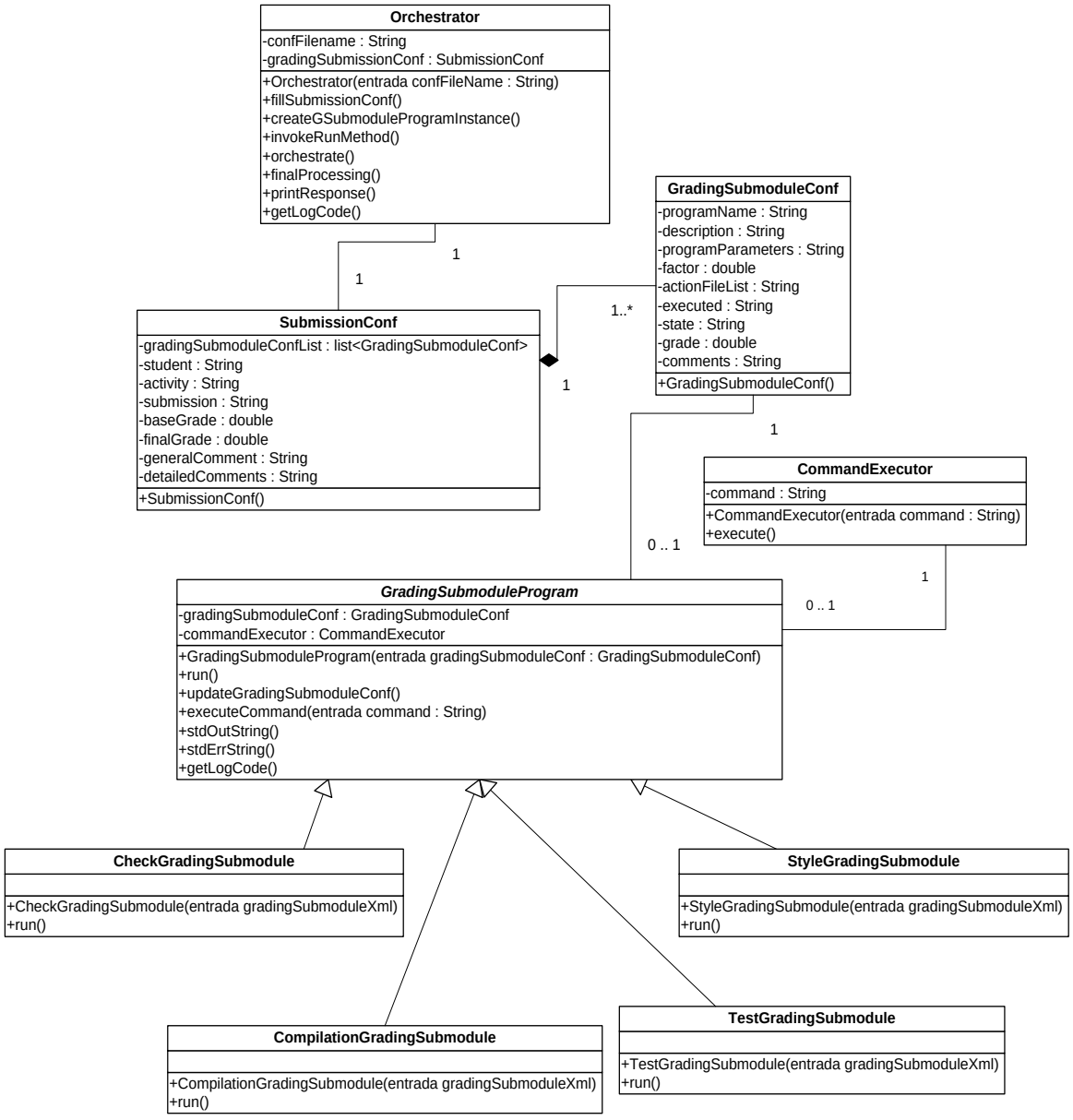


Figura 2.3 Diagrama de clases para el orquestador y los submódulos de calificación [1].

La clase *GradingSubmoduleProgram* es abstracta porque actúa como intermediaria entre el orquestador y cualquier submódulo de calificación que el profesor o el administrador añadirá. Existen cuatro clases hijas de *GradingSubmoduleProgram*, las cuales permiten comprobar la estructura de un conjunto de archivos (*CheckGradingSubmodule*), compilar un conjunto de archivos de código fuente en Java (*CompilationGradingSubmodule*), probar un conjunto de archivos de código fuente contra casos de prueba (*TestGradingSubmodule*) y evaluar el estilo de un

archivo de código fuente en Java (*StyleGradingSubmodule*). Además, esta clase obliga a cada clase hija a implementar la operación abstracta run(). Esta operación debe definir: cómo evaluar un código fuente en Java (enviado por el estudiante), cómo se calculará la calificación final y cómo se constituirá los comentarios de retroalimentación.

La clase *CommandExecutor* permite ejecutar comandos del sistema, obtener los resultados de ejecución y obtener mensajes de error.

La clase *AnyGradingSubmoduleProgram* sirve representar cualquier submódulo de calificación añadido por el administrador o por el profesor.

La clase *Orchestrator* controla todo el proceso de calificación y requiere de una instancia de *SubmissionConf* porque tiene toda la información sobre el envío o *submission*. Para rellenar cualquier dato dentro de esta instancia se usa el archivo de archivo de configuración XML.

2.5 FUNCIONAMIENTO DEL CALIFICADOR

El diagrama de secuencia mostrado en la Figura 2.4 es útil para comprender mejor el funcionamiento. Esta figura sólo incluye objetos y funciones realizadas.

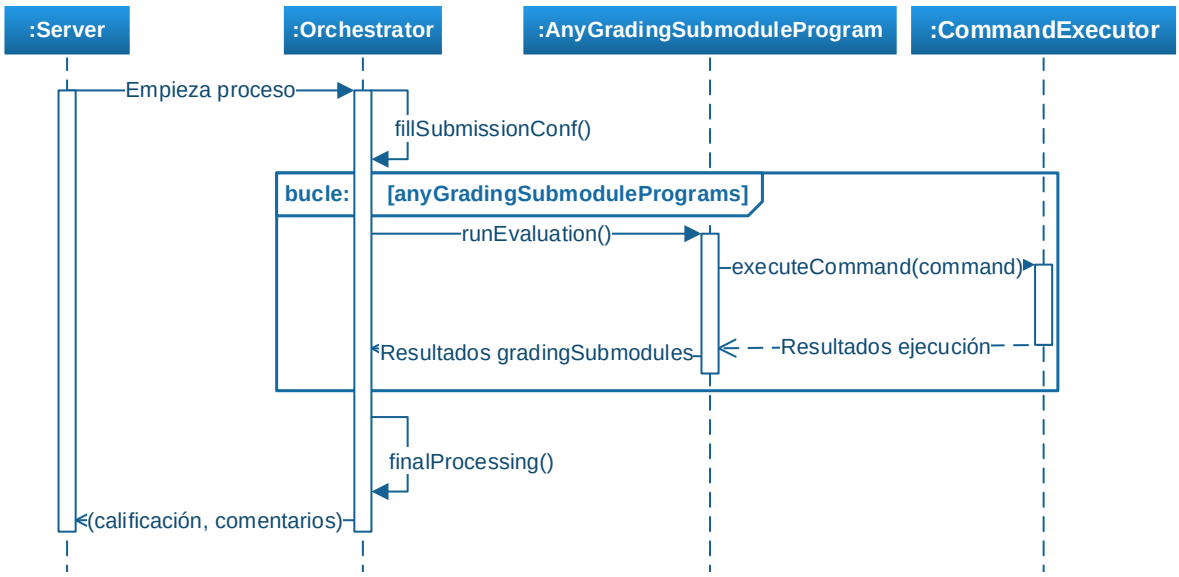


Figura 2.4 Funcionamiento del Calificador.

Algunas cosas por tomar en cuenta de esta Figura son:

- Aunque el archivo de ejecución llama a la Clase *Orchestrator*, el proceso de calificación es iniciado por una instancia del Servidor de Calificación. El archivo de ejecución es sólo un intermediario, por lo tanto, no es un objeto; este último se debe crear para mantener la integración del Calificador con la plataforma Edx.
- Las clases *SubmissionConf* y *GradingSubmoduleConf* no se han representado porque son archivos de configuración.
- La clase *Orchestrator* realiza llamadas a operaciones *fillSubmissionConf()* y *finalProcessing().*
- Finalmente, el Servidor de Calificación inicia el proceso y espera para obtener el resultado con la calificación final y los comentarios para enviar la retroalimentación.

Los requisitos para usar el Calificador son tener todos los archivos necesarios dentro del Servidor y que un programa inicie el proceso llamando a una instancia de la clase *Orchestrator*.

2.6 CORRECTIVOS REALIZADOS EN EL CALIFICADOR

El calificador propuesto en [1] fue personalizado para que pueda ser integrado con Edx. Para el presente proyecto, al Calificador se le realizó algunos correctivos en las clases: *CheckGradingSubmodule.java*, *TestingGradingSubmodule.java, StyleGradingSubmodule.java* y *Orchestator.java*. para la integración con la plataforma Edx usando el módulo *External Grader*. Dichos correctivos se describen en la sección 4.3.1.1.

2.7 ESCENARIO DE PRUEBA

Para probar el funcionamiento del Calificador se creó un escenario en el IDE (*Integrated Development Environment*)[10] Eclipse. Luego de realizar todas estas configuraciones se exportó el Calificador como un archivo JAR ejecutable. Su nombre es *Evaluation.jar* y se almacenó en un repositorio de Github, este se puede observar o descargar en [58].

[10] **IDE:** Un entorno de desarrollo integrado es una aplicación informática que proporciona servicios integrales para facilitarle al desarrollador o programador el desarrollo de software.

2.7.1 DESCRIPCIÓN DEL ESCENARIO

Este escenario, sirvió para verificar el funcionamiento de cada uno de los submódulos del calificador y realizar las correcciones presentas en la sección anterior. Se ha tomado un ejercicio de la asignatura de Programación Orientada a Objetos (POO) y su enunciado es:

TEMA: Herencia - Matemáticas

NOTA: Todas las clases deben poseer: Identificador, Atributos, Constructores, Métodos y propiedades.

Construir una clase final MateEnPOO que amplíe las declaraciones de métodos estáticos de la clase *Math* y que incluya funciones que devuelvan, respectivamente, el máximo, el mínimo, el sumatorio, la media aritmética y la media geométrica de un array de números reales dado como parámetro.

Realizar una clase Aplicación (*main*) en la que pruebe todas las funciones sobreescritas.

2.7.2 ARCHIVOS NECESARIOS PARA EL CALIFICADOR

2.7.2.1 Archivo de configuración XML

Para realizar las pruebas de funcionamiento del Calificador, como se especificó en sección Proceso de Calificación, es necesario el archivo de configuración XML que se presenta en el Código 2.1. Para el presente escenario este archivo posee 3 submódulos de calificación:

- Submódulo de Compilación – *CompilationGradingSubmodule*, en esta sección en la etiqueta <*factor*> se ha definido un factor de calificación de 10 sobre 100 y en la etiqueta <*action-file-list*> se escribe los archivos que se van a compilar (MateEnPOO.java y PruebaMateEnPOO.java).
- Submódulo de Casos de Prueba – *TestingGradingSubmodule*, en esta sección en la etiqueta <*factor*> se ha definido un factor de calificación de 70 sobre 100 y en la etiqueta <*action-file-list*> se escribe el archivo *Corrector*.java, el cual ejecuta pruebas sobre los métodos de la clase MateEnPOO.java y se muestra en la sección 2.7.2.2.

- Submódulo de Estilo – *StyleGradingSubmodule*, en esta sección en la etiqueta <*factor*> se ha definido un factor de calificación de 20 sobre 100 y en la etiqueta <*action-file-list*> se escribe los archivos (MateEnPOO.java y PruebaMateEnPOO.java) que se van a verificar su estilo con el archivo *checkstyle.jar*.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<ns2:submissionConf xmlns:ns2="es.upm.dit.tfm.grad">
   <student>Student indentification</student>
   <activity>Activity indentification</activity>
   <submission>Submission indentification</submission>
   <base-grade>100</base-grade>
   <final-grade>0.0</final-grade>
   <general-comment/>
   <detailed-comments/>
   <submodules>
      <submodule>
         <program-name>es.upm.dit.tfm.grad.sub.CompilationGradingSubmodule</program-name>
         <description>CompilationSubmodule</description>
         <program-parameters />
         <factor>10.00</factor>
         <action-file-list>Matematica/MateEnPOO.java,Matematica/PruebaMateEnPOO.java</action-file-list>
         <executed>no</executed>
         <state>failed</state>
         <grade>0.0</grade>
         <comments />
      </submodule>
      <submodule>
         <program-name>es.upm.dit.tfm.grad.sub.TestingGradingSubmodule</program-name>
         <description>TestingSubmodule</description>
         <program-parameters>Corrector.java</program-parameters>
         <factor>70.00</factor>
         <action-file-list>Matematica/Corrector.java,Matematica.Corrector</action-file-list>
         <executed>no</executed>
         <state>failed</state>
         <grade>0.0</grade>
         <comments />
      </submodule>
        <submodule>
         <program-name>es.upm.dit.tfm.grad.sub.StyleGradingSubmodule</program-name>
         <description>StyleGradingSubmodule</description>
         <program-parameters>librerías/vpl/libs/checkstyle.jar;javadoc.xml;5;5</program-parameters>
         <factor>30.00</factor>
         <action-file-list>Matematica/MateEnPOO.java,Matematica/PruebaMateEnPOO.java</action-file-list>
         <executed>no</executed>
         <state>failed</state>
         <grade>0.0</grade>
         <comments />
      </submodule>
   </submodules>
</ns2:submissionConf>
```

Código 2.1 Archivo de configuración XML.

La descripción de cada uno de los campos del archivo de configuración XML, se detalla en la sección de pruebas de funcionalidad.

2.7.2.2 Clase Corrector.java

La clase Corrector.java es utilizada por el submódulo *Testing Grading Submo*dule para ejecutar algunos casos de prueba sobre las funciones del código respuesta

enviado por el estudiante. Las librerías necesarias para su funcionamiento son: para entrada y salida de datos (java.io.*), y para contar los casos de prueba correctos (org.junit.*) [55]. Dentro de la clase *Corrector.java* se crean funciones o métodos para probar que se calcule correctamente: el mínimo, el máximo, la suma, la media aritmética y la media geométrica de un grupo de números. La estructura de esta clase se muestra en el Código 2.2.

```
package Matematica;
import static org.junit.Assert.*;

import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.OutputStream;
import java.io.PrintStream;
import java.io.PrintWriter;
import java.util.*;

import org.junit.Before;
import org.junit.Test;
import org.junit.runner.JUnitCore;
import org.junit.runner.Result;
import org.junit.runner.notification.Failure;
import Matematica.MateEnPOO;

public class Corrector {

        double [] x = {0.0,1.0,2.0,3.0,4.0,5.0,6.0,7.0,8.0,9.0};
        float delta = (float) 0.001;
        MateEnPOO m;

        @Test (timeout=500)
        public void test_min() {
        }

        @Test (timeout=500)
        public void test_max() {
        }

        @Test (timeout=500)
        public void test_sum() {
        }

        @Test (timeout=500)
        public void test_mediaAritmetica() {
        }

        @Test (timeout=500)
        public void test_mediaGeometrica() {
        }

        public static void main(String[] args) {

        }

}
```

Código 2.2 Estructura de la clase *Corrector.java*.

En el Código 2.3 se muestra la implementación del método *test_min*, el cual ya se sabe la respuesta que se debe obtener del vector (x) que es tipo de dato (*double*), el cual previamente fue declarado e inicializado. Dentro del condicional se resta el

valor obtenido del valor esperado y si es igual o cercano al valor de cero está correcto, caso contrario se envía un error con el mensaje “Error en min.”.

```
@Test (timeout=500)
public void test_min() {
        try{
                if( Math.abs(m.min(x)-0.0) > delta )
                        fail("<p>Error en min. </p>");
        }catch(Exception e){
                fail("<p>Error en min. </p>");
        }
}
```

Código 2.3 Método *test_min* de la clase *Corrector.java*.

En el Código 2.4 se muestra la implementación del método *test_max*, el cual fue implementado de manera similar que el método *test_min*.

```
@Test (timeout=500)
public void test_max() {
        try{
                if( Math.abs(m.max(x)-9.0) > delta )
                        fail("<p>Error en max. </p>");
        }catch(Exception e){
                fail("<p>Error en max. </p>");
        }
}
```

Código 2.4 Método *test_max* de la clase *Corrector.java*.

En el Código 2.5 se muestra la implementación del método *test_sum*, el cual fue implementado de manera similar que el método *test_min*.

```
@Test (timeout=500)
public void test_sum() {
        try{
                if( Math.abs(m.sum(x)-45.0) > delta )
                        fail("<p>Error en max. </p>");
        }catch(Exception e){
                fail("<p>Error en sum. </p>");
        }
}
```

Código 2.5 Método *test_sum* de la clase *Corrector.java*.

En el Código 2.6 se muestra la implementación del método *test_mediaAritmetica*, el cual fue implementado de manera similar que el método *test_min*.

