DESARROLLO DE UNA APLICACIÓN WEB PARA LA AUTOMATIZACIÓN DE ENTREGA Y CORRECCIÓN DE TRABAJOS DE PROGRAMACIÓN

Proyecto de Desarrollo

ÓSCAR AUGUSTO CHAMAT

200745403

chamatoscar@gmail.com

Ángel García Baños, Ph.D

angel.garcia@correounivalle.edu.co

Facultad de Ingeniería

Escuela de Ingeniería de Sistemas y Computación

Programa Académico de Ingeniería de Sistemas

Santiago de Cali, Junio 11 de 2014

Tabla de Contenido

Índice de cuadros

Índice de figuras

Índice de algoritmos

Resumen

Diferentes herramientas de software son utilizadas actualmente para ayudar al proceso educativo como por ejemplo; Scratch para aprender conceptos de programación, Calcwav para ayudar a personas discapacitadas o incluso la realización de exámenes por medios virtuales. Todas éstas ayudan tanto al profesor en cuanto al tiempo de verificación y entrega de resultados como al estudiante en el aumento de la velocidad de aprendizaje.

Haciendo énfasis en la enseñanza de la programación son notorios algunos problemas que se dan en el momento de la calificación de pruebas prácticas ya que la misma se realiza "a mano" y esto junto con la cantidad de estudiantes de un curso (generalmente grande) crea problemas que van desde la demora en la entrega de los resultados de las tareas hasta calificaciones erróneas o vacías para algunos estudiantes.

La implementación de una herramienta de software que realice la calificación automática de las pruebas de programación permitirá una calificación más rápida y mas flexible de las mismas incluso los estudiantes podrían conocer al momento de una entrega los resultados frente a un conjunto (completo o una parte) de las pruebas esperadas para los ejercicios. Por otro lado al profesor se le pueden facilitar calificaciones de errores sintácticos o problemas del estilo de programación para una valoración personal de cada nota.

Abstract

Different software tools are currently used to help the educational process such as; Scratch to learn programming concepts, Calcwav to help people with disabilities or even conducting academical tests by virtual means. All these help both the teacher reducing the time of verification and delivery of notes and the student increasing the speed of learning.

In the matter of the teaching of programming some problems are notorious at the time of the practical tests rating since this process is done "by hand" and this together with the number of students in a course (usually large) creates problems ranging from the late delivery of the results of tasks to the delivery of wrong or empty scores for some students.

The implementation of a software tool to perform automatic grading of programing tests will allow this process to be faster and more flexible even the students might know at the time of delivery the performance of their solution against a set (full or part) of the tests exercises. In addition qualifications for syntax errors or problems in the programming style can be provided to the teacher for a personal assessment of each note.

1. Introducción

1.1. Objetivo General

Desarrollar una aplicación Web que permita la calificación automática de trabajos de programación.

1.2. Objetivos Específicos

- Buscar una aplicación o escribir el código necesario que cumpla las siguientes condiciones y tenga los siguientes módulos:
  - Integración con el software manejo educativo moodle 2.5.
  - Una interfaz que permita a un docente crear trabajos de programación.
  - Un modulo que permita la subida de archivos a la aplicación, para su calificación, configurable para indicar en qué trabajo de programación será calificado.
  - Un sistema que muestre los errores especificados por el compilador del código fuente.
  - Un sistema que permita la creación de pruebas por medio de entradas y salidas por archivos de texto.
- Elaborar un sistema de evaluación de la documentación de código fuente que permita verificar si los archivos mencionados cumplen con un formato configurable en la aplicación.
- Crear un sistema de evaluación de estilos de programación a través de la comparación de los archivos con una versión de los mismos pero formateada con ayuda de un code beautifier.
- Modificar la interfaz de creación de tareas para poder especificar que tipos de estilos de programación y de documentación son calificables para cada tarea.
- Modificar la interfaz de creación de tareas para que la misma acepte calificación de proyectos de archivos varios para los lenguajes java 1.6, scheme 5.0 o C++ 4.1.2.
- Realizar una tabla estadística con los resultados generados en cada una de las pruebas para un estudiante con un trabajo de programación.
- Se desarrollará la aplicación bajo la metodología de desarrollo conocida como XP (Extreme Programing).

1.3. Resultados Esperados

Objetivo específico	Resultado Esperado	Sección
1. Buscar una aplicación o escribir el código necesario que cumpla las condiciones mencionadas.	Un sistema que mediante la integración con el software educativo moodle 2.5 resuelva los siguientes módulos de la aplicación: 1. Una interfaz que permita a un docente crear trabajos de programación. 2. Un modulo que permita la subida de archivos a la aplicación, para su calificación en un trabajo de programación especifico. 3. Un modulo que permita la calificación de los trabajos de programación teniendo entradas y salidas esperadas utilizando entrada y salida estándar. 4. Un sistema que muestre los errores dados por el compilador del código fuente.	2.2.1, 5.2.2, 6.1
2. Elaborar un sistema de evaluación de la documentación de código fuente, que permita verificar si los archivos mencionados cumplen con un formato especificado en la aplicación.	1. Especificación del formato con el cual se comparará la documentación de los archivos de código fuente para los lenguajes: java 1.6, scheme 5.0 o C++ 4.1.2. 2. Sistema de comparación de la documentación de los archivos de código fuente contra el formato especificado y configurable para su correspondiente lenguaje. 3. Documentación de código que permita la reutilización de el modulo(métodos y variables públicos).	2.2.3, 6.1.2

Objetivo específico	Resultado Esperado	Sección
3. Crear un sistema de evaluación de estilos de programación a través de la comparación de los archivos con una versión de los mismos pero formateada con ayuda de un code beautifier.	1. Especificación de los dos formatos más populares de programación con los cuales se comparará el estilo de los archivos de código fuente para los lenguajes: java 1.6, scheme 5.0 o C++ 4.1.2 . 2. Sistema de comparación de los estilos de los archivos de código fuente contra el formato especificado para su correspondiente lenguaje. 3. Documentación de código que permita la reutilización de el modulo(métodos y variables públicos).	2.2.4, 6.1.1
4. Modificar la interfaz de creación de tareas para poder especificar que tipos de estilos de programación y de documentación son calificables para cada tarea.	1. Modificación del sistema de creación de trabajos de programación que permita al mismo la funcionalidades de: para un trabajo de programación con su lenguaje estipulado entre java 1.6, scheme 5.0 o C++ 4.1.2 se pueda escoger un estilo de programación y un tipo de documentación que pueda ser aplicado a el lenguaje en particular en un trabajo de programación particular. 2. Manual de uso del modulo donde se especifica la creación de un trabajo de programación especifico bajo un lenguaje de programación mostrando la selección de un estilo de documentación y uno de programación para la calificación de ese trabajo y además mostrando como configurar qué ejercicios va a tener el trabajo. 3. Documentación de código que permita la reutilización de el modulo(métodos y variables públicos).	6.1.1, 6.1.2, 2.2.5

Objetivo específico	Resultado Esperado	Sección
5. Modificar la interfaz de creación de tareas para que la misma acepte calificación de proyectos de archivos varios para los lenguajes java 1.6, scheme 5.0 o C++ 4.1.2.	1. Modificación del sistema de creación de trabajos de programación para que permita las funcionalidades de: para un lenguaje estipulado entre java 1.6, scheme 5.0 o C++ 4.1.2 se pueda generar un trabajo de programación que al enviar una solución se pueda en forma de varios archivos(como por ejemplo diferentes clases en diferentes archivos en el caso de Java). 2. Manual de uso del modulo donde se especifica la creación de un trabajo de programación especifico bajo un lenguaje de programación mostrando para la calificación de ese trabajo y además mostrando como configurar qué ejercicios va a tener el trabajo. 3. Documentación de código que permita la reutilización de el modulo(métodos y variables públicos).	5.2.1, 2.2.2, 6.3

Objetivo específico	Resultado Esperado	Sección
6. Realizar una tabla estadística con los resultados generados en cada una de las pruebas que permita a un usuario autorizado visualizar y cambiar la nota de cada uno de los apartados anteriormente mencionados (documentación, estilo, errores de código fuente y errores en los resultados) para generar una nota final. Y que a un usuario no autorizado para generar notas le permita visualizar la nota obtenida.	1. Módulo que usando los módulos desarrollados para la calificación de: la documentación, el estilo, los errores del código fuente y los resultados de las pruebas unitarias genere los datos necesarios para llenar la tabla descrita. 2. Interfaz gráfica que revele un informe de los resultados de las pruebas. 3. Manual de uso del módulo donde se muestre el proceso de generación de una nota para un estudiante. 4. Manual de uso del módulo donde se muestre el proceso de consulta de una nota para un estudiante en un trabajo de programación especifico. 5. Documentación de código que permita la reutilización del módulo (métodos y variables públicos).	2.2.6, 6.2
7. Se desarrollará la aplicación bajo la metodología de desarrollo conocida como XP (Extreme Programing).	1. Artefactos entregables en la metodología de desarrollo XP[31, 32]; Diagramas de Clases, Historias de Usuario y Casos de Uso.	5.1

Cuadro 1: Resultados Esperados

2. Desarrollo Del Trabajo

2.1. Antecedentes o Estado del Arte

A continuación se presentan algunas aplicaciones que implementan algunas de las funcionalidades esperadas para este proyecto de grado o que son importantes de un punto teórico para el mismo.

2.1.1. Jueces Online

Un juez online es un sistema que permite probar programas de los concursos de programación. Muchos de estos jueces online pueden realizar sus propios concursos.

El sistema de estos jueces puede compilar, ejecutar y probar códigos con datos pre-construidos. Estas son unas de las características que se espera del proyecto a implementar pero en general estos jueces solo tienen en cuenta tres cosas: una respuesta correcta a las entradas planteadas, el tiempo y la memoria utilizados esto deja fuera algunas de las características importantes de este proyecto entre las cuales están:

- No tienen en cuenta un sentido estricto ni de documentación ni de legibilidad con lo que respecta a el código fuente.

- Los envíos en estos jueces solo pueden estar compuestos de un archivo de texto plano esto puede llegar a complicar la organización.

- En cuanto a la copia de código fuente, aunque hay algunas normas sobre esta práctica, en general los jueces online no la castigan.

- Para finalizar los jueces online no manejan una calificación máxima por ejercicio por lo que lo único que presentan es un ranking de las personas que tienen envíos a estos ejercicios organizados según los criterios de evaluación ya mencionados y que en caso se empate utilizan la fecha de envío del ejercicio como comparación.

Sin embargo estos jueces tienen algunas ventajas a tener en cuenta:

- El manejo de los problemas presentados en tiempo de compilación es bastante sencillo: básicamente consiste en mostrar la salida que da el compilador del lenguaje al usuario pero teniendo en cuenta la supresión de los warnings esto es una buena idea ya que procesar la salida de un compilador además de ser complicado no genera un aporte importante de información comparada con la información que da el propio compilador.

- Permiten una visualización gráfica del desempeño del usuario a lo largo del tiempo que lleve registrado el usuario al juez lo que para el caso del proyecto a desarrollar podría ser interesante si se aplica a un trabajo de programación en particular.

Entre algunos ejemplos importantes están:

Uva online judge[12]

Es un juez online muy conocido ya que es el juez oficial de la universidad de Valladolid y tiene el apoyo directo de la Competición Internacional Universitaria ACM de Programación(ACM-ICPC) que es una competición anual de programación y algorítmica entre universidades de todo el mundo patrocinada por IBM y una de los más importantes eventos en lo que respecta a la programación competitiva.

El funcionamiento de este juez es bastante estándar según lo anteriormente descrito.

USACO[13]

Es el sitio online que hace parte del programa de entrenamiento para la United States of America Computing Olympiad y que junto con el campamento de verano de la misma organización sirve como proceso de elección y de entrenamiento para el equipo que representa a Estados Unidos en la Olimpiada Internacional De Informática (IOI).

La particularidad de este juez es que sus ejercicios se encuentran organizados por temas pero no todos los temas se encuentran disponibles desde el principio y sólo se liberan cuando se ha logrado realizar todos los ejercicios de los temas anteriores además cuenta con lecturas de teoría y explicaciones sobre los temas y las técnicas utilizadas en los ejercicios del entrenamiento.

Topcoder[14]

Este sitio se conoce como el juez en donde compiten los mejores programadores en lo que respecta a programación competitiva. Se basa completamente en competencias online que cuentan con premios en dinero para algunos de sus participaciones, esto las hace muy atractivas y accesibles a muchos programadores de muchos países. Muchos de los concursos organizados para este juez tienen patrocinadores muy importantes en el mundo tecnológico como por ejemplo: Nasa, IBM o Google que aportan a las competencias no solo con premios en dinero sino que además utilizan el mismo portal de Topcoder para encontrar y contratar nuevos talentos.

El sitio cuenta también con foros sobre las diferentes competencias y muchos otros temas entre los cuales hay algunos tutoriales sobre las técnicas utilizadas en las competencias.

El funcionamiento del juez es básicamente el estándar su principal diferencia reside en dos cosas:

1. Hay un sistema de puntos que funcionan bajo los criterios ya explicados: memoria, velocidad, y tiempo de envío pero que tiene como particularidad que para los ejercicios la cantidad de puntos posibles tiene que ver con su dificultad siendo los mas fáciles lo que menos puntos aportan.

2. Cuando algún otro usuario tiene un envío a un ejercicio se puede crear una prueba para el envío en particular de este ejercicio que si genera una respuesta incorrecta genera un aumento en los puntos propios y una reducción en los puntos del usuario cuyo envío fallo, esto se conoce como desafió

La creación de unos cuantos tutoriales o el enlace a ellos podría ser interesante para la aplicación de este proyecto aunque no sea necesaria para la primer versión.

Codeforces[15]

Este juez esta organizado como una red social. Se pueden ver las participaciones de los amigos en los diferentes concursos organizados aquí así como su estatus. El resto de su funcionamiento es muy parecido a el de TopCoder: tiene un sistema de puntos, los ejercicios se organizan por dificultad, se pueden desafiar envíos, etc.

Sphere Online Judge (spoj)[16]

Es un juez bastante normal pero sirve para organizar concursos de forma muy simple con los problemas que este juez cuenta: se puede restringir el acceso a estos concursos para ciertos usuarios además de otras opciones de configuración. La forma y la facilidad en como estos concursos son organizados es una de las funcionalidades que se esperan implementar en el sistema.

Livearchive[17]

Básicamente tiene el mismo formato y acceso de uva.online pero es bastante importante ya que solo tiene problemas que han aparecido en la acm-icpc en las competencias regionales y mundiales en lo que a lo demás concierne ese es su único aporte.

Otros

Dado que la competencia internacional Universitaria ACM-ICPC es un evento muy conocido y que las competencias de programación cada vez se están haciendo mas populares hay muchos otros jueces online que pertenecen a diferentes países o a diferentes grupos de entrenamiento algunos ejemplos son:

Programming Challenges [18]

The Caribbean Online Judge (COJ) [19]

Pekin University online Judge for ICPC [20]

Timus Online Judge [21]

Teddy Online Judge [22]

2.1.2. Aplicaciones

Además de los jueces online hay algunas aplicaciones que permiten el acceso a algunas de las funcionalidades de los jueces online de forma más extensible y configurable tanto así que se puede incluso añadir lenguajes diferentes para el envió de los ejercicios, entre algunos ejemplos se encuentran:

- **PC2[23]**: Es una pequeña aplicación desarrollada en java que sirve para organizar concursos de programación en una red lan. La dificultad que presenta esta aplicación es que debe ser instalada en cada uno de los computadores que van a hacer parte de la competencia sean o no jueces.
- **DOM**[24]: Esta es una aplicación desarrollada en PHP que permite implementar un juez online en cualquier computador con el que se pueden organizar concursos, crear usuarios, configurar diferentes lenguajes entre otros. Ya que estas aplicaciones son desarrolladas con la esperanza de imitar un juez online cuentan con sus mismos defectos y ventajas y aunque son altamente configurables todo su funcionamiento está directamente pensado para que funcione como cualquier juez mencionado lo que hace muy difícil y poco elegante su extensión a otra forma de funcionamiento.

2.1.3. Aplicaciones Para Cursos

Más profundamente se encuentran algunas aplicaciones que han sido pensadas para la calificación de trabajos de programación en un curso de programación pero aunque estas se amolden a un estilo de notas académicas no tienen en cuenta ni calificaciones en estilo de programación ni de documentación. Primero mencionamos algunas aplicaciones pensadas para un entorno educativo que utilizan como base para su funcionalidad a DOM Judge (discutido anteriormente) este soluciona algunas de las partes complicadas de la creación de aplicaciones de este tipo como por ejemplo la automatización de la compilación y ejecución de los códigos o la comparación de sus salidas con las respuestas esperadas, algunos ejemplos son:

- **Schemeassessment[25]**: Esta aplicación es bastante interesante ya que se encarga de la tarea no trivial de calificar trabajos de programación escritos en Scheme para esto cuenta con algunos códigos de ayuda que funcionan como comparadores de cadenas para igualar las respuestas de los algoritmos con las respuestas esperadas; según su documentación al lidiar con el lenguaje lo trata como si fuera un lenguaje compilado usando como herramienta a MzScheme[7](que ahora es nombrado Racket[8]) todo esto para que sea compatible con DOM Judge.

- **Onlinejudge o Epaile[10]:** Es un proyecto muy bien desarrollado ya que fue parte de un Verano de código de Google(Google Summer Code). Además de apoyarse en DOM Judge esta escrito como un plugin de Moodle; es fácil de instalar en Moodle 1.X pero no en moodle 2.X, viene configurado para poder manejar trabajos de programación en c, c++, java, haskell, pascal.

También hay algunas aplicaciones que funcionan como plugin en el sistema de manejo de cursos educativos Moodle lo que les permite una fácil organización de los usuarios y de los diferentes cursos así como también un acceso más simple a los resultados algo que funcionaria muy bien para la aplicación que se esta construyendo, entre ellos están:

- **Javabrat[9]**: Esta desarrollado para calificar trabajos de programación en los lenguajes Java y Scala no presenta innovaciones en lo que respecta a la funcionalidad de las aplicaciones discutidas anteriormente y cuenta con su misma filosofía; dar retroalimentación instantánea a los usuarios que envían sus soluciones lo que aunque no es un concepto revolucionario en el ambiente de jueces en linea y de la programación competitiva si lo es en un ambiente educativo.
- **Moodle Online Judge u Online Judge 2[26]**: Es la evolución de la aplicación epaile (mucho de su implementación se encuentra basado en la anterior), funciona básicamente igual a Epaile pero no se apoya en DOM Judge más bien utiliza un servicio web que sirve para compilar y ejecutar programas -Ideone[33]- lo que le facilita trabajar con más de 40 idiomas diferentes(java, c, c++, pascal, scheme, etc.) esto bajo algunas restricciones de cuantos ejercicios se pueden calificar al mes o que los envíos pueden tener un solo archivo. Se presenta como una gran aplicación para explorar ya que esta es compatible con Moodle 2.X y se encarga de muchos de los problemas que tiene la implementación del sistema objetivo.

Para terminar hay algunas aplicaciones que no dependen ni de DOM Judge ni de Moodle; en su mayoría son herramientas desactualizadas o proyectos muertos pero no por eso dejan de presentar un acercamiento interesante hacia el cumplimiento del objetivo general de este proyecto, algunos ejemplos:

- **Web-CAT[2]**: Soporta scripts completamente adaptables y generación de pruebas automáticas para cualquier trabajo de programación. Web-CAT además de los modelos tradicionales de calificación automatizados es compatible con la calificación de tareas donde los estudiantes generan sus propias pruebas. Ayuda a fomentar el desarrollo basado en pruebas (también denominado "Test First Codification"), web-CAT permite a un estudiante presentar sus casos de prueba, junto con la solución. El sistema

intenta calificar la validez de las prueba y la exhaustividad de prueba así como la correctitud del código.

- **AutoGrader[27]**: Solo soporta programas en Java pero además soporta análisis estático de código usando PMD[3, 4] esto para detectar posibles bugs, código muerto o código subóptimo o sobrecomplicado. Esta escrito completamente en Java.

- **Curator[1]**: Es uno de los primeros sistemas online de calificación automática (junio 2000) la calificación no era directamente instantánea en el sistema sino que los resultados eran enviados vía e-mail.

