Universidad de las Ciencias Informáticas

Facultad 1

Título: Sistema para la Calificación Automática en Competencias de Programación

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autores: Beatriz Lazo Tamayo

Ernesto Soto Gómez

Tutores: MCs. Yeneit Delgado Kios

Dr. Edistio Yoel Verdecia Martínez

La Habana. Junio, 2013

Declaración de autoría

Declaramos ser autores del presente trabajo de diploma “Sistema para la calificación automática en competencias de programación” y solo autorizamos a la Universidad de las Ciencias Informáticas (UCI) los derechos del uso del mismo en su beneficio. Para que así conste firmamos la presente a los ___ días del mes de _____ del año _____.

Beatriz Lazo Tamayo

_____

Firma del Autor

Ernesto Soto Gómez

_____

Firma del Autor

MCs. Yeneit Delgado Kios

_____

Firma del Tutor

Dr. Edistio Yoel Verdecia Martínez

_____

Firma del Tutor

Agradecimientos

A todos los profesores que de uno u otra forma contribuyeron a nuestra formación como ingenieros. A los tutores, por su paciencia y dedicación. A los miembros del tribunal y al oponente que influyeron en el perfeccionamiento de esta investigación.

Beatriz

Quiero agradecer en primer lugar a mi mami Nirma por haber cumplido el rol de madre y padre de manera ejemplar. Por todos los sacrificios que ha realizado por mí, por sus noches de desvelo cuando he estado enferma o preocupada, por sus consejos y su amor. A mi abuelita Luisa, quien ha sido mi segunda mamita y me ha dado siempre su amor incondicional. A mi abuelito Nelio, quien con su rectitud ha contribuido a formar parte de la mujer que soy. A mi tía Uly, quien me ha dado ejemplo de constante superación. A mi hermanito Dani, quien ha contribuido a que me esfuerce cada vez más para ser un buen ejemplo. A mi padrastro Orlando y a mis tíos Eduard y Osmany. A mi novio Luis por darme paz, seguridad y mucho amor. A mis abuelos y tíos paternos. A mi amiga Melvis con quien he compartido muchas noches de insomnio estudiando. A Ernesto, quien no solamente es mi compañero de tesis, sino mi gran amigo. En fin a todos los que de alguna manera han contribuido, no solo a mi formación como ingeniera, sino como el ser humano que hoy soy.

Ernesto

Quiero agradecer en primer lugar a mis irremplazables e incomparables amigos de siempre: mis padres y mi hermano Kiki, sin los cuales no fuera la persona que soy hoy. A Ramón Ropero por su amistad incondicional y verdadera. A José Antonio (Tony) Rey y Mirian Mercedes (Miri) Vargas por ser, más que mis maestros y familia, parte de mis mejores amigos. A mis tías Lola y Mirian, quienes siempre han estado presentes ante cualquier situación. A mi Tía Sara por acogerme en su casa y soportar mis despistes. A José Jairán (Cepi) Breijo y a Beatriz (Betty) Lazo por apoyarme incondicionalmente como amigos genuinos. A Betty un agradecimiento especial, porque mejor compañera de tesis no la iba a tener nunca. A Jorge Luis (Jorgito) Fonseca por su gran amistad y por colaborar en ideas previas a la de esta tesis. A Jorge Ruiz quien fue el de la idea del nombre Wormhole. Resumiendo, y porque pienso que la amistad es lo más grande, le agradezco a todos mis amigos.

Dedicatoria

Beatriz

A la memoria de mi papá Osmel, quien sé que ha de estar muy orgulloso de mí…

Ernesto

Le dedico el resultado de este trabajo, tal si fuera una poesía (que no lo es), a mis padres Nubia Berta Gómez Iribar y Emilio Soto García, y a mi hermano Emilio Soto Gómez; porque, si de dedicatorias se trata, ellos se merecen todas las del mundo. Porque estoy orgulloso de ellos y porque de entre todos mis amigos, no hay ninguno más grande.

Resumen

La realización de competencias de programación de máquinas de cómputo persigue la mejora del proceso de enseñanza-aprendizaje en los estudiantes que cursan carreras afines. La calificación en este tipo de eventos se realiza de manera automática por los denominados jurados en línea.

Actualmente estos sistemas, en su mayoría, solamente permiten la realización de competencias de programación de un mismo tipo, las reglas que determinan un tipo de competencia están atadas al jurado en línea que se utilice. Además no existe un sistema único que concentre estos tipos de competencias y que permita la configuración de nuevos tipos.

Se identificó entonces la necesidad de desarrollar un jurado en línea que permita la configuración de nuevos tipos de competencias, incluyendo las existentes actualmente; así como concentrar estos tipos en un único sistema, posibilitando la gestión y reutilización de los mismos.

El presente documento recoge los resultados de la investigación realizada, describiéndose las principales características de los sistemas analizados, la arquitectura y el diseño del sistema propuesto. Se describen además las herramientas, tecnologías utilizadas y los artefactos generados en el proceso de desarrollo.

Finalmente se obtiene un jurado en línea para la calificación automática en competencias de programación que soluciona la problemática que lo originó mediante el cumplimiento de los objetivos trazados.

Palabras claves: jurado en línea, competencia de programación, reglas de la competencia, motor de calificación.

Índice de contenido

Figuras

Tablas

Introducción

La construcción de *software* constituye un campo esencial de la Informática. Disímiles son las ramas de la computación en las que, de una u otra forma, se deben desarrollar habilidades y aplicar los conocimientos adquiridos durante el estudio de esta disciplina. Para potenciar el aprendizaje y motivación hacia la programación de computadoras, de quienes cursan carreras afines, se destacan las competencias relacionadas precisamente con la programación de máquinas de cómputo.

Este tipo de concursos es realizado principalmente para incrementar la calidad algorítmica mediante la motivación competitiva. Son efectuados a nivel de consola, o sea, sin mediación de una interfaz gráfica, debido a que se centran en el desarrollo de algoritmos y su codificación, sin estéticas, estimulando así únicamente la capacidad de razonamiento orientado a la solución de problemas.

A nivel internacional crece continuamente el número de eventos de este tipo, entre los que destacan: el renombrado concurso perteneciente a la ACM (del inglés Association for Computer Machinery), conocido como ACM-ICPC (del inglés International Collegiate Programing Contest)[1], en el cual participan alrededor de 2000 universidades de más de 80 países y la IOI (del inglés International Olympiad in Informatics)[2], en la que participan cuatro estudiantes por cada país y alrededor de 81 países.

En Cuba, al igual que en el resto del mundo, la realización de competencias de programación de máquinas de cómputo ha aumentado en los últimos años. Desde el preuniversitario se preparan a los estudiantes aptos para formar parte de la preselección y selección nacional de programación que participa en la IOI. A nivel universitario se realizan eventos como la competencia regional de ACM-ICPC, de la cual la Universidad de las Ciencias Informáticas (UCI) ha sido sede en varias ocasiones, para seleccionar a los estudiantes que participarán en este concurso a nivel internacional. En la UCI se realizan otras competencias para contribuir a la formación de los estudiantes, entre ellas, la Copa Pascal y la Copa UCI.

Tanto en concursos nacionales como internacionales, la calificación de las soluciones a los problemas en este tipo de eventos se realiza de manera automática. Estos sistemas automáticos son comúnmente llamados Jurados en Línea (u Online Judges en inglés), pues son aplicaciones web que compilan y ejecutan código y cuyo fin es calificar las respuestas a problemas de programación de máquinas de cómputo, de los que se conoce un juego de datos de entrada y el o los posibles juegos de datos de salida. Los administradores de estos sistemas organizan sus propias competencias, así cuentan, generalmente, con un conjunto de ejercicios públicos cuya resolución contribuye a la puntuación del usuario dentro de un *ranking* general. Cada jurado es libre de implementar las reglas de estas competencias, las cuales definen el tipo de competencia y no se refieren solamente a los métodos o fórmulas que se usen para asignar o sustraer determinada cantidad de puntos a determinado usuario, sino también a las condiciones

[1] http://icpc.baylor.edu
[2] http://ioinformatics.org

de comienzo y finalización de una competencia, a la forma en que se muestra el *ranking* y los parámetros por los que se ordena, a la forma en que se muestra el resultado de una calificación, o sea, si se brinda más o menos información acerca del mismo, se refieren además a las restricciones de calificación manejadas para cada lenguaje permitido para cada ejercicio, a la cantidad de veces que un usuario puede emitir una posible solución a un problema determinado, entre otras características que permitan definir el comportamiento de una competencia dentro del sistema y las acciones permitidas a los concursantes de la misma en el sistema.

En la UCI actualmente se cuenta con varios jurados en línea entre ellos la Cátedra de Programación Avanzada (CPAV) y el COJ (del inglés Caribbean Online Judge). Cada jurado tiene su propia manera de asignar puntos y de mostrar los resultados, sin embargo la que más se utiliza es el estilo ACM-ICPC. Esta forma de asignar puntos se basa, principalmente, en la cantidad de veces que ha sido respondido correctamente un determinado problema, o sea, mientras mayor sea la cantidad de soluciones correctas a un ejercicio, la puntuación recibida por los concursantes al resolverlo es menor, sin embargo, mientras menor sea la cantidad de soluciones correctas a un problema, mayor es la puntuación recibida por resolver dicho problema. Los resultados de la evaluación son mostrados con mensajes que brindan la menor información posible para no dar pista alguna del posible error que puede haber llevado a una respuesta incorrecta. Solamente se notifica si hubo error de compilación o de ejecución, si el programa no cumple con las restricciones de tiempo, memoria o tamaño de fichero de código fuente, o si el mismo no resuelve el problema.

Esta manera de asignar puntos tiene como ventaja el hecho de establecer una medida de cuáles pudieran ser los ejercicios menos complicados, debido a que los ejercicios que más hayan sido resueltos serían los que menos punteen. Sin embargo esta medida no es definitiva, pues pudiera existir un ejercicio fácil de resolver pero con un enunciado poco comprensible, o simplemente la atención de los concursantes se puede tornar hacia aquellos ejercicios que más rápidamente han comenzado a devaluarse, ejercicios que no tienen por qué ser los menos complicados, sino simplemente puede que sean los que primero comenzaron a resolverse. Este método contempla solamente la asignación de puntos a ejercicios que han sido resueltos completamente. Se conoce que existen problemas que, en la actualidad, no poseen una solución factible, pudiéndose encontrar únicamente soluciones aproximadas en un tiempo razonable, entre ellos se encuentran el problema del ciclo hamiltoniano de costo mínimo (problema del viajante) y el problema de satisfacibilidad *booleana*, entre otros. Una competencia que asigne más puntos a la mejor solución parcial no podría usar el método antes descrito; tampoco lo podría hacer una competencia que se utilice solamente de entrenamiento, donde se pudieran mostrar muchos más detalles en los resultados de calificación que los comunes para una evaluación. De nada serviría crear nuevas versiones de los jurados existentes que puedan soportar tipos de competencias específicos, pues las reglas de un tipo de

competencia determinada pueden ser disímiles en dependencia del objetivo del tipo de competencia y en el momento en que se quieran adoptar nuevas reglas, será necesario entonces crear otra nueva versión del *software*.

De acuerdo a la problemática explicada con anterioridad se define el siguiente **problema científico**: ¿cómo perfeccionar la gestión y configuración de competencias de programación de diferentes tipos?

A partir de ello la presente investigación enmarca su **objeto de estudio** en: el proceso de evaluación en línea de soluciones a problemas en concursos de programación.