```
@Test (timeout=500)
public void test_mediaAritmetica() {
        try{
                if( Math.abs(m.mediaAritmetica(x)-4.5) > delta )
                        fail("<p>Error en mediaAritmetica. </p>");
        }catch(Exception e){
                fail("<p>Error en mediaAritmetica.</p>");
        }
}
```

Código 2.6 Método *test_mediaAritmetica* de la clase *Corrector.java*.

En el Código 2.7 se muestra la implementación del método *test_mediaGeometrica*, el cual fue implementado de manera similar que el método *test_min*.

```
@Test (timeout=500)
public void test_mediaGeometrica() {
        try{
                if( Math.abs(m.mediaGeometrica(x)-0.0) > delta )
                        fail("<p>Error en mediaGeometrica. </p>");
        }catch(Exception e){
                fail("<p>Error en mediaGeometrica.</p>");
        }

}
```

Código 2.7 Método *test_mediaGeometrica* de la clase *Corrector.java*.

El método *main* de la clase *Corrector.java* cumple las siguientes funciones:

- Ejecuta todos los métodos (*test_*)* definidos anteriormente.
- Verifica que los constructores de todos los métodos (*test_*)* estén correctos.
- Si existe errores al ejecutar los métodos, estos generan mensajes que son recuperados.
- Si un método supera el *timeout* se envía un mensaje indicando el método que tiene este problema.
- Finalmente, se imprime el número de métodos (*test_**) evaluados, métodos (*test_**) fallados y los comentarios enviados en caso de errores.

En el Código 2.8 se presenta la implementación del método *main*.

```
	public static void main(String[] args) {
		//Pasamos las pruebas
		Result result = JUnitCore.runClasses(Corrector.class);
		String fails = "", cadena;

Ubuntu		//Si han fallado todos los tests es que su constructor va mal
		if(result.getFailureCount() == result.getRunCount()){
			if(result.getFailures().get(0).getMessage() == null){
				fails = "<p>No se han podido realizar pruebas a su entrega: Compruebe que el
constructor de su clase funciona de forma correcta. </p>";
			}
		}
		//Recuperamos los mensajes de las pruebas que han fallado
		else {
			for (Failure failure : result.getFailures()) {
				cadena = failure.getMessage();
				//Si se produjo un timeout necesitamos saber en que metodo para cambiar el mensaje
				if(cadena!=null && cadena.contains("timed out")){
					String metodo = failure.getTestHeader().split("_")[0];
					cadena = "<p>Error detectado al probar el metodo "+metodo+": Ha excedido el
tiempo de prueba. Compruebe que no tiene bucles infinitos en su codigo.</p> ";
				}
				fails += cadena;
			}
		}

		//System.out.println(result.wasSuccessful());
		//Para depurar en Eclipse
		System.out.println(result.getRunCount());
		System.out.println(result.getFailureCount());
		System.out.println(fails);
	}
```

Código 2.8 Método *main* de la clase *Corrector*.java.

2.7.3 SOLUCIÓN DEL ESCENARIO PLANTEADO

Se muestra la solución del ejercicio planteado, para la cual se debe crear dos archivos; el primero es la clase final MateEnPOO.java, que se muestra en el Código 2.9; y el segundo PruebaMateEnPOO.java, que se muestra en el Código 2.10 y posee el método *main* para probar los métodos de clase MateEnPOO.java.

```
package Matematica;
public final class MateEnPOO {
	/**
	* @param v
	* @return menor
	*/
	 public static double min (double [] v) {
		double menor = v[0];
		for (int i=0; i<v.length; i++) {
		if (menor>v[i]) { menor=v[i]; }
		}
		return menor;
	 }
	/**
	* @param v
	* @return mayor
	*/
	 public static double max (double [] v) {
		double mayor = v[0];
		for (int i=0; i<v.length; i++) {
		if (mayor<v[i]) { mayor=v[i]; }
		}
		return mayor;
	 }
```

Código 2.9 Estructura de la clase MateEnPoo.java.