- **Praktomat[28]**: Está creado con el propósito de mejorar la calidad del código de los estudiantes. Se concentra en las revisiones de pares(peer review), permite a los demás usuarios comentar sobre el código escrito por otros estudiantes - una característica interesante para agregar en trabajo posterior a este proyecto-. Los estudiantes pueden enviar su código la cantidad de veces que quieran antes de la fecha de entrega limite así pueden cambiar su código teniendo en cuenta cosas que aprendieran del código de los demás o de los comentarios sobre su propio código. Praktomat está desarrollado en Phyton y su filosofía de pruebas consiste en un conjunto de publico de las mismas para probar la funcionalidad básica y el entendimiento del problema y un conjunto privado de pruebas que generalmente tienen en cuenta casos complicados de tratar con algoritmos comunes o que sobrepasan la memoria o el tiempo de algoritmos no tan óptimos.

- **Tamarin[29]**: Tamarin es un sistema de calificación en línea simple para uso en cursos introductorios de programación. Es modular: para cada trabajo de programación se puede configurar un calificador de este modo se puede adaptar para manejar cualquier tipo de archivo enviado o lenguaje de programación. Actualmente Tamarin ofrece principalmente soporte para trabajos en Java. El núcleo de la librería "Java grader" proporciona herramientas que permiten que un envió sea calificado fácilmente de forma segura usando un sandbox.

- **Kattis[5]**: Kattis está escrito en una combinación de Python, PHP y SQL y se ejecuta en Solaris. Tiene una solución de seguridad diseñada especialmente para evitar copia. Todo el código enviado a la aplicación se almacena para su revisiones posteriores si son necesarias.El concepto educativo de Kattis es que los estudiantes sean capaces de presentar su código y obtener rápidamente la respuesta sabiendo si su código funciona. También Katty ayuda a los estudiantes en la práctica de la búsqueda de errores en el código. Está diseñado para calificar c, c++ y java.

2.1.4. Indentadores

En el desarrollo de este trabajo de grado no solo se usan conceptos derivados de los jueces online sino que también se añade la funcionalidad de la calificación de la indentación o la "presentación" de un código entregado con este propósito se sacan a colación los code-beautifiers o indentadores.

La funcionalidad de los code-beautifiers es básicamente la de hacer el código más legible insertando espacios, tabuladores y caracteres de retorno en el código generalmente respetando algunas reglas, algunas de las aplicaciones investigadas son:

- **Pretty Diff[34]:** Pretty Diff es una página web que permite la indentación online de código fuente, no especifica cuales lenguajes es capaz de indentar, no cuenta con un servicio web al cual se le puedan hacer llamados y las posibilidades de configurar el estilo por el cual se quiere indentar son pobres. Su código fuente es descargable[35] pero no es interesante por las pocas posibilidades de configuración que permite.
- **Gist Github[36]:** Es una página web para indentación online de código fuente, cuenta con más de 50 lenguajes diferentes para indentar pero las posibilidades de configurar el estilo por el cual se quiere indentar son solo dos; indentar con espacios o tabuladores y el tamaño del tabulador pero no permite definir un estilo de indentación como tal.
- **Pretty Printer[37]:** De nuevo una página web para indentación online de código fuente, cuenta con los lenguajes PHP, Java, C++, C, Perl, JavaScript, CSS para indentar y las posibilidades de configurar el estilo por el cual se quiere indentar son alrededor de 10 lo que permitiría definir un estilo de programación aunque no cuenta con estilos predeterminados como tal (gnu, Linux,etc.).
- **PHP Formatter[38]:** Otra página web para indentación online de código fuente, no especifica que lenguajes es capaz de indentar pero esta dirigido a indentar código php, las posibilidades de configurar el estilo por el cual se quiere indentar van más allá de las básicas (espacios de separación entre los operadores, o entre los paréntesis, etc) incluso pudiendo utilizar estilos predeterminados; PEAR, k&r, allman, gnu, etc.
- **Jsbeautifier[38]:** Otra página web para indentación online de código fuente, es capaz de indentar javascript y html, las posibilidades de configurar el estilo por el cual se quiere indentar son alrededor de 4 y cuenta con la posibilidad de descargarlo como plugin para vim para la ejecución local pero no cuenta con los lenguajes necesarios para ser interesante para la aplicación desarrollada.

- **Artistic Style[39]:** Artistic style es una aplicación de licencia gnu instalable para indentación de código fuente, es capaz de indentar C, C++, C++/CLI, C# y Java, las posibilidades de configurar el estilo para indentar son muy variadas y su principal ejecución es desde la linea de comandos esto presenta facilidad en lo que respecta a su ejecución desde un lenguaje de programación pero no cuenta con lenguajes tipo Lisp y su extensión de funcionalidad para incorporarlos es complicada.

- **Beautify PHP[40]:** Es una aplicación de licencia gnu instalable para indentación de código fuente, es capaz de indentar código en PHP las posibilidades de configurar el estilo para indentar son pocas pero tiene dos estilos de utilización de las llaves; PEAR y C-style.

- **Otros indentadores de php**: dws-format-filter[41], PhpBeautifier[42] son todos proyectos desarrollados en php que tienen la misma funcionalidad y comportamiento que Beautify PHP.

- **Universal Indent GUI[43]:** Es una aplicación de licencia gnu instalable que permite probar otras aplicaciones para indentación como Artistic Style de forma simple y gráfica.

- **Vim[44]:** Vim es un editor de texto de licencia gnu instalable que tiene la posibilidad de indentación automática de código fuente, es capaz de indentar C, C++, Java, Python C#, Lisp, y muchos otros lenguajes. Las posibilidades de configurar el estilo son muy variadas pero su ejecución desde la linea de comandos o haciendo un llamado desde un lenguaje de programación no esta adecuadamente documentada lo que complica su ejecución.

- **Eclipse**[45]**:** Eclipse es un editor de código fuente desarrollado en Java del cual se pueden usar sus clases internas para indentar códigos de diferentes lenguajes[46] entre los cuales no se incluyen los derivados de Lisp entonces su desarrollo aunque posible sería bastante complicado.

- **Emacs-Batch[47, 48]:** Emacs es un editor de texto de licencia gnu instalable que tiene la posibilidad de indentación automática de código fuente, es capaz de indentar C, C++, Java, Python C#, Lisp, y muchos otros lenguajes, las posibilidades de configurar el estilo son muy variadas además cuenta con muchos estilos de indentación construidos dentro de la aplicación (gnu , k&r - estilo Kernighan y Ritchie para el código C-, bsd, whitesmith, stroustrup ,ellemtel , Java - El estilo para indentar Java - ,awk) y la posibilidad de extensión mediante scripts. La ejecución desde la linea de comandos desde un lenguaje de programación es poco común pero se encuentran suficientes ejemplos para su aplicación.

- **Indent[69]:** Indent gnu es una aplicación de licencia gnu instalable para indentación de código fuente, es capaz de indentar C, C++, y Java, las posibilidades de configurar el estilo para indentar son muy variadas y su

principal ejecución es desde la linea de comandos esto presenta facilidad en lo que respecta a su ejecución desde un lenguaje de programación pero no cuenta con lenguajes tipo Lisp. Esta aplicación lleva es de las únicas que lleva el concepto de indentación hasta la aplicación del pretty print (incluir nuevas lineas además de tabuladores y espacios) por lo que ha sido utilizado en este trabajo basándose en la extensión de Emacs-Batch.

- **pretty-print Racket[70]:** pretty-print del lenguaje de programación Racket es más bien una función que una aplicación pero desde mi experiencia propia es la única capaz de aplicar una buena indentación incluyendo la introducción correcta de nuevas lineas en Racket Lisp. Debido a que es una función se creo un script propio para su aplicación sobre el código fuente.

2.1.5. Generadores De Documentación De Código

Al enfrentar la necesidad de dar una calificación a que tan bien documentado se encuentra un archivo de código fuente se investigo primero las aplicaciones obvias; los generadores automáticos de documentación que toman las etiquetas y generan la documentación en formato HTML, entre algunos de ellos:

- **Javadoc[49]:** Javadoc es una herramienta para la generación de documentación de la API en formato HTML a partir de los comentarios de los documentos en el código fuente, su estilo de documentación fue adoptado por Doxygen y es el más conocido en lo que respecta a desarrollo de software. La licencia del código fuente de java expresa que este solo puede ser usado para referencias[50] entonces utilizar el mismo para tratar de calificar documentación no es practico.
- **Doxygen[51]:** Doxygen es la herramienta estándar para la generación de documentación de C++ con anotaciones, pero también es compatible con otros lenguajes de programación como C, Objective-C, C #, PHP, Java, Python, IDL (Corba, Microsoft, y los sabores UNO / OpenOffice), Fortran, VHDL, Tcl, y hasta cierto punto D. La variedad de lenguajes sobre los que permite generar la documentación hace que el estilo de anotaciones para la documentación sea bastante popular y casi un estándar en el desarrollo de software. El código fuente de la aplicación como tal es abierto y podría ser modificado para la calificación de la documentación pero debido a que esta desarrollado en C++ su modificación en lo que respecta a la corrección de errores sería complicado.
- **Smartcomments[52]:** Básicamente tiene la misma funcionalidad que Doxygen con el mismo estilo de anotaciones solo que aplicable para javascript. Su código fuente es libre pero su funcionalidad limitada hace que no sea interesante para desarrollar.

- **Natural Docs[53]:** "Natural Docs es un generador de documentación de código abierto para múltiples lenguajes de programación[54]". Presenta una opción al estilo de documentación de Javadoc que es definido en la página web de la aplicación como una sintaxis natural que se lee como ingles simple en otras palabras tiene algunas palabras claves en ingles pero no utiliza símbolos como "@"; tratar de extender el código fuente del mismo para calificar documentación podría ser una buena opción ya que su código es libre pero dado que esta escrito en C# aumentaría la complejidad del proyecto al incluir otro lenguaje a la cantidad ya variada de lenguajes que se utilizan.

2.1.6. Generadores De Arboles Sintácticos De Código Fuente

Cuando se investigaron a profundidad los generadores de documentación automáticos se hizo evidente que era necesario una funcionalidad más potente que la de los anteriores por lo cual se investigaron parsers o scanners de código fuente como:

- **JavaCC [55]:** JavaCC es generador de parsers o scanners de código fuente que cuando se le provee la gramática del lenguaje sirve para generar el árbol sintáctico de códigos de diferentes lenguajes, la definición de la gramática es un asunto complejo que requiere un muy buen conocimiento y documentación sobre el lenguaje.
- **Eclipse**[45]**:** Eclipse es un editor de código fuente desarrollado en Java del cual se pueden usar sus clases internas para generar el árbol sintáctico de códigos de diferentes lenguajes[56, 57] entre los cuales no se incluyen los derivados de Lisp (Racket / drscheme es un derivado de Lisp).

2.2. Desarrollo

El sistema desarrollado en este proyecto depende del software educativo moodle[58] que es extensivamente utilizado en varias universidades y no solo en los cursos que tienen que ver con programación sino que además aloja los cursos de todas las facultades de la universidad - cálculos, idiomas, físicas - por lo que se debe evitar lo más posible poner en juego la seguridad del servidor en donde se despliega moodle. En el asunto de guardar la seguridad del software educativo moodle se decidió no ejecutar los códigos de las entregas en el mismo computador que ejecuta el software educativo moodle, bajo esta decisión se utilizó la siguiente arquitectura de sistema:

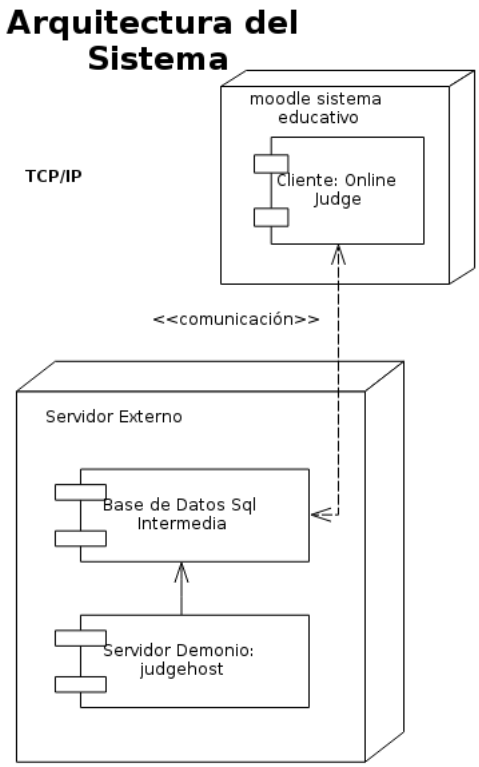

Figura 1: Arquitectura del sistema

2.2.1. Cliente

Del lado de la aplicación que se encuentra embebida en moodle se utilizo como ejemplo y base la aplicación Online Judge 2 plugin for Moodle 2[26]. La aplicación Online Judge 2 plugin for Moodle 2 permitía hacer el envió de un archivo de código fuente para que este fuera compilado y ejecutado frente a una cantidad de pruebas que siguen el estilo utilizado en ACM ICPC[30]; básicamente los programas reciben entradas por medio de entrada estándar (flujos) y para estas entradas se tiene un conjunto de salidas esperadas.

Algoritmo De Pruebas De Código

En el desarrollo de la aplicación se conserva el estilo de pruebas ACM ICPC ya que permite al usuario "profesor" (al que genere las pruebas) concentrarse menos en como debe ser desarrollada una solución y más en que debe solucionar. También ayuda a que el usuario estudiante no genere métodos demasiado largos que lo solucionan todo porque no es necesario especificarle al estudiante los nombres

Figura 2: Aplicación base del cliente

de los métodos y esto le da la libertad de desarrollar sus soluciones mientras las mismas cumplan con las técnicas necesarias para convertir las entradas en las salidas.

Los archivos bases para el cliente fueron tomados del repositorio publico: https://github.com/hit-moodle/moodle-local_onlinejudge/

Estos archivos son luego copiados en directorio_moodle/local/onlinejudge para su instalación como se en la sección 5.2 Instalación.

2.2.2. Servidor

La aplicación base Online Judge 2 plugin for Moodle 2 en su forma original solo permite enviar soluciones consistentes de 1 archivo de código fuente y depende de un servicio web privado para la ejecución de sus códigos[33] lo que limita su rendimiento y su seguridad por lo que se hizo necesaria la extensión de su funcionalidad.

Para la extensión de la funcionalidad de la aplicación base se incluyo la conexión con un servidor demonio que a su vez extiende la funcionalidad de domjudge[59]; ya que domjudge está en su versión 3.4.2 y tiene más de 9 años de desarrollo se trato de no modificar la organización del mismo conservando su comunicación a través de una base de datos que guarda la información de los envíos sin calificar. La conexión atraves de la base de datos permite que en momentos de alta demanda (como en el final de un plazo de entrega) los envíos de las soluciones sean puestos en cola y no hagan que el sistema colapse.

Utilizar como base a domjudge permite ejecutar más de un servidor para la calificación de los envíos de las soluciones ya que el acceso a la base de datos esta desarrollado para que pueda ser concurrente e incluso se puede configurar un sistema chroot que permite que los archivos del computador que corre el servidor domjudge estén seguros.

Calificación Envíos Multiarchivos

Cuando se termina de desarrollar la conexión de los componentes discutidos la ejecución de los envíos multiarchivos para C++ se soluciona fácilmente ya que se encuentra desarrollado en la funcionalidad del domjudge. Para el caso de Java la configuración de la seguridad y las preferencias del sistema tienen que ser tratados más extensamente pero para scheme / Racket la ejecución y la compilación del lenguaje es un problema de investigación ya que solo ha sido tratado por alrededor de 2 aplicaciones de calificación automática de tareas[25](la segunda es un proyecto no oficial de un estudiante de la universidad). La ejecución / compilación de scheme fue solucionada con la correcta utilización de los compiladores e interpretadores nuevos del lenguaje y la introducción de una regla: el archivo que contiene "el método main" o en otras palabras el que inicia la ejecución tiene que llamarse "main" con cualquier extensión valida además para conservar la estructura de las pruebas se debe utilizar la función read de Racket para permitir la lectura de los datos provistos por la consola del sistema.

El sistema de la ejecución de los envíos de los estudiantes tiene la posibilidad de poner un tiempo máximo de ejecución. El sistema termina la ejecución cuando este tiempo ha sido cumplido reportando este hecho en la calificación lo que permite que los estudiantes desarrollen sus soluciones cumpliendo a cabalidad las técnicas necesarias para asegurar la velocidad del código y no desarrollar por ejemplo un "mergesort" que corre en $O(n^2)$.

2.2.3. Calificación de la Documentación

El proyecto actual no solo enfrenta la posibilidad de que los estudiantes generen código correcto, eficiente y fácil de calificar sino que también busca que los códigos desarrollados presenten documentación y vayan influyendo en los estudiantes a generar códigos documentados y además bien presentados, esto se discutirá luego.

Aunque hay aplicaciones que se encargan de generar la documentación automáticamente en formato HTML mediante anotaciones hechas sobre el código fuente no hay discusiones (en tesis o investigaciones anteriores) sobre como dar una valoración numérica a que tan bien o que tan mal esta documentado un archivo de código fuente.

Al enfrentar el tema de la calificación lo primero fue hacer una investigación de las aplicaciones discutidas anteriormente que se encuentra documentada en la sección 2.1.5 - Estado del arte, Generadores De Documentación De Código - de esta investigación se obtuvo lo siguiente:

1. Las funcionalidades de los generadores automáticos de documentación no son del todo adecuadas para ser utilizadas como calificadores de la documentación en este proyecto porque las opciones disponibles para su uso habrían aumentado la complejidad del proyecto incluyendo otro lenguaje de programación al ya variado desarrollo actual. También aunque estás herramientas tenían la capacidad de reconocer los nombres de los métodos (no en todos los lenguajes necesarios) no se encargan de la parte más complicada al calificar códigos fuente; el reconocimiento de los nombres de los argumentos. Por las deficiencias de capacidad en los generadores automáticos de documentación se investigaron generadores de parsers o scanners de código fuente[55, 56] los que permitían generar las capacidades necesarias pero aumentaban la complejidad del proyecto haciendo que el tiempo para su desarrollo no fuera suficiente. Por último se decidió desarrollar un parser que podía calificar las etiquetas de documentación necesarias excluyendo las que tienen que ver con la documentación de métodos y variables.

2. El estilo de documentación más conocido viene del aplicado en java que consta de anotaciones @author, @version, @copyright, etc, no solo es conocido por ser usado en uno de los más populares lenguajes de desarrollo sino que además y gracias a Doxygen[51] es aplicable en muchísimos más lenguajes (C++, C, Objective-C, C#, PHP, Java, Python, IDL - Corba, Microsoft, y los sabores UNO / OpenOffice -, Fortran, VHDL, Tcl) por lo que el mismo fue escogido para el proyecto actual. Al escoger el estilo de Javadoc se espera que el conocimiento adquirido por el estudiante al aplicar las etiquetas desde primer semestre (el estilo se conserva para Racket /scheme) le sea útil cuando haga algún desarrollo en cualquier otro lenguaje de programación.

Estilos Utilizado Y Algoritmo De Calificación

Se califica el estilo primero mirando las etiquetas validas: @etiqueta; la etiqueta tiene que estar presente en el archivo de código fuente y tiene que tener algún contenido diferente de espaciadores en la misma linea que ocupa. Luego se hace el radio de las las etiquetas validas sobre las etiquetas posibles:

Algoritmo 1 Radio de etiquetas validas

$$calificacion = \frac{etiquetas-validas}{etiquetas-posibles}$$

El estilo de documentación es configurable mediante el link "Manejar los estilos de programación" de un trabajo de programación creado en moodle bajo la aplicación entregada como se muestra en la sección 6.1.2.

Implementación

Teniendo en cuenta los estilos previamente configurados para un trabajo de programación la calificación de los mismos es realizada en la parte del servidor que como se describe aparte esta desarrollado en php basado en DomJudge[24] pero por razones de facilidad la ejecución del algoritmo de calificación fue desarrollado en Java. Por lo anterior la llamada del mismo se hace mediante la interfaz que tiene php con la consola del sistema(en el archivo judgedaemon.main.php).

Como se menciona la ejecución del algoritmo es desarrollada en Java siguiendo la estructura de un parser de código creando un árbol sintáctico del código fuente que organiza y hace asequibles los comentarios del archivo.

Este Parser fue desarrollado teniendo en cuenta el patrón de diseño factory para así poder fácilmente tratar los diferentes tipos de lenguajes y así no duplicar el código para cada uno de estos como se puede apreciar fácilmente en su diagrama de clases en la sección 5.1.3 en la página 47 con las tres clases FileNavigator, CPPFileNavigator, SchemeFileNavigator y sus relaciones.

Este parser se implementa teniendo en cuenta que nodos del árbol sintáctico tienen una estructura de comienzo y otra de final de nuevo visible en su diagrama de clases en la sección 5.1.3 en la página 47 con la estructura Node y su árbol la clase Tree.