El **objetivo general** se declara como: desarrollar un jurado en línea que permita, además de las funcionalidades básicas, la gestión y configuración de las reglas de las competencias de programación.

Se desglosa el objetivo general en los siguientes **objetivos específicos**:

- Caracterizar los jurados en línea, las técnicas y herramientas que pueden ser utilizados en su implementación.
- Realizar el análisis y el diseño del jurado en línea.
- Implementar el jurado en línea propuesto.
- Realizar pruebas de *software* al jurado en línea propuesto para garantizar su calidad de acuerdo a las exigencias de la metodología de desarrollo utilizada.

Específicamente se tiene como **campo de acción**: el proceso de configuración del proceso de evaluación en línea.

De esta manera la **idea a defender** es: el desarrollo de un jurado en línea que permita, además de las funcionalidades básicas, la gestión y configuración de las reglas de las competencias de programación, permitirá perfeccionar la gestión y configuración de competencias de programación de diferentes tipos.

Para el cumplimiento de cada uno de los objetivos planteados se definen las siguientes **tareas de la investigación**:

- Análisis de las soluciones existentes en el mundo y en particular en la UCI para la calificación automática en línea.
- Selección de los lenguajes de programación, marcos de trabajo, sistema gestor de bases de datos y protocolos de comunicación a utilizar para el desarrollo del jurado en línea.
- Identificación de los entornos de desarrollo integrado y herramientas de modelado necesarias para el desarrollo del jurado en línea.
- Selección de la metodología a utilizar para el desarrollo del jurado en línea.
- Definición de los requisitos funcionales y no funcionales del jurado en línea.
- Definición de la arquitectura del sistema.
- Diseño de la base de datos del sistema.
- Desarrollo de los componentes del sistema.

- Integración de los componentes del sistema.
- Diseño y realización de pruebas al jurado en línea.

La investigación se sustenta en los siguientes **métodos científicos:**

Métodos teóricos

El método analítico-sintético se utiliza para sintetizar los elementos más importantes de las fuentes bibliográficas analizadas y de esta manera poder identificar las tecnologías, estándares y herramientas a usar para el desarrollo del jurado en línea.

El uso del método de modelación, para la representación abstracta de determinadas características del sistema, se materializa mediante la construcción de diagramas y modelos a lo largo del desarrollo de la investigación.

Métodos empíricos

Se utiliza el método de entrevista para obtener información, experiencias, y puntos de vistas que contribuyan al desarrollo de la investigación y que aporten conocimientos específicos a la misma.

Justificación de la investigación

La realización de competencias de programación de máquinas de cómputo es de suma importancia para la mejora del proceso de enseñanza-aprendizaje en los estudiantes que cursan carreras afines, pues este tipo de eventos desarrolla el pensamiento algorítmico necesario para la creación de *software* mediante la solución de problemas de alta complejidad en un período de tiempo determinado. Estas competencias promueven además valores como el trabajo en equipo y el compañerismo.

La calificación en estos eventos se realiza de manera automática por los jurados en línea. Actualmente la mayoría de estos sistemas solamente permiten la realización de competencias de un mismo tipo, las reglas que determinan un tipo de competencia están atadas al jurado en línea que se utilice. Además no existe un sistema único que concentre estos tipos de competencias y que permita la configuración de nuevos tipos.

El jurado en línea que se propone resolvería los inconvenientes mencionados con anterioridad, pues permitiría la configuración de nuevos tipos de competencias, incluyendo las existentes actualmente, lo que sería útil para promover más habilidades entre los concursantes debido a la posibilidad de crear competencias con objetivos diversos, al tiempo que aumentaría la motivación de los estudiantes, desde el más instruido hasta el que no lo es tanto, a participar en estos eventos en dependencia del tipo de competencia. El sistema en cuestión permitiría también concentrar estos tipos posibilitando la gestión y reutilización de los mismos.

El presente documento está estructurado en tres capítulos:

Capítulo 1: Fundamentación teórica: este capítulo contiene una base teórica para entender el problema planteado, en él se describen los jurados en línea existentes, sus aportes y limitaciones, así como las herramientas y tecnologías a utilizar.

Capítulo 2: Propuesta de solución: en este capítulo se presentan las fases de Planificación y Diseño definidas por la metodología XP. Se realiza la propuesta de solución y se planifica el proceso de desarrollo de *software*. Se definen, entre otros, las historias de usuarios, los requisitos no funcionales, la arquitectura del sistema, el diseño de clases y el diseño de la base de datos.

Capítulo 3: Codificación y prueba: en este capítulo se definen las tareas de ingeniería, se codifica la solución diseñada y se realizan las pruebas definidas para el sistema.

Capítulo 1: Fundamentación teórica

1.1 Introducción

En el presente capítulo se realiza un análisis de los principales jurados en línea existentes en el mundo y en Cuba para determinar sus limitaciones y los aportes al sistema propuesto. Además se identifican, luego de un profundo análisis, la metodología de desarrollo de *software*, tecnologías y herramientas necesarias para la implementación de este jurado en línea.

1.2 Estado del arte de los jurados en línea

Un jurado en línea es una aplicación web que incluye la descripción de varios problemas que pueden ser solucionados mediante distintas técnicas de programación. Estas aplicaciones evalúan de manera automática las soluciones enviadas por los usuarios y emiten una calificación (1). Se asocian normalmente con entornos académicos, pues se emplean como calificadores en competencias de programación, así como para el entrenamiento diario de los estudiantes en materia de programación de máquinas de cómputo, estimulando el aprendizaje de esta importante disciplina dentro de la Informática.

La implementación de jurados en línea ha crecido a nivel mundial en los últimos años, algunos de los más reconocidos son el jurado en línea de la Universidad de Valladolid (UVA)[3], el Sphere Online Judge (SPOJ)[4], el jurado en línea de la Universidad de Pekín (PKU)[5] y el de la Universidad de los Urales (Timus)[6]. En Cuba el desarrollo de este tipo de aplicaciones no ha quedado al margen, siendo la UCI el mayor exponente en la construcción de jurados en línea propios, destacándose el jurado en línea de la Cátedra de Programación Avanzada (CPAV)[7] y el Caribbean Online Judge (COJ)[8].

A continuación se describen cada uno de los jurados en línea anteriormente enunciados, para así determinar sus limitaciones y sus aportes al jurado en línea propuesto.

1.2.1 Soluciones existentes en el mundo

1.2.1.1 Jurado en línea de la Universidad de Valladolid (UVA)

El jurado en línea de la Universidad de Valladolid (UVA) es en la actualidad el más prestigioso en la comunidad universitaria a nivel mundial. Es uno de los jurados más antiguos y el que abarca la mayor cantidad de problemas, incluso es usado en ocasiones como fuente para los demás jurados. Ha sido utilizado en disímiles competencias de ACM-ICPC. Una de las principales características del UVA es que

[3] http://uva.onlinejudge.org/
[4] http://www.spoj.pl/
[5] http://poj.org/
[6] http://acm.timus.ru/
[7] http://cpav.uci.cu
[8] http://coj.uci.cu

los problemas se organizan en extensos volúmenes que se nutren en su mayoría de las competencias de la ACM, convirtiéndose prácticamente en referencia obligada para los interesados en este tema (1).

Este jurado en línea solo soporta el envío de soluciones implementadas en los lenguajes de programación C, C++, Java y Pascal. Las posibles respuestas a emitir por el UVA luego de calificar una solución son: In Queue (QU)[9], Accepted (AC)[10], Wrong Answer (WA)[11], Compile Error (CE)[12], Runtime Error (RE)[13], Time Limit Exceeded (TL)[14], Memory Limit Exceeded (ML)[15], Presentation Error (PE)[16], Output Limit Exceeded (OL)[17], Can't Be Judged (CJ)[18], Submission Error (SE)[19] y Restricted Function (RF)[20]. El UVA permite obtener algunas estadísticas como los últimos 50 envíos, el *ranking* de los usuarios de acuerdo a problemas que han tratado de resolver, problemas que han resuelto y total de envíos, así como las estadísticas anuales de envíos de cada lenguaje de programación. Este jurado posibilita además la visualización de diferentes aspectos, como un *ranking* con el número de envíos y sus autores en la hora actual, en el último día, semana, mes y año; los usuarios que han sido líderes del *ranking* y el tiempo que cada usuario ha sido líder, entre otros.

1.2.1.2 Sphere Online Judge (SPOJ)

El Sphere Online Judge (SPOJ) se encuentra afiliado a la Universidad de Tecnología de Gdansk en Polonia. Además del idioma inglés este jurado ofrece su contenido en polaco, portugués y vietnamita. Ha sido anfitrión de más de 2400 competencias tanto oficiales como la IOI, como propias de los usuarios (2).

Posee soporte para el envío de soluciones implementadas en cualquiera de los 45 lenguajes de programación de los que dispone, entre ellos Pascal, Java, Python y C#, así como brinda soporte para diferentes compiladores. El SPOJ además contiene varios volúmenes de problemas disponibles en distintos idiomas, dentro de los cuales se incluyen, en su mayoría, los problemas presentados en las diversas ediciones del ICPC (3). Dichos problemas son divididos en categorías y en dependencia de ello cambian o no las características de las soluciones a enviar así como las características de la calificación del jurado. Las posibles respuestas emitidas por el SPOJ al calificar una solución son: “AC”, “WA”, “CE”, “RE” y “TL”. Este jurado en línea permite la configuración de la asignación de puntos dentro de una

[9] In Queue (QU): El jurado está ocupado y calificará la solución cuando esté listo.
[10] Accepted (AC): El programa cumple con todos los parámetros establecidos.
[11] Wrong Answer (WA): La solución es incorrecta de acuerdo a los datos de entrada y salida probados.
[12] Compile Error (CE): El compilador no puede compilar la solución.
[13] Runtime Error (RE): El fallo ocurre durante la ejecución.
[14] Time Limit Exceeded (TL): El programa trata de ejecutarse durante mucho tiempo.
[15] Memory Limit Exceeded (ML): El programa trata de usar más memoria de la permitida por el jurado.
[16] Presentation Error (PE): Las salidas del programa son correctas, pero se presentan de manera inadecuada.
[17] Output Limit Exceeded (OL): El programa trata de escribir demasiada información.
[18] Can't Be Judged (CJ): El jurado no posee juegos de datos de entrada y salida para probar la solución a ese problema.
[19] Submission Error (SE): El proceso de envío no fue correcto.
[20] Restricted Function (RF): El programa trata de usar alguna función que el jurado considera perjudicial para el sistema.

competencia usando directamente C++ (4). Permite además la visualización de diferentes *rankings*: por usuarios de acuerdo a la máxima puntuación obtenida que se calcula sumando la recibida por cada tipo de problema enviado, por países, por lenguajes de programación, por instituciones y por usuarios de acuerdo a la cantidad de problemas adicionados. Estos *rankings* no cambian solamente cuando el usuario realiza una operación, sino cada vez que se realiza una operación sobre uno de los problemas que tributan a la puntuación del usuario, asegurando que los problemas más complejos puntúen más y que los usuarios que pasen cierto tiempo sin realizar envíos desciendan y asciendan los usuarios más activos. Las soluciones enviadas por los concursantes son calificadas por alguno de los clústeres[21], el que su creador escoja, y de acuerdo a ello será la velocidad de la calificación y el límite de la memoria (5).

1.2.1.3 Jurado en línea de la Universidad de los Urales (Timus)

El jurado en línea de la Universidad de los Urales (Timus) en Rusia, es un proyecto desarrollado y mantenido por estudiantes que tiene las funcionalidades básicas de este tipo de aplicación y una interfaz intuitiva y fácil de usar. Cuenta con dos versiones del mismo *software*, en ruso y en inglés. Es la mayor fuente en Rusia de ejercicios de diferentes competencias de programación (6). Una de las características de este jurado es su solución de comunicación interna mediante una *web board*[22].