```java
package Matematica;
/**
 * Programa MateEnPOO
 */
public class PruebaMateEnPOO {
        public static void main (String [] args) {
                double [] x = new double[10];
                for (int i=0; i<x.length; i++) {
                x[i] = i;
                System.out.println("x["+i+"] = "+x[i]);
        }
        System.out.println("Minimo : " + MateEnPOO.min(x));
        System.out.println("Maximo : " + MateEnPOO.max(x));
        System.out.println("Sumatorio : " + MateEnPOO.sum(x));
        System.out.println("Media arit. : " + MateEnPOO.mediaAritmetica(x));
        System.out.println("Media geom. : " + MateEnPOO.mediaGeometrica(x));
        }
}
```

Código 2.10 Estructura de la clase PruebaMateEnPoo.java.

2.7.4 RESULTADO DEL CALIFICADOR

La Figura 2.5 se muestra la respuesta obtenida por el Calificador.

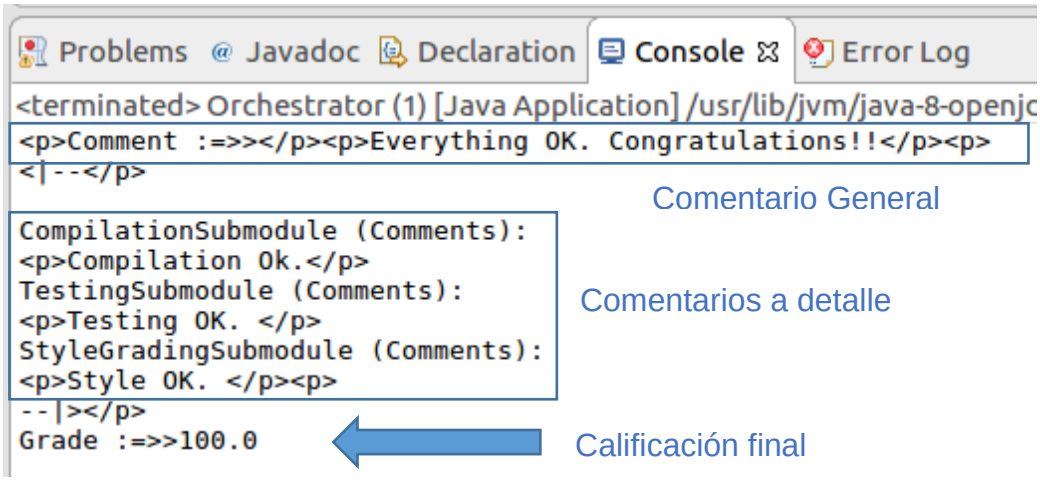


Figura 2.5 Respuesta del Calificador al escenario propuesto.

CAPÍTULO 3: DISEÑO DEL MÓDULO

En este capítulo se establecen los requerimientos para el sistema a desarrollar, mediante el uso de historias de usuario. Se definen los actores que interactúan en el sistema y se describen los servicios que ofrece el mismo.

Adicionalmente, se describirán los artefactos diseñados para la realización del proyecto, se incluyen: las vistas, los diagramas de actividades y secuencia, según se apliquen al proyecto.

XP es la metodología a utilizar durante todo el proceso de desarrollo, esto se debe a su capacidad de respuesta ante imprevistos y su retroalimentación continua en el transcurso de la presente investigación. Puede usarse solo algunas características de XP, las cuales sean necesarias para el sistema.

3.1 ANÁLISIS DE REQUERIMIENTOS

Los requerimientos se han divido en funcionales y no funcionales.

3.1.1 REQUERIMIENTOS NO FUNCIONALES

Son aquellos requerimientos que no se refieren directamente a las funciones específicas que proporciona el sistema sino a las propiedades adicionales. En la Tabla 3.1 se detallan los requerimientos no funcionales.

REQUERIMIENTOS NO FUNCIONALES	
N.	DETALLE
1	En el sistema debe implementarse una herramienta de calificación automática para el lenguaje de programación JAVA, debido a que este lenguaje es usado en la asignatura de POO de la carrera de Electrónica y Redes de Información de la Escuela Politécnica Nacional.
2	El número mínimo de usuarios que debe soportar el sistema es 30, debido a que es el promedio de estudiantes que se matriculan en la asignatura de POO de la carrera de Electrónica y Redes de Información de la Escuela Politécnica Nacional.
3	El sistema deberá implementarse con herramientas basadas en software libre
4	El sistema deberá ser escalable, es decir permitirá añadirle nuevos componentes.
5	La base del sistema deberá ser Edx.

Tabla 3.1 Requerimientos no funcionales del sistema.

3.1.2 REQUERIMIENTOS FUNCIONALES

Los requerimientos funcionales de un sistema describen lo que el sistema debe hacer, los cuales se detallan en la Tabla 3.2.

REQUERIMIENTOS FUNCIONALES	
N.	DETALLE
1	El sistema permitirá crear, modificar y eliminar usuarios.
2	El sistema permitirá asignar un rol (administrador, profesor o estudiante) a cada usuario.
3	El sistema permitirá crear y asignar credenciales de acceso a cada usuario.
4	El sistema permitirá soportar herramientas de calificación automática de ejercicios de Programación Orientada a Objetos.
5	El sistema permitirá al profesor crear, modificar y eliminar contenido en un curso.
6	El sistema permitirá al administrador crear, modificar y eliminar cursos.
7	El sistema permitirá al administrador asignar estudiantes y profesores a un determinado curso.
8	El sistema permitirá al estudiante el acceso y uso del contenido en un curso.
9	El sistema permitirá descargar un reporte de las calificaciones obtenidas por los estudiantes.
10	El sistema permitirá al estudiante ver su progreso en el curso.
11	El sistema permitirá la inscripción de usuarios y enviar la confirmación al correo electrónico.
12	El sistema permitirá al estudiante editar y enviar su código solución escrito en JAVA.
13	El sistema permitirá al administrador y profesor configurar las métricas de calificación.
14	El sistema permitirá calificar el código enviado por los estudiantes en base a las métricas configuradas por el profesor o administrador.

Tabla 3.2 Requerimientos funcionales del sistema.

Tomando en cuenta la información analizada en la sección 2.3, se definen los usuarios, roles y los requisitos, que deberá cumplir el sistema.

3.1.3 USUARIOS Y ROLES

- Administrador, es el encargado de la administración del sistema, realiza las acciones CRUD de Usuarios, Cursos y Herramientas de Calificación. Adicionalmente, definirá aspectos de presentación (tema de la plataforma y creación de cursos) y de autenticación (autorización de accesos y permisos).
- Profesor, es el encargado de la gestión del contenido (crear problemas y subir material de aprendizaje) del curso.
- Estudiante, es quien accede al sistema Web a través de un host y utiliza los contenidos creados por el profesor. Además, podrá editar y enviar su código fuente escrito en JAVA para recibir una retroalimentación inmediata.

3.1.4 HISTORIAS DE USUARIO

En la metodología XP, el primer punto de la etapa de planificación consiste en realizar la especificación de requerimientos a través de la recopilación de historias de usuario.

3.1.4.1 Formato de las historias de usuario

En la Tabla 3.3 se muestra el formato de las historias de usuario, misma que se describe a continuación:

Número de HU:		Fecha:	
Nombre de HU:			
Tipo de Actividad:		Estimación:	
Prioridad:		Iteración:	
Descripción:			
Responsable:			

Tabla 3.3 Formato de las historias de usuario.

- Número de HU, este campo es el identificador de la historia de usuario. Su estructura está conformada por las siglas “HU” y el número de la historia de usuario (dos dígitos).
- Fecha, este campo es la fecha en la cual se escribe la historia de usuario.
- Nombre de HU, este campo es un título descriptivo del requerimiento.
- Tipo de actividad, este campo indica el tipo de actividad y puede tomar uno de los siguientes valores: Nueva, si el requerimiento se va a crear por primera vez; Cambio: para volver a definir el requerimiento; y Afinamiento, para mejorar un requerimiento.
- Estimación, este campo es el número de horas estimadas para la implementación de la historia de usuario.
- Prioridad, este campo indica el grado de importancia de la historia de usuario dentro del desarrollo del sistema. Puede tener tres valores: 1 si es alta, 2 si es media o 3 si es baja.
- Descripción, este campo contiene una explicación breve y concisa del requerimiento.
- Responsable, este campo contiene el nombre del programador responsable de codificar la historia de usuario.
- Iteración, este campo es la prioridad de la historia de usuario.

3.1.5 DESCRIPCIÓN DE LAS HISTORIAS DE USUARIO

A continuación, se definen las historias de usuario planteadas para el presente proyecto.

3.1.5.1 Módulo de Administración de Usuarios

En las Tablas 3.4, 3.5 y 3.6 se presentan las historias de usuario para la administración de Estudiantes, Profesores y Administradores respectivamente.

Número de HU:	HU01	Fecha:	5/6/2017
Nombre de HU:	Administración de Estudiantes		
Tipo de Actividad:	Nueva	Estimación:	
Prioridad:	1	Iteración:	1
Descripción:	El administrador realiza las acciones CRUD de los estudiantes.		
Responsable:	Darwin Poveda		

Tabla 3.4 Historia de usuario para la administración de Estudiantes.

Número de HU:	HU02	Fecha:	5/6/2017
Nombre de HU:	Administración de Profesores		
Tipo de Actividad:	Nueva	Estimación:	
Prioridad:	1	Iteración:	1
Descripción:	El administrador realiza las acciones CRUD de los profesores.		
Responsable:	Darwin Poveda		

Tabla 3.5 Historia de usuario para la administración de Profesores.

Número de HU:	HU03	Fecha:	5/6/2017
Nombre de HU:	Administración de Administradores		
Tipo de Actividad:	Nueva	Estimación:	
Prioridad:	1	Iteración:	1
Descripción:	El administrador realiza las acciones CRUD de los administradores.		
Responsable:	Darwin Poveda		

Tabla 3.6 Historia de usuario para la administración de Administradores.

3.1.5.2 Módulo de Administración de Cursos

En las Tablas 3.7 y 3.8 se presentan las historias de usuario para la administración de Cursos y de Contenido respectivamente. La tabla 3.9 presenta la historia de usuario para la inscripción de usuarios a la plataforma.

Número de HU:	HU04	Fecha:	5/6/2017
Nombre de HU:	Administración de Cursos		
Tipo de Actividad:	Nueva	Estimación:	
Prioridad:	1	Iteración:	1
Descripción:	El administrador realiza las acciones CRUD de los cursos.		
Responsable:	Darwin Poveda		

Tabla 3.7 Historia de usuario para la administración de Cursos.

Número de HU:	HU05	Fecha:	5/6/2017
Nombre de HU:	Administración de Contenido		
Tipo de Actividad:	Nueva	Estimación:	
Prioridad:	1	Iteración:	1
Descripción:	El administrador y profesor realiza las acciones CRUD del contenido del curso.		
Responsable:	Darwin Poveda		

Tabla 3.8 Historia de usuario para la administración de Contenido.

Número de HU:	HU06	Fecha:	5/6/2017
Nombre de HU:	Inscripción de usuarios		
Tipo de Actividad:	Nueva	Estimación:	5
Prioridad:	2	Iteración:	2
Descripción:	Se inscriben los usuarios y reciben la confirmación al correo electrónico.		
Responsable:	Darwin Poveda		

Tabla 3.9 Historia de usuario para la inscripción de usuarios a la plataforma.

3.1.5.3 Módulo de Autenticación

En la Tabla 3.10 se presenta la historia de usuario para la autenticación.

Número de HU:	HU07	Fecha:	5/6/2017
Nombre de HU:	Autenticación		
Tipo de Actividad:	Nueva	Estimación:	
Prioridad:	2	Iteración:	2
Descripción:	Se valida al estudiante, profesor y administrador, antes de acceder al sistema.		
Responsable:	Darwin Poveda		

Tabla 3.10 Historia de usuario para la autenticación.

3.1.5.4 Módulo de Administración de Herramientas

En la Tabla 3.11 se presenta la historia de usuario para la integración de la herramienta y en la Tabla 3.12 se presenta la historia de usuario para la configuración de la herramienta.

Número de HU:	HU08	Fecha:	5/6/2017
Nombre de HU:	Integración de la herramienta		
Tipo de Actividad:	Afinamiento	Estimación:	20
Prioridad:	2	Iteración:	3
Descripción:	El administrador configura el sistema, para la integración del mismo con el Calificador.		
Responsable:	Darwin Poveda		

Tabla 3.11 Historia de usuario para la integración de la herramienta.

Número de HU:	HU09	Fecha:	5/6/2017
Nombre de HU:	Configuración de la herramienta		
Tipo de Actividad:	Afinamiento	Estimación:	20
Prioridad:	2	Iteración:	3
Descripción:	El administrador crea un escenario para evaluar el Calificador propuesto en [1] y corregir la arquitectura del mismo.		
Responsable:	Darwin Poveda		

Tabla 3.12 Historia de usuario para la configuración de la herramienta.

3.1.5.5 Módulo de Reportes

En la Tabla 3.13 se presenta la historia de usuario para la visualización de gráficos estadísticos.

Número de HU:	HU10	Fecha:	5/6/2017
Nombre de HU:	Visualización de gráficos estadísticos		
Tipo de Actividad:	Nueva	Estimación:	
Prioridad:	3	Iteración:	3
Descripción:	El estudiante, el profesor y el administrador visualizan el progreso de un estudiante del curso, a través de un gráfico estadístico.		
Responsable:	Darwin Poveda		

Tabla 3.13 Historia de usuario para la visualización de gráficos estadísticos.

En la Tabla 3.14 se presenta la historia de usuario para la visualización de calificaciones.

Número de HU:	HU11	Fecha:	5/6/2017
Nombre de HU:	Visualización de calificaciones		
Tipo de Actividad:	Nueva	Estimación:	
Prioridad:	2	Iteración:	3
Descripción:	El profesor y el administrador pueden descargar un archivo (.csv), de todas las de las calificaciones de los estudiantes.		
Responsable:	Darwin Poveda		

Tabla 3.14 Historia de usuario para la visualización de calificaciones.

3.1.5.6 Módulo de Interfaz de estudiante

En la Tabla 3.15 se presenta la historia de usuario para el acceso al contenido del curso. En la Tabla 3.16 se presenta la historia de usuario para el envío de tareas a calificar.

Número de HU:	HU12	Fecha:	5/6/2017
Nombre de HU:	Acceso al contenido del curso		
Tipo de Actividad:	Nueva	Estimación:	
Prioridad:	2	Iteración:	2
Descripción:	El estudiante puede visualizar el contenido del curso.		
Responsable:	Darwin Poveda		

Tabla 3.15 Historia de usuario para el acceso al contenido del curso.

Número de HU:	HU13	Fecha:	5/6/2017
Nombre de HU:	Envío de tareas a calificar		
Tipo de Actividad:	Nueva	Estimación:	20
Prioridad:	2	Iteración:	2
Descripción:	El estudiante puede editar en un cuadro de texto su código solución en JAVA. Luego, enviar a calificar y esperar la retroalimentación inmediata.		
Responsable:	Darwin Poveda		

Tabla 3.16 Historia de usuario para el envío de tareas a calificar.

3.1.5.7 Módulo de Interfaz de profesor

En la Tabla 3.17 se presenta la historia de usuario para la configuración de la rúbrica de calificación.

Número de HU:	HU14	Fecha:	5/6/2017
Nombre de HU:	Configuración de la rúbrica de calificación		
Tipo de Actividad:	Nueva	Estimación:	
Prioridad:	2	Iteración:	2
Descripción:	El profesor y administrador pueden crear, modificar o borrar las métricas utilizadas para el proceso de calificación.		
Responsable:	Darwin Poveda		

Tabla 3.17 Historia de usuario para la configuración de la rúbrica de calificación.

3.1.6 ANÁLISIS DE REQUERIMIENTOS DE USUARIO

En base a las historias de usuario expuestas, se definen los siguientes módulos, como se puede ver en la Tabla 3.18.

MÓDULOS	N. DE HU	NOMBRE DE HU	PRIORIDAD	ITERACIÓN
Administración de Usuarios	HU01	Administración de Estudiantes	1	1
	HU02	Administración de Profesores	1	1
	HU03	Administración de Administradores	1	1
Administración de Cursos	HU04	Administración de Cursos	1	1
	HU05	Administración de Contenido	1	1
	HU06	Inscripción de usuarios a la plataforma	1	1
Autenticación	HU07	Autenticación	2	2
Administración de Herramientas	HU08	Integración de la herramienta	2	2
	HU09	Configuración de la herramienta	2	2
Reportes	HU10	Visualización de gráficos estadísticos	3	3
	HU11	Visualización de calificaciones	2	3
Interfaz de estudiante	HU12	Acceso al contenido de un curso	2	2
	HU13	Envío de tareas a calificar	2	2
Interfaz de profesor	HU14	Configuración de la rúbrica de calificación	2	2

Tabla 3.18 Módulos del Sistema.

3.1.7 HISTORIAS DE USUARIO POR ITERACIÓN

3.1.7.1 Primera iteración

En esta iteración se realizan módulos de administración más importantes del sistema. En la Tabla 3.19 se observan los módulos que forman parte de la primera iteración.

MÓDULOS	N. DE HU	NOMBRE DE HU	PRIORIDAD	ITERACIÓN
Administración de Usuarios	HU01	Administración de Estudiantes	1	1
	HU02	Administración de Profesores	1	1
	HU03	Administración de Administradores	1	1
Administración de Cursos	HU04	Administración de Cursos	1	1
	HU05	Administración de Contenido	1	1
	HU06	Inscripción de usuarios a la plataforma	1	1

Tabla 3.19 Historias de usuario de la primera iteración.

3.1.7.2 Segunda iteración

En esta iteración se realiza la autenticación de usuarios, así como la administración de las herramientas de Calificación para que puedan ser usadas por el profesor y el estudiante. Adicionalmente se crea la interfaz de profesor y estudiante. En la Tabla 3.20 se observan los módulos que forman parte de la segunda iteración.

MÓDULOS	N. DE HU	NOMBRE DE HU	PRIORIDAD	ITERACIÓN
Autenticación	HU07	Autenticación	2	2
Administración de Herramientas	HU08	Integración de la herramienta	2	2
	HU09	Configuración de la herramienta	2	2
Interfaz de estudiante	HU12	Acceso al contenido de un curso	2	2
	HU13	Envío de tareas a calificar	2	2
Interfaz de profesor	HU14	Configuración de la rúbrica de calificación	2	2

Tabla 3.20 Historias de usuario de la segunda iteración.

3.1.7.3 Tercera iteración

En esta iteración se realiza los reportes para que los estudiantes puedan ver su avance dentro del curso en gráficos estadísticos y para que los profesores puedan observar y descargar la lista de calificaciones. En la Tabla 3.21 se observan los módulos que forman parte de la tercera iteración.

MÓDULOS	N. DE HU	NOMBRE DE HU	PRIORIDAD	ITERACIÓN
Reportes	HU10	Visualización de gráficos estadísticos	3	3
	HU11	Visualización de calificaciones	2	3

Tabla 3.21 Historias de usuario de la tercera iteración.

3.1.8 HERRAMIENTAS TELEMÁTICAS UTILIZADAS

El sistema se sustentará en la plataforma Edx de acuerdo con el requerimiento no funcional 5 de la Tabla 3.1 y en la sección 1.5 se muestra su justificación.

3.1.8.1 Parámetros para uso de la plataforma

Los parámetros para usar la plataforma Edx, fueron definidos en función de los siguientes módulos, que se muestran en la Tabla 3.23:

MÓDULOS	N. DE HU	NOMBRE DE HU
Administración de Usuarios	HU01	Administración de Estudiantes
	HU02	Administración de Profesores
	HU03	Administración de Administradores
Administración de Cursos	HU04	Administración de Cursos
	HU05	Administración de Contenido
	HU06	Inscripción de usuarios a la plataforma
Autenticación	HU07	Autenticación
Reportes	HU10	Visualización de gráficos estadísticos
	HU11	Visualización de calificaciones
Interfaz de estudiante	HU12	Acceso al contenido de un curso

Tabla 3.22 Módulos con los que cumple la plataforma Edx.

Todos los módulos e historias de usuario de la Tabla 3.23 los cumple la plataforma Edx. Adicionalmente, permite una fácil integración con recursos externos y provee flexibilidad para añadir nuevas funcionalidades gracias a su diseño modular.

3.2 DISEÑO

En esta sección se determina el funcionamiento completo del sistema, y se describen los diferentes componentes a través de algunos artefactos diseñados.

Edx provee un componente llamado *External Grader* detallado en la sección 1.6.4 el cual permite enviar respuestas de los estudiantes y recibir retroalimentación inmediata desde un Calificador. Se aprovechará de este componente para el diseño de los módulos de administración de herramientas, interfaz de usuario e interfaz de profesor.

3.2.1 MÓDULO DE ADMINISTRACIÓN DE HERRAMIENTAS

Este módulo es el encargado de la configuración de la herramienta de calificación automática propuesta en [1] y de integración de la misma con el MOOC Edx.

La configuración de la herramienta se realiza por el administrador, para lo cual usando el IDE Eclipse y el lenguaje de programación JAVA se ha creado un escenario para analizar la arquitectura del Calificador, de cada una de sus capas, de cada uno de sus componentes y de cada elemento de software. Este requerimiento se lo cumplió en la sección 2.6. El detalle de las correcciones realizadas en el Calificador se mostrará en el Capítulo de Implementación y Evaluación. Las clases modificadas fueron: *CheckGradingSubmodule.java, TestingGradingSubmodule.java, StyleGradingSubmodule.java* y *Orchestrator.java.*

3.2.2 MÓDULO DE INTERFAZ DE PROFESOR

Este módulo es el encargado de que el profesor o el administrador puedan crear, modificar o borrar las métricas utilizadas para el proceso de calificación. Después de configurar el proceso, se lo envía al Calificador.

3.2.2.1 Proceso para administrar y configurar el proceso de calificación

El profesor debe administrar las métricas de calificación y enviar al LMS de la plataforma Edx. Dicha configuración se almacena en la Base de Datos. Luego es enviada al *socket submission-queue* de *XQueue*, el cual creará un mensaje de petición HTTP al Calificador. En el Calificador existe un servicio web que recibe la petición, crea el archivo de configuración XML y envía una respuesta afirmativa en el mensaje de respuesta HTTP. Esta respuesta se almacena en la Base de Datos y es enviada a la plataforma para que pueda observar el profesor. En la Figura 3.1 se observa el diagrama de secuencia para administrar y configurar el proceso de calificación.

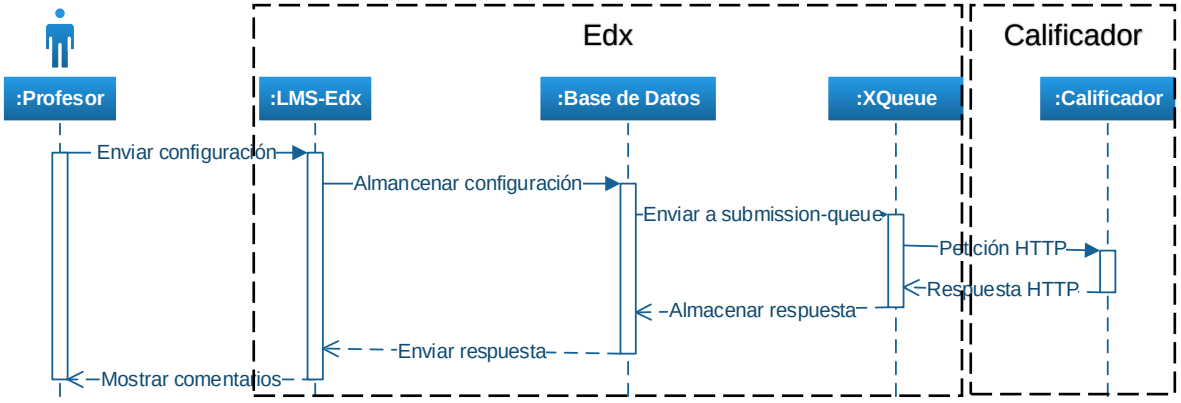


Figura 3.1 Diagrama de secuencia para administrar y configurar el proceso de calificación.

3.2.2.2 Interfaz de Administración y Configuración del proceso de calificación

Para que el profesor pueda administrar y configurar el proceso de calificación de cada problema, se diseñó una interfaz. En la Figura 3.2 se presenta un boceto de dicha interfaz.

Figura 3.2 Interfaz de Administración y Configuración del proceso de calificación.

3.2.2.3 Proceso para crear una tarea

Primero el profesor debe acceder al *Studio* de Edx. Luego debe crear una tarea usando el componente *External Grader* y enviar al *Studio* de la plataforma Edx. La tarea es almacenada en la Base de Datos y devuelve una respuesta afirmativa. En el *Studio* y LMS de Edx aparece creada la tarea. En la Figura 3.3 se presenta el diagrama de secuencia para crear una tarea.

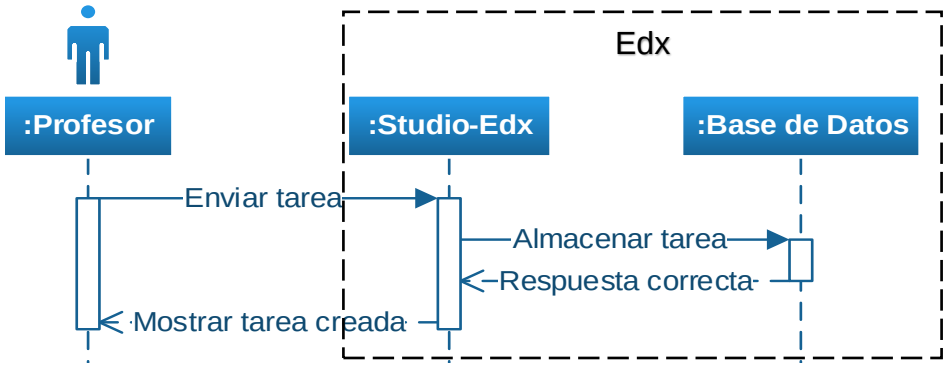


Figura 3.3 Diagrama de secuencia para crear una tarea.

3.2.2.4 Interfaz para crear una tarea

Para que el profesor pueda crear una tarea o problema, se diseñó una interfaz. En la Figura 3.4 se presenta un boceto de dicha interfaz.

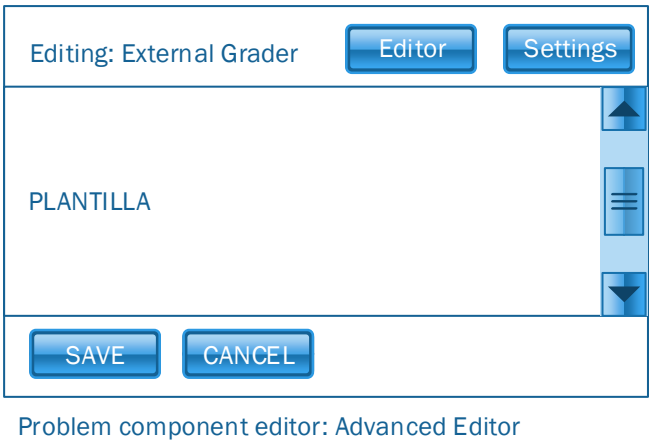


Figura 3.4 Interfaz para crear una tarea o problema.

3.2.3 MÓDULO DE INTERFAZ DE ESTUDIANTE

Este módulo es el encargado de permitir a los estudiantes el acceso al contenido del curso, así como el poder editar y enviar a calificar su código fuente en JAVA.

3.2.3.1 Proceso para enviar una tarea

El estudiante debe editar su código fuente JAVA de respuesta a una tarea, usando un cuadro de texto de *External Grader* y enviar al LMS de la plataforma Edx. El código de respuesta se almacena en la Base de Datos. Luego se envía a *java-queue* de *XQueue*, el cual crea un mensaje de petición HTTP al calificador. En el calificador existe un servicio web que recibe la petición, empieza el proceso de calificación, envía la calificación final y los comentarios en un mensaje de respuesta HTTP. Esta respuesta se almacena en la Base de Datos y es enviada a la plataforma para que el estudiante pueda observar la retroalimentación de la tarea. En la Figura 3.5 se muestra el diagrama de secuencia para enviar una tarea.

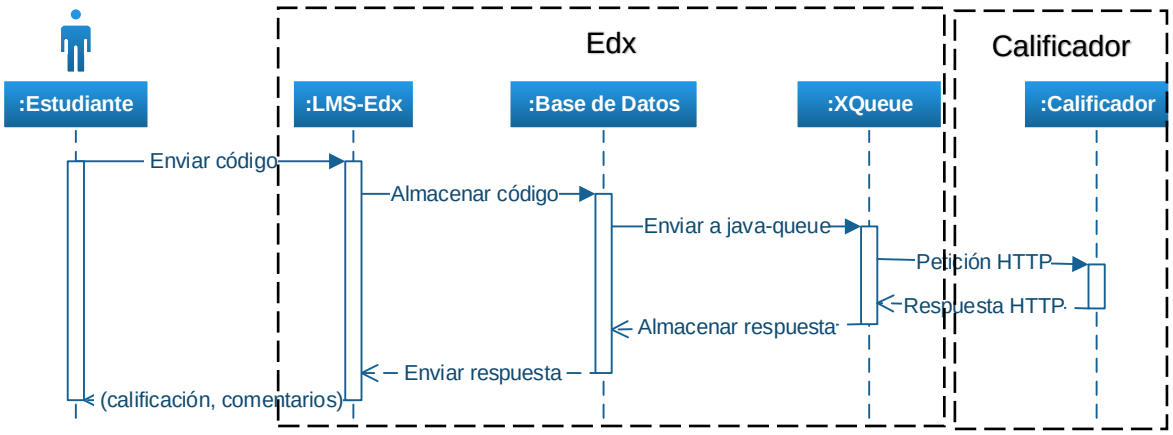


Figura 3.5 Diagrama de secuencia para enviar una tarea.

3.2.3.2 Interfaz para el envío de una respuesta a una tarea

Se ha diseñado una interfaz para el envío de una respuesta a una tarea por parte de los estudiantes. En esta interfaz se tiene un cuadro de texto en la cual el estudiante puede editar su código fuente en Java y dar clic en *Submit* para enviar a calificar. En la Figura 3.6 se presenta la interfaz de envío de una respuesta.

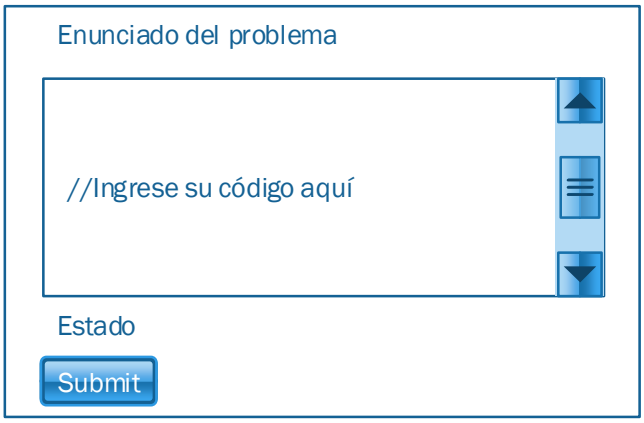


Figura 3.6 Interfaz de envío de una tarea.

La interfaz de retroalimentación a una tarea enviada muestra el estado del envío el cual puede ser *Correct* o *Incorrect*. Adicionalmente se muestra la calificación final y los comentarios recibidos del calificador. En la Figura 3.7 se muestra la interfaz de retroalimentación a un envío.

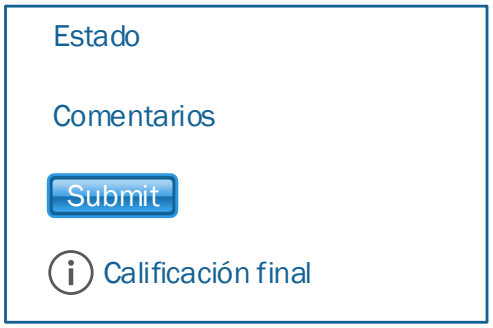


Figura 3.7 Interfaz de retroalimentación a una tarea enviada.

CAPÍTULO 4: IMPLEMENTACIÓN Y EVALUACIÓN

En este capítulo se presentará la implementación del módulo de calificación automática y de los artefactos diseñados en la sección 3.2. Se reportarán un conjunto de pruebas que permitan verificar la funcionalidad de todo el sistema.

Las interfaces que usa el sistema fueron creadas usando el componente *External Grader* de la MOOC Edx.

4.1 INTERACCIÓN ENTRE LOS COMPONENTES DEL SISTEMA

Para la integración de Edx con el Calificador propuesto en [1], se usaron los componentes: LMS, Studio, Base de Datos, *XQueue* de Edx y el Calificador. Estos componentes interactúan entre sí, como se muestra en la Figura 4.1. El Calificador fue implementado por separado de la plataforma Edx.

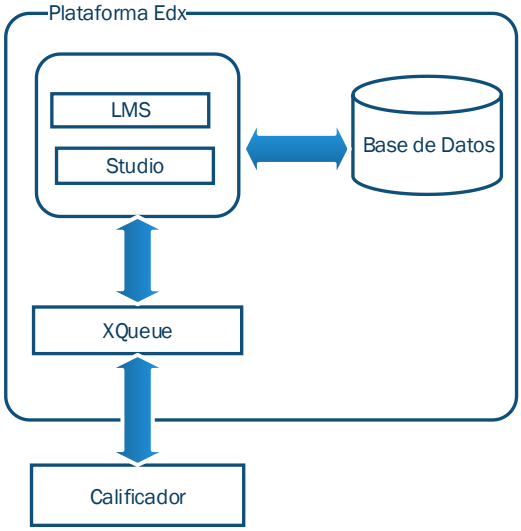


Figura 4.1 Diagrama de interacciones del presente proyecto.

4.2 TECNOLOGÍAS USADAS

Para el desarrollo del sistema se optó por las siguientes tecnologías:

Java

Java (versión 8 *update* 151) es lenguaje de programación utilizado para implementar todas las clases del Calificador. Se creó un escenario en el IDE Eclipse para probar su funcionamiento.

Python

BaseHTTPServer de Python 3, se usó para implementar tres servidores HTTP (servidores Web). Python tiene las siguientes características [59]: es un lenguaje de programación multiparadigma, usa una escritura dinámica. Además, Python da facilidad de extensión; es decir, se pueden escribir nuevos módulos en C o C++.

Ubuntu

Ubuntu (versión 17.04) es el sistema operativo utilizado para la instalación de Edx y del Calificador.

Edx

Edx es la plataforma MOOC usada como base para el presente proyecto. La instalación de Edx (versión Ficus.3) se presenta en el ANEXO 2.

Vagrant [60]

Vagrant (versión 2.0.1) es una herramienta para desarrolladores, la cual permite instalar y configurar software en una máquina virtual para simular un servidor en el que se alojará la aplicación Web

VirtualBox [61]

VirtualBox (versión 5.2.2) es un software virtualizador completo de uso general para arquitecturas x86/amd64, encaminado para servidores, escritorio y al uso incorporado.

4.3 IMPLEMENTACIÓN DE MÓDULOS

En esta sección se muestra la implementación de los módulos de: administración de herramientas y de administración de cursos. Los demás módulos no se implementan ya que fueron justificados en la sección 3.1.7.1. Las interfaces no son implementadas debido a que el componente *External Grader* de Edx cumple con el boceto diseñado y los requerimientos del presente proyecto.

4.3.1 ADMINISTRACIÓN DE HERRAMIENTAS

4.3.1.1 Configuración de la herramienta

El calificador propuesto en [1] fue creado para trabajar en la plataforma Moodle con el módulo VPL (*Vitual Programming Lab*)[11]. Para la presente investigación, al Calificador se le realizó algunos correctivos para la integración con la plataforma Edx usando el módulo *External Grader*. Dichos correctivos se describen a continuación:

- En la clase *CheckGradingSubmodule.java* se esperaba como respuesta del estudiante un archivo comprimido (extensión .zip), en el cual se deben incluir todos archivos para ser calificados. En el presente proyecto los envíos se realizan escribiendo el código de respuesta en un *TextBox* de *External Grader*, por lo tanto, las líneas de código cuya funcionalidad era recibir un archivo comprimido y la descompresión del mismo fueron comentadas, como se puede ver en el Código 4.1.