2.2.4. Calificación de la Indentación

Se discute ahora la calificación de la indentación los archivos de código fuente. De nuevo dar la medición de que tan bien esta indentado un archivo de código fuente no está definido en previos trabajos de investigación por lo que se considera entonces hacer la comparación de cada uno de los archivos entregados con una versión "perfectamente indentada" bajo un estilo predefinido y dar un valor porcentual al resultado de esta comparación.

Algoritmo-Solución De Indentación De Archivos

Para realizar la comparación anteriormente discutida el primer paso obvio es la generación del archivo indentado para encargarse de esta tarea se buscaron diferentes aplicaciones capaces de indentar archivos de código fuente que fueran lo suficientemente configurables y que presentaran simplicidad frente a su ejecución desde un lenguaje de programación. De las diferentes aplicaciones investigadas

que son referenciadas en la sección 2.1.4 - Estado del Arte Indentadores - la única que presentaba la posibilidad de indentar archivos de código fuente de Racket / drscheme con una buena documentación era Emacs-Batch[47, 48] que aunque para su configuración fue necesario el aprendizaje de un nuevo lenguaje - Elisp[60] - cuenta con varios estilos predeterminados de indentación para los lenguajes que se basan en un estilo C (Java y C++) además permite definir el tamaño de un indentador y generar scripts de formas simples para modificar el estilo.

El código necesario sobre este lenguaje permitía generar las configuraciones de los diferentes estilos en los diferentes lenguajes de forma simple (cambiando solo el nombre del método que se llama):

Esto visible en los archivos .el en la carpeta <basepath>/judgehost/lib/indentacion/.

Un ejemplo de estas funciones se encuentra en la página 24 (Cuadro de Algoritmo, Funciones de indentación).

Pero incluso al usar Emacs-Batch[47, 48, 61] los resultados de los códigos no eran lo esperado ya que solo incluían espacios o tabuladores (identaban) pero se quería que el código se comparara a un código bien presentado sin importar su organización original para lo cual se debían incluir o borrar retornos de linea (esto se conoce como pretty-print). Para lograr este cometido se uso indent gnu[69] para Java y C++ y una función propia de Racket (pretty-print[70]) los cuales generaban una buena indentación con un tiempo de ejecución relativamente corto.

Luego de generar la indentación es necesaria la medición de la diferencia entre el código fuente sin indentar y el código indentado pero el tratamiento de esta diferencia no es un tema trivial; Las aplicaciones que se encargan de revisar la diferencia entre dos archivos[62, 63] lo hacen de dos formas:

1. Directamente carácter a carácter interpretando un archivo de código como una cadena muy larga lo que haría que cualquier falta de un espacio hiciera el resto del archivo completamente diferente.
2. Interpretan el archivo eliminando toda indentación (caracteres de espacios, tabuladores y caracteres de retorno) y hacen una comparación palabra a palabra del archivo, ya que este proceso elimina la indentación y eso es exactamente lo que se está calificando.

Algoritmo De Cuantificación De La Comparación Entre Archivos

Como solución se utiliza la implementación de python de un algoritmo publicado a finales de 1980 por Ratcliff y Obershelp bajo el nombre de "gestalt pattern

Algoritmo 2 Funciones de indentación

```
defun gnu ()
    (interactive)
     (save-buffer)
     (setq sh-indent-command (concat
                              "indent -gnu -ts8 --no-tabs"
                              buffer-file-name
                              )
           )
     (mark-whole-buffer)
     (universal-argument)
     (shell-command-on-region
      (point-min)
      (point-max)
      sh-indent-command
      (buffer-name)
      )
     (save-buffer)
)

defun default ()
    (interactive)
     (save-buffer)
     (setq sh-indent-command (concat
                              "Racket ../indent-racket.rkt "
                              )
           )
     (mark-whole-buffer)
     (universal-argument)
     (shell-command-on-region
      (point-min)
      (point-max)
      sh-indent-command
      (buffer-name)
      )
     (save-buffer)
)
```

matching". La idea es encontrar la subsecuencia contigua más larga. La misma idea se aplica entonces de forma recursiva a las piezas de las secuencias a la izquierda y a la derecha de la subsecuencia coincidente. En otras palabras primero se encuentra la secuencia más larga de código perfectamente igual entre las dos secuencias (indentado bien) luego se llama el mismo algoritmo recursivamente a la izquierda y a la derecha del código perfectamente indentado y así sucesivamente.

La medición se realiza utilizando la función ratio(), de la Clase SequenceMatcher de Python[64] que cuenta la cantidad de subsecuencias comunes y las contrasta contra las subsecuencias posibles obteniendo el valor de la medida. Aunque la medición de este algoritmo tampoco permite dar un resultado exacto y correcto matemáticamente de que tan bien esta indentado un archivo es un acercamiento bastante bueno ya que nos permite saber si un archivo está indentado perfectamente en comparación al estilo y también nos permite ver como se puede alejar lentamente de una nota perfecta(según las pruebas). También el profesor puede cambiar la nota asignada al estilo.

Código Python (<basepath>/judgehost/lib/indentacion/checkDiff.py):

Algoritmo 3 Código comparación indentación

```
import sys
import time
from difflib import SequenceMatcher
text1 = open(sys.argv[1]).read()
text2 = open(sys.argv[2]).read()
m = SequenceMatcher(None, text1, text2)
print(m.ratio())
```

El estilo de documentación es configurable mediante el link “Manejar los estilos de programación” de un trabajo de programación creado en moodle bajo la aplicación entregada como se ve en la sección 6.1.1.

Implementación

Al utilizar las funcionalidades de emacs y las librerías de Python la única parte faltante de la solución de la calificación es su utilización desde el código en php que lo llama lo cual es representado en el diagrama de clase en la sección 5.1.3 en la página 46 donde se pone la interacción entre judgedaemon.main y los archivos mencionados.

Definición De Los Estilos Utilizados

Se definen estos estilos de Indentación escogidos para C++ y Java como:

Estilo de K & R (tomado de wikipedia[65])

El estilo de K & R, llamado así porque se utilizó en el libro de Kernighan y Ritchie "El Lenguaje de Programación C", se utiliza comúnmente en C. También se utiliza para C++, C#, Java y otros lenguajes de programación de llave.

Al aplicar K&R cada función tiene su llave de apertura en la línea siguiente en el mismo nivel de indentación de su cabecera, las declaraciones dentro de las llaves son indentadas y la llave de cierre al final está en el mismo nivel de indentación que la cabecera de la función en una línea propia. Los bloques dentro de una función, sin embargo, tienen su llave de apertura en la misma línea que sus respectivas sentencias de control; las llaves de cierre se mantienen en una línea propia, a no ser que estén seguidas por una cosa o mientras palabra clave.

En este estilo de una sentencia de control con una sola sentencia en su ámbito de aplicación puede omitir las llaves. El lenguaje de programación C se refiere a esta practica como suelo fértil para errores de programación lógicos y la desalienta.

Algoritmo 4 Estilo K & R

```
int main(int argc, char *argv[])
{
    ...
  while (x == y) {
        something();
        somethingelse();

        if (some_error) {
            do_correct();
        } else {
            continue_as_usual();
        }
   }
      finalthing();
    ...
}
```

Estilo Linux[66]

El estilo del kernel es conocido por su uso extensivo en el código fuente del kernel de Linux. Linus Torvalds recomienda encarecidamente a todos los contribuyentes a seguirlo. Una descripción detallada del estilo (que no sólo tiene en cuenta la indentación, sino también las convenciones de nomenclatura, los comentarios y varios otros aspectos también) se puede encontrar en kernel.org. Este estilo toma prestados algunos elementos de K & R.

El estilo Linux utiliza tabuladores (con tabuladores establecidos a 8 caracteres) para el indentado. Las llaves de apertura deben ser puestas en la misma línea que la declaración de la función separadas por un espacio. Las etiquetas en una sentencia "switch" están alineados con el bloque que lo contiene (sólo hay un nivel de sangría). Una sentencia de control que sea seguida por una sola linea (como if, while o do-while) no necesita llaves ni de apertura ni de cierre. Sin embargo, si al menos una de las sub-sentencias en un "if-else" requiere llaves, entonces ambas subsentencias deben envolverse dentro de llaves. La longitud máxima de una línea es de 80 caracteres.

Algoritmo 5 Estilo Linux

```
int power(int x, int y) {
        int result;
        if (y < 0) {
                return 0;
       } else {
                for (result = 1; y; y--)
                        result *= x;
                return result;
        }
}
```

Y para Lisp como:

Estilo Lisp[67]

Este estilo de programación se basa en insertar corchetes(paréntesis, llaves o corchetes) de cierre en la última línea de un bloque. Este estilo hace de la indentación la única manera de distinguir los bloques de código pero tiene la ventaja de que no contiene líneas poco informativos. Esto podría ser fácilmente llamado el estilo de Lisp (porque este estilo es muy común en el código Lisp) o el estilo de Python (Python no tiene brackets, pero el diseño es muy similar, como lo demuestran los dos bloques de código siguientes).

Algoritmo 6 Estilo Lisp

```
;; In Lisp
(dotimes (i 10)
  (if (evenp i)
      (do-something i)
      (do-something-else i)))
```

Aunque para Java y C++ se definen solo 2 estilos de indentación en este documento en la aplicación para Java y C++ se incluyen algunos otros estilos para su calificación (gnu, bsd). En caso de querer añadir otro estilo de indentación el proceso sería muy parecido a incluir un nuevo lenguaje (sección 5.2.3) pero solo se realizarían los pasos que tiene que ver con las tablas (en la base de datos) que tienen como nombre indentation* y los archivos que están en las carpetas con el mismo nombre.

2.2.5. Configuración De Los Estilos De Indentación Y Documentación Para Un Trabajo De Programación

La utilización y los pantallazos de como queda la implementación se muestran de forma más exacta en las secciones: 6.1.1,6.1.2 con sus manuales de uso. El código responsable de esta funcionalidad se realiza teniendo en cuenta la estructura de desarrollo utilizado en moodle.

Cada uno de estos menús se basan en la extensión de la funcionalidad de una clase llamada moodleform que hace parte de las librerías que presenta moodle para facilitar el desarrollo sobre esta plataforma y que es visible en la sección 5.1.3 en la página 48 con las clases moodleform indentation_style_form, testcase_form, documentation_style_form.

2.2.6. Gráfica De Calificación De Un Trabajo De Programación

Las calificaciones son realizadas del lado del servidor y guardadas en la base de datos que funciona como comunicación entre el cliente y el servidor. La implementación de estas calificaciones tienen un buen nivel de dificultad primero empezando por la arquitectura del sistema (2.2 en la página 17) necesario con los códigos, los diferentes algoritmos mencionados y las relaciones entre los mismos. Además de lo mencionado para que el sistema tenga la capacidad de ser seguro y mostrar las calificaciones en tiempo real tiene que componerse de 2 demonios uno del lado del servidor (sección 5.1.3 en la página 46) y otro del lado del cliente (sección 5.1.3 en la página 49).

La ejecución de cada uno de estos demonios aunque puede funcionar con los otros procesos del sistema sin mostrar salida presenta algo de información importante cuando se les ejecuta en la consola.

El correcto funcionamiento de esta interfaz se basa en un funcionamiento en tiempo real; esto quiere decir que el “status” que muestra la interfaz cambia a “compilando” y “ejecutando”, etc esto incluye la necesidad de un programa demonio del lado del cliente el cual es básicamente un ciclo infinito que tiene como

paradas los updates o los envíos de señales al programa para su sincronización este demonio utiliza archivos como semáforos.

Para la visualización de las notas se utiliza un archivo interesante que permite crear un widget y mostrar los detalles de un envió aplicándole diferentes estilos este archivo es el que nombra como details.php y lo podemos ver en el diagrama de clase del cliente en moodle(en la sección 5.1.3 en la página 48).

Para más información los archivos generados por la calificación se encuentran en <basepath>/judgehost/judgins/ el resto de la ruta depende del nombre de la maquina y del usuario que la ejecuta además de que cada envió se guarda en una carpeta aparte:

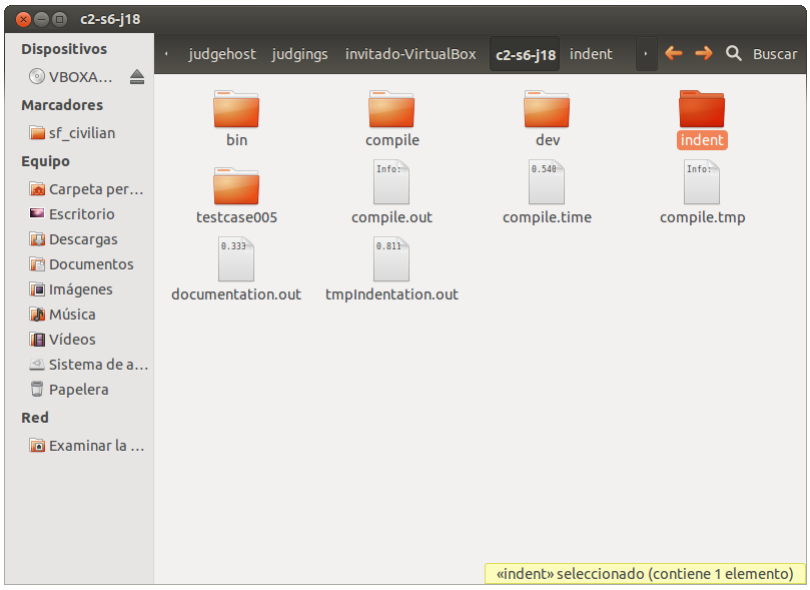

Figura 3: Archivos extra calificación

En la carpeta indent se encuentran los archivos de código fuente indentados y en general hay un conjunto de archivos generados por cada parte de la calificación.

Notificación De Falla

Debido a que el demonio por el lado del servidor es susceptible a fallas entonces se incluye en el archivo cron.php que es visible en el diagrama de clase en la sección 5.1.3 en la página 48 un método que tiene en cuenta cuanto tiempo ha estado esperando un envió para ser calificado. Si es mayor a el "Lapso de tiempo para considerar que el demonio judgehost esta caído (segundos)" que se puede ver en las ajustes del plugin (5.2.2 en la página 66) entonces la rutina cron.php envía correos a las personas configuradas de nuevo en los ajustes del plugin.

3. Conclusiones

- La planeación en problemas de largo alcance es una tarea complicada que solo se mejora con experiencia y con constante reimaginación de como lidiar con todos los problemas posibles que se pueden encontrar en cualquier despliegue.
- Un sistema de calificación automático es un punto critico en la educación ya que promete mejorar de forma considerable la calidad de los estudiantes de sistemas por que les permite llevar la teoría a la verdadera practica con parámetros claros y realmente probados sobre sus programas. Además al facilitar la calificación aumenta de forma significativa la cantidad de practica que puede tener un estudiante programando.
- El hecho de que los trabajos sobre la valoración numérica de los estilos ya sea de indentación o de documentación del código sean prácticamente inexistentes presenta un vació interesante en las ciencias de la computación y como se están midiendo dos valores que pueden tener tanta importancia en la escalabilidad de cualquier aplicación.
- La utilización de esta herramienta permite que los profesores se eviten mucho de trabajo repetitivo y poder concentrarse en áreas más interesantes de la enseñanza e incluso reducen altamente la posibilidad del error humano en las calificaciones de los estudiantes.
- La utilización de código libre permite que una sola persona cree aplicaciones complejas y con algún nivel de confiabilidad con un esfuerzo menor al requerido al desarrollar estas aplicaciones desde el inicio.
- Las practicas de desarrollo de software basadas que alientan realizar las pruebas primero(TDD, BDD[11, 6, 76, 68] y recientemente XP[31, 32]) se nos han venido vendiendo como la solución a todos los problemas del software actual, pero como todo ingeniero de software debe saber no existe un "Golden Hammer" el enfoque excesivo en las pruebas no es la solución a todo. Este acercamiento (en mi opinión y la de otros desarrolladores [71]) puede causar que la calidad de los diseños se vea reducida incluyendo muchos objetos de servicio y/o compromisos en el diseño que reducen la calidad del proyecto.

4. Trabajo Futuro

Teniendo en cuenta que la aplicación no ha sido terminada se presenta este apartado como ideas de extensión del proyecto para nuevos proyectos.

- Extender la funcionalidad de las opciones de calificación del estilo de indentación para permitir más opciones y por lo tanto permitir la especificación de estilos más flexibles que se podrían acercar más a las diferentes configuraciones permitidas por los formateadores de estilo del entorno de programación eclipse[45]. Y también poder extender las posibilidades de la calificación de los comentarios.

- Permitir que los estilos de programación e indentación puedan ser descargables y que se puedan compartir entre profesores.

- Incluir en el entorno de calificación restricciones sobre la cantidad de memoria permitida en un programa sin que estos sean los estándares del lenguaje de programación utilizado en el trabajo de programación.

- La posibilidad de un curso autocalificable para aprender técnicas sobre algún tema en especial, en donde podría tomar como ejemplo portal USACO[13].

- La posibilidad de ordenar las envíos de soluciones por tiempos de ejecución y por uso de memoria para conocer cuales son las mejores y además incluir la posibilidad de calificar la limpieza del código.

- Mediante la adición del tiempo de envió de un trabajo de programación como algo calificable se podrían organizar maratones de programación para todas las personas con acceso al la aplicación de forma simple.

- Incluir más lenguajes de programación basándose en algunos de los scripts ya existentes para su compilación y extendiendo la funcionalidades implementadas para la calificación del estilo y de la documentación.

- Dar la posibilidad a los usuarios estudiantes de incluir pruebas o "testcases" para un trabajo de programación en concreto y así no solo aumentar la cantidad de pruebas realizables para un trabajo de programación sino también poder incluir una bonificación para la persona que cree estos nuevos casos de pruebas.

- Incluir un foro que tenga información de links e información sobre las diferentes técnicas de programación usadas en cada uno de los trabajos de programación cuyo funcionamiento sea parecido a Algorithm Tutorials de topcoder[14].

- Permitir la capacidad de no solo generar pruebas para el código que tengan en cuenta su salida para una entrada dada sino también que se puedan realizar pruebas unitarias sobre el código mandado y algunas pruebas automáticas sobre el funcionamiento del software como por ejemplo las realizadas por selenium. Lo anterior no solo para los usuarios que crean los trabajos de programación sino también permitir que los estudiantes envíen sus propias pruebas de código.

- Incluir la posibilidad de usar PMD[3, 4] sobre el código enviado esto para detectar posibles bugs, código muerto o código subóptimo o sobrecomplicado.
- Mediante la adición de la posibilidad de revisar el código de diferentes envíos hechos por otros estudiantes a un trabajo de programación en concreto los estudiantes podrían enviar su código la cantidad de veces que quieran antes de la fecha de entrega limite así pueden cambiar su código teniendo en cuenta cosas que aprendieran del código de los demás o de los comentarios sobre su propio código.

Referencias

[1] http://courses.cs.vt.edu/curator/, 2004. Online; acceso junio-2013.

[2] http://web-cat.org/, 2006. Online; acceso junio-2013.

[3] http://pmd.sourceforge.net/, 2006. Online; acceso junio-2013.

[4] http://en.wikipedia.org/wiki/PMD_(software)/, 2006. Online; acceso junio-2013.

[5] http://kattis.csc.kth.se/, 2006. Online; acceso junio-2013.

[6] http://sebasjm.wordpress.com/2009/09/23/behavior-driven-development/, 2009. Online; acceso junio-2013.

[7] http://plt-scheme.org/software/mzscheme/, 2010. Online; acceso junio-2013.

[8] http://racket-lang.org/, 2010. Online; acceso junio-2013.

[9] http://scholarworks.sjsu.edu/etd_projects/51/, 2010. Online; acceso junio-2013.

[10] https://github.com/hit-moodle/onlinejudge/blob/master/README.txt, 2011. Online; acceso junio-2013.

[11] http://phpunit.de/manual/3.7/en/behaviour-driven-development.html, 2011. Online; acceso junio-2013.

[12] http://uva.onlinejudge.org/, 2012. Online; acceso julio-2012.

[13] http://ace.delos.com/usacogate, 2012. Online; acceso junio-2012.

[14] http://community.topcoder.com/tc , 2012. Online; acceso junio-2012.

[15] http://codeforces.com/, 2012. Online; acceso febrero-2012.

[16] http://www.spoj.pl, 2012. Online; acceso junio-2012.

[17] http://livearchive.onlinejudge.org/, 2012. Online; acceso junio-2012.

[18] http://www.programming-challenges.com/pg.php?page=index, 2012. Online; acceso junio-2012.

[19] http://coj.uci.cu/index.xhtml, 2012. Online; acceso junio-2012.