El Timus define una serie de reglas de seguridad como el uso de una sola cuenta para cada usuario y no publicar o distribuir soluciones. Soporta la implementación de soluciones en los lenguajes de programación Java, C/C++, Pascal y C#. La dificultad de un problema es asignada en dependencia del número de usuarios que lo hayan solucionado y de la antigüedad del problema. Mientras menor sea la cantidad de usuarios que resuelvan el problema, mayor es la dificultad, de esta manera el *rating* de un usuario es la suma de las dificultades de todos los problemas que haya resuelto. Los problemas en este jurado en línea se encuentran agrupados por volúmenes, por eventos y por categorías, así se pueden filtrar y buscar problemas relacionados por ejemplo con programación dinámica, geometría y teoría de números, entre otras clasificaciones. El Timus se ha utilizado en más de 90 competencias (7), entre ellas algunas de ACM-ICPC. Este jurado en línea permite la visualización de un *ranking* de usuarios de acuerdo al *rating* y la cantidad de problemas resueltos.

1.2.1.4 Jurado en línea de la Universidad de Pekín (PKU)

El jurado en línea de la Universidad de Pekín (PKU) y el Timus, anteriormente descrito, poseen prácticamente la misma interfaz gráfica y funcionalidades básicas en común. El PKU ha sido utilizado para disímiles competencias, entre ellas las de entrenamiento para la ACM-ICPC y la IOI (8).

---

[21] En este documento se refiere a una agrupación lógica de máquinas computadoras que se comportan con características similares y que se utilizan como calificadores de las soluciones enviadas por los usuarios.

[22] Una *web board* es una herramienta (*software*) para el uso de salas de conferencias en línea.

Este jurado propone tres directrices fundamentales para implementar una solución, ellas son hacer exactamente lo que el problema pida, ni más ni menos, pues en estas aplicaciones la calificación es automática y aunque la respuesta sea lógicamente correcta si no está en el formato adecuado es inútil, la segunda es que la solución acceda solamente a la entrada estándar, la salida estándar y a la memoria y por último escribir el código lo más estandarizado posible. Soporta el envío de soluciones escritas en los lenguajes de programación C/C++, Java, Pascal y Fortran, y el uso de diferentes compiladores. Las posibles respuestas a emitir por el PKU al calificar una solución son: “AC”, “WA”, “CE”, “RE”, “TL”, “ML”, “PE”, “OL”, System Error[23] y Validator Error[24]. Este jurado incluye más de 3000 problemas organizados en diferentes volúmenes. Contiene un módulo de estadísticas de envíos y posibilita la visualización de un *ranking* de usuarios de acuerdo a un *rating* que se calcula mediante el porciento que representan los problemas resueltos sobre el total de problemas enviados. Brinda la opción de descargar una versión libre de la aplicación, lo cual marcó los primeros pasos para el desarrollo y despliegue de jurados en línea en la UCI (9).

1.2.2 Soluciones existentes en Cuba

1.2.2.1 Caribbean Online Judge (COJ)

El Caribbean Online Judge (COJ) es un sistema creado por estudiantes y profesores de la UCI. Surge ante los problemas de conectividad que limitaban en Cuba y se encuentra disponible desde junio del 2010 en la red nacional del Ministerio de Educación Superior (MES) y en Internet desde que la UCI se unió al movimiento ACM-ICPC. Permite el entrenamiento para competencias como la ACM-ICPC y la IOI. Incluye un foro para el intercambio de experiencias y la discusión de temas de interés (10). Ha sido utilizado hasta la fecha para la realización de más de 200 competencias ya sean nacionales, como la Copa Pascal de programación, o internacionales, como la competencia regional de ACM-ICPC (11).

Este jurado en línea posibilita realizar la evaluación de soluciones de manera distribuida mediante la utilización de clústeres (9). Soporta el envío de soluciones implementadas en los lenguajes de programación C/C++, Java, C#, Pascal, Python, Perl, Ruby y PHP, así como el uso de diferentes compiladores. Propone una serie de directrices para implementar las soluciones, entre ellas respetar y cumplir estrictamente las indicaciones de entrada y salida indicadas en la descripción del problema y nunca exceder los límites de ejecución y del uso de la memoria. Las posibles respuestas emitidas por el COJ al calificar una solución son: “AC”, “WA”, “CE”, “RE”, “TL”, “ML”, “PE”, “OL”, Judging (JDG)[25],

[23] System Error: El jurado falló al ejecutar la solución.

[24] Validator Error: El programa que se encarga de verificar ha mostrado un comportamiento anormal durante la validación de la salida producida por la solución.

[25] Judging (JDG): El jurado está ocupado, por lo que revisará la solución cuando esté listo.

Unqualified (UQD)[26], Invalid Function (IVF)[27] y Size Limit Exceeded (SLE)[28]. Este jurado en línea contiene más de 1000 problemas que puntúan de acuerdo al porcentaje de total de veces que han sido aceptados contra total de veces que han sido enviados. Permite visualizar una serie de *rankings*, tanto de usuarios como por instituciones y países. El COJ brinda además estadísticas sobre la cantidad de respuestas emitidas en la calificación, por tipo de respuesta por cada lenguaje de programación.

1.2.2.2 Cátedra de Programación Avanzada (CPAV)

La Cátedra de Programación Avanzada (CPAV) es un jurado en línea desarrollado por estudiantes y profesores de la UCI en el año 2008. Ha sido utilizado en varias competencias de programación como la Copa Troya y la Copa UCI de programación (9). Entre sus características se encuentran que permite el envío de soluciones implementadas en los lenguajes de programación C#, C++, Java, Python y Pascal. Incluye un foro para el intercambio entre los usuarios. Agrupa los problemas en diferentes volúmenes. La CPAV permite además la visualización de un *ranking* general, y *rankings* por equipo, por año, por facultad y por lenguajes de programación. Muestra además estadísticas de los usuarios, permitiendo la comparación entre los mismos. La puntuación de los problemas varía en dependencia de la cantidad de envíos aceptados que tenga el problema, mientras mayor sea la cantidad, menos puntúa (12).

1.2.3 Aportes y limitaciones de las soluciones existentes

Los jurados en línea descritos anteriormente constituyen opciones para realizar la calificación automática en competencias de programación. Algunos de estos sistemas son utilizados ampliamente y cuentan con un considerable tiempo de existencia, lo que ha permitido el aumento de sus colecciones de ejercicios y de los lenguajes de programación que soportan, así como de funcionalidades útiles tales como la calificación distribuida y la utilización de chequeadores[29] externos.

Estos jurados en línea aportan las funcionalidades básicas y comunes a este tipo de sistemas, sin embargo aunque algunos han acogido la celebración de competencias de renombre, los mismos, normalmente, solo soportan un número estático de tipos de competencias. Las reglas de la competencia predominantes en estos sistemas son las usadas en competencias tipo ACM-ICPC[30] y no es posible la realización de nuevos tipos de competencias con reglas que puedan ser configuradas en el sistema sin necesidad de modificar o extender el mismo modificando o añadiendo código fuente directamente. El único de estos jurados que se acerca a estos objetivos es el SPOJ, que permite la configuración de la

[26] Unqualified (UQD): El programa no puede ser ejecutado en ese momento.
[27] Invalid Function (IVF): El programa trata de hacer alguna operación que no está permitida por el jurado.
[28] Size Limit Exceeded (SLE): El código excede el límite propuesto para él.
[29] Un chequeador es un programa que verifica que la salida generada por un código juzgado sea la correcta sintáctica y semánticamente. Por lo general, un jurado en línea, por defecto, chequea que la salida emitida por el código juzgado sea exactamente igual a una salida patrón que se posee de antemano.
[30] http://icpc.baylor.edu/info/Regional+Rules.

asignación de puntos dentro de una competencia, aunque usando directamente C++ como lenguaje de programación (4) para agregar una nueva configuración.

La configuración y gestión de nuevos tipos de competencias permitiría la realización de disímiles competencias con objetivos diversos, lo que sería útil para promover más habilidades entre los concursantes, al tiempo que aumentaría la motivación de los estudiantes, desde el más instruido hasta el que no lo es tanto, a participar en estos eventos en dependencia del tipo de competencia. Un sistema que concentre estos tipos permitiría además la gestión y reutilización de los mismos.

No sería sabio modificar alguno de estos sistemas para lograr el objetivo antes mencionado debido a que los mismos constituyen jurados especializados, creados y optimizados con el fin de ser usados únicamente en determinados tipos de eventos, por lo que cada vez que se desee crear un nuevo tipo de competencia, habría que modificar o añadir código fuente. Sería más sencilla la construcción de un nuevo jurado diseñado explícitamente para soportar esta característica que está tan ligada al núcleo del sistema en sí, de esta manera se podrían crear nuevos tipos de competencia con reglas que pueden ser configuradas en tiempo de ejecución del sistema.

1.3 Herramientas y tecnologías propuestas para la solución del problema

1.3.1 Indicadores para identificar herramientas y tecnologías candidatas

Para la elección de las herramientas candidatas se definieron una serie de indicadores que se ajustan a las necesidades generales del jurado en línea. En primer lugar se han elegido aquellas que hayan sido liberadas bajo una licencia de *software* libre o de código abierto, de acuerdo a la migración hacia este tipo de *software* que se lleva a cabo en el país. Usar tecnologías libres trae consigo además facilidades de uso y promoción del producto en posibles contextos internacionales, sin costo adicional, debido a las libertades de uso, estudio, modificación y distribución que proporcionan (13).

Se consideraron también como posibles herramientas, aquellas de las cuales los desarrolladores tuvieran conocimiento previo, para así elegir fácilmente las mejores opciones de acuerdo a la experiencia de uso y favorecer el cumplimiento del cronograma de desarrollo y la fecha de terminación del producto final.

Otro de los indicadores se relaciona con el propósito para el que fueron diseñadas las herramientas. Se escogieron aquellas que fueran de uso general, o sea, que pudieran utilizarse para el desarrollo de disímiles tipos de soluciones, debido a que la implementación del jurado en línea se enmarca tanto en el desarrollo web como en el de escritorio. Tener en cuenta este indicador favorece el uso de la menor cantidad de herramientas de un mismo tipo, lo que conlleva a la obtención de un producto final más fácil de mantener, facilitando así el proceso de soporte o la realización de futuras versiones.

Se consideró además la fiabilidad de las herramientas candidatas, o sea, que ya hayan sido usadas anteriormente de manera exitosa en proyectos de mediana o mayor envergadura, demostrando así la

robustez de las mismas. Se valoró también que dichas candidatas tuvieran un buen soporte y mantenimiento, es decir, que contaran con un grupo de personas oficialmente encargadas de mejorarlas y documentarlas.

Para el caso específico de los lenguajes de programación candidatos se tuvo en cuenta además que implementaran, como mínimo, el paradigma de Programación Orientada a Objetos (POO). Este paradigma permite la resolución de problemas de alta complejidad (14), como lo es el desarrollo de un jurado en línea.

Para los servidores distribuidores de mensajes se tuvo en cuenta además que los mismos implementaran el protocolo de comunicación AMQP (del inglés Advanced Message Queuing Protocol)[31] en su versión 0.8 como mínimo.