```
//List of parameters
//String[] programParameters;
//programParameters = this.getGradingSubmoduleConf().getProgramParameters().split(";");
//String zipFilename = programParameters[0];// will be the name of the zip file that contains the source files

String stdout = new String();   //It will store information for standard output as String after every command ex
                                //It is necessary call stdOutStringExecution() method in GradingSubmoduleProgram
String command;
String[] checkFileList;
checkFileList = this.getGradingSubmoduleConf().getActionFileList().split(",");  //List of files to execute the a

//command = "unzip -q -o "+zipFilename;
//this.executeCommand(command);
```

Código 4.1 Correctivo en la clase *CheckGradingSubmodule.java*.

- En la clase *TestingGradingSubmodule.java* se utiliza el archivo *Corrector.java*, el cual posee casos de prueba para evaluar el código enviado por el estudiante. Para compilar el archivo *Corrector.java* se necesita la herramienta JUnit [55]. Esta herramienta permite crear casos de prueba, haciendo instancias de las clases para probar su correcto funcionamiento. Su resultado es el número pruebas correctas e incorrectas, las cuales son almacenadas en un archivo de texto. Este último proceso del submódulo

[11] **VPL:** es la manera de gestionar (edición, ejecución y evaluación) las tareas de programación en Moodle.

tenía errores. Para dar solución a esto, luego de compilar el archivo Corrector.java usando dos archivos JAR ejecutables de JUnit, se aumentó una sentencia para guardar el resultado en el archivo *resultTesting.txt*. Este archivo se creó en el mismo directorio donde se encuentra el Calificador, tal como se muestra en el Código 4.2. De este archivo se extrae el número de pruebas correctas e incorrectas, con las cuales se calcula la calificación del submódulo.

```
command = "java -classpath :junit-4.11.jar:hamcrest-core-1.3.jar "+testFileList[1]+" > resultTesting.txt"
this.executeCommand(command);
```

Código 4.2 Correctivo en la clase *TestingGradingSubmodule.java*.

- En la clase *StyleGradingSubmodule.java* se usa la herramienta *CheckStyle* [56]. Esta herramienta permite verificar que un archivo de código fuente Java (extensión .java) posea todos comentarios y etiquetas necesarias, en formato *Javadoc* [40]. Esta herramienta genera como resultado la ubicación del comentario o la etiqueta faltante del código que está siendo calificado. Se encontró que no funcionaba correctamente el contador de comentarios y etiquetas faltantes en el código para el idioma español. El problema se solucionó añadiendo un operador lógico OR (||) dentro del condicional, para que adicionalmente se busque coincidencias en el idioma español, como se puede ver en el Código 4.3.

```
if(results[i].contains("home") && results[i].contains("com") || results[i].contains("comment") ){
    missComments += 1;
}
if(results[i].contains("home") && results[i].contains("tiqueta")|| results[i].contains("tag")){
    missTags += 1;
```

Código 4.3 Correctivo en la clase *StyleGradingSubmodule.java*.

- Finalmente, en la clase *Orchestator.java* se aumentó la etiqueta HTML para definir un párrafo ( ), como se ve en el Código 4.4. Este cambio permitió generar un resultado organizado al momento de mostrar los comentarios al estudiante en la plataforma Edx.

```
String finalResponse =  "<p>Comment :=>></p>"+this.getSubmissionConf().getGeneralComment()+
                        "<p>\n<|--</p>"+
                        "\n"+this.getSubmissionConf().getDetailedComments()+// Detailed mes
                        "<p>\n--|></p>"+
                        "\nGrade :=>>"+this.getSubmissionConf().getFinalGrade();
```

Código 4.4 Correctivo en la clase *Orchestrator.java*.

4.3.1.2 Integración de la herramienta

Se detalla el código y funcionamiento de los archivos *JavaGrader.py*, *SubmissionConf.py* y *Corrector.py* los cuales fueron creados y son necesarios para la integración de Edx con el Calificador.

El archivo de ejecución ***JavaGrader.py***, recibe de *XQueue* toda la información de la tarea enviada en un objeto JSON, empieza el proceso de calificación y devuelve la retroalimentación de la tarea. *BaseHTTPServer* de Python 3, se utilizó para implementar un servidor HTTP (servidor Web). La estructura del código de *JavaGrader.py* se muestra en el Código 4.5.

```
import BaseHTTPServer
import json
import subprocess
import os
import re

class HTTPHandler(BaseHTTPServer.BaseHTTPRequestHandler):

    def do_POST(self):


def grade(problem_name, student_response):


def process_result(result):


def get_info(body_content):


if __name__ == "__main__":
```

Código 4.5 Estructura del archivo *JavaGrader.py*.

Se creó la clase *HTTPHandler* usando como parámetro *BaseHTTPRequestHandler*, la cual se utiliza para manejar las peticiones HTTP que llegan al servidor. *BaseHTTPRequestHandle*r, no puede responder a ninguna

solicitud HTTP, por lo que se creó funciones como *GET* o *POST*, para manejar cada petición. Esta clase posee 5 métodos o funciones:

- *do_POST*, este método creó un objeto JSON que contiene el estado del proceso, la calificación final y los comentarios, como se presenta en el Código 4.6. Este objeto es enviado como respuesta a *XQueue*.

```
def do_POST(self):
    #Define the post for the result
    body_len = int(self.headers.getheader('content-length', 0))
    body_content = self.rfile.read(body_len)
    problem_name, student_response = get_info(body_content)
    result = grade(problem_name, student_response)
    self.send_response(200)
    self.end_headers()
    self.wfile.write(result)
```

Código 4.6 Método *do_POST* de *JavaGrader.py*.

- *process_result***,** este método captura el estado, la calificación final y los comentarios devueltos por el Calificador, como se muestra en el Código 4.7.

```
def process_result(result):
    #Define the paramaters for the result
    score = result["score"]
    if score == 1 :
        correct = True
    else:
        correct = False
    msg = result["msg"]
    result = {}
    result.update({"correct": correct, "score": score, "msg": msg})
    result = json.dumps(result)
    return result
```

Código 4.7 Método *process_result* de *JavaGrader.py*.

- *grade*, con la información del método *get_info*, se crean todos los códigos fuente de JAVA (.java) enviados por el estudiante dentro del paquete indicado. Adicionalmente, este método ejecuta el proceso de calificación, enviando como parámetro el archivo de configuración Xml, como se exhibe en el Código 4.8. Finalmente, dicha función borra del paquete los códigos

fuente (.java) y de bytes (.class) de Java. Los códigos fuente del estudiante fueron creados por el servicio de *JavaGrader.py*, en cambio los códigos de bytes fueron creados por el Calificador en la compilación.

```
def grade(problem_name, student_response):
    #Split name of files
    program=student_response.split("*Codigo")
    #Write all the java files
    problem_names = problem_name["problem_name"].split(",")
    for i in range(len(program)-1):
        program_name = "/edx/Evaluation/{0}/{1}.java".format(problem_names[0], problem_names[i+1])
        program_code = program[i+1].encode('utf-8')
        source_file = open(program_name, 'w')
        source_file.write(program_code)
        source_file.close()
    result = {}
    #Run the External Grader with the parameters in file submissionConf.xml
    submmission_name = "/edx/Evaluation/{0}/submissionConf.xml".format(problem_names[0])
    p = subprocess.Popen(["java", "-jar", "Evaluation.jar", submmission_name], stdout=subprocess.PIPE, stderr=subprocess.PIPE)
    out, err = p.communicate()
    out = out.split("Grade :=>>")
    #Inicializate the parameters for the result
    message = out[0]
    out1 = out[1].split('\n')
    score = float(out1[0])/100
    result.update({"score": score, "msg": message})
    result = process_result(result)
    #remove student's program from disk
    for i in range(len(program)-1):
        program_name = "/edx/Evaluation/{0}/{1}.java".format(problem_names[0], problem_names[i+1])
        os.remove(program_name)
        program_name = "/edx/Evaluation/{0}/{1}.class".format(problem_names[0], problem_names[i+1])
        os.remove(program_name)
    return result
```

Código 4.8 Método *grade* de *JavaGrader.py*.

- *get_info*, este método extrae del objeto JSON el código fuente JAVA de respuesta enviado, el nombre del paquete y el nombre de los archivos de JAVA a crear, como se presenta en el Código 4.9.

```
def get_info(body_content):
    #Extract the information of Json Object
    json_object = json.loads(body_content)
    json_object = json.loads(json_object["xqueue_body"])
    problem_name = json.loads(json_object["grader_payload"])
    student_response = json_object["student_response"]
    return problem_name, student_response
```

Código 4.9 Método *get_info* de *JavaGrader*.py

- *__name__ == "__main__",* este método crea un servidor HTTP que escucha todas las peticiones de los clientes en el *socket* (dirección IP: *localhost*, puerto: 1710), como se indica en el Código 4.10.

```
if __name__ == "__main__":
    #The server listen for ever in his port
    server = BaseHTTPServer.HTTPServer(("localhost", 1710), HTTPHandler)
    print 'Starting JavaGrader server on port 1710...'
    server.serve_forever()
```

Código 4.10 Método *__name__ == "__main__"* de *JavaGrader.py*.

El archivo de ejecución ***SubmisssionConf.py***, recibe de *XQueue* un objeto JSON con toda la configuración del proceso de calificación. Usando dicho objeto se creó un archivo de configuración XML. Al igual que el archivo *JavaGrader.py*, se usó *BaseHTTPServer* de Python 3, para crear un servidor Web. Los siguientes pasos describen el funcionamiento del archivo *SubmisssionConf.py*:

- Se creó un servidor HTTP que escucha todas las peticiones de los clientes en el *socket* (dirección IP: *localhost*, puerto: 1730).
- Se extrajo del objeto JSON toda la configuración del proceso de calificación, el nombre del paquete y el nombre del archivo XML a crear.
- Con la información anterior se creó el archivo de configuración XML dentro del paquete indicado.
- Se creó un objeto JSON que contiene el estado del proceso y un comentario de creación exitosa del archivo de configuración XML. Este objeto es enviado como respuesta a *XQueue*.

La estructura del archivo *SubmissionConf.py* es similar al de *JavaGrader.py*. Los cambios en el código del archivo *SubmissionConf.py* usado fueron en los métodos:

- *__name__ == "__main__",* este método creó un servidor HTTP que escucha todas las peticiones de los clientes en el *socket* (dirección IP: *localhost*, puerto: 1730), como se presenta en el Código 4.11.

```
if __name__ == "__main__":
    #The server listen for ever in his port
    server = BaseHTTPServer.HTTPServer(("localhost", 1730), HTTPHandler)
    print 'Starting SubmissionConf.py Server on port 1730...'
    server.serve_forever()
```

Código 4.11 Método *__name__ == "__main__"* de *SubmissionConf.py*.

- *grade***,** con la información del método *get_info*, se creó el archivo de configuración XML dentro del paquete indicado, como se muestra en el Código 4.12.