[20] http://poj.org/, 2012. Online; acceso junio-2012.

[21] http://acm.timus.ru/, 2012. Online; acceso junio-2012.

[22] http://teddy.itc.mx/ , 2012. Online; acceso junio-2012.

[23] http://www.ecs.csus.edu/pc2/pc2desc.html, 2012. Online; acceso junio-2012.

[24] http://domjudge.sourceforge.net/ , 2012. Online; acceso junio-2012.

[25] http://paginas.fe.up.pt/ ei05058/doku.php?id=thesis:schemeassessment, 2012. Online; acceso junio-2013.

[26] https://github.com/hit-moodle/moodle-local_onlinejudge/blob/master/README.md, 2012. Online; acceso junio-2013.

[27] http://sc.lib.muohio.edu/handle/2374.MIA/251?show=full, 2012. Online; acceso junio-2013.

[28] http://diycomputerscience.com/blog/post/2012/02/01/using-automated-grading-systems-for-programming-courses, 2012. Online; acceso junio-2013.

[29] https://code.google.com/p/tamarin/wiki/Philosophy, 2012. Online; acceso junio-2013.

[30] http://www.acm.org/ , 2012. Online; acceso abril-2012.

[31] http://www.slideshare.net/MrSMAk/extreme-programming-12047889, 2013. Online; acceso diciembre-2013.

[32] http://xprogramming.com/what-is-extreme-programming/, 2013. Online; acceso diciembre-2013.

[33] http://ideone.com/, 2013. Online; acceso junio-2013.

[34] http://prettydiff.com/, 2013. Online; acceso noviembre-2013.

[35] https://github.com/austincheney/Pretty-Diff, 2013. Online; acceso noviembre-2013.

[36] https://gist.github.com/, 2013. Online; acceso noviembre-2013.

[37] http://prettyprinter.de/index.php, 2013. Online; acceso noviembre-2013.

[38] http://beta.phpformatter.com/, 2013. Online; acceso noviembre-2013.

[39] http://astyle.sourceforge.net/, 2013. Online; acceso noviembre-2013.

[40] http://www.bierkandt.org/beautify/beautify_php.php, 2013. Online; acceso noviembre-2013.

[41] https://github.com/chrisryan/dws-format-filter, 2013. Online; acceso noviembre-2013.

[42] https://github.com/clbustos/PHP_Beautifier, 2013. Online; acceso noviembre-2013.

[43] http://sourceforge.net/projects/universalindent/, 2013. Online; acceso noviembre-2013.

[44] http://vim.wikia.com/wiki/Indenting_source_code, 2013. Online; acceso noviembre-2013.

[45] http://eclipse.org/, 2013. Online; acceso octubre-2013.

[46] http://www.eclipsezone.com/eclipse/forums/t88960.html, 2013. Online; acceso noviembre-2013.

[47] http://www.gnu.org/software/emacs/, 2013. Online; acceso octubre-2013.

[48] http://www.cslab.pepperdine.edu/warford/BatchIndentationEmacs.html, 2013. Online; acceso octubre-2013.

[49] http://www.oracle.com/technetwork/java/javase/documentation/index-jsp-135444.html, 2013. Online; acceso noviembre-2013.

[50] http://www.oracle.com/technetwork/java/javase/terms/license/index.html, 2013. Online; acceso noviembre-2013.

[51] www.doxygen.org, 2013. Online; acceso noviembre-2013.

[52] http://smartcomments.github.io/, 2013. Online; acceso noviembre-2013.

[53] http://www.naturaldocs.org/, 2013. Online; acceso noviembre-2013.

[54] http://www.naturaldocs.org/about.html, 2013. Online; acceso noviembre-2013.

[55] https://java.net/projects/javacc/, 2013. Online; acceso noviembre-2013.

[56] http://goo.gl/icl4Mq, 2013. Online; acceso noviembre-2013.

[57] http://www.eclipse.org/articles/article.php?file=Article-JavaCodeManipulation_AST/index.html, 2013. Online; acceso noviembre-2013.

[58] https://moodle.org/, 2013. Online; acceso noviembre-2013.

[59] http://www.domjudge.org/docs/admin-manual-3.html#ss3.1, 2013. Online; acceso noviembre-2013.

[60] http://www.gnu.org/software/emacs/manual/html_node/elisp/, 2013. Online; acceso noviembre-2013.

[61] http://www.emacswiki.org/emacs/IndentingC, 2013. Online; acceso octubre-2013.

[62] http://linux.about.com/library/cmd/blcmdl1_diff.htm, 2013. Online; acceso noviembre-2013.

[63] http://linux.about.com/library/cmd/blcmdl1_sdiff.htm, 2013. Online; acceso noviembre-2013.

[64] http://docs.python.org/2/library/difflib.html, 2013. Online; acceso noviembre-2013.

[65] http://en.wikipedia.org/wiki/Indent_style#K.26R_style, 2013. Online; acceso noviembre-2013.

[66] http://en.wikipedia.org/wiki/Indent_style#Kernel_style, 2013. Online; acceso noviembre-2013.

[67] http://en.wikipedia.org/wiki/Indent_style#Lisp_style, 2013. Online; acceso noviembre-2013.

[68] http://es.wikipedia.org/wiki/Desarrollo_guiado_por_pruebas, 2013. Online; acceso junio-2013.

[69] http://www.gnu.org/software/indent/, 2014. Online; acceso marzo-2014.

[70] http://docs.racket-lang.org/reference/pretty-print.html, 2014. Online; acceso marzo-2014.

[71] http://david.heinemeierhansson.com/2014/tdd-is-dead-long-live-testing.html, 2014. Online; acceso marzo-2014.

[72] http://my.safaribooksonline.com/book/software-engineering-and-development/extreme-programming/9781449399849/ivdot-extreme-programming-artifacts/extreme_programming_artifacts, 2014. Online; acceso marzo-2014.

[73] goo.gl/T2ISzZ, 2014. Online; acceso marzo-2014.

[74] http://phpunit.de/, 2014. Online; acceso marzo-2014.

[75] http://docs.moodle.org/dev/PHPUnit, 2014. Online; acceso marzo-2014.

[76] http://www.drdobbs.com/open-source/behavior-driven-development-with-easyb/232800187, 208. Online; acceso junio-2013.

5. Anexos

5.1. Metodología

La metodología escogida para el desarrollo de este trabajo de grado es Extreme Programming(XP)[31, 72, 73], se realizaron dos iteraciones debido a estimaciones demasiado bajas y por lo tanto atrasos en las entregas; esto llevo a la eliminación de unas historias de usuario y la generación de un nuevo plan de entregas.

5.1.1. Historias De Usuario

Número de historia: A1	**Titulo:** Aplicación Base Moodle	**Fecha:** Abril 6 de 2013
Descripción de la historia: Buscar una aplicación o escribir el código necesario que cumpla las condiciones mencionadas.		
Anotaciones: Un sistema que mediante la integración con el software educativo moodle 2.5 resuelva los siguientes módulos de la aplicación: 1) Una interfaz que permita a un docente crear trabajos de programación.2)Un modulo que permita la subida de archivos a la aplicación, para su calificación en un trabajo de programación especifico. Un modulo que permita la calificación de los trabajos de programación teniendo entradas y salidas esperadas utilizando entrada y salida estándar. Un sistema que muestre los errores dados por el compilador del código fuente.		

Número de historia: A2	**Titulo:** Formato De Calificación De La Documentación	**Fecha:** Abril 6 de 2013
Descripción de la historia: Generar la especificación del formato con el cual se comparará la documentación de los archivos de código fuente para los lenguajes: java 1.6, scheme 5.0 o C++ 4.1.2.		
Anotaciones:		

Número de historia: A3	**Titulo:** Sistema De Calificación De La Documentación	**Fecha:** Abril 6 de 2013
Descripción de la historia: Crear un sistema de comparación de la documentación de los archivos de código fuente contra el formato especificado y configurable para su correspondiente lenguaje.		
Anotaciones:		

Número de historia: A4	**Titulo:** Documentación Del Sistema De Calificación De La Documentación	**Fecha:** Abril 6 de 2013
Descripción de la historia: Documentación de código que permita la reutilización de el modulo(métodos y variables públicos).		
Anotaciones:		

Número de historia: A5	**Titulo:** Formato De Calificación De La Indentación	**Fecha:** Abril 6 de 2013
Descripción de la historia: Especificación de los dos formatos más populares de programación con los cuales se comparará el estilo de los archivos de código fuente para los lenguajes: java 1.6, scheme 5.0 o C++ 4.1.2 .		
Anotaciones:		

Número de historia: A6	**Titulo:** Sistema De Calificación De La Indentación	**Fecha:** Abril 6 de 2013
Descripción de la historia: Crear un sistema de comparación de los estilos de los archivos de código fuente contra el formato especificado para su correspondiente lenguaje.		
Anotaciones:		

Número de historia: A7	**Titulo:** Documentación Del Sistema De Calificación De La Indentación	**Fecha:** Abril 6 de 2013
Descripción de la historia: Documentación de código que permita la reutilización de el modulo(métodos y variables públicos).		
Anotaciones:		

Número de historia: A8	**Titulo:** Configuración De Un Trabajo De Programación	**Fecha:** Abril 6 de 2013
Descripción de la historia: Modificación del sistema de creación de trabajos de programación que permita al mismo la funcionalidades de: para un trabajo de programación con su lenguaje estipulado entre java 1.6, scheme 5.0 o C++ 4.1.2 se pueda escoger un estilo de programación y un tipo de documentación que pueda ser aplicado a el lenguaje en particular en un trabajo de programación particular.		
Anotaciones:		

Número de historia: A9	**Titulo:** Manual De Configuración De Un Trabajo De Programación	**Fecha:** Abril 6 de 2013
Descripción de la historia: Manual de uso del modulo donde se especifica la creación de un trabajo de programación especifico bajo un lenguaje de programación mostrando la selección de un estilo de documentación y uno de programación para la calificación de ese trabajo y además mostrando como configurar qué ejercicios va a tener el trabajo.		
Anotaciones:		

Número de historia: A10	**Titulo:** Documentación Modulo De Configuración De Un Trabajo De Programación	**Fecha:** Abril 6 de 2013
Descripción de la historia: Documentación de código que permita la reutilización de el modulo(métodos y variables públicos).		
Anotaciones:		

Número de historia: A11	**Titulo:** Calificación De Varios Archivos	**Fecha:** Abril 6 de 2013
Descripción de la historia: Modificar el sistema de creación de trabajos de programación para que permita las funcionalidades de: para un lenguaje estipulado entre java 1.6, scheme 5.0 o C++ 4.1.2 se pueda generar un trabajo de programación que al enviar una solución se pueda en forma de varios archivos(como por ejemplo diferentes clases en diferentes archivos en el caso de Java).		
Anotaciones:		

Número de historia: A12	**Titulo:** Manual De Creación De Un Trabajo De Programación	**Fecha:** Abril 6 de 2013
Descripción de la historia: Manual de uso del modulo donde se especifica la creación de un trabajo de programación especifico bajo un lenguaje de programación mostrando para la calificación de ese trabajo y además mostrando como configurar qué ejercicios va a tener el trabajo.		
Anotaciones:		

Número de historia: A13	**Titulo:** Documentación Modulo De Calificación De Varios Archivos	**Fecha:** Abril 6 de 2013
Descripción de la historia: Documentación de código que permita la reutilización de el modulo(métodos y variables públicos).		
Anotaciones:		

Número de historia: A14	**Titulo:** Generar Notas Individuales	**Fecha:** Abril 6 de 2013
Descripción de la historia: Módulo que usando los módulos desarrollados para la calificación de: la documentación, el estilo, los errores del código fuente y los resultados de las pruebas unitarias genere los datos necesarios para llenar la tabla descrita.		
Anotaciones:		

Número de historia: A15	**Titulo:** Mostrar Notas Individuales	**Fecha:** Abril 6 de 2013
Descripción de la historia: Interfaz gráfica que revele un informe de los resultados de las pruebas.		
Anotaciones:		

Número de historia: A16	**Titulo:** Manual De Uso Del Modulo De Generación De Notas Individuales	**Fecha:** Abril 6 de 2013
Descripción de la historia: Manual de uso del módulo donde se muestre el proceso de generación de una nota para un estudiante.		
Anotaciones:		

Número de historia: A17	**Titulo:** Manual De Uso Del Modulo De Consulta De Notas Individuales	**Fecha:** Abril 6 de 2013
Descripción de la historia: Manual de uso del módulo donde se muestre el proceso de consulta de una nota para un estudiante en un trabajo de programación especifico.		
Anotaciones:		

Número de historia: A18	**Titulo:** Documentación De Código Del Modulo De Notas Individuales	**Fecha:** Abril 6 de 2013
Descripción de la historia: Documentación de código que permita la reutilización del módulo (métodos y variables públicos).		
Anotaciones:		

Número de historia: A19	**Titulo:** Modulo De Notas Por Trabajo De Programación	**Fecha:** Abril 6 de 2013
Descripción de la historia: Realizar una tabla estadística con los resultados generados en cada una de las pruebas para todos los estudiantes con un trabajo de programación teniendo en cuenta los apartados mencionados (documentación, estilo, errores de código fuente y errores en los resultados). Y que sea visualizable para cualquier usuario que tenga que ver con el curso a el cual se asigno el trabajo de programación.		
Anotaciones: 1) Módulo que usando los módulos desarrollados para la calificación de: la documentación, el estilo, los errores del código fuente y los resultados de las pruebas unitarias genere los datos necesarios para llenar la tabla descrita. 2) Interfaz gráfica que revele un informe de los resultados de las pruebas para todos los estudiantes en un trabajo de programación especifico. 3) Manual de uso del módulo donde se muestre el proceso de consulta del desempeño (osea poder visualizar las diferentes notas para todos los estudiantes) para todos los estudiantes en un trabajo de programación especifico. 4) Documentación de código que permita la reutilización del módulo (métodos y variables públicos).		

Número de historia: A20	**Titulo:** Modulo De Notas Por Asignatura	**Fecha:** Abril 6 de 2013
Descripción de la historia: Realizar una tabla estadística con los resultados generados en cada una de las pruebas para todos los estudiantes con todos los trabajo de programación teniendo en cuenta los apartados mencionados (documentación, estilo, errores de código fuente y errores en los resultados). Y que sea visualizable para cualquier usuario que tenga que ver con el curso a el cual se asigno el trabajo de programación.		
Anotaciones: 1) Módulo que usando los módulos desarrollados para la calificación de: la documentación, el estilo, los errores del código fuente y los resultados de las pruebas unitarias genere los datos necesarios para llenar la tabla descrita. 2) Interfaz gráfica que revele un informe de los resultados de las pruebas para todos los estudiantes para todos los trabajos de programación realizados a lo largo del curso.3) Manual de uso del módulo donde se muestre el proceso de consulta del desempeño (osea poder visualizar las diferentes notas para todos los estudiantes) para todos los estudiantes en todos los trabajo de programación realizados hasta la fecha de la consulta en el curso.4) Documentación de código que permita la reutilización del módulo (métodos y variables públicos).		

Primer Plan de Entregas

	Semana																										
Historia de Usuario	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
A1	■	■	■	■	■	■	■	■																			
A2, A3, A4									■	■	■	■															
A5, A6, A7												■	■	■													
A8, A9, A10															■	■											
A11,A12, A13																	■	■	■	■							
A14, A15, A16, A17, A18																					■	■	■				
A19																								■	■		
A20																										■	■

Cuadro 2: Primer plan de entregas

El proyecto primero tuvo una iteración larga que contó de muchos spikes por que muchas funcionalidades del proyecto no tenían precedente en su implementación y muchas otras ni siquiera tenían base teórica. Esto influyo sobre las estimaciones y a lo largo en un atraso en el calendario de entregas. Por uno de los principios de XP se volvió a generar un nuevo plan de entregas y se revisaron las historias de usuario necesarias. En esta revisión se eliminaron las historias de usuario A19 y A20 y teniendo en cuenta el nuevo plan de entregas se tienen las historias de usuario:

Número de historia: B1	**Titulo:** Artefactos XP	**Fecha:** Enero 20 de 2014
Descripción de la historia: Generación de los artefactos de documentación más populares para Extreme Programming.		
Anotaciones:		

Número de historia: B2	**Titulo:** Gráficas De Notas	**Fecha:** Enero 20 de 2014
Descripción de la historia: Dar la funcionalidad completa a las maqueta de las gráfica desarrollada para mostrar las calificaciones por estudiante desarrollando un script demonio que permita a la gráfica mostrar las calificaciones generadas en tiempo real. Tener en cuenta mostrar los datos de compilaciones fallidas, de tiempos de ejecución y de resultados de indentación y documentación al generar estas notas.		
Anotaciones:		

Número de historia: B3	**Titulo:** Traducción Plugin	**Fecha:** Enero 20 de 2014
Descripción de la historia: Ya que la aplicación base instalada en moodle como cliente tenia el ingles como idioma en el que mostraba su interfaz se prosiguió desarrollando en este mismo lenguaje entonces es necesaria su traducción al español.		
Anotaciones:		

Número de historia: B4	**Titulo:** Versión Final Instaladores	**Fecha:** Enero 20 de 2014
Descripción de la historia: Luego de terminar la funcionalidad de la aplicación hay que mejorar la calidad de los instaladores e incluir la creación del demonio de la actividad 2.		
Anotaciones:		

Número de historia: B5	**Titulo:** Manuales De Usuario	**Fecha:** Enero 20 de 2014
Descripción de la historia: Modificar los manuales de usuario entregados para incluir el total de las funcionalidades desarrolladas al final y cumplir con todos los resultados esperados.		
Anotaciones:		

Número de historia: B6	**Titulo:** Revisión Documentación De Código	**Fecha:** Enero 20 de 2014
Descripción de la historia: Revisar y corregir la documentación de los códigos entregados.		
Anotaciones:		

Número de historia: B7	**Titulo:** Generación Documentación De Código	**Fecha:** Enero 20 de 2014
Descripción de la historia: Luego de revisar la documentación generarla para poder entregarla en formato html junto con sus códigos fuente.		
Anotaciones:		

Número de historia: B8	**Titulo:** Conclusiones	**Fecha:** Enero 20 de 2014
Descripción de la historia: Generar las conclusiones del trabajo desarrollado.		
Anotaciones:		

Número de historia: B9	**Titulo:** Corrección Del Documento	**Fecha:** Enero 20 de 2014
Descripción de la historia: Poner al día todo el documento teniendo en cuenta todas las actividades que completan este proyecto y hacer las correcciones necesarias.		
Anotaciones:		

Segundo Plan de Entregas

	Semana							
Actividad	01	02	03	04	05	06	07	08
B1								
B2								
B3								
B4								
B5								
B6								
B7								
B8								
B9								