Para los servidores web candidatos se previó además que soportaran la tecnología CGI (del inglés Common Gateway Interface)[32], lo que permite la implementación de la aplicación web en el lado del servidor utilizando lenguajes de escritorio. Actualmente, las aplicaciones no interactúan con el estándar CGI directamente, sino mediante marcos de trabajo (conocidos en inglés como *frameworks*) para desarrollo web (15).

A continuación se caracterizan los candidatos escogidos de acuerdo a los indicadores descritos.

1.3.2 Lenguajes de programación

Los lenguajes de programación son idiomas artificiales diseñados para la creación de programas cuyo fin es representar algoritmos precisos que controlen el comportamiento de los dispositivos de *hardware* y de *software*. Agrupan símbolos, instrucciones y reglas sintácticas y semánticas que expresan su estructura. Sus expresiones facilitan la escritura, prueba y mantenimiento del código fuente, siendo fácilmente escritas y leídas por personas durante el proceso de programación (16).

1.3.2.1 Python

Python es un lenguaje de *script* [33] (o interpretado) de propósito general. Es multiplataforma, permite tipado dinámico y es fuertemente tipado. Soporta la POO y la programación imperativa y en menor medida el paradigma de programación funcional. Posee la licencia de código abierto Python Software Foundation License, la cual es compatible con la licencia GNU GPL (del inglés GNU General Public License), lo que lo convierte en *software* libre. Es mantenido por una amplia comunidad de desarrolladores denominada

[31] AMQP es un protocolo de estándar abierto en la capa de aplicaciones de un sistema de comunicación que estipula el comportamiento tanto del servidor que provee los mensajes como del cliente de la mensajería. Es muy útil en lo que a cómputo distribuido se refiere, entre otras aplicaciones.

[32] CGI es una tecnología de la World Wide Web (www) que permite a un cliente (navegador web) solicitar datos de un programa ejecutado en un servidor web. Es un mecanismo de comunicación entre el servidor web y una aplicación externa.

[33] Un script o archivo de procesamiento por lotes es un programa usualmente simple, que por lo general se almacena en un archivo de texto plano.

Python Software Foundation. Posee una extensa colección de bibliotecas para disímiles usos, como el acceso a bases de datos, la interacción con servidores distribuidores de mensajes que implementen AMQP, y para desarrollo web. Según el desarrollador de Python Tim Peters en "El Zen de Python"[34], este lenguaje tiene entre sus principios: explícito es mejor que implícito, simple es mejor que complejo, disperso es mejor que denso, los errores nunca deberían dejarse pasar silenciosamente, si la implementación es difícil de explicar, es una mala idea; estos principios convierten a Python en un lenguaje con una sintaxis muy limpia por lo que es muy fácil de aprender. Grandes organizaciones han empleado este lenguaje en algunos de sus productos. La Administración Nacional de Aeronáutica y del Espacio de los Estados Unidos (conocida por sus siglas en inglés como NASA) usa Python para varios de sus sistemas de *software* y lo ha adoptado como lenguaje de *scripting* estándar para su Sistema de Planificación Integrado; Yahoo lo utiliza para administrar sus grupos de discusión, mientras que Google lo ha empleado para implementar muchos de los componentes de su rastreador web y motor de búsqueda (17).

1.3.2.2 Java

Java es un lenguaje de programación desarrollado por la empresa Sun Microsystems, que luego fue adquirida por la empresa Oracle. Un programa escrito en Java consiste en una serie de instrucciones que son interpretadas por un *software* denominado máquina virtual de Java. Existen varias implementaciones de este *software*, entre las que se encuentra Java Sun que posee la licencia libre GNU GPL versión 2.0. Java es un lenguaje sencillo capaz de generar código de tamaño muy reducido, por lo que posee una curva de aprendizaje muy rápida. Soporta la POO y la programación imperativa. Es un lenguaje multiplataforma, de lo que se deriva la portabilidad de sus programas, lo que se logra gracias al Java Runtime Environment (JRE)[35]. Java es un lenguaje interpretado, fuertemente tipado, posee una sintaxis similar a C/C++ e integra además tratamiento genérico de tipos. Incluye librerías estándar para interfaces de usuario, objetos distribuidos, XML (del inglés Extensible Markup Language)[36], web, móviles, televisión, entre otras. La ejecución de programas desarrollados en Java puede ser, entre otras, ejecución como aplicación independiente, conocida en inglés como Stand-alone Application y ejecución como *applet*[37] (18). Actualmente Java se ejecuta en más de 850 millones de ordenadores personales de todo el mundo y en millones de dispositivos. De igual manera existe un gran número de aplicaciones y sitios web que no funcionan a menos que Java esté instalado (19).

---

[34] El Zen de Python es una enumeración de los principios de diseño y la filosofía de Python útiles para comprender y utilizar el lenguaje.

[35] El JRE es un conjunto de utilidades que permite la ejecución de programas Java. Está conformado básicamente por una máquina virtual de Java y un conjunto de bibliotecas Java y actúa como un "intermediario" entre el sistema operativo y Java.

[36] XML es un lenguaje de marcas que se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas. Es útil cuando varias aplicaciones se deben comunicar entre sí o integrar información.

[37] Un *applet* es un componente de una aplicación que se ejecuta en el contexto de otro programa, por ejemplo en un navegador web.

1.3.2.3 C++

El lenguaje C++ fue desarrollado en la compañía AT&T (del inglés American Telephone and Telegraph). Fue creado para extender el lenguaje de programación C. Al ser un lenguaje compilado, existen varias implementaciones de su compilador. Uno de ellos es el GCC (del inglés GNU Compiler Collection), que se distribuye bajo la licencia GNU GPL lo cual lo convierte en un compilador de libre distribución. Es la organización Free Software Foundation la que brinda el soporte para este compilador. C++ es de propósito general y mantiene las ventajas de C en cuanto a riqueza de operadores y expresiones, flexibilidad, concisión y eficiencia. Es compatible casi completamente con su antecesor, pero elimina algunas dificultades y limitaciones del C original, pues se le han añadido nuevos tipos de datos, clases, plantillas, mecanismo de excepciones, sistema de espacios de nombres, sobrecarga de operadores, referencias y operadores para manejo de memoria persistente. Es orientado a algoritmos, o sea es un lenguaje procedural. Es multiplataforma y multiparadigma debido a que es orientado a objetos y permite además la programación imperativa y genérica. Al igual que C, mantiene una considerable potencia para programación a bajo nivel, aunque se le han añadido elementos que le permiten también una programación con altos niveles de abstracción. El sistema de base de datos MySQL, el cliente de correo electrónico Thunderbird y el editor de páginas web Adobe Dreamweaver son ejemplos de *softwares* que han sido implementados en C++. Linux está escrito en el lenguaje de programación C, en la variante utilizada por el compilador GCC (que ha introducido extensiones y cambios al C estándar) (20).

1.3.3 Framework para desarrollo web

Un *framework* web, es un conjunto de componentes (por ejemplo clases en Java y descriptores y archivos de configuración en XML) que componen un diseño reutilizable que facilita y agiliza el desarrollo de sistemas web (21).

1.3.3.1 Django

Django es un *framework* de desarrollo web. Es de código abierto, bajo la licencia BSD (del inglés Berkeley Software Distribution) y es mantenido por la comunidad de desarrollo Django Software Foundation, lo que ayuda a que se mantenga actualizado, se detecten y corrijan sus errores y cuente con documentación actualizada y detallada. Django es conocido como "el *framework* web para perfeccionistas con fechas de entrega", debido a que facilita la tarea de desarrollo, a la vez que permite la implementación de un buen código. Propone estructurar los sitios web en proyectos, los cuales contienen la configuración general del sitio y en aplicaciones, las cuales contienen la funcionalidad en sí. Alienta a que las aplicaciones sean lo más desacopladas posible, de esta manera pueden reutilizarse en más de un proyecto. Django incluye muchas aplicaciones comunes a todos los sitios web, como la autenticación de usuarios o la administración del contenido del sitio. Su arquitectura está inspirada en el patrón Modelo-Vista-

Controlador (MVC), encargándose en gran parte de los controladores, y proveyendo herramientas para facilitar el desarrollo de las vistas y los modelos. Django posee además soporte nativo para PostgreSQL como gestor de bases de datos. Entre los sitios que usan este *framework* actualmente se encuentran, entre otros, Mozilla y OpenStack[38] (15).

1.3.4 Gestores de bases de datos

Un Sistema Gestor de Bases de Datos (SGBD), es el *software* que permite a los usuarios procesar, describir, administrar y recuperar los datos almacenados en una base de datos. Proporcionan un conjunto coordinado de programas, procedimientos y lenguajes que permiten a los usuarios la manipulación de los datos, garantizando además la seguridad de los mismos (22).

1.3.4.1 PostgreSQL

PostgreSQL es un SGBD objeto-relacional. Se encuentra distribuido bajo la licencia de *software* libre BSD. Es mantenido por la organización PostgreSQL Global Development Team y existe además una amplia comunidad de usuarios y programadores que colaboran activamente. PostgreSQL está disponible en un amplio rango de plataformas. Es altamente configurable y extensible a disímiles tipos de aplicación. Se centra en que el *software* sea robusto, de calidad, fácil de mantener y con un código bien comentado, además cumple con los estándares SQL de interoperabilidad y compatibilidad, y se destaca por su estabilidad, potencia, robustez y facilidad de administración. Utiliza el conocido modelo cliente-servidor. Usa multiprocesos en lugar de multihilos lo que garantiza la estabilidad del sistema, pues un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando. Entre sus principales características se encuentran el control concurrente multiversión (conocido por sus siglas en inglés como MVCC)[39], múltiples métodos de autenticación, uso de procedimientos almacenados, Application Programming Interface (API)[40] para programar en C/C++, Java y Python, entre otros. Funciona muy bien con grandes cantidades de datos y una alta concurrencia de usuarios accediendo paralelamente al sistema (23). Disímiles son las empresas y compañías que construyen sus productos, sitios web o herramientas utilizando este gestor, entre ellas Debian, Apple, Fujitsu, Red Hat y Sun Microsystems (24).

1.3.4.2 MySQL

MySQL es un SGBD relacional. Posee un licenciamiento dual pues se ofrece tanto bajo la licencia GNU GPL para cualquier uso compatible con esta licencia, como para aquellas empresas que quieran incorporarlo en productos privativos bajo una licencia comercial. Es ampliamente conocido y utilizado por

[38] http://www.openstack.org/

[39] El MVCC permite el acceso a una tabla sin necesidad de bloqueos aunque otros procesos estén escribiendo en la misma.

[40] Un API es un conjunto de funciones y procedimientos que ofrece una biblioteca para ser utilizado por otro *software* como una capa de abstracción.

su simplicidad y notable rendimiento. La empresa MySQL AB, además de ser la encargada de la venta de licencias privativas y poseer los derechos sobre la mayor parte del código, es la que ofrece el soporte. Este gestor es multiplataforma y tiene un diseño multihilo, lo que le permite soportar una carga considerable de manera eficiente. Entre sus principales características se encuentran el ser multiusuario, contar con varias API que posibilitan a aplicaciones escritas en diversos lenguajes de programación acceder a las bases de datos MySQL, incluyendo, entre otros, C++, Java y Python; uso de procedimientos almacenados y vistas actualizables. Según cifras del fabricante, en la actualidad existen más de seis millones de copias de MySQL funcionando y cuenta con usuarios como Google, Yahoo, Nokia y Wikipedia (25).

1.3.5 Servidores distribuidores de mensajes

Los servidores distribuidores de mensajes son sistemas que permiten a los programas comunicarse mediante el intercambio de mensajes. Garantizan la entrega, velocidad y seguridad de los mismos, así como la ausencia de mensajes no deseados (26).