```
def grade(problem_name, student_response):
    problem_names = problem_name["problem_name"].split(",")
    #Write the file SubmissionConf.xml
    program_name = "/edx/Evaluation/{0}/{1}".format(problem_names[0], problem_names[1])
    program_code = student_response.encode('utf-8')
    source_file = open(program_name, 'w')
    source_file.write(program_code)
    source_file.close()
    result = {}
    #Inicializate the parameters for the result
    message = "Ok, File SubmissionConf.xml created"
    score=0
    result.update({"score": score, "msg": message})
    result = process_result(result)
    return result
```

Código 4.12 Método *grade* de *SubmissionConf.py*.

El archivo de ejecución ***Corrector.py***, recibe de *XQueue* un objeto JSON con los casos de prueba a usar en *Testing Grading Submodule*. Usando dicho objeto se creó un archivo de código fuente JAVA. Al igual que el archivo *JavaGrader*.py, se usó *BaseHTTPServer* de Python 3, para crear un servidor Web. Los siguientes pasos describen el funcionamiento del archivo *Corrector*.py:

- Se creó un servidor HTTP que escucha todas las peticiones de los clientes en el *socket* (dirección IP: *localhost*, puerto: 1720).
- Se extrajo del objeto JSON todos los casos de prueba a usar en *Testing Grading Submodule*, el nombre del paquete y el nombre del archivo JAVA a crear.

- Con la información anterior se creó el archivo de código fuente JAVA, dentro del paquete indicado.
- Se creó un objeto JSON que contiene el estado del proceso y un comentario de creación exitosa del archivo Corrector.java. Este objeto es enviado como respuesta a *XQueue*.

La estructura del archivo *Corrector*.py es la misma que *SubmissionConf*.py y *JavaGrader*.py. Los cambios en el código del archivo *Corrector*.py usado fueron en los métodos:

- *__name__ == "__main__",* este método creó un servidor HTTP que escucha todas las peticiones de los clientes en el *socket* (dirección IP: *localhost*, puerto: 1720), como se presenta en el Código 4.13.

```
if __name__ == "__main__":
    #The server listen for ever in his port
    server = BaseHTTPServer.HTTPServer(("localhost", 1720), HTTPHandler)
    print 'Starting Corrector.py Server on port 1720...'
    server.serve_forever()
```

Código 4.13 Método *__name__ == "__main__"* de *Corrector.py*.

- **grade,** con la información del método *get_info*, se creó el archivo de código fuente JAVA, dentro del paquete indicado, como se muestra en el Código 4.14.

```
def grade(problem_name, student_response):
    problem_names = problem_name["problem_name"].split(",")
    #Write the file Corrector.java
    program_name = "/edx/Evaluation/{0}/{1}".format(problem_names[0], problem_names[1])
    program_code = student_response.encode('utf-8')
    source_file = open(program_name, 'w')
    source_file.write(program_code)
    source_file.close()
    result = {}
    #Inicializate the parameters for the result
    message = "Ok, File Corrector.java created"
    score=0
    result.update({"score": score, "msg": message})
    result = process_result(result)
    return result
```

Código 4.14 Método grade de *Corrector*.py.

Luego de instalar Edx en el Sistema Operativo Ubuntu, se realizaron algunas configuraciones, ya que para usar el Calificador es necesario que todos los archivos requeridos estén dentro del mismo. Para el presente proyecto, en *XQueue* de la plataforma Edx se creó tres nuevos *queues*: *submission-queue*, que se comunica con el servicio de *SubmisssionConf.py* para crear el archivo de configuración Xml; *corrector-queue*, que se comunica con el servicio de *Corrector.py* para crear el archivo *Corrector.java*; y *java-queue*, que se comunica con el servicio de *JavaGrader*.py para empezar el proceso de calificación. Para configurar los tres nuevos *queues*, se siguieron los siguientes pasos:

- Se editó el archivo xqueue.env.json.

  sudo vi /edx/app/xqueue/xqueue.env.json

- Dentro del archivo xqueue.env.json, se agregaron las siguientes líneas:

  "java-queue": "http://localhost:1710",

  "corrector-queue": "http://localhost:1720",

  "submission-queue": "http://localhost:1730",

- Se reinició el servicio *XQueue.*

  sudo /edx/bin/supervisorctl restart xqueue:

Luego de este paso se agregaron tres nuevos *queues* a *XQueue*. Para poner en funcionamiento el Calificador se siguieron los siguientes pasos:

- Se ingresó al directorio /edx y se descargó el proyecto *Evaluation* de Github de [16].

  cd /edx

  sudo git clone https://github.com/DarwinPoveda/Evaluation.git

El proyecto *Evaluation*, posee los archivos *Submission.py*, *Corrector.py*, *JavaGrader.py*, *Evaluation.jar* y otros archivos necesarios para el Calificador.

- Se accedió al directorio /edx/Evaluation/ y se ejecutó en segundo plano los tres archivos creados en Python.

cd /edx/Evaluation/

sudo python JavaGrader.py &

sudo python Corrector.py &

sudo python SubmissionConf.py &

- Se verificó que estén funcionando los tres servicios en segundo plano.

  jobs

Al finalizar este punto, el Calificador está funcionando y a la espera de una tarea para ser calificada.

4.3.2 ADMINISTRACIÓN DE CURSOS

4.3.2.1 Implementación para inscripción de usuarios a la plataforma

Para la implementación de la inscripción de usuarios y recibir la confirmación por correo electrónico, se habilitó SMTP (*Simple Mail Transfer Protocol*)[12] en la plataforma Edx. Por defecto, la versión de desarrollo de la plataforma Edx no envía mensajes de correo electrónico a los usuarios cuando éstos se inscriben. Por lo tanto, se configuró la plataforma Edx (versión Ficus.3) para que esta versión de desarrollo también haga uso de un servidor SMTP. Para habilitar SMTP se siguieron los siguientes pasos:

1. Se editó el archivo common.py.
   sudo vi /edx/app/edxapp/edx-platform/cms/envs/common.py
2. Al archivo common.py, se le modificaron los datos resaltados en negrita.
   EMAIL_BACKEND = 'django.core.mail.backends.**smtp**.EmailBackend'
   EMAIL_HOST = **'smtp.gmail.com'**
   EMAIL_PORT = **587**
   EMAIL_USE_TLS = **True**
   EMAIL_HOST_USER = **'proyectopoo2017@gmail.com'**
   EMAIL_HOST_PASSWORD = **'**********'**
   DEFAULT_FROM_EMAIL = **'proyectopoo2017@gmail.com'**
   DEFAULT_FEEDBACK_EMAIL = **'proyectopoo2017@gmail.com'**

[12] **SMTP:** es un protocolo de red utilizado para el intercambio de mensajes de correo electrónico entre computadoras u otros dispositivos.

SERVER_EMAIL = **'proyectopoo2017@gmail.com'**

3. Se editó el archivo aws.py.
   sudo vi /edx/app/edxapp/edx-platform/lms/envs/aws.py
4. Al archivo aws.py, se le modificaron los datos resaltados en negrita.
   EMAIL_HOST = ENV_TOKENS.get('EMAIL_HOST', **'smtp.gmail.com'**)
   EMAIL_PORT = ENV_TOKENS.get('EMAIL_PORT', **587**)
   EMAIL_USE_TLS = ENV_TOKENS.get('EMAIL_USE_TLS', **True**)
   EMAIL_HOST_USER = AUTH_TOKENS.get('EMAIL_HOST_USER', **'proyectopoo2017@gmail.com'**)
   EMAIL_HOST_PASSWORD = AUTH_TOKENS.get('EMAIL_HOST_PASSWORD', **'**********'**)
5. Luego se configuraron las variables de los siguientes archivos con la información anterior:
   - /edx/app/edxapp/cms.env.json
   - /edx/app/edxapp/cms.auth.json
   - /edx/app/edxapp/lms.env.json
   - /edx/app/edxapp/lms.auth.json
6. Se creó el archivo server-vars.yml en el directorio /edx/app/edx_ansible/.
   sudo vi /edx/app/edx_ansible/server-vars.yml
7. En el archivo server-vars.yml se agregaron las siguientes variables:
   EDXAPP_EMAIL_BACKEND:
   'django.core.mail.backends.smtp.EmailBackend'
   EDXAPP_EMAIL_HOST: 'smtp.gmail.com'
   EDXAPP_EMAIL_PORT: 587
   EDXAPP_EMAIL_USE_TLS: True
   EDXAPP_EMAIL_HOST_USER: 'proyectopoo2017@gmail.com'
   EDXAPP_EMAIL_HOST_PASSWORD: '**********'
8. Para guardar los cambios, se actualizó la plataforma Edx. Para esto se ejecutó el siguiente comando:
   sudo /edx/bin/update edx-platform master
9. Luego de ejecutar el paso anterior se cambió el puerto del LMS de Edx, para solucionar esto se modificó el archivo lms dentro del directorio

/etc/nginx/sites-available/. Dentro de este archivo se modificó el puerto del servidor al 80.

sudo vi /etc/nginx/sites-available/lms

10. Finalmente, se reinició el Servidor Web de Nginx.

sudo service nginx restart

4.4 PRUEBAS

4.4.1 PRUEBAS UNITARIAS

Estas pruebas se realizaron en el transcurso de la implementación para probar cada uno de los módulos implementados.

En la Tabla 4.1. se detallan las pruebas que se realizaron sobre la administración de las herramientas de calificación y administración de los cursos.

Módulo	Historia de Usuario	Resultado	Descripción
Administración de Herramientas	Configuración de la herramienta	✔	Se verificó el correcto funcionamiento del Calificador mediante la creación de un escenario de prueba.
	Integración de la herramienta	✔	Se comprobó que se comunica el Calificador con la plataforma Edx.
Administración de Cursos	Inscripción de usuarios a la plataforma	✔	Se verificó que la plataforma permite la inscripción y matriculación en curso, mediante el envío notificaciones al correo electrónico de los usuarios.

Tabla 4.1 Pruebas Unitarias.

4.4.2 PRUEBAS DE FUNCIONALIDAD

Las pruebas de funcionalidad fueron realizadas por los usuarios para asegurar que el sistema cumpla con los requisitos solicitados en las historias de usuario planteadas en la etapa de diseño.

Para comprobar que el sistema web era responsivo se utilizaron 2 dispositivos diferentes: computadora personal y teléfono inteligente.

El escenario para estas pruebas fue un ambiente donde el usuario tenga acceso a internet para ingresar al sistema web a través de los diferentes dispositivos.

En la Tabla 4.2 se detallan las características de hardware de los dispositivos.

Dispositivos	Características
Servidor	Marca: Dell
	Procesador: Core i7
	Memoria RAM: 16 GB
Computador personal	Marca: Lenovo
	Modelo: Thinkpad X260
	Procesador: Core i5
	Memoria RAM: 8 GB
Teléfono inteligente	Marca: Apple
	Modelo: iPhone SE
	Procesador: Chip A9
	Memoria RAM: 2 GB

Tabla 4.2 Características del hardware.

En la Tabla 4.3 se detallan las características de software de los dispositivos utilizados.

Dispositivos	Características
Servidor	Sistema Operativo: Ubuntu 17.04
Computador personal	Sistema Operativo: Windows 10
	Navegador: Google Chrome
Teléfono inteligente	Sistema Operativo: iOS 10.3.3
	Navegador: Safari

Tabla 4.3 Características del software.

Las pruebas se realizaron sobre la computadora personal y el teléfono inteligente. A continuación, se presentarán las pruebas realizadas.

4.4.2.1 Crear una tarea

Descripción del escenario de prueba

Se ha tomado un ejercicio de la asignatura de Programación Orientada a Objetos (POO) y su enunciado es:

TEMA: Ecuación de 2do. Grado

REALIZAR EL CÓDIGO DEL SIGUIENTE ENUNCIADO. DEBE ESTAR ESCRITO TODO LO NECESARIO PARA COMPILAR Y EJECUTAR CORRECTAMENTE EL PROGRAMA SIN ERRORES.

Realizar una clase llamada Raices, donde se representará los valores de una ecuación de 2º grado. Se tendrán los 3 coeficientes como atributos, llamados a, b y c. Las operaciones que se podrán hacer son las siguientes:

- getDiscriminante(): devuelve el valor del discriminante (*double*), el discriminante tiene la siguiente formula, (b^2)-4*a*c.
- tieneRaices(): devuelve un booleano indicando si tiene dos soluciones. Para que esto ocurra, el discriminante debe ser mayor o igual que 0.
- tieneRaiz(): devuelve un booleano indicando si tiene una única solución. para que esto ocurra, el discriminante debe ser igual que 0.
- obtenerRaices(): imprime las 2 posibles soluciones.
- obtenerRaiz(): imprime única raíz, que será cuando solo tenga una solución posible.
- calcular(): mostrará por consola las posibles soluciones que tiene nuestra ecuación, en caso de no existir solución, mostrarlo también.

Fórmula ecuación 2º grado: (-b±raiz((b^2)-(4*a*c)))/(2*a) Solo varia el signo delante de -b*/

El *main* a compilar es:

```
public class Principal {

    public static void main(String[] args) {

      System.out.println("\nEcuación 1:");

      Raices ecuacion=new Raices(4,4,4); //Se crea un objeto

      ecuacion.calcular(); //Se realizan cálculos

      System.out.println("\nEcuación 2:");

      Raices ecuacion2=new Raices(1,-9,14); //Se crea un objeto

      ecuacion2.calcular(); // Se realizan cálculos

      System.out.println("\nEcuación 3:");

      Raices ecuacion3=new Raices(4,-4,1); // Se crea un objeto
```

```
        ecuacion3.calcular(); // Se realizan cálculos
    }
}
```

La respuesta a la ejecución del *main()* realizada la clase Raices sería:

Ecuación 1:

No tiene soluciones

Ecuación 2:

Solución X1: 7.0, Solución X2: 2.0

Ecuación 3:

Única solución: 0.5

Se ha creado en Edx un escenario usando el componente *External Grader*. En el Código 4.15 se muestra la configuración básica de este componente con un escenario planteado.

```
<problem>
  <text>
    <p>
      <b>Escriba su codigo:</b>
    </p>
  </text>
  <coderesponse queuename="java-queue">
    <textbox mode="java" tabsize="4"/>
    <codeparam>
      <initial_display>
            </initial_display>
      <grader_payload>
{"problem_name": "Prueba1_2,Raices,Principal"}
            </grader_payload>
    </codeparam>
  </coderesponse>
</problem>
```

Código 4.15 Configuración de un ejercicio usando *External Grader*.

Esta configuración usa etiquetas HTML. Los campos que contiene esta configuración son:

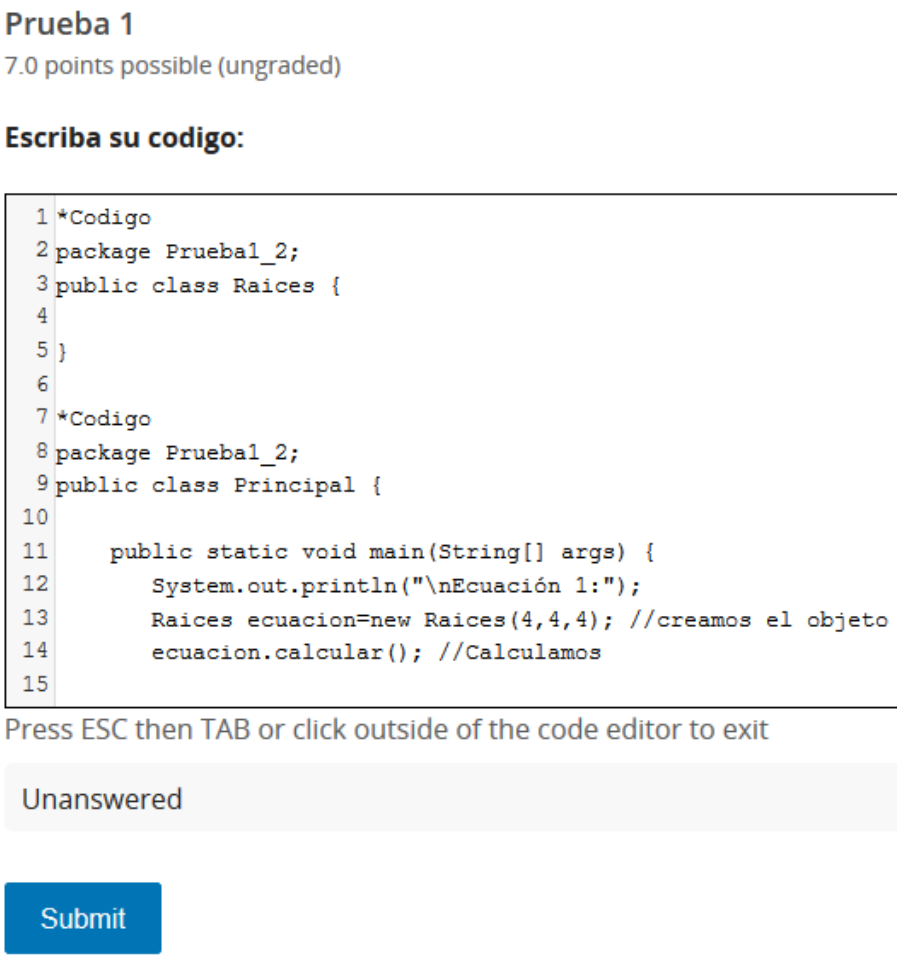
- *Text,* este campo contiene el enunciado del problema, se usa el formato HTML.
- *Coderesponse*, este campo contiene el nombre del *queue*, al cual se le va a enviar un objeto JSON. Este objeto contiene toda la información del envío y el código de respuesta editado por el estudiante.
- *Textbox*, este campo define el tamaño del cuadro de texto y el estilo (sea text, java, xml, etc.) para que el estudiante pueda editar su código. Si existen muchos archivos, se deben separar con la etiqueta (*Codigo).
- *Codeparam*, este campo contiene los parámetros para el envío de la respuesta del estudiante. Este a su vez posee dos campos:
    - *Initial_display*, en este campo se puede editar un texto inicial que aparece en el *TextBox*.
    - *Grader_payload*, contiene un objeto JSON para crear todos los archivos a calificar. El nombre del objeto es *problem_name* y el contenido es el nombre del paquete y los archivos a crearse para su calificación. El paquete por usarse y los nombres de los archivos, deben separarse con comas (,) y sin espacios.