Cuadro 3: Segundo plan de entregas

5.1.2. Diagrama De Casos De Uso

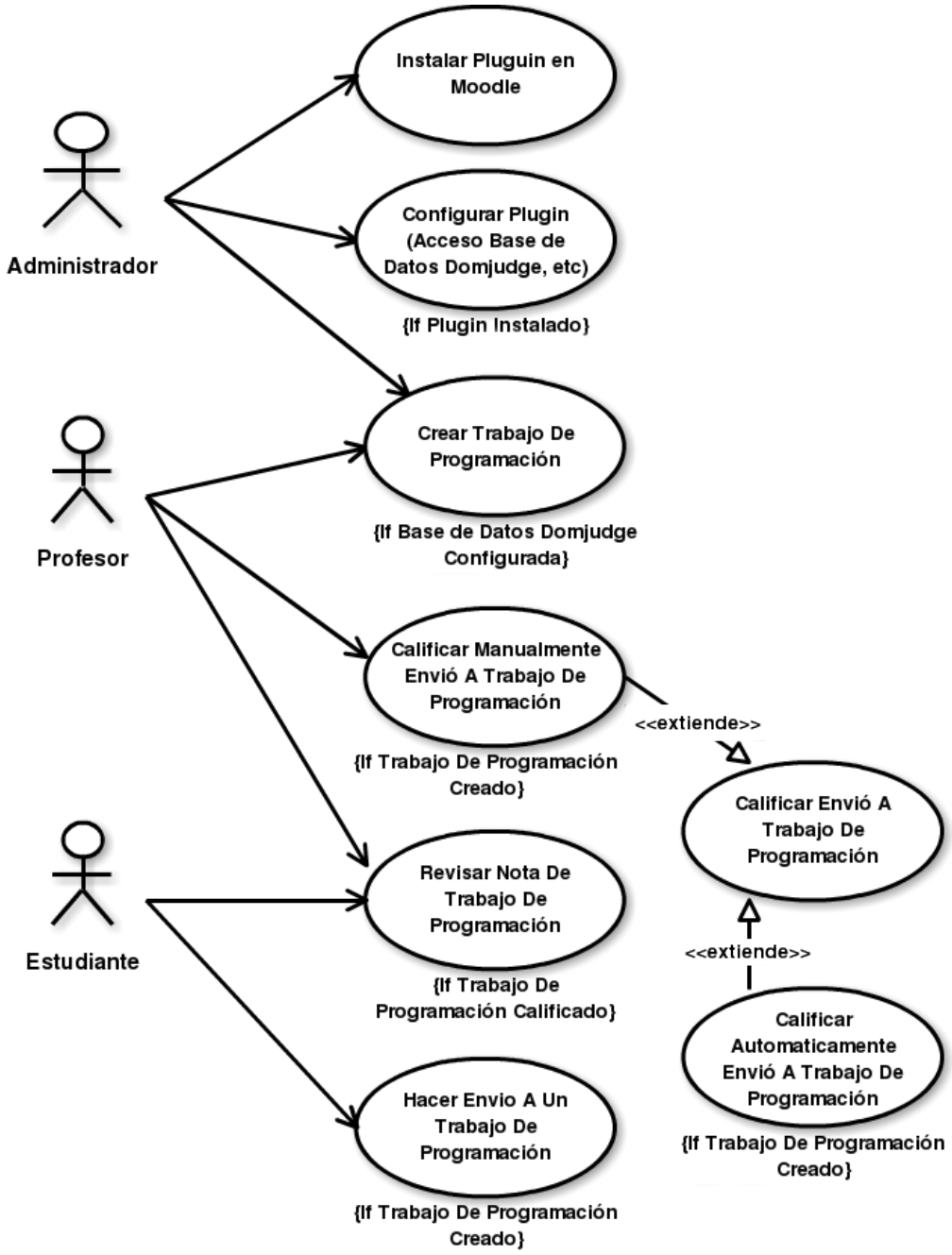

Figura 4: Diagrama de casos de uso

5.1.3. Diagramas de Clase

Diagrama De Clase Servidor

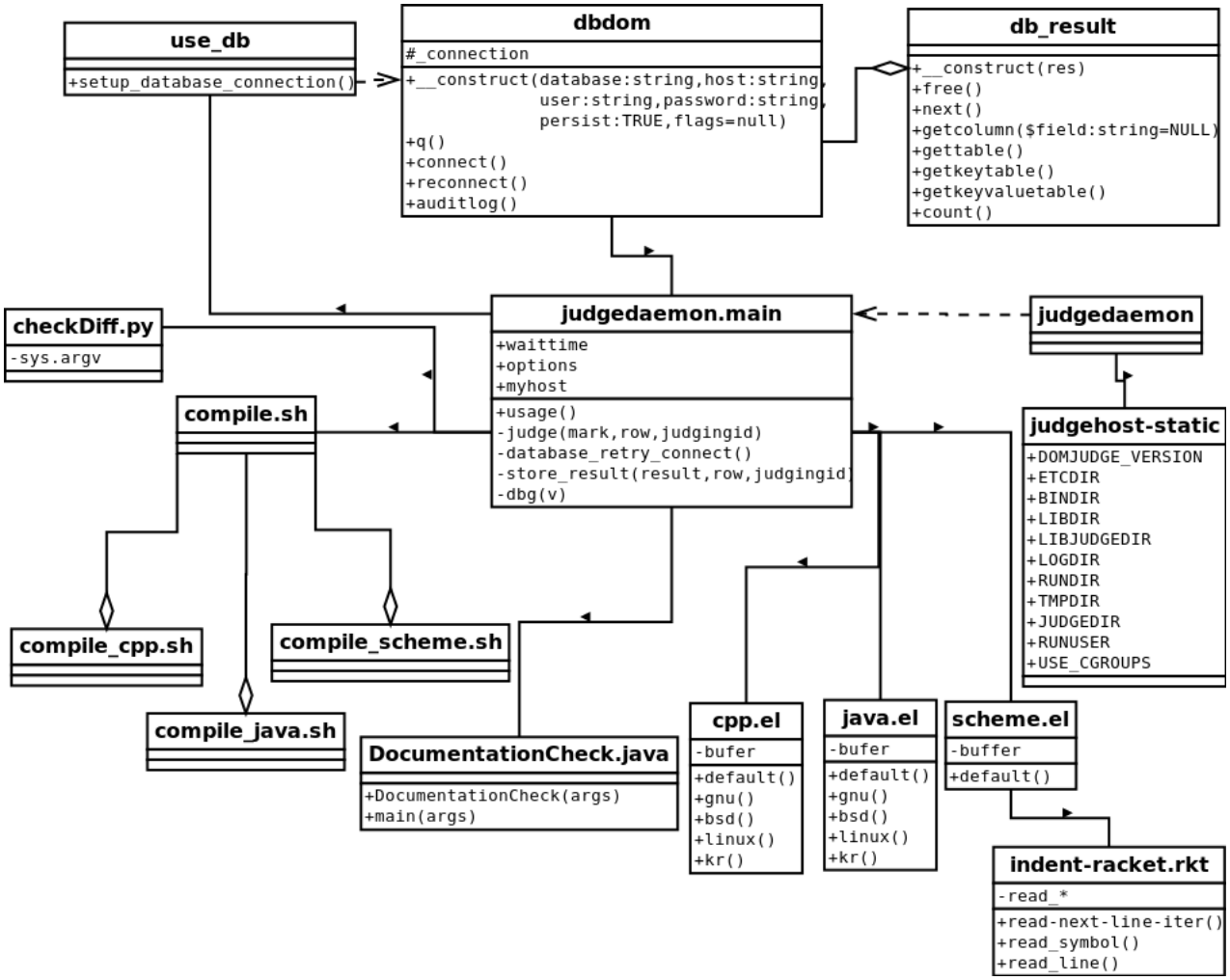

Figura 5: Diagrama de clase servidor

Diagrama De Clase Calificar Documentación(DocumentationCheck)

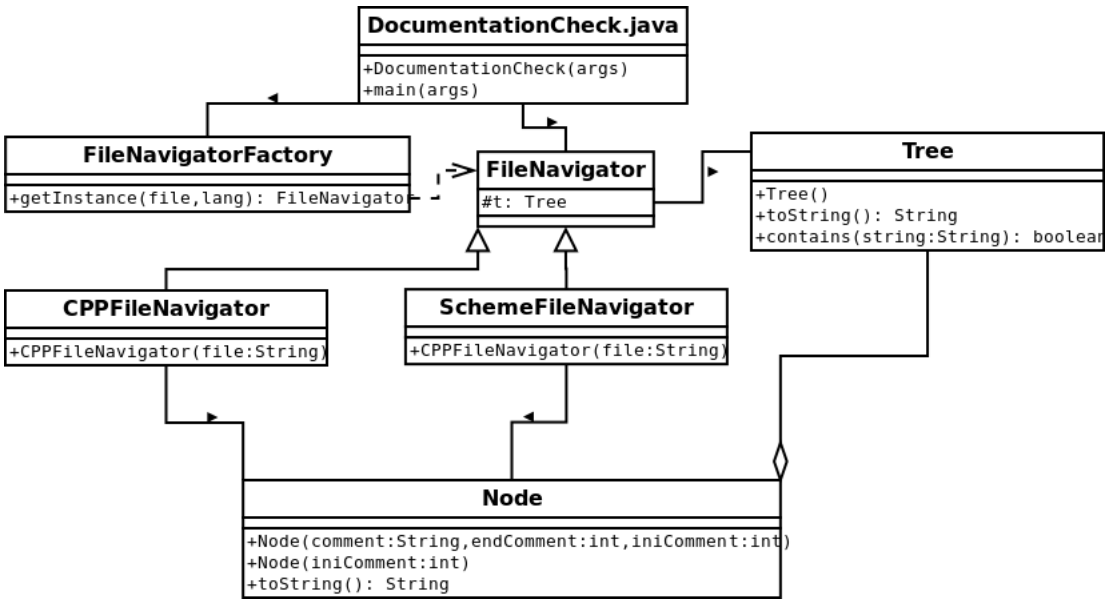

Figura 6: Diagrama de clase calificar documentación(DocumentationCheck)

Diagrama De Clase Cliente En Moodle

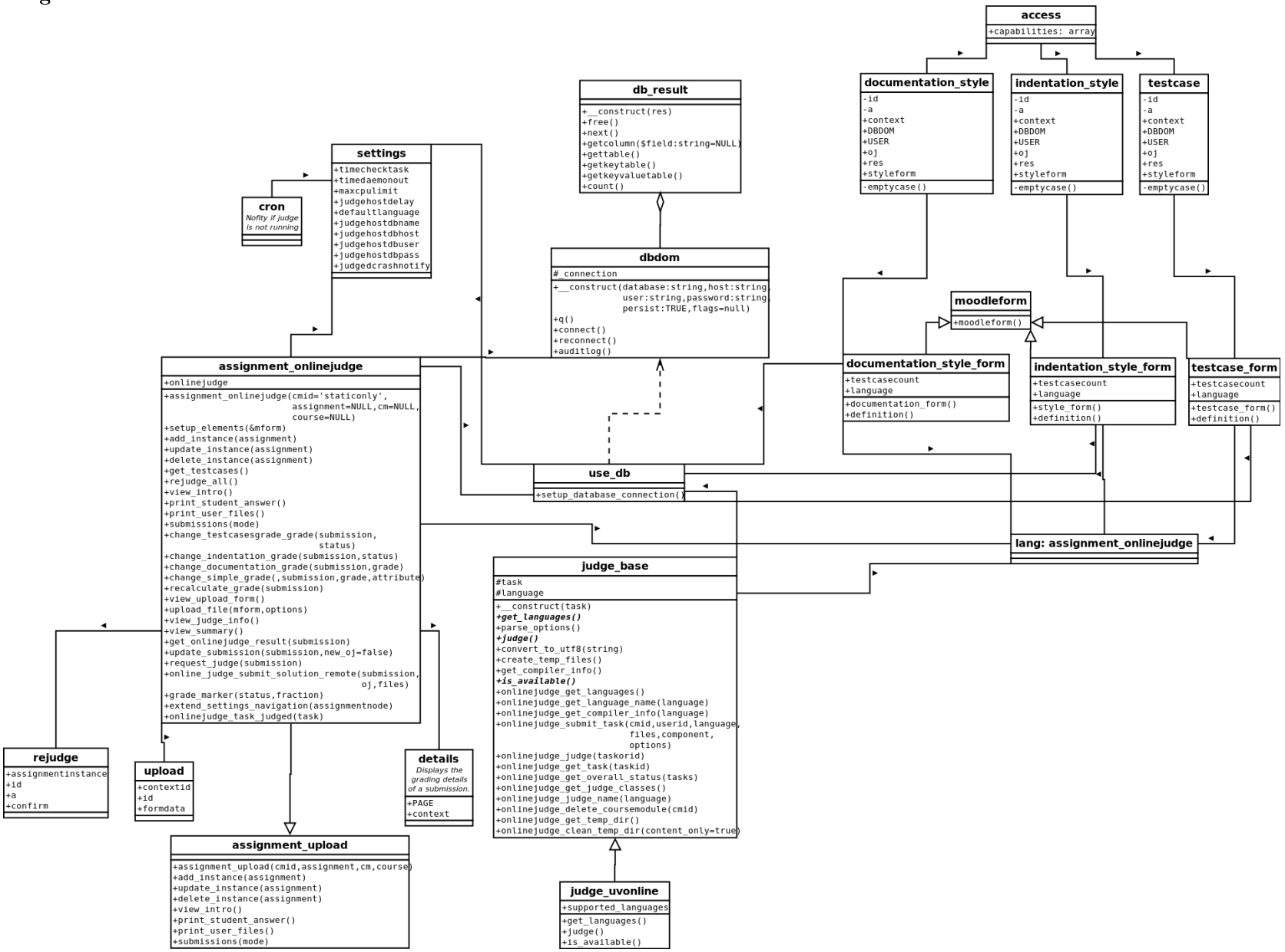

Figura 7: Diagrama de clase cliente en moodle

Diagrama De Clase Demonio Del Lado De Moodle

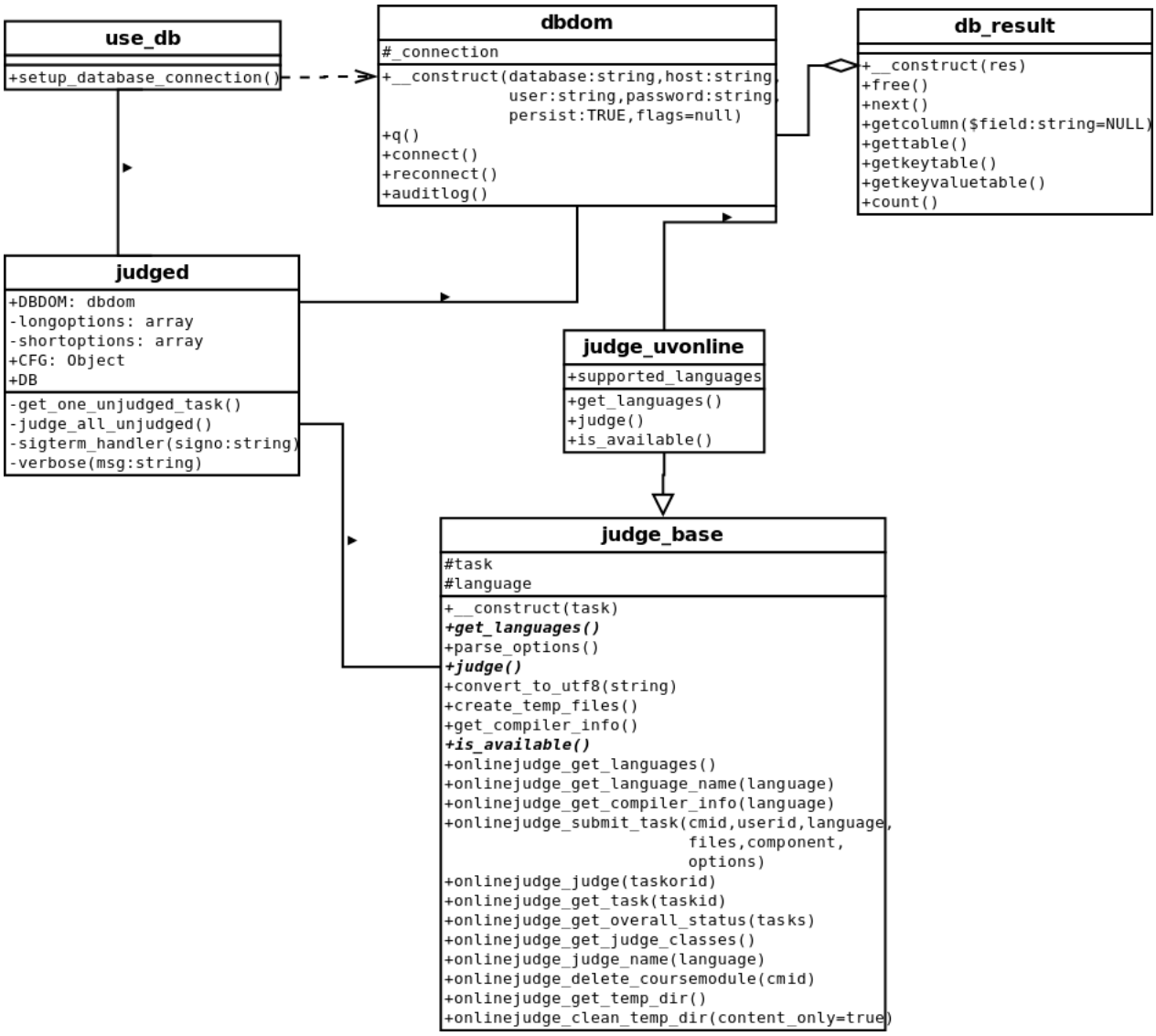

Figura 8: Diagrama de clase demonio del lado de moodle

5.1.4. Diagramas De Bases De Datos

Diagrama De La Base De Datos Del Servidor Judgehost

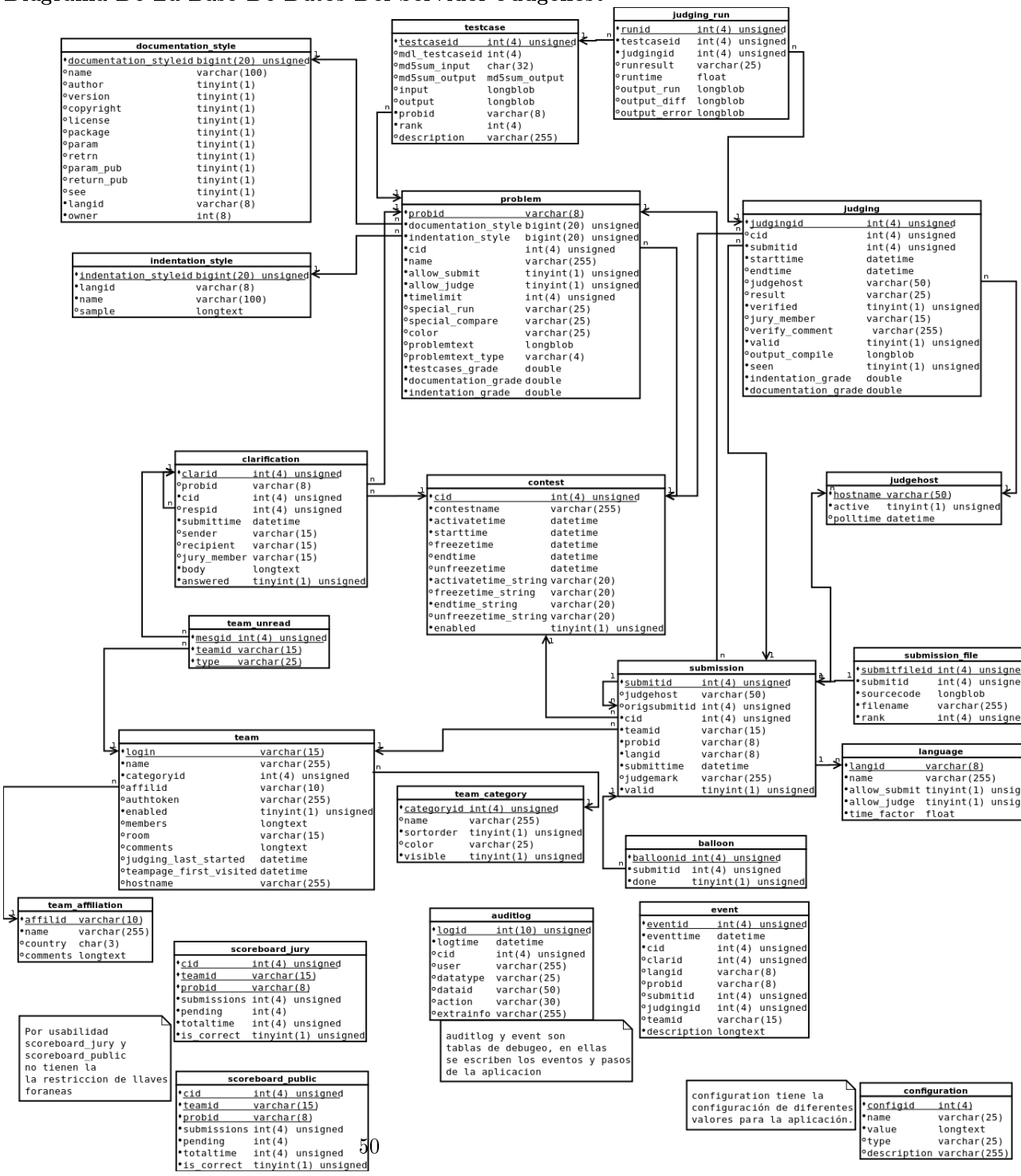

Figura 9: Diagrama de la base de datos del servidor judgehost

Diagrama De La Base De Datos Del Cliente En Moodle

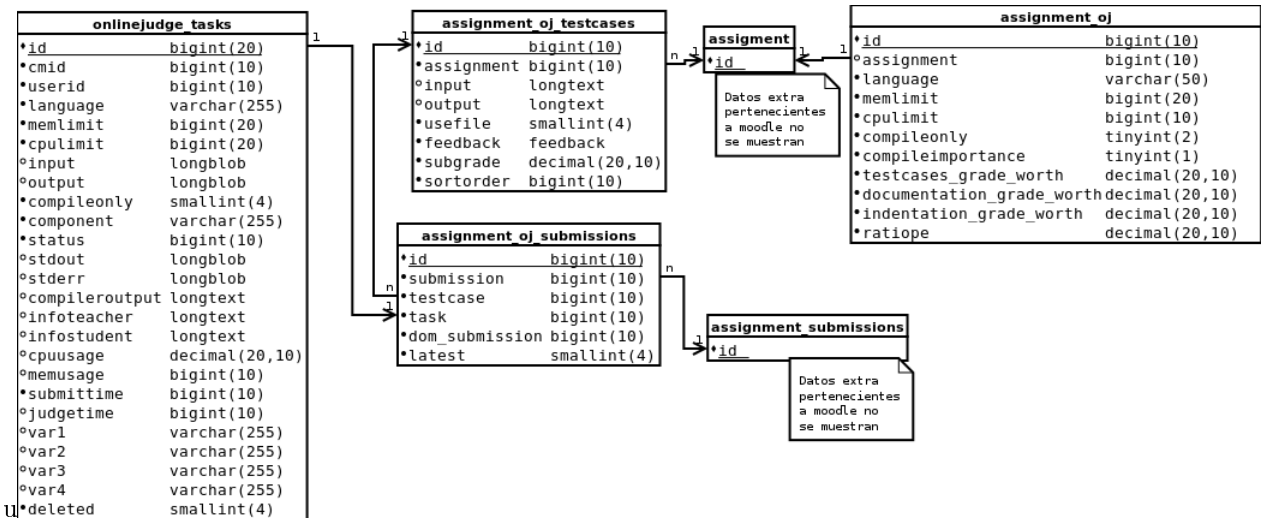

Figura 10: Diagrama de la base de datos del cliente en moodle

Entre los diagramas también cabria la arquitectura del sistema pero la misma ya es mostrada y explicada en la sección 2.2 en la página 17.