1.3.5.1 Protocolo AMQP

El estándar AMQP (Advanced Message Queuing Protocol) es un protocolo de estándar abierto en la capa de aplicaciones de un sistema de comunicación. Las características que definen al protocolo AMQP son la orientación a mensajes y a colas, el enrutamiento, la exactitud y la seguridad. El modelo de la arquitectura de AMQP está orientado a resolver problemas de procesamiento de datos mediante la abstracción del problema de productores-consumidores, donde los productores son los programas que envían mensajes y los consumidores aquellos que los reciben. Los mensajes constituyen cualquier conjunto de bytes que se desee enviar, son enviados a colas, garantizando que sean entregados en el mismo orden en que son enviados. Para recibir mensajes, la cola primeramente debe tener establecido un enlace o vínculo con un intercambiador que es el encargado de entregar los mensajes a las colas correspondientes. Parte de establecer la unión incluye especificar qué mensajes deben ser enviados a las colas (27). Tanto los mensajes, como las colas y los vínculos entre el intercambiador y las colas, poseen propiedades configurables, lo que hace de AMQP un protocolo flexible, al no tener una preconfiguración estática, así todos los recursos pueden ser creados y destruidos dinámicamente por los clientes. AMQP es un modelo simple que cuenta además con disímiles librerías en diversos lenguajes.

1.3.5.2 RabbitMQ

RabbitMQ es un servidor de mensajería que implementa el protocolo AMQP y está escrito en el lenguaje de programación Erlang. Es de código abierto bajo la licencia Mozilla Public License versión 1.1. Es comercialmente soportado por la organización Rabbit Technologies Ltd. (27). Permite una alta tolerancia a fallos pues soporta nativamente la persistencia, la cual consiste en almacenar los mensajes, colas e

intercambios en el disco duro de tal forma que si existe un fallo en el sistema no se pierdan los mismos (28) , y brinda notificación de entrega del mensaje. Con este servidor una sola cola de mensajes es capaz de almacenar cerca de 30.000 mensajes por segundo, dependiendo de la cantidad y carga de los clientes, así como de las propiedades y tamaño de los mensajes (29). El reconocido proyecto OpenStack lo emplea para controlar el envío de mensajes en sus diferentes servicios (30).

1.3.5.3 ApacheQpid

ApacheQpid es un servidor de mensajería que implementa el protocolo AMQP y está escrito en los lenguajes de programación C++ y Java. Es un proyecto de código abierto que pertenece a la organización Apache Software Foundation y se distribuye bajo la licencia Apache License 2.0 (26). Soporta la persistencia pero mediante *plugins*[41] que ni siquiera son mantenidos por Apache (31). La velocidad de este servidor en modo no persistente ronda en los 5000 mensajes por segundo, mientras que en modo persistente se observan cifras de 1100 mensajes por segundo (29).

1.3.6 Servidor web

Un servidor web es un programa que se ejecuta de manera continua en un ordenador a la espera de peticiones por parte de un cliente (navegador de Internet). La respuesta a estas peticiones puede ser una página web que será mostrada en el navegador o un mensaje si se detectó algún error (32).

1.3.6.1 Apache

El servidor web Apache es un servidor web HTTP (del inglés Hypertext Transfer Protocol)[42] de código abierto. Se desarrolla dentro de la Apache Software Foundation y es administrado en conjunto por una extensa comunidad de usuarios. Es altamente configurable y posee un diseño modular, de esta manera es muy fácil extender sus capacidades, pues existen disímiles módulos de Apache disponibles. Trabaja cómodamente con diferentes lenguajes de *scripting*, entre los que se encuentra Python. Este servidor web es simple pero robusto, pues aunque no cuenta con una interfaz gráfica para administrarlo, es altamente configurable. Apache brinda además una gran flexibilidad en el registro y seguimiento del contexto del propio servidor y permite personalizar los archivos de registros convenientemente. Es de uso general, pues incluye entre otros, soporte para SSL (del inglés Secured Socket Layer)[43], para CGI, que es usado de manera transparente por el *framework* para desarrollo web Django; para *hosts* virtuales y para autenticación por HTTP (33). Apache es en la actualidad el número uno entre los servidores web utilizados

[41] Un *plugin* es un módulo de *hardware* o *software* que añade una característica o un servicio específico a un sistema más grande.

[42] HTTP es el protocolo usado en cada transacción de la World Wide Web. Define la sintaxis y la semántica que utilizan los elementos de *software* de la arquitectura web para comunicarse.

[43] SSL es un protocolo diseñado para proveer comunicaciones encriptadas en Internet, haciendo posible que solo el servidor y el cliente entiendan un texto.

en el mundo, pues el número de sitios web que lo utilizan asciende a 152 millones, acaparando casi el 60 % del mercado de servidores web (34).

1.3.7 Sistemas operativos

Un sistema operativo es un conjunto coherente de código que gestiona el *hardware* de un sistema de cómputo creando facilidades para la ejecución sobre el mismo de todo tipo de programas. Es un administrador de los recursos de *hardware* del sistema que protege al usuario de los detalles y complejidades del mismo (35).

1.3.7.1 Debian

Debian es un sistema operativo de propósito general, desarrollado sobre Linux y basado en *software* libre bajo la licencia GNU GPL versión 2.0. Usuarios y desarrolladores activos alrededor del mundo ayudan voluntariamente y trabajan de manera conjunta conformando una comunidad conocida como Proyecto Debian. Este proyecto se encuentra representado por la organización Software Public Interest y aunque es libre permite a personas o empresas distribuirlo comercialmente siempre que respete los términos de su licencia (36). Se rige por tres documentos fundamentales, el Contrato Social de Debian[44], las Directrices de *software* libre de Debian [45] y la Constitución de Debian[46]. Posee un proceso de prueba del *software* muy robusto pues divide el desarrollo en las ramas inestable, en pruebas y estable (37). La instalación de este sistema puede configurarse fácilmente para cumplir diversas funciones, ya sea la de cortafuegos, simple estación de trabajo o servidor de red de alto rendimiento. Debian fue la primera distribución de Linux en incluir un gestor de paquetes, en este caso el APT (del inglés Advanced Packaging Tool)[47], que permite una fácil instalación y desinstalación del *software* y una actualización sin necesidad de reinstalación (o sea, de tipo *rolling release*). En caso de detección de problemas de seguridad en los paquetes ya distribuidos, se publican rápidamente parches para eliminarlos que pueden ser descargados e instalados sencillamente. Debian está diseñado para funcionar en casi todos los ordenadores personales, cada nueva versión soporta un mayor número de arquitecturas de ordenadores, entre las que se encuentran las basadas en Intel x86, AMD64 e Intel EM64T (38). Disímiles organizaciones usan Debian, entre ellas, el grupo empresarial CubaNíquel y el Banco Comercial de Venezuela (39).

1.3.7.2 Fedora

Fedora es un sistema operativo de propósito general basado en *software* libre. Usa Linux como núcleo. Es patrocinado por Red Hat y tiene como premisa fundamental la colaboración. Es mantenido por el Proyecto

[44] http://www.debian.org/social_contract
[45] http://www.debian.org/social_contract.es.html#guidelines
[46] http://www.debian.org/devel/constitution
[47] APT es un sistema de gestión de paquetes creado por el proyecto Debian que simplifica en gran medida, mediante líneas de comandos, la instalación y eliminación de programas en los sistemas GNU/Linux.

Fedora, una comunidad en la que sus desarrolladores trabajan con equipos alrededor del mundo que se denominan ramas de desarrollo, garantizando así una mejor experiencia y las mejoras pertinentes en beneficio de los usuarios y las comunidades de desarrollo de *software*. Cada año se liberan dos versiones de Fedora para garantizar un *software* vanguardista. Su distribución se realiza bajo los términos de la licencia GNU GPL (40). Este sistema operativo garantiza que sus actualizaciones estén disponibles para todas las variantes de GNU/Linux debido a que, en lugar de aplicar los parches específicos en su distribución, sus desarrolladores hacen cambios en las fuentes originales. Entre sus características de seguridad se destaca la Security-Enhaced Linux (SELinux)[48]. Posee además un administrador de paquetes denominado Yum y la alternativa de utilizar comandos APT si se está acostumbrado a trabajar con Debian o Ubuntu. Utiliza únicamente repositorios que disponen de paquetes de *software* libre o de código abierto. Soporta versiones actualizadas de lenguajes como Python y Perl, entre otros, y las plataformas x86 y x86-64 (41). Fedora ya es la base de distribuciones derivadas, como Linux para empresas de Red Hat (40).

1.3.7.3 Ubuntu

Ubuntu es un sistema operativo de propósito general basado en la herencia de Debian GNU/Linux. La visión para este sistema es en parte social, *software* libre bajo los términos de la licencia GNU GPL disponible para todos por igual, y en parte económica, con disímiles servicios provistos por Canonical Ltd. Ambas partes colaboran para producir un *software* de alta calidad, de esta manera el Proyecto Ubuntu es el trabajo compartido entre Canonical, otras compañías y cientos de voluntarios alrededor del mundo que intercambian experiencias para mejorarlo (42). Ubuntu puede utilizarse tanto para simples estaciones de trabajo como para servidores, pues tiene opciones preconfiguradas de instalación. Se caracteriza además por ciclos de liberación de seis meses. Cuenta con cientos de paquetes que son administrados con el uso de APT/Synaptic[49], con un *framework* de seguridad denominado AppArmor[50] y posee soporte para arquitecturas como x86 y AMD64 (43). Actualmente este sistema operativo es preinstalado en computadoras fabricadas por la internacionalmente reconocida Dell o Lenovo, entre otras empresas a nivel mundial (42).

1.3.8 Entornos de desarrollo integrado

Un entorno de desarrollo integrado (conocido por sus siglas en inglés como IDE) es un programa compuesto por una serie de herramientas que utilizan los programadores para desarrollar código. Puede

[48] SELinux es una arquitectura de seguridad integrada en el *kernel* 2.6.x usando los módulos de seguridad Linux.

[49] Synaptic es una interfaz gráfica para apt. Combina la simplicidad de la interfaz gráfica de usuario con la potencia de la herramienta de línea de comandos apt.

[50] AppArmor es un programa de seguridad para Linux que permite al administrador del sistema asociar a cada programa un perfil de seguridad que restrinja las capacidades de ese programa proporcionando el control de acceso obligatorio (conocido por sus siglas en inglés como MAC).

estar diseñado para utilizarse con uno o varios lenguajes de programación. Entre las herramientas que componen un IDE se encuentran: un editor de texto, un compilador, un intérprete y un depurador (44).

1.3.8.1 Eclipse

Eclipse es un entorno de desarrollo integrado desarrollado por completo en el lenguaje de programación Java. Es una de las herramientas que se desarrollan dentro del Proyecto Eclipse. Es mantenido por la organización Eclipse Foundation que fomenta una comunidad de código abierto así como un conjunto de productos, capacidades y servicios (45). Se distribuye bajo la licencia de *software* libre Eclipse Public License (46). Es un IDE multiplataforma que mediante la implementación de los *plugins* correspondientes, permite integrar diversos lenguajes de programación, pues no se encuentra orientado a ninguno en específico, así por ejemplo existen *plugins*, entre otros, para C/C++, Pascal y Python y permite además introducir herramientas UML (del inglés Unified Modeling Language)[51], ayudas en línea para librerías, entre otras. Integra la librería SWT (Standard Widget Toolkit)[52] que permite que la ejecución de las interfaces de usuario sea rápida y fluida. Entre sus principales características se encuentran: editor de texto, vistas, resaltado de sintaxis, refactorización, completamiento de código, compilación en tiempo real, pruebas unitarias con JUnit[53], *frameworks* para el desarrollo de aplicaciones gráficas así como para aplicaciones web. Eclipse ha sido utilizado para desarrollar varios IDE como el Java Development Toolkit (JDT)[54] y el compilador Eclipse Compiler for Java (ECJ)[55] (45).