```
*Codigo
package Prueba1_2;
public class Raices {

}

*Codigo
package Prueba1_2;
public class Principal {

    public static void main(String[] args) {
       System.out.println("\nEcuación 1:");
       Raices ecuacion=new Raices(4,4,4); //creamos el objeto
       ecuacion.calcular(); //Calculamos

       System.out.println("\nEcuación 2:");
       Raices ecuacion2=new Raices(1,-9,14); //creamos el objeto
       ecuacion2.calcular(); //Calculamos

       System.out.println("\nEcuación 3:");
       Raices ecuacion3=new Raices(4,-4,1); //creamos el objeto
       ecuacion3.calcular(); //Calculamos
    }
}
```

Código 4.16 Plantilla del escenario planteado.

Dentro de las etiquetas *initial_display* se colocó la plantilla que se muestra en el Código 4.16.

El resultado de la configuración se muestra en el Código 4.17.



Código 4.17 Escenario creado en Edx.

Solución del escenario planteado

Se muestra la solución del ejercicio planteado, para la cual se debe editar el código en el *TextBox* usando la etiqueta *Codigo. En los Códigos 4.18 y 4.19 se muestra la solución de clase Raíces.

```
*Codigo
package Prueba1_2;
/**
 * Clase Raices
 * Representa una ecuacion de 2 grado
 * @author discoduroderoer
 */
public class Raices {

    /*Atributos*/
    private double a;
    private double b;
    private double c;

        /**
         * @param a  g
         * @param b  g
         * @param c  g
         */
    public Raices(double a, double b, double c){
        this.a=a;
        this.b=b;
        this.c=c;
    }

    /**
     * Metodos para obtener las raices cuando hay 2 soluciones posibles
     */
    private void obtenerRaices(){

        double x1=(-b+Math.sqrt(getDiscriminante()))/(2*a);
        double x2=(-b-Math.sqrt(getDiscriminante()))/(2*a);

        System.out.println("Solucion X1");
        System.out.println(x1);
        System.out.println("Solucion X2");
        System.out.println(x2);
    }

    /**
     * Obtiene una unica raiz, cuando solo tiene la posibilidad de er una solucion
     */
    private void obtenerRaiz(){

        double x=(-b)/(2*a);

        System.out.println("Unica solucion");
        System.out.println(x);

    }
```

Código 4.18 Solución del escenario planteado. Parte 1 de 2.

```
    /**
     * Nos devuelve el valor del discriminante,
     * @return Math.pow(b, 2)-(4*a*c)
     */
    public double getDiscriminante(){
        return Math.pow(b, 2)-(4*a*c);
    }

    /**
     * Si el discriminante es mayor que 0 tiene mas de una raiz
     * (No hemos puesto >= ya que puede confundirse con una solucion)
     * @return
     */
    private boolean tieneRaices(){
        return getDiscriminante()>0;
    }

    /**
     * Si el discriminante es igual a cero tiene una sola raiz
     * @return
     */
    private boolean tieneRaiz(){
        return getDiscriminante()==0;
    }

    /**
     * Nos permite calcular las raices de una ecuacion de 2 grado
     */
    public void calcular(){

        if(tieneRaices()){
            obtenerRaices();
        }else if(tieneRaiz()){
            obtenerRaiz();
        }else{
            System.out.println("No tiene soluciones");
        }
    }
}
```

Código 4.18 Solución del escenario planteado. Parte 2 de 2.

4.4.2.2 Administración y configuración de los submódulos de calificación

Para usar el Calificador, el cual ha sido integrado con la plataforma Edx, es necesario crear todos los submódulos de calificación a usar. Estos submódulos deben crearse en el archivo de configuración XML. Para la presente investigación, se creó una plantilla para administrar y configurar el proceso de calificación, la misma que es única para cada uno de los problemas planteados. En dicha plantilla se establecieron tres submódulos de calificación: *CompilationGradingSubmodule*, para compilar un conjunto de archivos de código fuente en Java; *TestGradingSubmodule*, para probar un conjunto de archivos de código fuente

contra casos de prueba; y *StyleGradingSubmodule*, evaluar el estilo de un archivo de código fuente en Java. Esta plantilla solo es visible por el profesor o administrador de la plataforma. El formato de la plantilla para *Compilation Grading Submodule* de la Figura 4.2. se aplica para los tres submódulos.

Figura 4.2 Administración y configuración del proceso de calificación.

Luego de crear una actividad con ejercicios, debe ser publicada y así se podrá ver en el LMS de Edx. Se puede configurar el proceso de calificación, aumentado o disminuyendo submódulos en la plantilla. Si se configura el proceso de calificación, entonces se debe configurar el campo factor de cada submódulo para que entre todos sumen 100, como se muestra en el Código 4.20. Para el presente escenario se configuran los 3 submódulos de calificación de la siguiente manera:

- Submódulo de Compilación – *CompilationGradingSubmodule*, en esta sección en la etiqueta <*factor*> se ha definido un factor de calificación de 10 sobre 100 y en la etiqueta <*action-file-list*> se escriben los archivos que se van a compilar (Raices.java y Principal.java).
- Submódulo de Pruebas – *TestingGradingSubmodule*, en esta sección en la etiqueta <*factor*> se ha definido un factor de calificación de 70 sobre 100 y

en la etiqueta <*action-file-list*> se escribe el archivo Corrector.java, el cual ejecuta algunas pruebas sobre la clase Raices.java.

- Submódulo de Estilo – *StyleGradingSubmodule*, en esta sección en la etiqueta <*factor*> se ha definido un factor de calificación de 20 sobre 100 y en la etiqueta <*action-file-list*> se escriben los archivos (Raices.java) que se van a verificar su estilo con el archivo checkstyle.jar.