5.1.5. Pruebas Unitarias

Se listan las pruebas unitarias implementadas en este proyecto:

Número de Prueba Unitaria: PU1	**Titulo:** Test del generador	**Prerrequisitos:** Se recomienda localizar la base de datos de domjudge en el mismo equipo que se ejecutan las pruebas.(Se recomienda paciencia, los tests son lentos).
Descripción de lo que se Probará: Para simplificar la cantidad necesaria de código para los variados y diferentes tests de la aplicación se realiza un generador que simplifica pasos como generar assigments(tareas de programación) o configurar estilos. Esta prueba unitaria se utiliza para probar este modulo.		
Archivo: MOODLE_PATH/local/onlinejudge/tests/generator_test.php		**Anotaciones:**

Número de Prueba Unitaria:PU2, PU3, PU4.	**Titulo:** Test de envió aceptado.	**Prerrequisitos:** Se recomienda localizar la base de datos de domjudge en el mismo equipo que se ejecutan las pruebas.(Se recomienda paciencia, los tests son lentos).
Descripción de lo que se Probará: Se prueba que se puedan enviar diferentes soluciones a un trabajo de programación y que estas vuelvan con una calificación de "accepted" -correcta-.		
Archivo: MOODLE_PATH/local/onlinejudge/tests/accepted_cpp_test.php, MOODLE_PATH/local/onlinejudge/tests/accepted_java_test.php, MOODLE_PATH/local/onlinejudge/tests/accepted_scheme_test.php		**Anotaciones:** Se utiliza -siguiendo las guías de implementación de moodle- una prueba por archivo. En cada archivo se prueban los resultados de los diferentes lenguajes.

Número de Prueba Unitaria: PU5, PU6.	**Titulo:** Test de la modificación de la calificación	**Prerrequisitos:** Se recomienda localizar la base de datos de domjudge en el mismo equipo que se ejecutan las pruebas.(Se recomienda paciencia, los tests son lentos).
Descripción de lo que se Probará: Se prueba los diferentes métodos disponibles que permiten cambiar la calificación de un envió luego de haber sido calificado. Esto incluye la calificación de la indentación, la documentación y los casos de prueba de la tarea de programación.		
Archivo: MOODLE_PATH/local/onlinejudge/tests/change_grades_test, MOODLE_PATH/local/onlinejudge/tests/change_grades_scheme_test		**Anotaciones:** Se utiliza -siguiendo las guías de implementación de moodle- una prueba por archivo. En cada archivo se prueban los resultados de los diferentes lenguajes.(Solo dos archivos porque cambiar las notas en diferentes lenguajes siempre involucra al mismo código).

Número de Prueba Unitaria: PU7, PU8, PU9.	**Titulo:** Test de error de compilación	**Prerrequisitos:** Se recomienda localizar la base de datos de domjudge en el mismo equipo que se ejecutan las pruebas.(Se recomienda paciencia, los tests son lentos).
Descripción de lo que se Probará: Descripción de lo que se Probará: Se prueba que se pueda enviar diferentes soluciones a un trabajo de programación y que estas vuelvan con una calificación de "compilation-error" o que indica que ha habido un error en la compilación.		
Archivo: MOODLE_PATH/local/onlinejudge/tests/compilation_error_cpp.php, MOODLE_PATH/local/onlinejudge/tests/compilation_error_java.php, MOODLE_PATH/local/onlinejudge/tests/compilation_error_scheme.php		**Anotaciones:** Ya que se puede configurar para que el resultado de la configuración decida si se suma o no se suma la calificación de la indentación y la documentación a la nota total se tienen en cuenta los dos casos con dos trabajos de programación uno en el que se tiene en cuenta y otro en el que no. Debido a que scheme es interpretado solo se puede generar un runtime error o error en tiempo de ejecución. Se utiliza -siguiendo las guías de implementación de moodle- una prueba por archivo. En cada archivo se prueban los resultados de los diferentes lenguajes.

Número de Prueba Unitaria: PU10, PU11, PU12.	**Titulo:** Test de calificación de documentación	**Prerrequisitos:** Se recomienda localizar la base de datos de domjudge en el mismo equipo que se ejecutan las pruebas.(Se recomienda paciencia, los tests son lentos).
Descripción de lo que se Probará: Se prueba que al enviar diferentes soluciones a un trabajo de programación se reciban diferentes calificaciones dependiendo de que tan buena o mala sea la documentación del archivo. Se toman 3 niveles uno de calificación 0 otro de calificación media y otro de calificación perfecta. Además se tiene en cuenta la calificación de archivos que no compilan o que no son del lenguaje de programación (por ejemplo un archivo txt)para probar que los mismos se tienen en cuenta y no generan ciclos infinitos o fallos catastróficos.		
Archivo: MOODLE_PATH/local/onlinejudge/tests/documentation_cpp_test.php, MOODLE_PATH/local/onlinejudge/tests/documentation_java_test.php, MOODLE_PATH/local/onlinejudge/tests/documentation_scheme_test.php		**Anotaciones:** Se utiliza -siguiendo las guías de implementación de moodle- una prueba por archivo. En cada archivo se prueban los resultados de los diferentes lenguajes.

Número de Prueba Unitaria: PU13, PU14, PU15.	**Titulo:** Test de calificación de indentación	**Prerrequisitos:** Se recomienda localizar la base de datos de domjudge en el mismo equipo que se ejecutan las pruebas.(Se recomienda paciencia, los tests son lentos).
Descripción de lo que se Probará: Se prueba que al enviar diferentes soluciones a un trabajo de programación se reciban diferentes calificaciones dependiendo de que tan buena o mala sea la indentación del archivo. Se toman 3 niveles uno de calificación 0 otro de calificación media y otro de calificación perfecta. Además se tiene en cuenta la calificación de archivos que no compilan o que no son del lenguaje de programación (por ejemplo un archivo txt) para probar que los mismos se tienen en cuenta y no generan ciclos infinitos o fallos catastróficos.		
Archivo: MOODLE_PATH/local/onlinejudge/tests/indentation_cpp_test.php, MOODLE_PATH/local/onlinejudge/tests/indentation_java_test.php, MOODLE_PATH/local/onlinejudge/tests/indentation_scheme_test.php		**Anotaciones:** Se utiliza -siguiendo las guías de implementación de moodle- una prueba por archivo. En cada archivo se prueban los resultados de los diferentes lenguajes.

Número de Prueba Unitaria: PU16, PU17, PU18.	**Titulo:** Test de error de cantidad de memoria excedida	**Prerrequisitos:** Se recomienda localizar la base de datos de domjudge en el mismo equipo que se ejecutan las pruebas.(Se recomienda paciencia, los tests son lentos).
Descripción de lo que se Probará: Se prueba que se pueda enviar diferentes soluciones a un trabajo de programación y que estas vuelvan con una calificación de "memory_limit_exceed" que indica que el nivel de memoria usado en este envió es mayor a el configurado.		
Archivo: MOODLE_PATH/local/onlinejudge/tests/memory_limit_cpp_test.php, MOODLE_PATH/local/onlinejudge/tests/memory_limit_java_test.php, MOODLE_PATH/local/onlinejudge/tests/memory_limit_scheme_test.php		**Anotaciones:** La cantidad de memoria desbordada es primero reconocida por el interpretador del archivo por lo que el resultado presentado en Java y en Scheme en error en tiempo de ejecución pero en "Salida estándar de errores" por caso de prueba se muestra claramente que es un exceso de memoria. En c++ esta prueba en algunos causa que el demonio domjudge judgehost termine inesperadamente. Se utiliza -siguiendo las guías de implementación de moodle- una prueba por archivo. En cada archivo se prueban los resultados de los diferentes lenguajes.

Número de Prueba Unitaria: PU19, PU20, PU21.	**Titulo:** Test de error de cantidad de output excedida	**Prerrequisitos:** Se recomienda localizar la base de datos de domjudge en el mismo equipo que se ejecutan las pruebas.(Se recomienda paciencia, los tests son lentos).
Descripción de lo que se Probará: Se prueba que se pueda enviar diferentes soluciones a un trabajo de programación y que estas vuelvan con una calificación de "output_limit_exceed" que indica que el nivel de salida de texto generado en este envió es mayor a el configurado como limite para el demonio.		
Archivo: MOODLE_PATH/local/onlinejudge/tests/output_limit_cpp_test.php, MOODLE_PATH/local/onlinejudge/tests/output_limit_java_test.php, MOODLE_PATH/local/onlinejudge/tests/output_limit_scheme_test.php		**Anotaciones:** La ejecución de estas pruebas muestra que este estado es demasiado improbable para alcanzar(por lo que no se ejecuta con las demás pruebas) ya que antes de que se consiga el output se consigue un error por memoria o por tiempo, incluso utilizando las formas más eficientes de impresión del lenguaje y subiendo el tiempo a 1 minuto por caso de prueba. Se utiliza -siguiendo las guías de implementación de moodle- una prueba por archivo. En cada archivo se prueban los resultados de los diferentes lenguajes.

Número de Prueba Unitaria: PU22, PU23, PU24.	**Titulo:** Test de error de presentación.	**Prerrequisitos:** Se recomienda localizar la base de datos de domjudge en el mismo equipo que se ejecutan las pruebas.(Se recomienda paciencia, los tests son lentos).
Descripción de lo que se Probará: Se prueba que se puedan enviar diferentes soluciones a un trabajo de programación y que estas vuelvan con una calificación de "presentation-error". Esto significa que la salida es "correcta" pero tiene indentadores de más.		
Archivo: MOODLE_PATH/local/onlinejudge/tests/presentation_error_cpp_test.php, MOODLE_PATH/local/onlinejudge/tests/presentation_error_java_test.php , MOODLE_PATH/local/onlinejudge/tests/presentation_error_scheme_test.php		**Anotaciones:** Se utiliza -siguiendo las guías de implementación de moodle- una prueba por archivo. En cada archivo se prueban los resultados de los diferentes lenguajes.

Número de Prueba Unitaria: PU25, PU26, PU27.	**Titulo:** Test de error en tiempo de ejecución.	**Prerrequisitos:** Se recomienda localizar la base de datos de domjudge en el mismo equipo que se ejecutan las pruebas.(Se recomienda paciencia, los tests son lentos).
Descripción de lo que se Probará: Se prueba que se puedan enviar diferentes soluciones a un trabajo de programación y que estas vuelvan con una calificación de "runtime-error" o un error en tiempo de ejecución.		
Archivo: MOODLE_PATH/local/onlinejudge/tests/runtime_error_cpp_test.php, MOODLE_PATH/local/onlinejudge/tests/runtime_error_java_test.php, MOODLE_PATH/local/onlinejudge/tests/runtime_error_scheme_test.php		**Anotaciones:** Se utiliza -siguiendo las guías de implementación de moodle- una prueba por archivo. En cada archivo se prueban los resultados de los diferentes lenguajes.

Número de Prueba Unitaria: PU28, PU29, PU30.	**Titulo:** Test de la implementación de la seguridad del demonio judgehost	**Prerrequisitos:** Se recomienda localizar la base de datos de domjudge en el mismo equipo que se ejecutan las pruebas.(Se recomienda paciencia, los tests son lentos).
Descripción de lo que se Probará: Se prueba que los envíos a un trabajo de programación no tienen la capacidad de acceder a archivos importantes para el sistema y genere un error en tiempo de ejecución o los encuentre vacíos (dependiendo del lenguaje).		
Archivo: MOODLE_PATH/local/onlinejudge/tests/security_cpp_test.php, MOODLE_PATH/local/onlinejudge/tests/security_java_test.php, MOODLE_PATH/local/onlinejudge/tests/security_scheme_test.php		**Anotaciones:** Se utiliza -siguiendo las guías de implementación de moodle- una prueba por archivo. En cada archivo se prueban los resultados de los diferentes lenguajes.

Número de Prueba Unitaria: PU31, PU32, PU33.	**Titulo:** Test de error de cantidad de tiempo excedida	**Prerrequisitos:** Se recomienda localizar la base de datos de domjudge en el mismo equipo que se ejecutan las pruebas.(Se recomienda paciencia, los tests son lentos).
Descripción de lo que se Probará: Se prueba que se puedan enviar diferentes soluciones a un trabajo de programación y que estas vuelvan con una calificación de "time-limit-exceed" o lo que significa que el tiempo de ejecución del envió ha superado el tiempo configurado como limite para el trabajo de programación. En el momento en que la condición anterior se presenta el sistema termina el programa para no entrar en un ciclo infinito.		
Archivo: MOODLE_PATH/local/onlinejudge/tests/time_limit_cpp_test.php, MOODLE_PATH/local/onlinejudge/tests/time_limit_java_test.php, MOODLE_PATH/local/onlinejudge/tests/time_limit_scheme_test.php		**Anotaciones:** Se utiliza -siguiendo las guías de implementación de moodle- una prueba por archivo. En cada archivo se prueban los resultados de los diferentes lenguajes.

Número de Prueba Unitaria: PU34, PU35, PU36.	**Titulo:** Test de respuesta incorrecta.	**Prerrequisitos:** Se recomienda localizar la base de datos de domjudge en el mismo equipo que se ejecutan las pruebas.(Se recomienda paciencia, los tests son lentos).
Descripción de lo que se Probará: Se prueba que se puedan enviar diferentes soluciones a un trabajo de programación y que estas vuelvan con una calificación de "wrong-answer" -respuesta incorrecta-. La respuesta del envió no es la esperada.		
Archivo: MOODLE_PATH/local/onlinejudge/tests/wrong_answer_cpp_test.php, MOODLE_PATH/local/onlinejudge/tests/wrong_answer_java_test.php, MOODLE_PATH/local/onlinejudge/tests/wrong_answer_scheme_test.php		**Anotaciones:** Se utiliza -siguiendo las guías de implementación de moodle- una prueba por archivo. En cada archivo se prueban los resultados de los diferentes lenguajes.

Número de Prueba Unitaria: PU37.	**Titulo:** Test de porcentajes 0	**Prerrequisitos:** Se recomienda localizar la base de datos de domjudge en el mismo equipo que se ejecutan las pruebas.(Se recomienda paciencia, los tests son lentos).
Descripción de lo que se Probará: Se prueba un caso limite en el que todos los porcentajes configurados en el trabajo de programación están puestos en 0 esto para asegurar el comportamiento esperado de la aplicación incluso en casos extremos.		
Archivo: MOODLE_PATH/local/onlinejudge/tests/zero_percent_calification_test.php		**Anotaciones:** No se incluye un archivo por cada lenguaje porque la parte que calcula la nota final no cambia con el lenguaje.

5.2. Instalación

Para facilidad o en caso de problemas se entrega como recurso la imagen de maquina virtual InstalaciónFuncionalAplicacionCalificacionDeTareas.ova en el dvd entregado que tiene instalada el software desarrollado en este proyecto, para utilizarla en Virtualbox siga las instrucciones de esta página: http://syconet.wordpress.com/2012/11/05/importar-y-exportar-maquinas-virtuales-con-virtualbox/.

Los datos para su acceso son:

Usuario de la maquina virtual: invitado

Password para el usuario: invitado

Para el acceso desde un navegador en la maquina virtual ingrese a http://localhost/~invitado/moodle/

Y en el acceso a moodle:

Usuario de la maquina moodle: admin

Password para el usuario en moodle: Administrador-1

Figura 11: Login Moodle

La instalación de domjudge esta en /home/invitado/public_html/domjudge y la moodle onlinejudge en /home/invitado/public_html/moodle/local/onlinejudge:

```
invitado@invitado-VirtualBox: ~/public_html/domjudge/judgehost
invitado@invitado-VirtualBox:~$ cd /home/invitado/public_html/domjudge/judgehost/
invitado@invitado-VirtualBox:~/public_html/domjudge/judgehost$
invitado@invitado-VirtualBox:~/public_html/domjudge/judgehost$ bin/judgedaemon
```

Para correr los dos demonios necesarios para la calificación de tareas (después de cualquier reinicio) simplemente abra una terminal (Nota: esto solo esta configurado en la imagen entregada en una instalación diferente considere poner estos demonios como servicios).

Para hacer una instalación desde cero se deben seguir los siguientes pasos:

5.2.1. Servidor

Para la instalación del servidor es necesario el sistema operativo Linux. Ejecute los siguientes pasos en el computador en el cual desee ejecutar el demonio que ejecuta los envíos de los estudiantes:

Instale Emacs, Python e indent para que puedan ser ejecutables desde la linea de comandos:

$apt-get install emacs python3 indent racket

Instale la versión modificada de domjudge[59] presente en el dvd (aplicación/domjudge-3.4.2):

$apt-get install make sudo php5-cli php5-mysql php5-json ntp xsltproc procps gcc g++ gcj openjdk-6-jre-headless openjdk-6-jdk ghc fp-compiler libboost1.48-all-dev

Copie y pegue la carpeta aplicación/domjudge-3.4.2 a una ubicación en su disco y diríjase a la misma luego:

$make clean

$./configure [–enable-fhs] –prefix=<basepath>.

$make judgehost domserver

$make install-judgehost install-domserver

Luego proseguimos con la **Instalación de la base de datos**

DOMjudge utiliza un servidor de base de datos MySQL para el almacenamiento de información.

La estructura de base de datos y los privilegios se incluyen en los archivos de volcado de MySQL en el subdirectorio sql. El nombre de la base de datos por defecto es domjudge. Esto se puede cambiar manualmente en el archivo <basepath>/domserver/etc/dbpasswords.secret: el nombre de la base de datos como se especifica en este archivo se utilizará durante la instalación.

La instalación de la base de datos se realiza con <basepath>/domserver/bin/dj-setup-database. Para esto, es necesario un servidor MySQL y un administrador de acceso instalado y configurado.

Corremos:

$dj-setup-database genpass

$dj-setup-database [-u <admin_user>] [<contraseña> -p | -r] install

ejemplo; $bin/dj-setup-database -u root -r install

El primer comando crea las credenciales de base de datos DOMjudge archivo etc/dbpasswords.secret (opcionalmente se puede cambiar la contraseña generada al azar, aunque no es necesario para el funcionamiento normal). Luego se crea la base de datos y los usuarios e inserta algunos datos default/ejemplo en la base de datos domjudge. La opción -r le pedirá una contraseña para mysql; cuando

no se especifica ningún usuario, el cliente mysql intentará leer las credenciales de $HOME/.my.cnf como de costumbre.El comando uninstall se puede pasar a dj-setup-database para eliminar la base de datos DOMjudge y sus usuarios; Esto elimina todos los datos![59].

Para ejecutar los envíos a las tareas bajo un usuario sin privilegios, este usuario tiene que ser añadido al(os) sistema(s) que actúan como judgehost. Este usuario no tiene un directorio de inicio o contraseña, por lo que el siguiente comando sería suficiente para añadir un usuario 'domjudge-run' con privilegios mínimos.

En RedHat:

$useradd -d /nonexistent -g nobody -M -n -s /bin/false domjudge-run

En Debian:

$useradd -d /nonexistent -g nogroup -s /bin/false domjudge-run

Para otros sistemas verifique los detalles de su comando useradd. Este usuario también debe estar configurado como el usuario con el que los programas se ejecutan a través de $configure –enable-runuser=USER; el valor predeterminado es domjudge-run.