1.3.8.2 Netbeans

Netbeans es un proyecto de código abierto sin restricciones para su uso comercial o no comercial, desarrollado en Java, fácil de usar y famoso entre los programadores de Java. Es mantenido por la empresa Sun MicroSystems y una comunidad de usuarios en constante crecimiento. Se distribuye bajo la licencia Common Development and Distribution License (CDDL) y GNU GPL versión 2.0 (47). Es un IDE multiplataforma. Está orientado a la modularidad, pues existe un considerable número de *plugins* para extenderlo. Gracias a los *plugins* es posible el uso de disímiles lenguajes de programación como C/C++, Java y Python, entre otros. Incluye herramientas de desarrollo visuales, herramientas de esquemas y modelado UML. Permite el desarrollo de aplicaciones de escritorio, web, móviles, entre otras, así como de todos los tipos de aplicaciones Java. Entre sus principales características se encuentran: editor de código,

[51] UML es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. Ofrece un estándar para describir un "plano" del sistema (modelo).

[52] SWT es una librería de *widgets* con la que se aprovechan los *widgets* nativos del sistema sobre el que se ejecuta.

[53] JUnit es un *framework* Open Source para la automatización de las pruebas (tanto unitarias, como de integración) en los proyectos de *software*, pues provee herramientas, clases y métodos que facilitan esta tarea.

[54] JDT es un subproyecto del proyecto Eclipse que desarrolla herramientas para la programación en Java. Esto incluye un compilador de Java, constructor incremental, editores, asistentes, asistente de contenido, entre otras características.

[55] ECJ es un compilador de Eclipse para Java, es un compilador incremental de código abierto utilizado por Eclipse JDT que proporciona las bibliotecas y herramientas relacionadas con la implementación de Java de GNU.

completamiento de código, resaltado de sintaxis, instalación y actualización simple, refactorización y control de versiones. Actualmente NetBeans se ha establecido como la plataforma preferida de muchos de los desarrolladores tecnológicos para sus empresas, por ejemplo la Sun Mycrosistems lo ha utilizado para realizar algunos de sus productos (48).

1.3.9 Selección definitiva de herramientas y tecnologías a utilizar

Se seleccionó Python, en su versión 2.7 o superior, como lenguaje de programación principal y C++, en la versión 4.7 de GCC o superior, para la implementación de las partes claves del sistema que necesiten una elevada eficiencia. Para realizar esta selección se consideró que la eficiencia era el factor de mayor importancia[56]. Inicialmente se valoró el uso de C++ para la implementación de todo el sistema debido a que es el más eficiente de los candidatos (49) (50) (51) (52), sin embargo, esta posibilidad se desechó debido a la alta complejidad del lenguaje y a la carencia de un *framework* potente para el desarrollo web; entonces la elección se centró entre Java y Python. Se escogió finalmente Python y no Java debido principalmente a tres factores: integración con C++, *framework* para desarrollo web y complejidad del lenguaje. Se tomó en cuenta la integración con C++ para de esta manera asegurar la eficiencia en las partes que más lo requieran. Existe la posibilidad de implementar librerías en C++ que pueden ser integradas con Python como si las mismas se hubieran escrito en este último, lográndose así una integración casi transparente (53). Python cuenta además con Django, un potente y probado *framework* para el desarrollo web que se utiliza en su versión 1.5 o superior (15). En cuanto a la complejidad del lenguaje, Python es sin dudas el más sencillo, con una sintaxis tan comprensible como un seudocódigo pero al mismo tiempo tan potente como los lenguajes de alto nivel actuales (54) (55) (56) (57). Python, además del paradigma orientado a objetos, permite la utilización de gran parte del paradigma de programación funcional, lo que simplifica aún más su sintaxis al mismo tiempo que agrega funcionalidad (55).

Se seleccionó Debian, en su versión 6.0 o superior, como sistema operativo para los servidores. Para realizar esta selección se tuvo en cuenta que el candidato elegido favoreciera la eficiencia, parámetro fundamental a tener en cuenta en un sistema de alta respuesta como lo es un jurado en línea. En este sentido se descartó Ubuntu, ya que es una distribución basada a su vez en Debian (58), lo que trae consigo la inclusión de programas y configuraciones que no son necesarias en todos los escenarios y que se mantienen ejecutándose sobre el sistema aunque no se usen. La mayoría de estos programas no pueden desinstalarse de manera sencilla sin desestabilizar el sistema, pues se encuentran muy ligados al mismo; por lo que es más sencillo instalar sobre Debian estrictamente las aplicaciones que sean necesarias, razón por la cual se consideró que la instalación inicial del sistema operativo debía contar solo

[56] Un jurado en línea debe ser un sistema con un tiempo de respuesta rápido, pues en una competencia de programación un segundo de espera puede ser determinante dentro de la misma.

con un conjunto mínimo de aplicaciones para hacerlo funcionar, de manera tal que pudiera configurarse para un uso especializado mediante la posterior instalación de programas específicos. Lo anteriormente planteado requiere un mayor tiempo de configuración inicial, pero permite aumentar la eficiencia del sistema así como disminuir su complejidad. Debian puede moldearse y configurarse como se desee y posee además opciones para la instalación de programas comúnmente usados (36). En este momento del análisis se valoraron dos opciones: Debian y Fedora. Debian posee un poderoso gestor de paquetes y es una distribución de tipo *rolling release*, lo que incrementa la seguridad y disponibilidad de los servidores al ser posible la actualización con regularidad de sus sistemas operativos sin necesidad de detenerlos por mucho tiempo. Aunque Fedora también cumple con esta última característica, Debian, a diferencia de Ubuntu y Fedora, posee un proceso de prueba del *software* más sofisticado lo cual lo convierte en un sistema extremadamente estable. Este proceso consiste en dividir el desarrollo en las ramas inestable, en pruebas y estable. Cada paquete pasa por todas las ramas y en cada una de ellas se le realizan pruebas exhaustivas hasta llegar a la rama estable (37), por lo que la versión estable de Debian tarda en liberarse, pero el resultado es de una confiabilidad elevada; debido a estas razones se decidió finalmente usar Debian y no Fedora como sistema operativo para los servidores.

Se seleccionó PostgreSQL, en su versión 9.0 o superior, como gestor de bases de datos y se descartó la posibilidad de utilizar MySQL. Para realizar esta selección se tuvo en cuenta que existiera un balance entre integridad, eficiencia y funcionalidades para la consulta de los datos. PostgreSQL es eficiente en escenarios donde el tráfico de datos sea alto y exista una alta concurrencia de usuarios (23), pues gracias al acceso concurrente multiversión, en este tipo de circunstancias no presenta incoherencias en la escritura de los datos. Es además altamente tolerante a fallos debido al uso de multiprocesos en lugar de multihilos. Las características anteriores son únicas de PostgreSQL, lo que lo convierte en la mejor opción para un sistema con los escenarios antes descritos y de tiempos de respuesta rápidos como lo es un jurado en línea.

Para la comunicación entre los diferentes componentes del sistema se seleccionó RabbitMQ, en su versión 2.8.4 o superior, como servidor distribuidor de mensajes. Para realizar esta selección se tuvo en cuenta principalmente la eficiencia. ApacheQpid fue descartado pues normalmente es más lento que RabbitMQ en circunstancias similares, enviando entre 5000 y 1100 mensajes por segundo mientras que RabbitMQ es capaz de enviar 30000 mensajes por segundo (29).Se consideró además el hecho de que el sistema fuera altamente tolerante a fallos, o sea, que se perdiera la menor cantidad de mensajes, lo que disminuye el riesgo de que el sistema no responda debido a la espera infinita de respuesta de un mensaje previo. En este sentido RabbitMQ soporta nativamente la persistencia, almacenando los mensajes, colas e intercambios en el disco duro de tal forma que si existe un fallo en el sistema no se pierdan los mismos, (28) mientras que ApacheQpid soporta esta característica mediante *plugins* que no son mantenidos por

Apache y cuya configuración puede ser trabajosa (31). RabbitMQ soporta además notificación de entrega, lo que asegura que un mensaje solo será borrado cuando un consumidor haya terminado de procesarlo. A diferencia de ApacheQpid, RabbitMQ puede configurarse de manera fácil y rápidamente, además de que el trabajo con sus librerías clientes es muy simple, contándose con una amplia y comprensible documentación[57], así como una serie de intuitivos tutoriales[58] que facilitan el estudio de la herramienta, de las librerías clientes para interactuar con el servidor y del protocolo de comunicación AMQP. Además ApacheQpid se encuentra en un estado de desarrollo en el que aún no ha alcanzado la versión 1.0 (59), mientras que RabbitMQ ya cuenta con la versión 3.0 (60) , razón por la que se considera un servidor más robusto.

Se seleccionó Apache, en su versión 2.0 o superior, como servidor web. Apache constituía el único candidato de acuerdo a los indicadores que se definieron para escoger los mismos. No se consideraron otros servidores como candidatos principalmente porque no cumplían con el indicador de ser de uso general o de contar con alta robustez o soporte y mantenimiento. Tal es el caso, por ejemplo, de Lighttpd (61), el cual a pesar de ser muy eficiente (62), constituye un servidor web muy ligero con una comunidad de desarrollo muy reducida (63) y con muy pocas funcionalidades (64). Un caso similar es el de Cherokee, que fue desarrollado hace relativamente muy poco tiempo (65).

Se seleccionó Eclipse, en su versión 3.8, como entorno de desarrollo integrado debido principalmente a la carencia de un *plugin* robusto para Netbeans que permitiera el trabajo en Python (66). Para Eclipse existe el *plugin* Pydev, el cual se libera bajo la licencia de código abierto Eclipse Public License y es mantenido por la empresa Appcelerator Inc. Pydev brinda integración con Django, completamiento de código, análisis del código, refactorización, depuración, analizador sintáctico y semántico e integración de pruebas unitarias (67).

1.4 Metodologías de desarrollo de software

Desarrollar un *software* de calidad conlleva a un sinnúmero de actividades y etapas, las cuales deben ser gestionadas de manera efectiva por la metodología de desarrollo de *software* adecuada. Una metodología de desarrollo de *software* es básicamente una filosofía que tiene el objetivo de planificar, estructurar y controlar el proceso de desarrollo de *software*, guiándolo hasta el logro del producto final (68).

Los requisitos de un *software* a otro son muy variados y cambiantes, por lo que existen disímiles metodologías para la creación del mismo que se clasifican en dos grandes grupos, el de las metodologías pesadas o tradicionales y el de las metodologías ligeras o ágiles.

[57] http://www.rabbitmq.com/documentation.html
[58] http://www.rabbitmq.com/getstarted.html

1.4.1 Metodologías pesadas

Estas metodologías se denominan pesadas precisamente porque se centran en la definición detallada y rigurosa de los roles, artefactos y tareas a realizar y generan una extensa documentación. Son efectivas y necesarias en proyectos con un tamaño considerable, aunque el costo de implementar un cambio es alto y no ofrecen una buena solución para proyectos donde el entorno es volátil (68). Entre las principales metodologías tradicionales se encuentran RUP (del inglés Rational Unified Process) y MSF (del inglés Microsoft Solution Framework), las cuales se describen a continuación.