```
<?xml version="1.0" encoding="UTF-8"?>
<ns2:submissionConf xmlns:ns2="es.upm.dit.tfm.grad">
   <student>Student indentification</student>
   <activity>Activity indentification</activity>
   <submission>Submission indentification</submission>
   <base-grade>100</base-grade>
   <final-grade>0.0</final-grade>
   <general-comment/>
   <detailed-comments/>
   <submodules>
      <submodule>
        <program-name>es.upm.dit.tfm.grad.sub.CompilationGradingSubmodule</program-name>
        <description>CompilationSubmodule</description>
        <program-parameters />
        <factor>10.00</factor>
        <action-file-list>Prueba1_2/Raices.java,Prueba1_2/Principal.java</action-file-list>
        <executed>no</executed>
        <state>failed</state>
        <grade>0.0</grade>
        <comments />
      </submodule>
      <submodule>
        <program-name>es.upm.dit.tfm.grad.sub.TestingGradingSubmodule</program-name>
        <description>TestingSubmodule</description>
        <program-parameters>Corrector.java</program-parameters>
        <factor>70.00</factor>
        <action-file-list>Prueba1_2/Corrector.java,Prueba1_2.Corrector</action-file-list>
        <executed>no</executed>
        <state>failed</state>
        <grade>0.0</grade>
        <comments />
      </submodule>
      <submodule>
        <program-name>es.upm.dit.tfm.grad.sub.StyleGradingSubmodule</program-name>
        <description>StyleGradingSubmodule</description>
        <program-parameters>librerías/vpl/libs/checkstyle.jar;javadoc.xml;5;5</program-parameters>
        <factor>20.00</factor>
        <action-file-list>Prueba1_2/Raices.java,Prueba1_2/Principal.java</action-file-list>
        <executed>no</executed>
        <state>failed</state>
        <grade>0.0</grade>
        <comments />
      </submodule>
   </submodules>
</ns2:submissionConf>
```

Compilation Grading Submodule

Testing Grading Submodule

Style Grading Submodule

Código 4.19 Estructura del archivo de configuración XML.

El archivo de configuración XML es el mecanismo usado por el orquestador, para que éste conozca qué programas asociados a un submódulo de calificación son

necesarios en el proceso de calificación. XML *Mapper* (JAXB[13]) ha sido utilizado para analizar y crear el archivo de configuración que se muestra en el Código 4.20. Los campos relacionados a los envíos o *submission* son:

- *student*, este campo puede ser el nombre o un identificador asignado por la plataforma del estudiante.
- *activity*, este campo es el nombre de la actividad dentro de la plataforma.
- *submission*, este campo es el identificador de la actividad.
- *base grade*, es la base sobre la cual se calcula la calificación final.
- *final grade*, es la calificación final del envió o *submission*.
- *general comment*, este campo almacena un comentario corto del envío.
- *detailed comments*, este campo almacena los comentarios de cada submódulo de calificación.

Cada submódulo de calificación posee los siguientes campos de información:

- *program name*, este campo contiene la ruta completa del programa asociado a un submódulo de calificación, dicha ruta incluye todo el paquete y la extensión .class no está incluida.
- *description*, este campo contiene una breve descripción del submódulo en cuestión. Este indica la función principal que hace el submódulo. incorporar
- *program parameters*, este campo contiene datos necesarios para el submódulo. Los parámetros pueden ser una ruta a un archivo, números o cadenas de texto. Si existe muchos parámetros, se deben separar con puntos y comas (;) y sin espacios entre ellos.
- *factor*, este campo es un porcentaje que representa el tamaño en el cálculo de la calificación final de un submódulo. La suma de este campo en todos los submódulos debe ser siempre 100.
- *action file list*, este campo tiene una lista de rutas de los archivos que van a ser usados por el submódulo. Si existen muchos archivos, se deben separar con comas (,) y sin espacios.
- *executed*, este campo muestra si el submódulo ha sido ejecutado.

[13] ***Java Architecture for XML Binding*** **(JAXB):** permite incorporar datos XML y ser procesados dentro de una aplicación Java.

- *state*, este campo indica si la ejecución del submódulo es correcta, es decir, obtiene la nota completa; o fallo, es decir, si tuvo algunos errores y obtiene una parte de la nota completa.
- *grade*, este campo es la calificación de cada uno del submódulo. Este es un valor numérico entre 0.00 y 100.00 con 2 decimales.
- *comments*, este campo es la información a detalle como resultado de la ejecución del submódulo.

4.4.2.3 Retroalimentación del Calificador

Para el estudiante, el proceso es muy simple. Primero, edita su código solución para una tarea. Luego selecciona Enviar y espera la retroalimentación del Calificador. Para probar el funcionamiento de todos los submódulos de calificación se ha realizado algunas pruebas:

- **Errores de Compilación**, en la Figura 4.3 se muestra la retroalimentación en el caso de fallas en las complicaciones del código respuestas editado por el estudiante. Se puede dar esto cuando el nombre de clase no coincida con el planteado en el ejercicio, errores de sintaxis o librerías no declaradas.

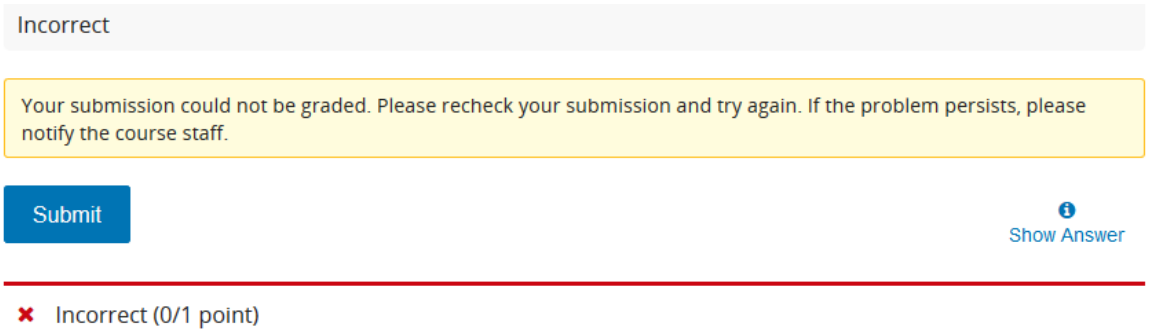


Figura 4.3 Retroalimentación del Calificador con errores de complicación.

- **Sin errores,** se obtiene como retroalimentación del código solución enviado al Calificador: un estado Correcto, un comentario general que todos los submódulos o métricas están correctos y también se muestra detalle los submódulos ejecutados (*CompilationSubmodule, TestingSubmodule y StyleGradingSubmodule*) que se encuentra de manera correcta. Adicionalmente se ha obtenido como calificación final 1/1. En la Figura 4.4. se muestra a detalle la retroalimentación.

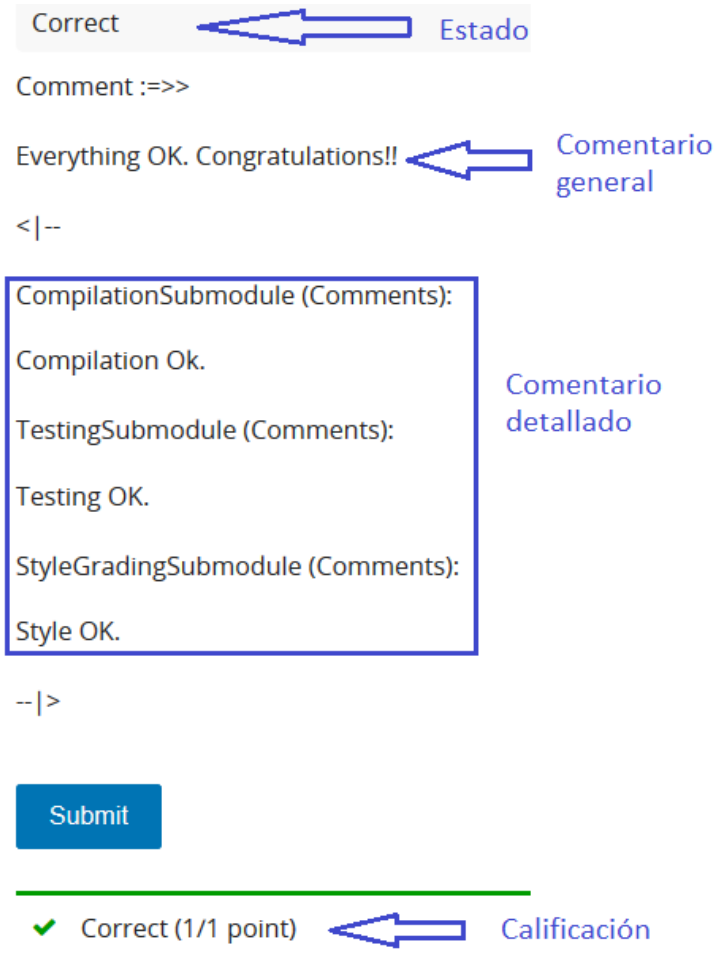


Figura 4.4 Retroalimentación del Calificador 100%.

- **Errores de Estilo**, si el código enviado por el estudiante presenta comentarios o etiquetas faltantes usando JavaDoc [62], se genera una retroalimentación como se muestra en la Figura 4.5. Por lo tanto, se obtiene de retroalimentación: un estado Incorrecto, un comentario general indicando que el código enviado, no pasó todas las métricas de calificación, además en los comentarios a detalle de los submódulos se observa que *CompilationSubmodule* y *TestingSubmodule* están correctos, pero en *StyleGradingSubmodule* le falta una etiqueta en la clase Raices.java en la línea 46, mostrando que le falta agregar el parámetro "*@param*" para la variable "c".

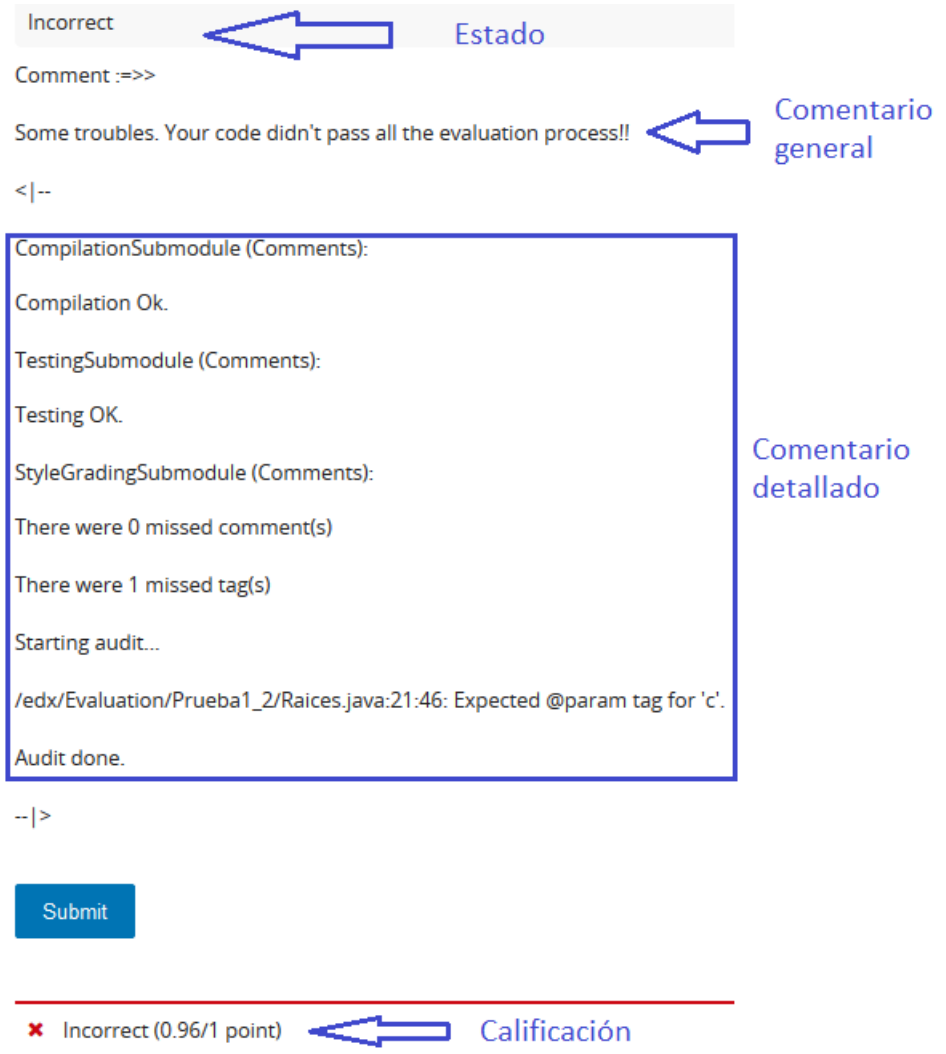


Figura 4.5 Retroalimentación del Calificador con errores de estilo.

- Errores en los casos de prueba, si las funciones editadas por el estudiante no tienen un correcto funcionamiento, se genera un comentario de la función que tiene errores para que sea corregida, esta retroalimentación se presenta en la Figura 4.6. Por lo tanto, se obtiene de retroalimentación: un estado Incorrecto, un comentario general indicando que el código enviado, no pasó todas las métricas de calificación, además en los comentarios a detalle de los submódulos se observa que *CompilationSubmodule* y *StyleGradingSubmodule* están correctos, pero en *TestingSubmodule* se evaluaron tres pruebas o *test*, pero fallo uno mostrando el mensaje “Error en discriminante1”. Este error indica que se evaluó un método o función para calcular el discriminante y estuvo incorrecto, ya que no coincide la respuesta esperada con la que genera dicha función.

Figura 4.6 Retroalimentación del Calificador con errores en los casos de prueba.

En la Tabla 4.4 se presentan los resultados obtenidos en cada uno de los dispositivos utilizados.

Actividad	Computadora Personal	Teléfono Inteligente	Observación
Crear una tarea	✔	✔	Ninguna
Administración y configuración de los submódulos de calificación	✔	✔	Ninguna
Retroalimentación del Calificador	✔	✔	Ninguna

Tabla 4.4 Resumen de pruebas de funcionalidad.

4.4.3 PRUEBAS DE FUNCIONALIDAD Y ACEPTACIÓN DE LOS ESTUDIANTES

Se realizó una corta introducción respecto del uso de la solución a los estudiantes del docente colaborador del proyecto, que cursaban la asignatura de Programación Orientada a Objetos.

Se pidió a los estudiantes que usen la herramienta de Calificación Automática durante un mes, en el cual se probaba la solución y se corregía cualquier inconveniente originado. Finalmente, la solución quedo estable y se procedió a realizar una encuesta a los estudiantes con el objetivo de conocer sus experiencias al usar la solución y así determinar el grado de funcionalidad de la misma, y la aceptación que tuvo entre los estudiantes. Es necesario mencionar que se pidió a los estudiantes que se tome en cuenta únicamente los ejercicios de Herencia y Excepciones, debido a que ellas ya estaban completamente estables.

Porcentaje de la Población Encuestada

Realizaron la encuesta 21 estudiantes que terminaron el semestre Octubre 2016 - Marzo 2017 (2017-A). Teniendo como población de 24 estudiantes, entonces el porcentaje de la población que llenó la encuesta es:

$$\boldsymbol{n} = \frac{21}{24} * 100 = \ 87.5\%$$

Diseño de la encuesta

La encuesta diseñada consta de ocho preguntas: seis de ellas son de opción múltiple, las cuales se componen de cinco alternativas de respuesta, y dos preguntas abiertas, frente a la cual los estudiantes expresaron su opinión. La estructura de la Encuesta realizada en Google Forms se encuentra en el ANEXO 3.

Para determinar la funcionalidad de la solución se tuvieron en cuenta aspectos de importancia como: el funcionamiento de cada una de las herramientas que conforman la solución y el acceso a cada una de ellas.

Respecto a la aceptación de la solución se tuvo en cuenta: la presentación de la interfaz de la solución, tiempos de carga de las páginas, facilidad de uso y las experiencias de los estudiantes al usar la solución.

Tabulación, Presentación y Análisis de resultados

Los resultados para cada una de las preguntas de la encuesta aplicada se muestran por medio de tablas, para que de esta forma sea más clara su presentación y análisis. El resultado obtenido de Google Forms se encuentra en el ANEXO 4 y ANEXO 5.

Cabe mencionar que las preguntas 2 y 3 evalúan el tiempo de carga de la página web, en cambio la pregunta 1 y las preguntas desde la 4 a la 8 servirán para validar las pruebas de aceptación, es decir determinar qué tan fácil y agradable fue trabajar con la solución para los estudiantes, y se presentan a continuación:

- **Pregunta 1**

1. ¿Cómo calificaría la presentación de la interfaz del Calificador Automático?	**Muy mala**	**Mala**	**Regular**	**Buena**	**Muy Buena**
	1	**2**	**3**	**4**	**5**
	2	0	12	5	2

Tabla 4.5 Presentación de la interfaz visual del Calificador Automático.

De acuerdo con la Tabla 4.5 la mayor cantidad de respuestas obtenidas es la opción de regular, entonces se evidencia que la presentación de la interfaz del Calificador Automático fue regular.

- **Pregunta 2**

2. Especifique el tiempo percibido que se tarda la carga de una página dentro del Calificador Automático. (En el Laboratorio de Redes II)	**0-2 segundos**	**2-4 segundos**	**4-6 segundos**	**6-10 segundos**	**más de 10 segundos**
	1	**2**	**3**	**4**	**5**
Opción 1: Página de inicio de sesión (login)	6	5	3	6	1
Opción 2: Evaluación en línea	3	5	6	4	3
Opción 3: Navegación entre páginas	6	7	3	5	0

Tabla 4.6 Tiempo de carga de una página dentro del Calificador Automático en el Laboratorio de Redes II.

Respecto al tiempo que se tarda la carga de una página dentro del Calificador Automático en el Laboratorio de Redes II, como: *login*, evaluación en línea, y navegación entre páginas, se obtuvo que la mayor cantidad respuestas está en un tiempo de espera de 2-4 segundos, como se observa en la Tabla 4.6. Acorde a [63] estos valores se encuentran dentro del rango de tiempo aceptable para carga de páginas.

- **Pregunta 3**

3. Especifique el tiempo percibido que se tarda la carga de una página dentro del Calificador Automático. (Fuera del Laboratorio de Redes II)	**0-2 segundos**	**2-4 segundos**	**4-6 segundos**	**6-10 segundos**	**más de 10 segundos**
	1	**2**	**3**	**4**	**5**
Opción 1: Página de inicio de sesión (*login*)	5	9	2	5	0
Opción 2: Evaluación en línea	1	7	7	3	3
Opción 3: Navegación entre páginas	4	9	4	4	0

Tabla 4.7 Tiempo de carga de una página dentro Calificador Automático fuera del Laboratorio de Redes II.

La Tabla 4.7 muestra que el tiempo que se tarda la carga de una página del Calificador Automático, fuera del Laboratorio de Redes II, como: *login*, evaluación en línea, y navegación entre páginas, se obtuvo se obtuvo que la mayor cantidad respuestas está en un tiempo de espera de 2-4 segundos. Al igual como se mencionó previamente, el tiempo de espera obtenido, se encuentra acorde a los valores aceptables dados en [63]. Se observa que el tiempo de espera tanto dentro como fuera del Laboratorio de Redes II se mantiene, lo que reduce las posibilidades de que el estudiante pierda el interés en acceder a las páginas del Calificador Automático, fuera del campus de la EPN.

- **Pregunta 4**

De acuerdo con la Tabla 4.8, la mayor cantidad de respuestas obtenidas para la afirmación “El uso de la herramienta es fácil”, es la opción algo de acuerdo. Dicho valor evidencia que la mayoría de los estudiantes tiene poca dificultad al usar la herramienta y su uso fue simple.

4. Valore las siguientes afirmaciones del Calificador Automático en una escala del 1 al 5.	**Muy de acuerdo**	**Algo de acuerdo**	**Ni de acuerdo ni en desacuerdo**	**Algo en desacuerdo**	**Muy en desacuerdo**
	1	**2**	**3**	**4**	**5**
Opción 1: El uso de la herramienta es fácil.	4	7	4	3	3
Opción 2: La subida del código respuesta se realizó con facilidad.	3	7	1	7	3
Opción 3: La retroalimentación obtenida permitió mejorar las destrezas de programación.	2	6	6	5	2
Opción 4: La métrica de calificación de casos de prueba mejora las destrezas de programación.	2	7	4	6	2
Opción 5: La métrica de calificación de estilo mejora las destrezas de programación.	2	6	6	5	2
Opción 6: La métrica de calificación de compilación mejora las destrezas de programación.	4	5	3	5	4

Tabla 4.8 Valoración de afirmaciones sobre el Calificador Automático.

Respecto a la afirmación “La subida del código respuesta se realizó con facilidad” se concluye que están ni de acuerdo ni en desacuerdo. Los resultados obtenidos demuestran que la subida del código mostró dificultad, debido a que los estudiantes presentan una resistencia a usar una plantilla para completar el código solución de un determinado problema.

La mayor cantidad de respuestas fue para las opciones: algo de acuerdo y ni de acuerdo ni desacuerdo, respecto con la afirmación “La retroalimentación obtenida permitió mejorar las destrezas de programación.” Los resultados obtenidos evidencian que la retroalimentación dada por la herramienta fue útil y práctica.

La afirmación: “La métrica de calificación de casos de prueba mejora las destrezas de programación” obtuvo la mayor cantidad de respuestas para la opción algo de

acuerdo, este valor demuestra que los estudiantes consideran que la métrica de casos de prueba es necesaria.

La mayor cantidad de respuestas fue para las opciones: algo de acuerdo y ni de acuerdo ni desacuerdo, respecto a la afirmación “La métrica de calificación de estilo mejora las destrezas de programación”. Los resultados obtenidos demuestran que no hay una gran aceptación por parte de los estudiantes de la métrica de calificación de estilo.

La afirmación: “La métrica de calificación de compilación mejora las destrezas de programación” obtuvo la mayor cantidad de respuestas para las opciones algo de acuerdo y ni de acuerdo ni desacuerdo, este valor demuestra que los estudiantes consideran que la métrica de compilación no es tan necesaria.

De acuerdo con el análisis previo, se evidencia que el uso de la solución fue un poco simple, y no presentó dificultad para los estudiantes. Por lo tanto, se puede concluir que el Calificador Automático tuvo un impacto positivo y logro buena aceptación en los estudiantes.

Las pruebas de funcionalidad buscan determinar si la solución trabajaba correctamente. Para dichas pruebas se tomó en cuenta las preguntas 5 y 6 de la encuesta, cuyo análisis se presenta a continuación.

- **Pregunta 5**

5. Valore el funcionamiento de los siguientes procesos del Calificador Automático.	**Nunca funcionó**	**Algunas veces funcionó**	**Frecuentemente funcionó**	**Siempre funcionó**
	1	**2**	**3**	**4**
Opción 1: Evaluación en línea.	1	7	10	3
Opción 2: Despliegue de la descripción del Ejercicio.	0	4	7	10

Tabla 4.9 Valoración de los procesos del Calificador Automático.

De acuerdo con los valores obtenidos en la Tabla 4.9, en las opciones frecuentemente funcionó y siempre funcionó para el despliegue de la descripción del Ejercicio y evaluación en línea del Calificador Automático, se evidencia que dichos procesos siempre funcionaron, por lo tanto, el Calificador Automático realizó los procesos para los que fue previsto sin presentar inconvenientes.

- **Pregunta 6**

6. Considera usted que el uso del Calificador Automático en un semestre completo (desde el inicio hasta el fin del semestre) sería de utilidad para el aprendizaje de Programación Orientada a Objetos.	Muy de acuerdo	Algo de acuerdo	Ni de acuerdo ni en desacuerdo	Algo en desacuerdo	Muy en desacuerdo
	1	2	3	4	5
	3	4	7	4	3

Tabla 4.10 Utilidad del uso del Calificador Automático en un semestre completo.

La Tabla 4.10 muestra que la utilidad de usar el Calificador Automático en un semestre completo obtuvo la mayor cantidad de respuestas para la opción: ni de acuerdo ni desacuerdo, dichos valores evidencian que la mayoría de los estudiantes no están de acuerdo ni en desacuerdo en que el uso de la solución sería de utilidad para el aprendizaje.

- **Pregunta 7**

7.Está satisfecho con las métricas de calificación planteadas?	Si	No
	15	6

Tabla 4.11 Valoración de las métricas de calificación del Calificador Automático.

Respecto a las métricas de calificación planteadas en la solución, la mayoría de los estudiantes opina que están satisfechos, como se muestra en la Tabla 4.11. Estos resultados ratifican a la pregunta 4, sin embargo, aquí también se obtuvo que a los estudiantes no les agrada la métrica de estilo.