Runguard un programa que sirve para ejecutar y compilar los envíos tiene que ser capaz de tener privilegios de root para ciertas operaciones, como cambiar el runuser o creando un chroot. Además, la secuencia de comandos chroot-startstop.sh utiliza sudo para obtener privilegios en ciertas operaciones. Hay un archivo /etc/sudoers.d/ en etc/sudoers-domjudge que previamente contiene las reglas necesarias(aunque pueden cambiar en cada sistema). Usted puede poner las líneas del archivo al final de /etc/sudoers, o para las versiones modernas de sudo, colocar el archivo en /etc/sudoers.d/. Si usted cambia el usuario que ejecuta el judgehost, o las rutas de instalación, asegúrese de actualizar las normas sudoers en consecuencia. Tenga muy en cuenta esta configuración ya que mucho de la confiabilidad del sistema depende de la buena configuración de estos permisos.

Cuando el ajuste de chroot está habilitado, un shell POSIX estático tiene que estar disponible para copiarlo en el entorno chroot. Para Linux i386, se incluye un shell Dash estático, que viene preconfigurado. Para otras arquitecturas o sistemas operativos, un shell se tiene que agregar manualmente. Entonces sólo tiene que apuntar el lib/symlink-sh estático a este archivo. Si desea utilizar idiomas que no se pueden compilar y generar binarios enlazados estáticamente, por ejemplo, compilante a byte como Java o lenguajes interpretados como Python, entonces un entorno chroot completo debe ser construido y configurado. Véase http://www.domjudge.org/docs/admin-manual-8.html#problems:java-chroot para más detalles.

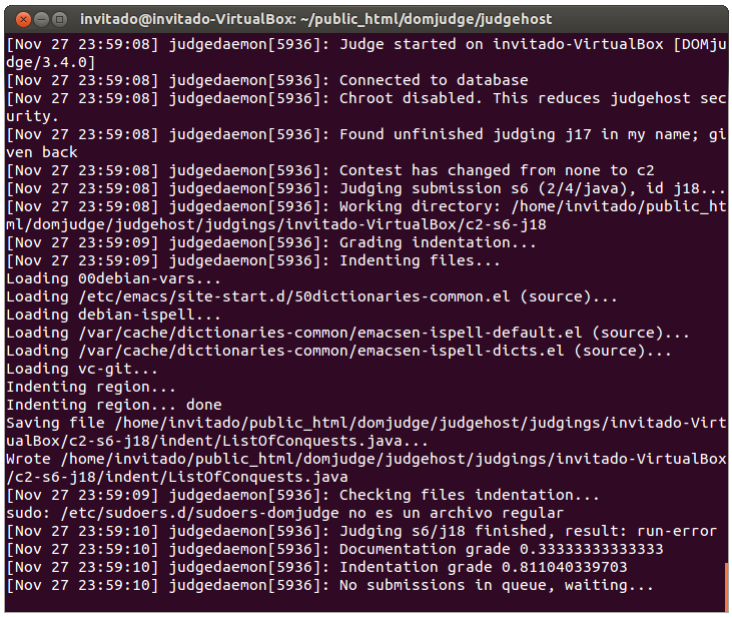

Figura 12: Ejecución demonio servidor

Al inicio, el judgehost se conectará a la domserver y agregará una entrada para sí mismo en la tabla de judgehosts, como activado por defecto.

Si hay problemas o necesita más información sobre la instalación de domjudge-judgehost el proceso de instalación exactamente el mismo que Domjudge[59] por lo cual puede buscar en foros sobre información de su instalación o remítase directamente a http://www.domjudge.org/docs/admin-manual-3.html.

Para probar que la instalación correcta correr:

$php <basepath>/judgehost/bin/judgedaemon

Para la calificación de las tareas el demonio (figura en la página 63, Ejecución demonio servidor) que corre con el comando :

$php <basepath>/judgehost/bin/judgedaemon

Se debe estar ejecutando por lo que se podría dejar corriendo cada vez que se inicie la maquina o ejecutarlo como un demonio.

5.2.2. Cliente

Requisitos Previos

En Linux

Moodle 2.0 o superior

php-cli

(opcional pero recomendable) make, gcc y g+ +.

(opcional pero recomendado) PCNTL y extensión POSIX para php-cli

En Windows

Moodle 2.0 o superior

php-cli

Instalación Cliente

MOODLE_PATH significa la ruta raíz de la instalación de moodle.

Para una instalación limpia en Moodle 2.3 o 2.4 (en lugar de un sitio actualizado) :

Es necesario habilitar el tipo de asignación 2.2 en la administración:

Administración del sitio > Plugins > módulos de actividad > Administrar actividades

En Linux

Si el directorio existe MOODLE_PATH/local/onlinejudge , elimínelo.

Asegúrese de que el nombre del directorio de este plugin es onlinejudge.

Si no es así , cambie su nombre.

Copie el directorio aplicación/onlinejudge presente en el cd a MOODLE_PATH/local/

Ejecute MOODLE_PATH/local/onlinejudge/cli/install_assignment_type.

Entre su sitio moodle como admin /admin/index.php para que se puedan instalar/actualizar.

Desde la consola ejecute;

$sudo -u www-data php MOODLE_PATH/local/onlinejudge/cli/judged.php

Para lanzar el demonio. El -u puede cambiar dependiendo del sistema operativo.

En Windows

Si existe la carpeta MOODLE_PATH\local\onlinejudge elimínela.

Asegúrese de que el nombre de la carpeta de este plugin es onlinejudge si no es así , cambie su nombre.

Copie la carpeta aplicación/onlinejudge presente en el cd a MOODLE_PATH\local\

En la carpeta MOODLE_PATH\local\onlinejudge\cli y ejecute install_assignment_type.bat.

Entre a su sitio moodle como admin /admin/index.php para terminar de instalar el plugin.

Desde la consola ejecute;

$php.exe MOODLE_PATH/local/onlinejudge/cli/judged.php -v

Para lanzar el demonio. El -u puede cambiar dependiendo del sistema operativo.

Configuración general

Asegúrese de configurar correctamente los datos de acceso a la base de datos de domjudge utilizando los datos escritos en <basepath>/domserver/etc/dbpasswords.secret y recuerde que algunos de estos datos pueden perjudicar la confiabilidad del servidor judgehost. Cada apartado de la configuración tiene comentarios suficientes para entender su utilidad:

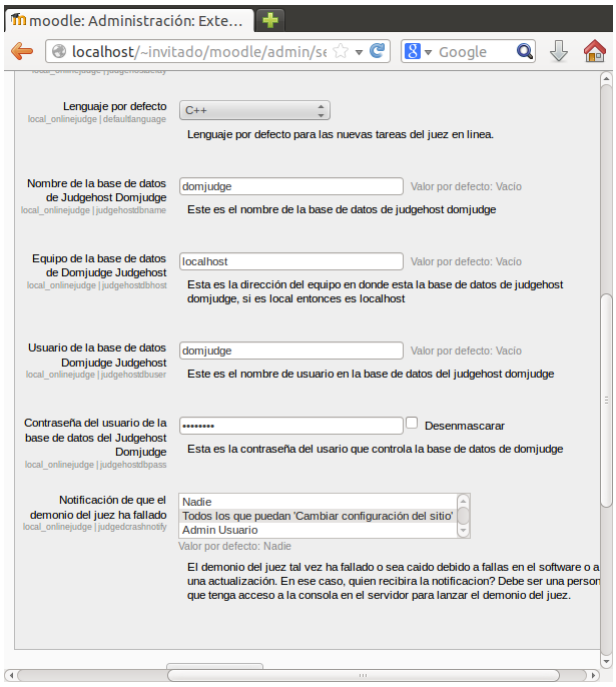

Figura 13: Ajustes de moodle

Recuerde que estos datos pueden ajustarse desde /Administración del sitio/Extensiones/Extensiones locales/Juez Online.

5.2.3. Configuración De Lenguajes

Para instalar un nuevo lenguaje usted tiene que ingresarlo en la tabla language de la base de datos de domjudge (en la sección 5.1.4 en la página 50) con los registros allow_submit y allow_judge en 1(o verdadero), además tiene que haber por lo menos una entrada en las tablas documentation_style e indentation_style que tenga en su registro langid a el lenguaje a añadir.

Además debe de existir un archivo .el del lado del servidor como se ve en la sección 5.1.3 en la página 46 en el diagrama de clases para cpp.el o java.el este archivo tiene que llamarse <langid>.el y ser capaz de indentar el código tomándolo como buffer de emacs un ejemplo de este archivo que serviría para varios lenguajes aunque no con muy buenos resultados sería:

Este archivo iría en la ubicación: <basepath>/judgehost/lib/indentation/

Algoritmo 7 Script de indentación lenguaje nuevo

```
(defun default ()
   "Format the whole buffer."
   (untabify (point-min) (point-max))
   (indent-region (point-min) (point-max) nil)
   (save-buffer)
)
```

Luego viene la parte de la compilación y la ejecución: La configuración de los compiladores de los idiomas soportados se debe hacer por separado. Para cada idioma soportado un shell-script llamado compile_<langid>.sh debe ser creado y colocado en lib/judge en los judgehosts, donde <langid> es el ID del lenguaje como se especifica en la base de datos. Para obtener más información, véase por ejemplo compile_c.sh y compile.sh en lib/judge para revisar la sintaxis. Tenga en cuenta que se incluyen desde hace ya los idiomas más comunes.

Los lenguajes interpretados y los binarios no enlazados estáticamente se pueden, en principio, también utilizar, pero requiere que todas las dependencias se añaden al entorno chroot.

Los lenguajes interpretados no generan un ejecutable y, en principio, no es necesario un paso de compilación. Sin embargo, para poder utilizar los lenguajes interpretados (también Java de Oracle), un script se debe generar durante la etapa de compilación, que funcionará como el ejecutable: el script ha de ejecutar el intérprete de la fuente. Ver compile_perl.sh y compile_java_javac.sh o compile_scheme.sh en lib/judge.

DOMjudge soporta el lenguaje de Oracle (Sun) Java dentro de un entorno chroot. Para ello primero se debe construir un entorno chroot que incluya las bibliotecas de Java. Esto se puede lograr con el dj_make_chroot script incluido: ejecutar como root y pasar como argumentos el directorio de destino para construir en el entorno chroot y, como segundo argumento de la arquitectura de la máquina de destino. Inicie el script sin argumentos para la información de uso. Ver también las secciones de instalación de un judgehost y Problemas: Java & chroot.

5.2.4. Instalación de la suite de pruebas

Si se desea ejecutar las diferentes pruebas unitarias que vienen desarrolladas con este proyecto primero hay que realizar unos cuantos pasos más de instalación. Primero hacemos la instalación de la suite de pruebas phpunit[74] de moodle[75];

Para la **instalación de la suite de pruebas en moodle** instalamos el gestor de dependencias de composer:

$cd MOODLE_PATH

$curl -s https://getcomposer.org/installer | php

En Windows vas a la url https://getcomposer.org/download/ y descargas el archivo Composer-Setup.exe.

Ejecutas el instalador de composer:

$cd MOODLE_PATH

$php composer.phar install –dev

(Si eso te da problemas de conexión intente ...)

$php composer.phar install –dev –prefer-source

Nuestra integración PHPUnit requiere una base de datos dedicada y un dataroot. En primer lugar, añada un nuevo directorio dataroot y prefijo en su config.php (usted puede encontrar ejemplos en http://docs.moodle.org/dev/PHPUnit#Common_Unit_Test_Prol config-dist.php)

Ejemplo:

"$CFG->phpunit_prefix = 'phpu_';

$CFG->phpunit_dataroot = '/home/example/phpu_moodledata';"

Para inicializar el entorno de pruebas se usa el siguiente comando:

$cd MOODLE_PATH

$php admin/tool/phpunit/cli/init.php

Este comando tiene que ser repetido después de cualquier actualización, si se va a instalar o desinstalar un plugin o si se añaden más tests.

En caso de problemas vaya a http://docs.moodle.org/dev/PHPUnit#Installation_of_PHPUnit_via_Compose Troubleshooting

También es necesaria la **instalación de una base de datos de pruebas en el servidor de domjudge** judgehost. Primero diríjase a la carpeta en donde tiene instalado domjudge a la que nos referiremos como <basepath>.

Luego diríjase a la carpeta <basepath>/domserver/bin/ y ejecute:

$dj-setup-database [-u <admin_user>] [<contraseña> -p | -r] install-tests

ejemplo; $bin/dj-setup-database -u root -r install-tests

Para la ejecución de las pruebas unitarias diríjase a 6.4 en la página 83.

6. Manuales

Se presenta el manual de uso de las principales funcionalidades del proyecto desarrollado con tres apartados:

1. Creación de un trabajo de programación bajo la aplicación desarrollada desde moodle.
2. Envió de una solución a esta tarea desarrollada.
3. Calificación de una tarea encolada.

6.1. Creación De Un Trabajo De Programación Bajo La Aplicación Desarrollada Desde Moodle.

Para la creación de un trabajo de programación diríjase en un navegador a la dirección en donde tenga instalado el moodle que contiene el plugin desarrollado:

(En la imagen de la maquina virtual entregada http://localhost/~invitado/moodle/)

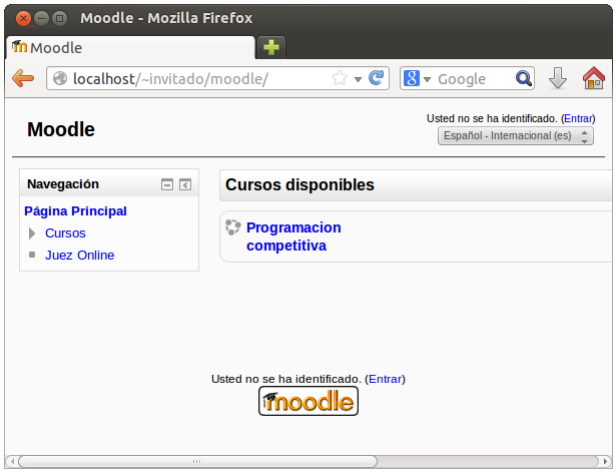

Figura 14: Pagina inicial moodle

Luego diríjase presionando el link “Entrar” diríjase a la página de login de la aplicación y entre las credenciales de un usuario que tenga la capacidad de editar un curso existente en el sistema moodle y presione enter para loguearse:

(En la imagen virtual login: “admin” password: “Administrador-1” para su simplicidad inicie el navegador firefox)

Figura 15: Login Moodle

Diríjase entonces al curso en el cual tiene permisos de edición o si se logueó como usuario administrador puede crear un curso. Los cursos disponibles deben aparecer cuando va de nuevo a la url raíz de su moodle .

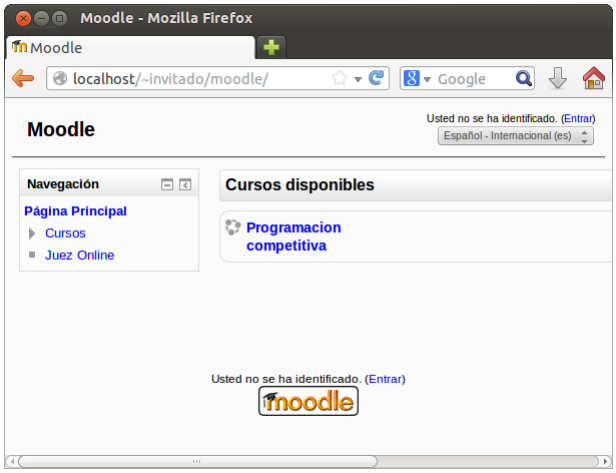

Figura 16: Cursos accesibles de Moodle

Figura 17: Activar edición Moodle

Presione el botón Activar Edición en la esquina derecha arriba del curso para poder añadir una nueva actividad (figura: Activar edición Moodle en la página 71)

Presione el link "Añadir una actividad o recurso" (figura: Añadir actividad en la página 71)

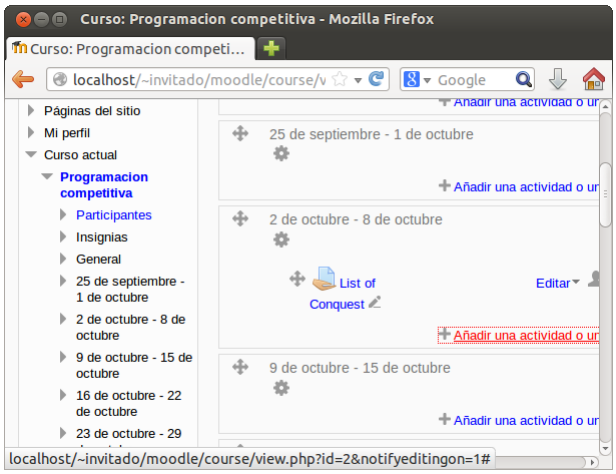

Figura 18: Añadir actividad

Luego en el menú que se despliega como pop-up busque bajo la actividad Tarea 2.2 la opción de "Juez Online" y escójala y pulse agregar (figura: Escogiendo

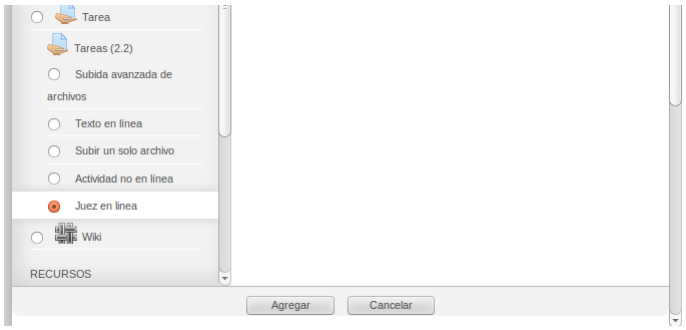

Figura 19: Escogiendo actividad Moodle

Figura 20: Editando actividad Moodle

actividad Moodle en la página 72).

Esto lo remitirá a una interfaz de entrada muy similar a la que es utilizada para crear una tarea normal de moodle (figura: Editando actividad Moodle en la página 72)

Pero incluye más opciones (figura: Editando actividad Moodle en la página 73):

- Escoger el lenguaje en que se quiere generar el trabajo de programación "Lenguaje de programación".
- Y escoger el tiempo máximo de ejecución que se permite para el envió en un caso de pruebas.

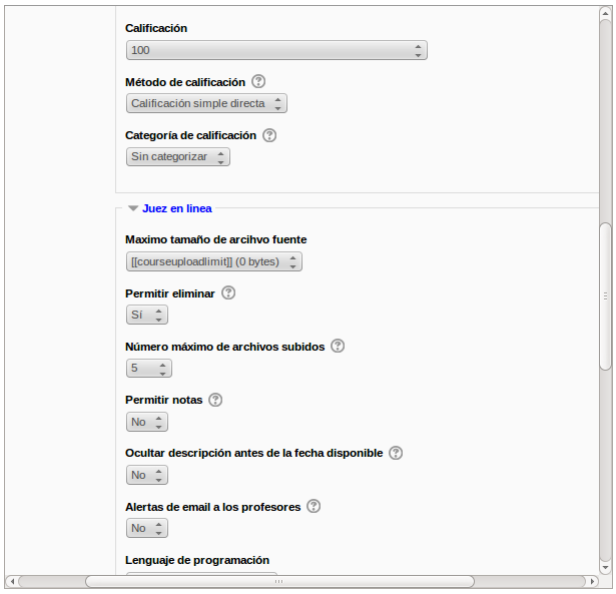

Figura 21: Editando actividad Moodle

Luego de escoger las opciones a su gusto presione el botón "Guardar cambios y regresar al curso" para así dejar creado el trabajo de programación.

Cuando el trabajo de programación está creado no se pueden realizar todavía envíos de soluciones hasta que no se hayan guardado casos de pruebas para el mismo (figura, Trabajo de programación sin casos de prueba, en la página 74)

Para guardar los casos de prueba de un trabajo de programación se entra al trabajo de programación y yendo al link "Manejar los casos de prueba" presente en su menú lateral (figura, Manejar casos de pruebas menú lateral Moodle, en la página 74).