1.4.1.1 Rational Unified Process (RUP)

RUP se encuentra dentro del grupo de metodologías tradicionales o pesadas. Desarrollada por Rational Software, se encuentra integrada con toda la suite Rational de herramientas. Puede ser adaptada y extendida para satisfacer las necesidades de la organización que la adopte (69).

Las cuatro fases del ciclo de vida de RUP son: Inicio, en la que se define el modelo del negocio y el alcance del proyecto; Elaboración, durante la que se establecen los cimientos de la arquitectura; Construcción, en la que se alcanza la capacidad operacional del producto; y Transición, en la que, tal como indica su nombre, se pone el producto en manos de los usuarios finales. Durante este ciclo las actividades a desarrollar se organizan en nueve flujos de trabajo, de los que seis se denominan flujos de ingeniería y el resto, flujos de apoyo. Los flujos de trabajo son: Modelamiento del negocio, Requerimientos, Análisis y diseño, Implementación, Prueba, Instalación, Administración del proyecto, Administración de configuración y cambios y Ambiente.

La siguiente figura (69) muestra las fases y flujos de RUP:

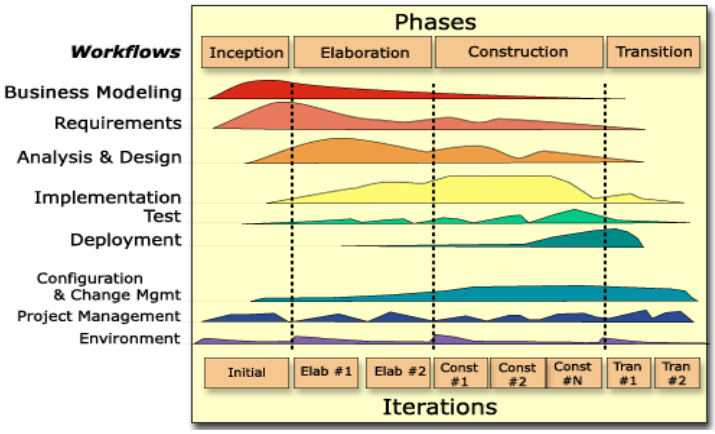

Figura 1. Fases y flujos de RUP

RUP requiere un grupo amplio de programadores, por lo que los equipos son de considerable tamaño. Esta metodología se caracteriza por tres rasgos esenciales: proceso dirigido por casos de usos, pues estos guían las cuatro fases de RUP, integrando todo el trabajo durante el transcurso de las mismas; centrado en la arquitectura, pues brinda especial atención al establecimiento temprano de una arquitectura

sólida que no se vea demasiado afectada ante la ocurrencia de un cambio durante las últimas etapas del desarrollo del *software*; y es un proceso iterativo e incremental, pues propone que el trabajo se divida en partes más pequeñas conocidas como miniproyectos. Cada miniproyecto constituye una iteración, o sea, un recorrido a lo largo de todos los flujos de trabajo fundamentales, del cual se obtiene un incremento del producto (69).

1.4.1.2 Microsoft Solution Framework (MSF)

MSF es una metodología pesada un poco más flexible, que pudiera definirse como un compendio de las mejores prácticas en cuanto a administración de proyectos se refiere. No constituye una metodología rígida de administración de proyectos, sino una serie de modelos adaptable a cualquier proyecto de tecnología de información. Tiene como principios base: promover comunicaciones abiertas, trabajar para una visión compartida, otorgar autoridad y poder de decisión a los equipos de trabajo, establecer responsabilidades claras y compartidas, focalizarse en agregar valor al negocio, permanecer ágil y esperar los cambios, invertir en calidad y aprender de todas las experiencias. Está compuesta por catorce corrientes de trabajo, compuestas a su vez por actividades más pequeñas, entre estas corrientes se encuentran: definir la visión del producto, crear requerimientos de calidad de servicio, planificar iteraciones, crear arquitectura de la solución e implementar tareas de desarrollo (70).

MSF propone cinco fases para guiar el proceso de desarrollo de *software*: Visión y alcance, en la que se persigue la unificación del equipo tras una visión común; Planificación, cuando la mayor parte de la planeación para el proyecto es terminada; Desarrollo, donde se realiza la mayor parte de la construcción de los componentes en términos de documentación y código; Estabilización, la cual conlleva a realizar pruebas sobre la solución; e Implantación, en la que se estabiliza la instalación y se obtiene la aprobación final del cliente (71).

La siguiente figura (71) muestra las fases de MSF:

Figura 2. Fases de MSF

Esta metodología propone el modelo de equipos de MSF para compensar desventajas de las estructuras de los equipos de los proyectos tradicionales. De esta manera los equipos organizados bajo este modelo pueden ser pequeños y multidisciplinarios, de tres a cuatro personas, o grandes proyectos que requieran cincuenta o más personas. Cuenta con seis roles igualmente importantes en su aporte al proyecto y cada rol puede estar compuesto por una o más personas (70).

1.4.2 Metodologías ágiles

Estas metodologías ofrecen una alternativa para guiar el proceso de desarrollo de *software* con un enfoque hacia la interacción con el cliente y el desarrollo incremental del *software.* Proponen mostrar prototipos funcionales del producto al usuario final en pequeños períodos de tiempo, así el cliente puede proponer cambios en cualquier momento del proceso (72). Se centran en que la creación de un producto de *software* que funcione, es más importante que escribir una extensa documentación. Entre estas metodologías se encuentran Scrum y XP (del inglés Extreme Programming), las cuales se describen a continuación.

1.4.2.1 Scrum

Scrum es una metodología de desarrollo simple que no se basa en el seguimiento estricto de un plan, sino que permite la adaptación continua de acuerdo a las circunstancias durante la evolución del proceso de desarrollo de *software*. Es por ello que se emplea en entornos que trabajen con requisitos inestables y que requieran rapidez y flexibilidad. Define un marco para la gestión de proyectos por lo que se utiliza en otras prácticas de ingeniería de *software* tales como RUP o XP. Scrum tiene dos características fundamentales, la primera es que propone el desarrollo de *software* mediante iteraciones de treinta días, denominadas *sprints*, donde cada *sprint* resulta en un incremento ejecutable para mostrar al cliente y la segunda es la realización de reuniones, destacándose las reuniones diarias de quince minutos, lo que permite coordinar y gestionar las actividades sistemáticamente. Esta metodología propone que la gestión del proceso de desarrollo de *software* hasta la obtención del producto final, se realice teniendo en cuenta prácticas como: revisión de las iteraciones con todos los implicados en el proyecto, desarrollo incremental, desarrollo evolutivo y colaboración entre los involucrados. Se compone de cinco actividades importantes: Planes de lanzamientos, Distribución, revisión y ajuste de los estándares de producto, *Sprint*, Revisión del *sprint* y Cierre (73).

La siguiente figura (73) muestra el trabajo en una iteración de Scrum:

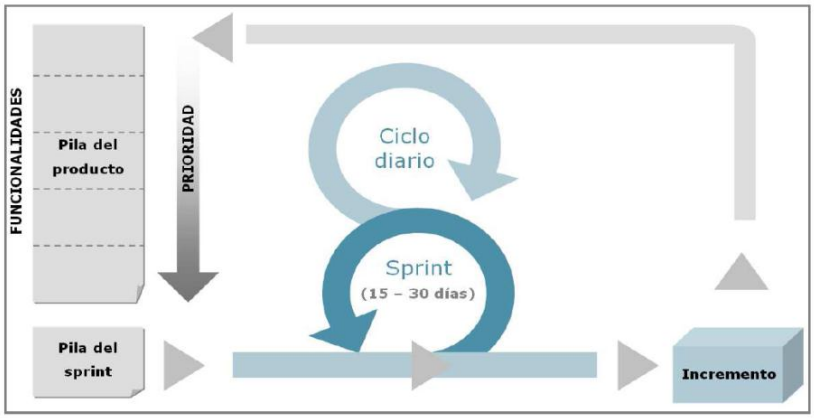

Figura 3. Iteración de Scrum

Los requisitos en Scrum conforman una lista denominada Pila del producto, que se origina inicialmente y crece y evoluciona con el avance del proyecto. La lista de trabajos que realiza el equipo durante cada *sprint* se denomina Pila del *sprint*, lo cual conlleva a la obtención del Incremento. Scrum agrupa a todas las personas involucradas en el proyecto en cuatro roles fundamentalmente. El tamaño aconsejable para los equipos es de cinco a nueve personas (73).

1.4.2.2 Extreme Programming (XP)

XP se centra en la retroalimentación entre el cliente y el equipo de desarrollo y pone énfasis en la adaptabilidad antes de la previsibilidad, por lo que es adecuada para proyectos con requisitos imprecisos, muy cambiantes y con un considerable riesgo técnico. Entre las prácticas que propone esta metodología para lograr un producto final de calidad se encuentran: entregas pequeñas, o sea producir continuamente prototipos funcionales del sistema; la metáfora, que describe cómo debería funcionar el sistema, pues no se cuenta con una arquitectura definida desde el inicio, lo que le proporciona flexibilidad al proceso; las pruebas unitarias, que verifican el código ante cada modificación; la refactorización, que consiste en mejorar la legibilidad del código; la programación en parejas; la propiedad colectiva del código y la integración continua, que plantea que cada parte de código se integre al sistema en cuanto esté lista. Estas prácticas proporcionan dinamismo y eficiencia al proceso de desarrollo de *software*. Además de las prácticas mencionadas anteriormente, esta metodología se caracteriza por las denominadas historias de usuario que especifican los requisitos funcionales y no funcionales del *software* descritos por el cliente y que se descomponen en tareas de programación implementadas por los programadores durante una iteración. XP propone además la existencia de siete roles principales, así como un equipo de desarrollo formado por dos a quince personas en correspondencia con el proyecto (72).

El trabajo con XP puede graficarse de la siguiente manera (74):

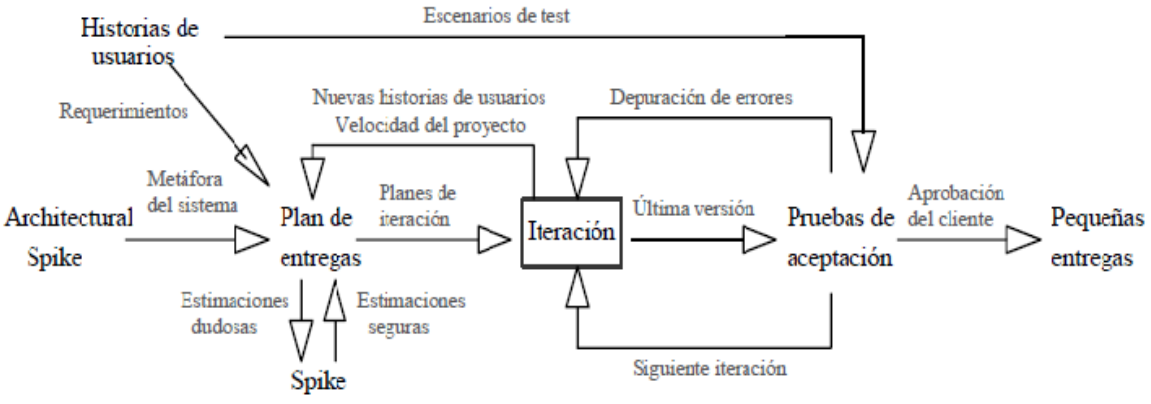

Figura 4. Flujo de trabajo en XP

Esta metodología consta de cuatro fases: Planificación del proyecto, en la que se planifica junto al cliente lo que se quiere lograr con el producto final; Diseño, en la que se persigue un diseño simple y se logran funcionalidades mínimas; Codificación, en la que se codifican las historias de usuario; y Pruebas, en la que se implanta el sistema y se realizan las pruebas de aceptación. Así se lleva a cabo el desarrollo de *software* con XP guiado por las variables coste, tiempo, calidad y alcance (74).