- **Pregunta 8**

8. Está satisfecho con la retroalimentación obtenida?	Si	No
	18	3

Tabla 4.12 Valoración de la retroalimentación obtenida del Calificador Automático.

Respecto a la retroalimentación obtenida de la solución, la mayoría de los estudiantes opina que están satisfechos, como se observa en la Figura 4.12.

4.4.4 PRUEBAS DE FUNCIONALIDAD Y ACEPTACIÓN DEL DOCENTE

La encuesta aplicada recogió las opiniones dadas por el docente colaborador del proyecto (la encuesta realizada se muestra en los ANEXO 6). Una vez recogida

dicha información, se realizó un análisis, cuyos resultados se presentan a continuación:

- El diseño y facilidad de uso, así como el tiempo de acceso al Calificador Automático es bueno. Estos resultados corroboran el resultado obtenido por parte de los estudiantes.
- El despliegue de la descripción del ejercicio y la evaluación en línea del Calificador Automático siempre funcionó.
- Las métricas que considera importantes son la de casos de prueba y compilación, en cambio la métrica de estilo la deja a un lado.
- Respecto al uso del Calificador Automático en un semestre completo está muy de acuerdo que sería de utilidad para el aprendizaje de Programación Orientada a Objetos.
- Finalmente, el docente encuestado considera que está satisfecho con las métricas de calificación planteada y la retroalimentación obtenida del Calificador Automático.

CAPÍTULO 5: CONCLUSIONES Y RECOMENDACIONES

En este capítulo se presentarán las conclusiones y las recomendaciones conforme a las experiencias y conocimientos adquiridos en el desarrollo del proyecto.

5.1 CONCLUSIONES

- Se alcanzó con éxito el objetivo principal del presente proyecto que fue implementar un módulo de calificación automática para una plataforma MOOC. Para lo cual, se creó un escenario para probar el funcionamiento del Calificador y se hizo los correctivos necesarios. Estos correctivos se realizaron sobre las clases: *CheckGradingSubmodule*.java, *StyleGradingSubmodule*.java, *TestingGradingSubmodule*.java, y *Orchestator*.java. Luego se diseñaron los procesos para crear una tarea, administrar y configurar el proceso de calificación, y enviar una tarea. Posterior a esto, se implementó la integración del Calificador con Edx, en base a los procesos diseñados. Finalmente, se muestra: la interfaz para administración y configuración del proceso de calificación, y la retroalimentación del Calificador a una tarea.

- Luego de una revisión de las herramientas MOOC usadas actualmente, se seleccionó a Edx como plataforma base para la presente investigación, debido a que posee: una arquitectura flexible, licencia GNU/GPL, fácil acceso a documentación y descarga de código fuente, proporciona un componente llamado *External Grader* y una estructura de curso con temarios.

- El Calificador Automático posee 3 métricas o submódulos de Calificación: *Compilation Grading Submodule*, para verificar que se compile el código en Java, *Testing Grading Submodule*, para ejecutar casos de prueba y *Style Grading Submodule*, para verificar el estilo del código fuente enviado por el estudiante.

- Se identificaron a través de encuestas, que los estudiantes consideran que las métricas de calificación: casos de prueba y de compilación, son importantes y necesarias, pero la métrica de estilo no es de su agrado.

- Se integró el Calificador Automático corregido con la plataforma Edx, mediante el uso de Servidores Web creados en Python, los cuales son: *JavaGrader*.py, el cual recibe peticiones de Edx, las envía al Calificador y reenvía la retroalimentación obtenida; *SubmissionConf*.py, para configurar las métricas de calificación de cada ejercicio; y *Corrector*.py, para ejecutar los casos de prueba sobre el código enviado por el estudiante.

- Se comprobó a través de encuestas, que el Calificador Automático es funcional y fue aceptado por los estudiantes y docente de la asignatura de Programación Orientada a Objetos de la carrera de Electrónica y Redes de Información en el periodo académico 2017-A.

5.2 RECOMENDACIONES

- Dar un mantenimiento preventivo al servidor que aloja al servidor Edx, el cual debe incluir un plan de respaldos.

- Se debe dar seguimiento a los estudiantes de tal forma que tengan una participación en la solución, dándoles una capacitación sobre el uso adecuado para la edición del código solución de los ejercicios planteados y él envió a calificar de los mismos dentro del Calificador Automático, a fin de obtener los resultados esperados.

- Se debe usar de preferencia la versión de Ubuntu 17.04, para sobre esta instalar la plataforma Edx con la versión Ficus.3, debido a que la versión 17.04 de Ubuntu es estable.

- Dadas las recomendaciones de las encuestas realizadas, se obtuvo que se debería permitir la subida de archivos en lugar de editar el código solución.

- Antes de poner a calificar la solución se debe configurar adecuadamente la dirección IP del servidor Edx y ejecutar en segundo plano los Servicios Web creados en Python para escuchar las peticiones de los estudiantes.

- Para la administración y configuración de las métricas o submódulos de calificación, se debe tomar en cuenta que existe una plantilla la cual debe ser modificada por el profesor en base a las necesidades del ejercicio planteado. Se recomienda que, si se cambia el factor de calificación de una métrica, se debe modificar las otras para entre todas sumen 100.

BIBLIOGRAFÍA

[1] J. C. Caiza, «Automatic Grading of Programming Assignments: Proposal and Validation of an Architecture», ESPAÑA/Escuela Técnica Superior de Ingenieros de Telecomunicaciones-Universidad Politécnica de Madrid/2013, 2013.

[2] S. Willman, R. Lindén, E. Kaila, T. Rajala, M.-J. Laakso, y T. Salakoski, «On study habits on an introductory course on programming», *Comput. Sci. Educ.*, vol. 25, n.º 3, pp. 276-291, jul. 2015.

[3] M. Amelung, K. Krieger, y D. Rösner, «E-Assessment as a Service», *IEEE Trans. Learn. Technol.*, vol. 4, n.º 2, pp. 162-174, abr. 2011.

[4] H. Le, «Interactive Computer Science Exercises In edX», 2016.

[5] T. Barrios y M. B. Marín, «Aprendizaje mixto a través de laboratorios virtuales», *Signos Univ.*, 2014.

[6] J. C. Rodríguez-del-Pino, E. Rubio Royo, y Z. Hernández Figueroa, «A Virtual Programming Lab for Moodle with automatic assessment and anti-plagiarism features», 2012.

[7] «Colleges and Universities that Use Moodle», *eClass4learning*, 25-may-2014. [En línea]. Disponible en: http://www.eclass4learning.com/colleges-universities-use-moodle/. [Accedido: 21-jul-2017].

[8] «MITx», *edX*, 12-nov-2013. [En línea]. Disponible en: https://www.edx.org/school/mitx. [Accedido: 21-jul-2017].

[9] «HarvardX», *edX*, 12-nov-2013. [En línea]. Disponible en: https://www.edx.org/school/harvardx. [Accedido: 21-jul-2017].

[10] J. C. Caiza y J. M. del Álamo Ramiro, «Programming assignments automatic grading: review of tools and implementations», en *7th International Technology, Education and Development Conference (INTED2013)*, Valencia, Spain, 2013, pp. 5691-5700.

[11] M. Guerrero, D. S. Guamán, y J. C. Caiza, «Revisión de Herramientas de Apoyo en el Proceso de Enseñanza-Aprendizaje de Programación», *Rev. Politécnica*, vol. 35, n.º 1, p. 84, feb. 2015.

[12] J. C. Almenara, M. del C. L. Cejudo, y J. A. M. Lozano, «Contributions to e-Learning from a Best Practices Study at Andalusian Universities», *Int. J. Educ. Technol. High. Educ.*, vol. 10, n.º 1, pp. 226–239, 2013.

[13] F. J. García-Peñalvo y A. M. S. Pardo, «Una revisión actualizada del concepto de eLearning. Décimo Aniversario/An updated review of the concept of eLearning. Tenth anniversary», *Educ. Knowl. Soc. Salamanca*, vol. 16, n.º 1, pp. 119-144, 2015.

[14] D. Weaver, C. Spratt, y C. S. Nair, «Academic and student use of a learning management system: Implications for quality», *Australas. J. Educ. Technol.*, vol. 24, n.º 1, 2008.

[15] I. Nafría, *Web 2.0: El usuario, el nuevo rey de Internet*. Gestión 2000, 2007.

[16] J. S. Ruiz, H. J. P. Díaz, J. A. Ruipérez-Valiente, P. J. Muñoz-Merino, y C. D. Kloos, «Towards the Development of a Learning Analytics Extension in Open edX», en *Proceedings of the Second International Conference on Technological Ecosystems for Enhancing Multiculturality*, New York, NY, USA, 2014, pp. 299–306.

[17] «Hypertext Transfer Protocol -- HTTP/1.1». [En línea]. Disponible en: http://www.rfc-editor.org/rfc/rfc2616.txt. [Accedido: 26-sep-2017].

[18] D. Raggett, A. Le Hors, I. Jacobs, y others, «HTML 4.01 Specification», *W3C Recomm.*, vol. 24, 1999.
[19] J. Cowan, J. Paoli, F. Yergeau, E. Maler, C. M. Sperberg-McQueen, y T. Bray, «Extensible Markup Language (XML) 1.1», *W3C CR CR-Xml11-20021015*, 2002.
[20] B. N. Director Managing, «What's the Difference Between a MOOC and an LMS? | Your Training Edge ®». [En línea]. Disponible en: http://www.yourtrainingedge.com/whats-the-difference-between-a-mooc-and-an-lms/. [Accedido: 01-jun-2017].
[21] S. L. M. Pedro Pernías Peco, «Los MOOC: orígenes, historia y tipos». [En línea]. Disponible en: http://www.centrocp.com/los-mooc-origenes-historia-y-tipos/. [Accedido: 17-jul-2017].
[22] «About | Coursera». [En línea]. Disponible en: https://about.coursera.org/. [Accedido: 02-jun-2017].
[23] D. C. Schmidt y Z. McCormick, «Producing and delivering a coursera MOOC on pattern-oriented software architecture for concurrent and networked software», 2013, pp. 167-176.
[24] P. Ruiz Martín, «Presente y futuro de los Massive Open Online Courses (MOOC): Análisis de la oferta completa de cursos de las plataformas Coursera, EdX, Miríada X y Udacity.», 2013.
[25] «Course Guides | Udacity». [En línea]. Disponible en: /sitemap/guides. [Accedido: 02-jun-2017].
[26] «Higher Education LMS Features | Canvas Learning Management System», *Instructure*. [En línea]. Disponible en: https://www.canvaslms.com/higher-education/features. [Accedido: 05-jun-2017].
[27] «About Us | Canvas Network». [En línea]. Disponible en: https://info.canvas.net/. [Accedido: 05-jun-2017].
[28] «Qué es una API y qué puede hacer por mi negocio». [En línea]. Disponible en: https://bbvaopen4u.com/es/actualidad/que-es-una-api-y-que-puede-hacer-por-mi-negocio. [Accedido: 15-ago-2017].
[29] «Learning Tools Interoperability | IMS Global Learning Consortium». [En línea]. Disponible en: /activity/learning-tools-interoperability. [Accedido: 06-jun-2017].
[30] «About Open edX | Open edX Portal». [En línea]. Disponible en: https://open.edx.org/about-open-edx. [Accedido: 01-jun-2017].
[31] «EdX Announces New Membership Structure; Expands edx.org», *edX*, 05-mar-2014. [En línea]. Disponible en: https://www.edx.org/press/edx-announces-new-membership-structure. [Accedido: 26-sep-2017].
[32] L. Yuan, S. Powell, J. CETIS, y others, «MOOCs and open education: Implications for higher education», 2013.
[33] L. Yuan y S. Powell, «Partnership Model for Entrepreneurial Innovation in Open Online Learning», 2015.
[34] «Acerca de Moodle - MoodleDocs». [En línea]. Disponible en: https://docs.moodle.org/all/es/Acerca_de_Moodle. [Accedido: 02-jun-2017].
[35] R. Stallman, «The gnu gpl and the american way», *GNU Org*, 2001.
[36] «2. Open edX Architecture — Open edX Developer's Guide documentation». [En línea]. Disponible en: https://edx.readthedocs.io/projects/edx-developer-guide/en/latest/architecture.html. [Accedido: 15-jun-2017].

[37] «CSS Tutorial». [En línea]. Disponible en: https://www.w3schools.com/css/. [Accedido: 26-sep-2017].
[38] G. Richards, S. Lebresne, B. Burg, y J. Vitek, «An analysis of the dynamic behavior of JavaScript programs», en *ACM Sigplan Notices*, 2010, vol. 45, pp. 1–12.
[39] «welcome to Mako!» [En línea]. Disponible en: http://www.makotemplates.org/. [Accedido: 26-sep-2017].
[40] «Welcome to Python.org», *Python.org*. [En línea]. Disponible en: https://www.python.org/. [Accedido: 26-sep-2017].
[41] «The Web framework for perfectionists with deadlines | Django». [En línea]. Disponible en: https://www.djangoproject.com/. [Accedido: 26-sep-2017].
[42] «9.14. External Grader — Building and Running an Open edX Course documentation». [En línea]. Disponible en: http://edx.readthedocs.io/projects/open-edx-building-and-running-a-course/en/latest/exercises_tools/external_graders.html. [Accedido: 02-jun-2017].
[43] «10.1. Problems, Exercises, and Tools — Building and Running an edX Course documentation». [En línea]. Disponible en: http://edx.readthedocs.io/projects/edx-partner-course-staff/en/latest/exercises_tools/create_exercises_and_tools.html. [Accedido: 02-jun-2017].
[44] «JSON». [En línea]. Disponible en: http://www.json.org/json-es.html. [Accedido: 15-ago-2017].
[45] Rafael Navarro Marset, «REST vs Web Services». [En línea]. Disponible en: http://files.distribuyendose.webnode.es/200000004-291172a0ec/RestVsWebServices.pdf. [Accedido: 19-jun-2017].
[46] «World Wide Web Consortium (W3C) - España». [En línea]. Disponible en: http://www.w3c.es/. [Accedido: 26-sep-2017].
[47] *IETF*. [En línea]. Disponible en: http://www.ietf.org/rfc/rfc3986.txt. [Accedido: 19-jun-2017].
[48] C. A. M. Machuca, «Estado del Arte: Servicios Web», *Univ. Nac. Colomb. Tesis Maest.*, 2010.
[49] L. D. Seta, «Introducción a los servicios web RESTful», *Dos Ideas*. [En línea]. Disponible en: https://dosideas.com/noticias/java/314-introduccion-a-los-servicios-web-restful. [Accedido: 19-jun-2017].
[50] W. made this site, «Project Name». [En línea]. Disponible en: http://www.adwe.es/general/colaboraciones/servicios-web-restful-con-http-parte-i-introduccion-y-bases-teoricas. [Accedido: 19-jun-2017].
[51] P. Letelier y P. Letelier, «Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP)», *www.cyta.com.ar/ta0502/v5n2a1.htm*, 15-abr-2006. [En línea]. Disponible en: http://www.cyta.com.ar/ta0502/b_v5n2a1.htm. [Accedido: 26-sep-2017].
[52] J. H. Canós y M. C. P. P. Letelier, «Metodologías ágiles en el desarrollo de software», 2012.
[53] J. Joskowicz, «Reglas y prácticas en eXtreme Programming», *Univ. Vigo*, p. 22, 2008.
[54] P. Mayer, A. Schroeder, y N. Koch, «MDD4SOA: Model-driven service orchestration», en *Enterprise Distributed Object Computing Conference, 2008. EDOC'08. 12th International IEEE*, 2008, pp. 203–212.

[55] «JUnit - About». [En línea]. Disponible en: http://junit.org/junit4/. [Accedido: 11-jul-2017].
[56] «checkstyle – Javadoc Comments». [En línea]. Disponible en: http://checkstyle.sourceforge.net/config_javadoc.html. [Accedido: 11-jul-2017].
[57] M. Dumas y A. H. Ter Hofstede, «UML activity diagrams as a workflow specification language», en *UML*, 2001, vol. 2185, pp. 76–90.
[58] «DarwinPoveda/Evaluation», *GitHub*. [En línea]. Disponible en: https://github.com/DarwinPoveda/Evaluation. [Accedido: 14-jul-2017].
[59] «Python (programming language) ~ Informatika & Komputer Manual Indonesia ~ colenak.ptkpt.net». [En línea]. Disponible en: http://colenak.ptkpt.net/_lain.php?_lain=3721. [Accedido: 07-jul-2017].
[60] «Documentation», *Vagrant by HashiCorp*. [En línea]. Disponible en: https://www.vagrantup.com/docs/index.html. [Accedido: 17-ago-2017].
[61] «VirtualBox – Oracle VM VirtualBox». [En línea]. Disponible en: https://www.virtualbox.org/wiki/VirtualBox. [Accedido: 17-ago-2017].
[62] «How to Write Doc Comments for the Javadoc Tool». [En línea]. Disponible en: http://www.oracle.com/technetwork/articles/java/index-137868.html. [Accedido: 11-jul-2017].
[63] «The Great Web Slowdown [INFOGRAPHIC]», *Radware Blog*, 25-feb-2014. [En línea]. Disponible en: https://blog.radware.com/applicationdelivery/wpo/2014/02/the-great-web-slowdown-infographic/. [Accedido: 17-ago-2017].
[64] «3.4.1. Installing Open edX Fullstack — Installing, Configuring, and Running the Open edX Platform documentation». [En línea]. Disponible en: http://edx.readthedocs.io/projects/edx-installing-configuring-and-running/en/latest/installation/fullstack/install_fullstack.html#installing-open-edx-fullstack. [Accedido: 03-jul-2017].

ANEXOS

ANEXO 1: TRABAJOS RELACIONADOS A PARTIR DEL 2010.

ANEXO 2. INSTALACIÓN DE EDX (VERSIÓN FICUS.3)[50]

Se puede instalar la versión Ficus.3 de EDX ficus de dos maneras: DevStack o Fullstack. Para el presente proyecto se usó la instalación Fullstack.

Prerrequisitos para la instalación

DevStack o Fullstack para ser instalados requieren el siguiente software:

- VirtualBox[14] 4.3.12 o posterior.
- Vagrant[15] 1.6.5 o posterior.
- Un cliente NFS (Network File System), si su sistema operativo no incluye uno. NFS permite que las máquinas virtuales DevStack y Fullstack compartan un sistema de archivos común con otros equipos.

Para la instalación se Fullstack se siguieron los siguientes pasos:

1. Se creó el directorio fullstack y se accedió hasta él desde el terminal.
   mkdir fullstack
   cd fullstack
2. Se definió la variable de entorno OPENEDX_RELEASE.
   export OPENEDX_RELEASE="open-release/ficus.3"
3. Se descargó el script de instalación.
   curl -OL https://raw.github.com/edx/configuration/$OPENEDX_RELEASE/util/install/install_stack.sh
4. Se ejecutó un conjunto de comandos para crear e iniciar la máquina virtual fullstack.
   bash install_stack.sh fullstack

Al final de este paso ya está creada el aula virtual, para comprobar la correcta instalación debe abrir en un buscador la plataforma Edx. En el buscador puede acceder de dos maneras: por la dirección ip 192.168.33.10 o por el alias *preview.localhost*.

[14] https://www.virtualbox.org/wiki/Downloads
[15] https://www.vagrantup.com/downloads.html

Esta versión posee unas cuentas por defecto precargadas, las cuales se muestran en la Tabla 1.

Cuenta	Descripción
staff@example.com	Esta es una cuenta de usuario para ser el administrador de un curso.
verified@example.com	Esta es una cuenta de estudiante, para verificar certificados.
audit@example.com	Esta es una cuenta de estudiante que se puede utilizar para acceder el LMS para probar un curso.
honor@example.com	Esta es una cuenta de estudiante que se puede utilizar para acceder el LMS para probar los certificados de código de honor.

Tabla 1. Cuentas por defectos de Edx

La contraseña de todas estas cuentas es “edx”.

ANEXO 3. ENCUESTA PARA LOS ESTUDIANTES

ANEXO 4. RESULTADOS DE LAS ENCUESTAS A LOS ESTUDIANTES

ANEXO 5. RESULTADO DE LA ENCUESTA DE LOS ESTUDIANTES EN EXCEL (CD ADJUNTO)

ANEXO 6. RESULTADO DE LA ENCUESTA DEL PROFESOR