Figura 22: Trabajo de programación sin casos de prueba

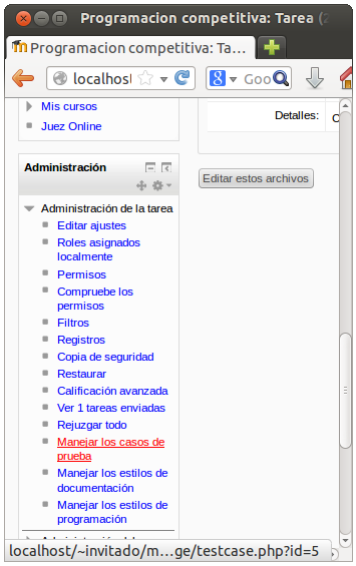

Figura 23: Manejar casos de pruebas menú lateral Moodle

Ahora en esta ventana cada caso de prueba es configurable con los siguientes apartados:

1. “Calificación” para que cada testcase tenga un porcentaje de la nota posible para los testcases.
2. “Entrada” una ventana para configurar las entradas para las soluciones.

3. “Salida” una ventana para configurar las salidas para las soluciones.

4. “Parte de la calificación otorgado por los casos de prueba” el porcentaje de la nota al que se le atribuyen todos los casos de pruebas.

5. “Añada 1 mas pruebas(s)” aumenta 1 caso de pruebas al formulario para que pueda ser llenado.

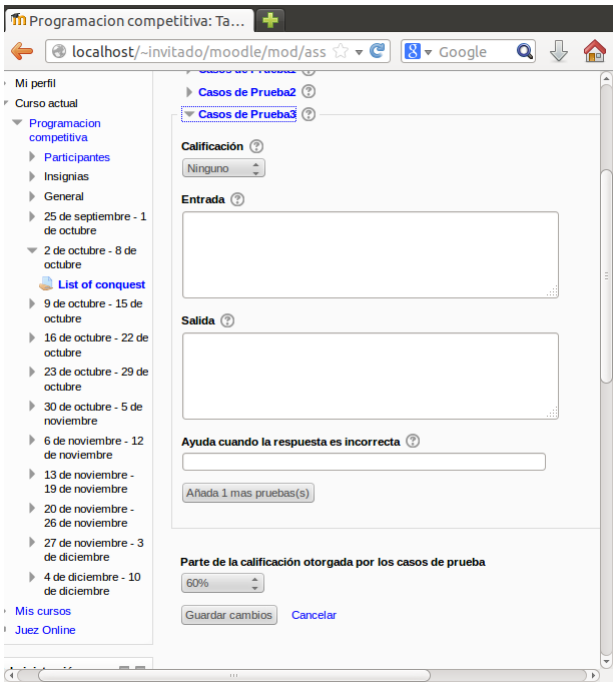

Figura 24: Casos de prueba de tarea

6.1.1. Configuración Indentación

En el mismo menú lateral se encuentra “Manejar los estilos de programación” que permite configurar con que estilo de indentación se quiere calificar un envió donde:

1. “Estilo de indentación para esta tarea” representa el nombre del estilo con el que se quiere calificar la indentación.

2. “Parte de la calificación otorgada por la indentación” representa el porcentaje de la nota configurable de la cantidad de la nota a dar a la solución.

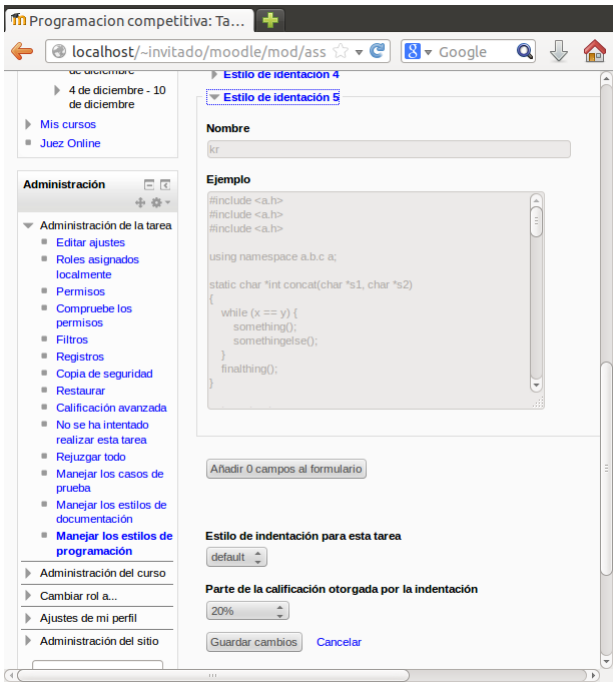

Figura 25: Configuración indentación de tarea

Cada "Ejemplo" tiene una muestra de código representativa que permite mostrar como se comporta el estilo en cada uno de estos casos.

6.1.2. Configuración Documentación

En el menú lateral se encuentra también "Manejar los estilos de documentación" que permite configurar con que estilo de documentación se quiere calificar un envió donde:

1. "Nombre" representa el nombre del estilo con el que se quiere calificar la indentación. Es cambiable y no es permitido tener más de un estilo con el mismo nombre para un solo lenguaje de programación
2. "Requiere la etiqueta @*" son las etiquetas que se van a pedir para la documentación.
3. "Parte de la calificación otorgada por la calificación" representa el porcentaje de la nota configurable de la cantidad de la nota a dar a la solución.

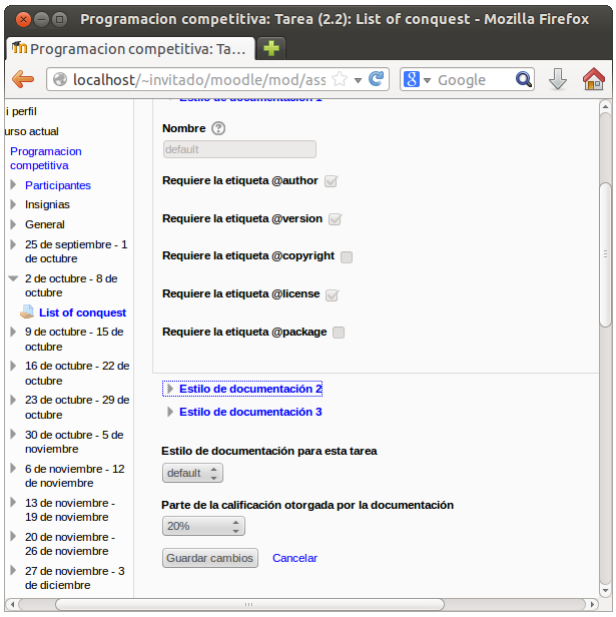

Figura 26: Configuración indentación de tarea

6.2. Envió De Una Solución A Esta Tarea Desarrollada.

Logueese como un usuario estudiante en alguno de los cursos en los que haya generado un trabajo de programación y diríjase a un curso en donde haya un trabajo de programación como se muestra en "6.1 Creación De Un Trabajo De Programación Bajo La Aplicación Desarrollada Desde Moodle" y de clic sobre el link con el nombre del trabajo de programación:

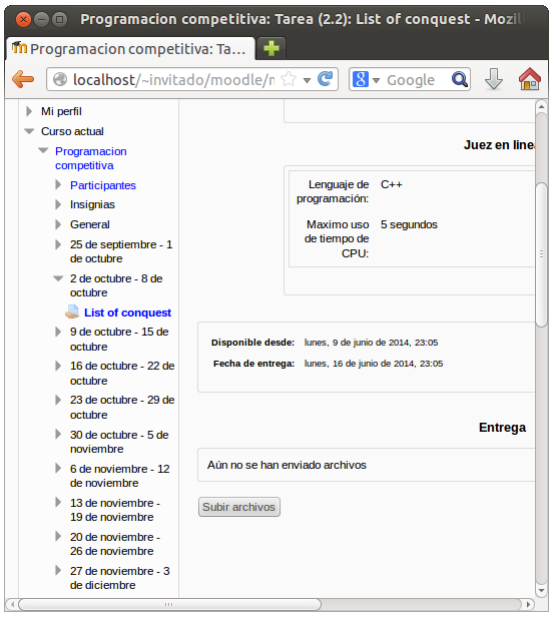

Figura 27: Envió De Trabajo De Programación Moodle

Para hacer un envió haga clic en "Subir archivos" y escoja los archivos (el sistema no acepta carpetas solo archivos de fuente) que quiere dar en su envió , luego al dar clic en guardar archivos se verá el envió del mismo en el sistema:

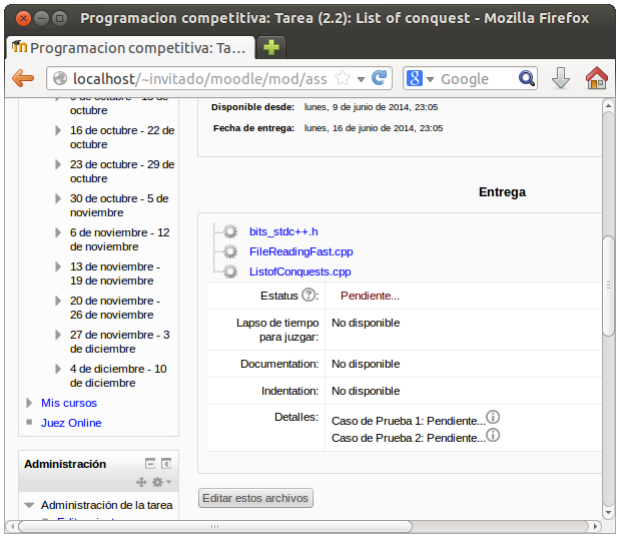

Figura 28: Envió De Trabajo De Programación Moodle

6.3. Calificación De Una Tarea Encolada.

Luego de hacer envíos por la interfaz del cliente mediante la interfaz de envió si los demonios han sido instalados correctamente los mismos deben en un tiempo razonable cambiar de estado mostrando las diferentes calificaciones:

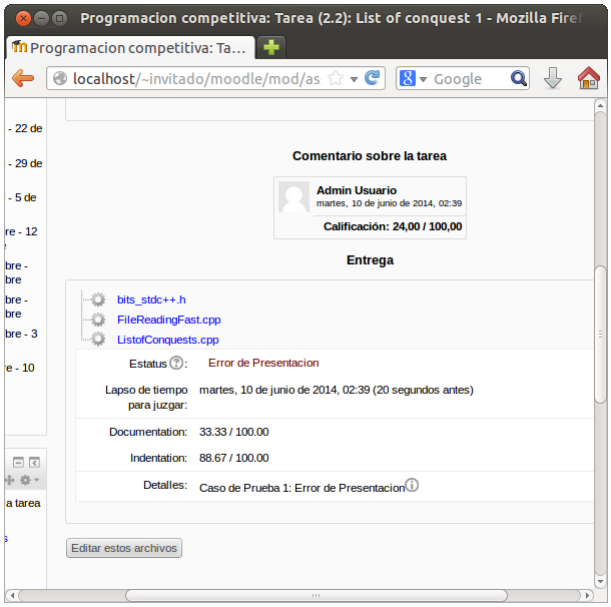

Figura 29: Calificación De Un Envió Moodle

Se puede incluso ver los detalles del envió presionando sobre el icono que esta encima de los detalles de cada caso de prueba (figura: Detalles De La Calificación De Un Caso de Prueba, en la página 80)

Además estas calificaciones son editables a mano al navegar por el enlace “Ver # tareas enviadas” (figura: Ver los envíos a las tareas, en la página 80)

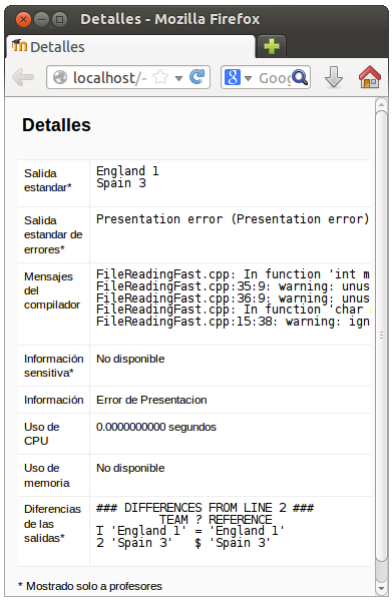

Figura 30: Detalles De La Calificación De Un Caso de Prueba

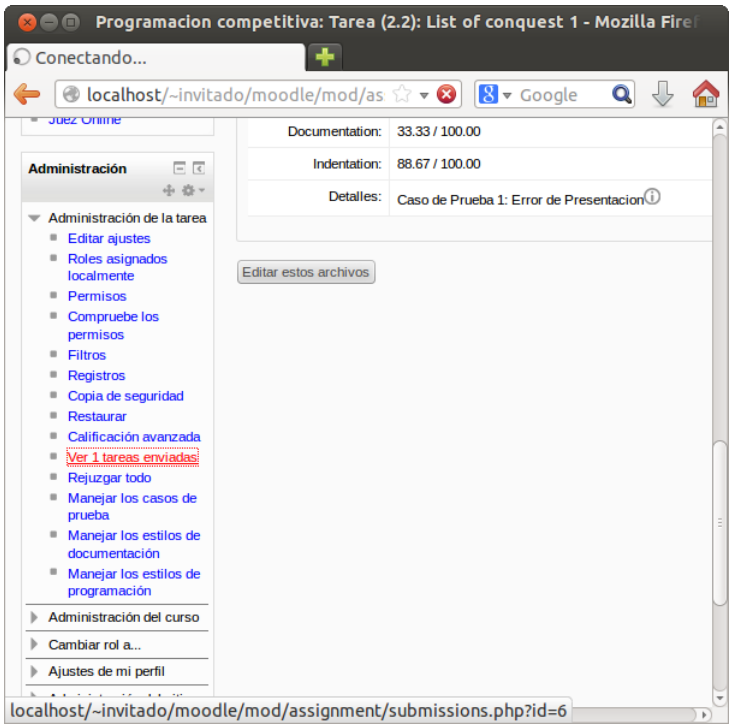

Figura 31: Ver Los Envíos A Las Tareas

Así se ven los diferentes envíos:

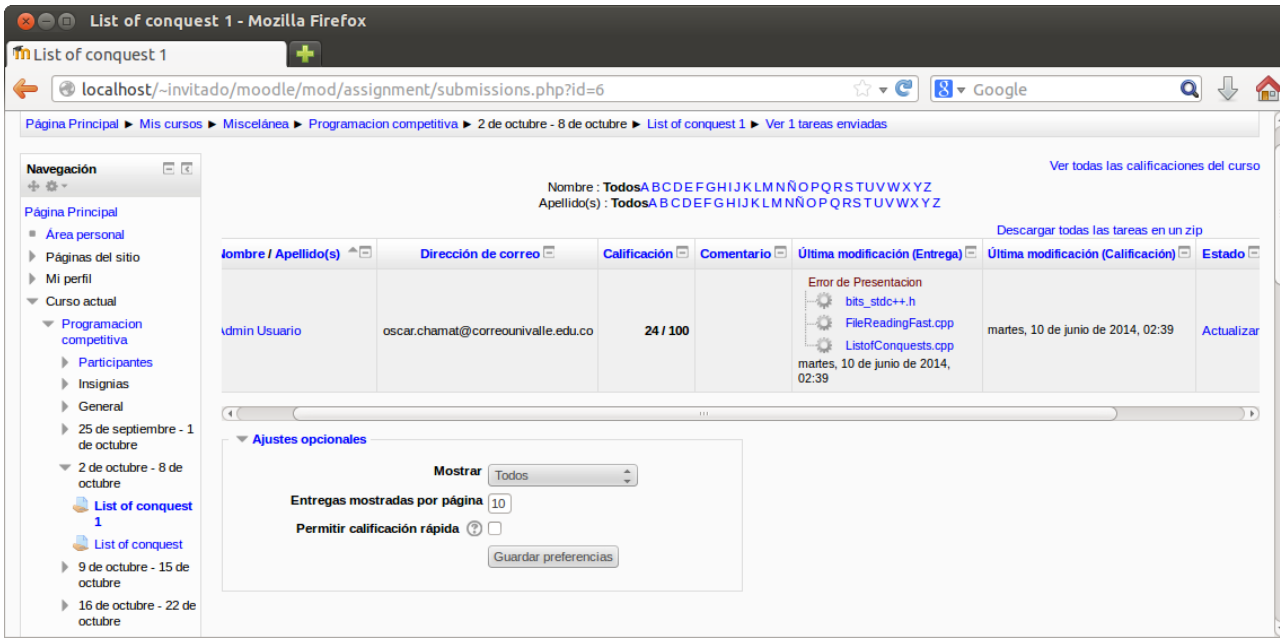

Figura 32: Calificaciones De Los Envíos Moodle

Y cuando se presiona el botón “Actualizar” sigue la siguiente pantalla;

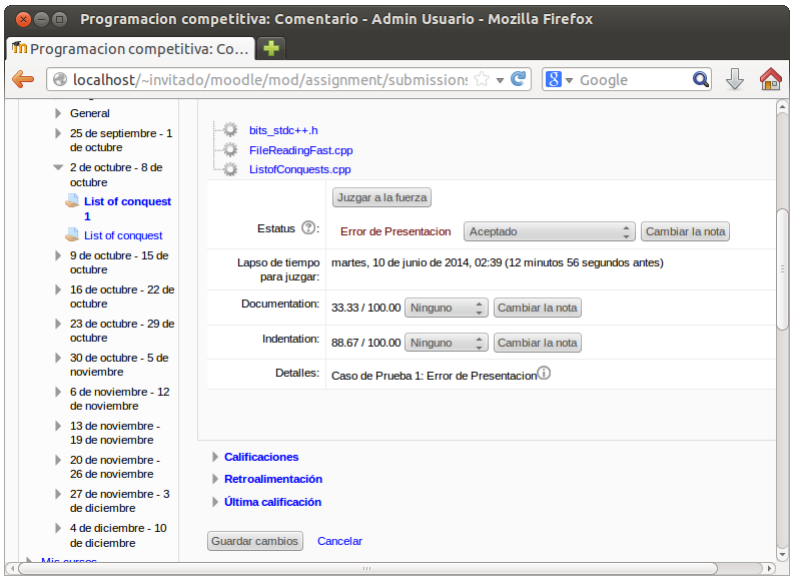

Figura 33: Cambiando Calificaciones

Desde esta pantalla se puede cambiar la calificación de cualquier apartado del envió y se puede enviar de nuevo para ser calificado automáticamente con "Juzgar A la fuerza".

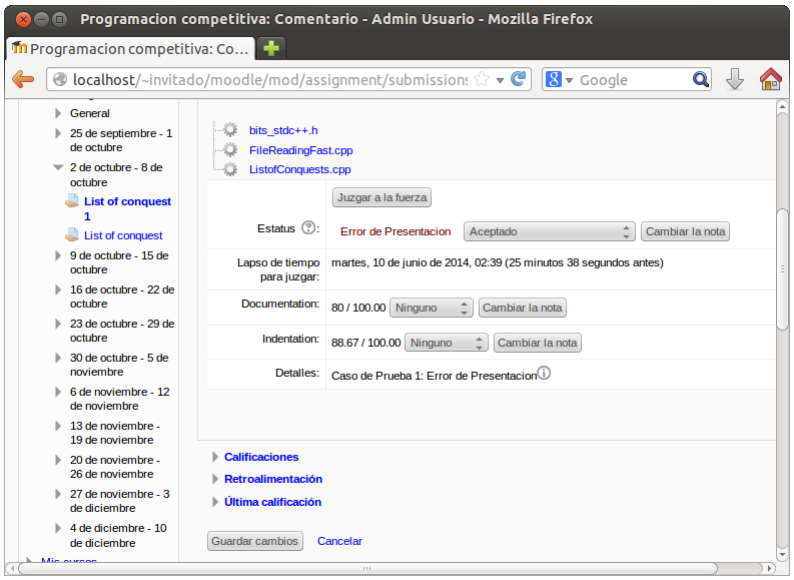

Figura 34: Cambiando Calificaciones

Figura 35: Ejecución de las pruebas

También se incluye la función de "Rejuzgar todo" para un trabajo de programación pero esto puede estresar demasiado el sistema.

6.4. Ejecución de las pruebas unitarias

Complete de forma exitosa lo explicado en la sección 5.2.4 en la página 67.

Ahora diríjase a el lugar donde instalo su servidor domjudge y ejecute (el orden de ejecución es importante primero el judgedaemon y luego las pruebas):

$php <basepath>/judgehost/bin/judgedaemon -t

Para ejecutar todas las pruebas:

$cd MOODLE_PATH/local/onlinejudge/tests

$./runtest.sh

Esto corre todos los test y muestra todos los resultados en lo que se puede ver si hay algún failure lo que significaría que hay un bug. (figura: Ejecución de las pruebas, en la página 83)

También se pueden ejecutar las pruebas individualmente:

$cd MOODLE_PATH

$vendor/bin/phpunit local_onlinejudge_compilation_error_cpp_testcase local/onlinejudge/tests/compilation_error_cpp_test.php

o $vendor/bin/phpunit local/onlinejudge/tests/compilation_error_cpp_test.php, si solo hay una clase por archivo.