1.4.3 Selección definitiva de la metodología a utilizar

Se determinó el uso de una metodología ágil como guía del proceso de desarrollo del jurado en línea. Para llegar a esta conclusión se tuvo en cuenta principalmente el tamaño del equipo de desarrollo. Debido a que es un equipo pequeño (dos personas) se descartó el uso de una metodología pesada. Además se consideró innecesaria la generación de una extensa documentación en cada etapa del proyecto debido a la posibilidad de la interacción directa entre los propios desarrolladores, los cuales, en este caso, juegan el papel de clientes al mismo tiempo. Con una metodología ágil se dirige el desarrollo de forma más directa sin perder organización, eficiencia y calidad. El uso de una metodología de este tipo favorece el desarrollo de pequeños prototipos funcionales cada determinado tiempo, lo que no solamente ordena y simplifica el trabajo al crear el *software* incrementalmente, sino que permite una retroalimentación continua, lo que posibilita la alta tolerancia al cambio disminuyendo así las complicaciones que puedan surgir en este sentido.

Posteriormente se seleccionó XP entre las metodologías ágiles candidatas. Tanto Scrum como XP, presentan características que favorecen el desarrollo de un *software* de calidad en un equipo con las características antes mencionadas. La elección de uno u otro no afectaría en gran medida el proceso de desarrollo de *software*, sin embargo se decidió el uso de XP debido a su dinamismo al planificar y desarrollar cada iteración, mientras que Scrum se basa en la realización de continuas reuniones durante todo el proceso de desarrollo que, por las características particulares del equipo de desarrollo del jurado en línea, se consideraron innecesarias. Además XP presenta características como la codificación en

parejas, las pruebas unitarias, la metáfora del sistema, la propiedad colectiva del código y la integración continua, las cuales se acoplan mejor a un equipo de dos personas, donde cada uno tendrá que asumir varios roles al mismo tiempo.

1.5 Conclusiones parciales

En este capítulo se realizó un estudio de la situación actual de los jurados en línea, tanto a nivel mundial como en Cuba, determinándose sus limitaciones y los aportes de estos al sistema propuesto. Además se definieron las herramientas y tecnologías a utilizar mediante indicadores tanto para la elección de candidatas como para la posterior selección definitiva de las mismas. La misma operación se realizó para el establecimiento de la metodología que regirá el proceso de desarrollo de *software*, resultando, en este caso, XP. Con esta base se da paso al próximo capítulo donde se describen las fases de Planificación y Diseño.

Capítulo 2: Propuesta de solución

2.1 Introducción

En el presente capítulo es expuesta la propuesta de solución al problema científico planteado utilizando la metodología de desarrollo de *software* Extreme Programming (XP). Se muestra la evolución de la solución durante las fases iniciales de Planificación y Diseño, y se presentan los diferentes artefactos generados en las mismas, los cuales constituyen punto de partida para la entrega final del jurado en línea.

2.2 Propuesta de solución

La siguiente figura expone el entorno en que se desarrolla el jurado en línea mediante la identificación de los principales conceptos que conforman el mismo, así como las relaciones entre ellos.

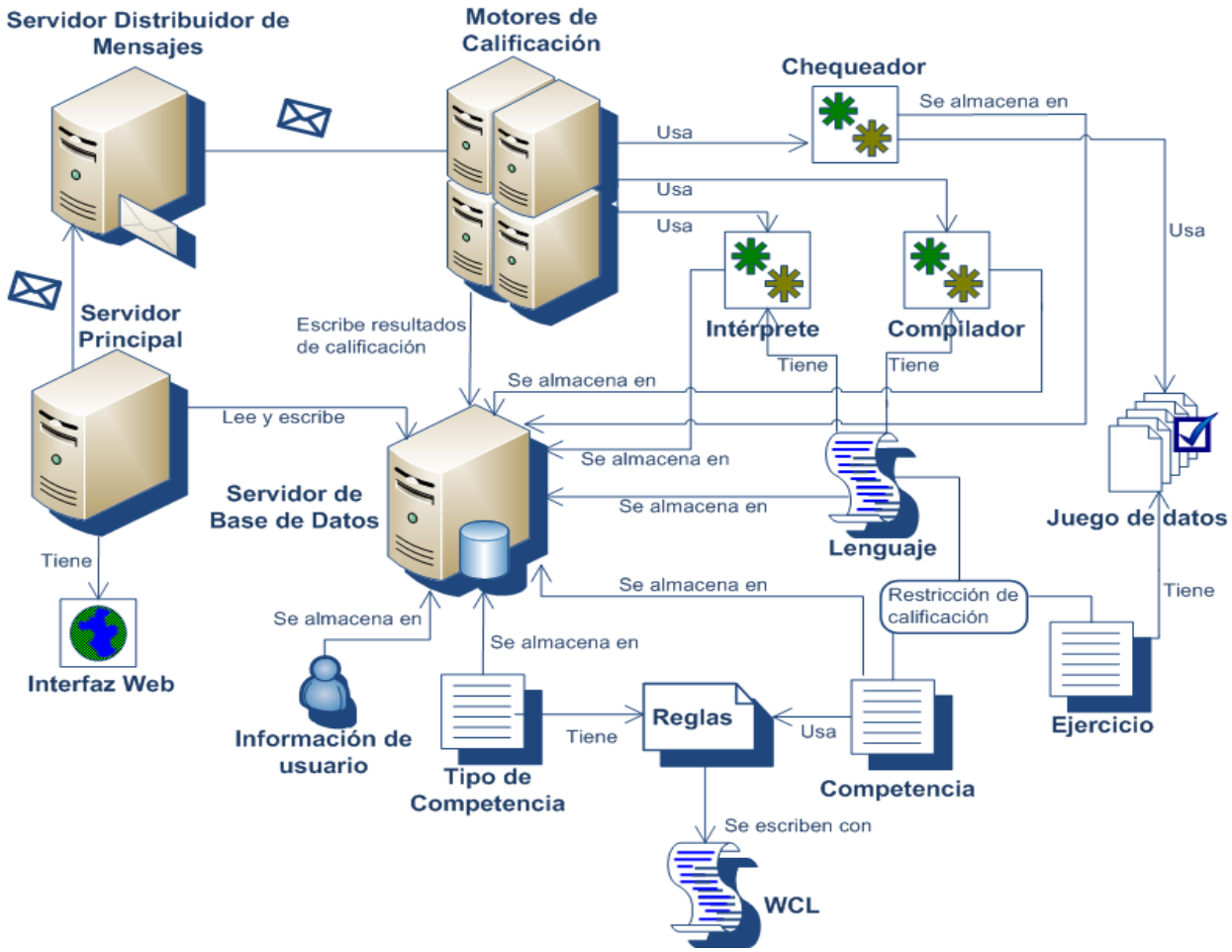

Figura 5. Propuesta de solución

2.2.1 Glosario de conceptos del entorno del sistema

Servidor principal: Se encarga de la gestión de todas las entidades de la base de datos y contiene la lógica principal.

Interfaz web: Interfaz mediante la cual se accede al sistema.

Base de datos del sistema: Contiene todos los datos persistentes del sistema.

Información de usuario: Entidad dentro de la base de datos que contiene los datos de una persona que interactúa con el sistema.

Tipo de competencia: Entidad dentro de la base de datos que contiene las reglas para desarrollar determinado tipo de competencia.

Competencia: Entidad dentro de la base de datos que contiene los datos de una competencia para la cual se utiliza el jurado en línea como calificador automático.

Reglas de una competencia: Representa las reglas que definen un tipo de competencia, desde la asignación de puntos y la forma en que se visualiza el *ranking*, hasta la forma en que se muestran los resultados de la calificación.

Ejercicio: Entidad dentro de la base de datos que representa un problema a ser solucionado dentro de una competencia.

Juego de datos: Entidad dentro de la base de datos que contiene los datos de entrada y de salida a ser utilizados para calificar un ejercicio.

Lenguaje: Entidad dentro de la base de datos que contiene los datos de un lenguaje de programación soportado por el sistema.

Restricción de calificación: Representa los límites de tiempo, memoria y tamaño de código, entre otros, con los que una solución a determinado ejercicio debe cumplir para que sea válida. Esta restricción puede establecerse, si se desea, en dependencia del lenguaje.

Motor de calificación: Se encarga de calificar una respuesta a un ejercicio.

Intérprete: Se encarga de interpretar el código de una solución a un ejercicio en caso de que el lenguaje con que se implementó la misma lo requiera.

Compilador: Se encarga de compilar el código de una solución a un ejercicio en caso de que el lenguaje con que se implementó la misma lo requiera.

Chequeador: Se encarga de determinar si la salida de una respuesta a un ejercicio es o no correcta sintáctica y semánticamente, verificando generalmente que la misma sea exactamente igual al juego de datos de salida que se posee de antemano.

Servidor distribuidor de mensajes: Se encarga de gestionar la comunicación entre el servidor principal y el o los motores de calificación.

WCL: De las siglas Wormhole Configuration Language. Es un minilenguaje creado por los desarrolladores del sistema que permite la configuración de las reglas de la competencia.

2.3 Historias de usuario

En la metodología de desarrollo de *software* XP los requisitos que debe cumplir el *software* son especificados por los clientes en las denominadas historias de usuario. Estas historias son descompuestas en tareas de programación y asignadas a los programadores. A continuación se describen las historias de usuario del sistema de acuerdo a la siguiente plantilla:

Historia de Usuario	
Número: *Número de la Historia de Usuario (HU), incremental en el tiempo.*	**Nombre:** *El nombre de la HU, sirve para identificarla fácilmente entre los desarrolladores y los clientes.*
Usuario: *El usuario del sistema que utiliza o protagoniza la historia.*	
Prioridad en Negocio: *Qué tan importante es para el cliente (Alta, Media o Baja).*	**Riesgo en Desarrollo:** *Qué tan difícil es para el desarrollador (Alto, Medio o Bajo).*
Iteración Asignada: *La iteración a la que corresponde.*	
Descripción: *La descripción de la historia, detallando las operaciones del usuario y opcionalmente las respuestas del sistema.*	
Observaciones: *Observaciones de interés, como glosario, información sobre usuarios, etc.*	

Tabla 1. Modelo de Historia de Usuario (HU)

Historia de Usuario	
Número: 1	**Nombre:** Autenticar usuario
Usuario: Todos	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Bajo
Iteración Asignada: 1	
Descripción: El usuario inserta su identificador y contraseña en el sistema para poder acceder a otras funcionalidades para las que esté autorizado.	
Observaciones:	

Tabla 2. HU Autenticar usuario

Historia de Usuario	
Número: 2	**Nombre:** Calificar solución
Usuario: Todos	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Medio
Iteración Asignada: 1	
Descripción: El usuario selecciona una competencia, dentro de la misma selecciona un ejercicio y envía la solución, introduciendo los siguientes datos: • Lenguaje de programación. • Código fuente. El sistema toma el código fuente y verifica las restricciones para el mismo. El código es compilado con	

el compilador del lenguaje, si este último se relaciona con alguno. Si no hubo error de compilación el sistema ejecuta (o interpreta, en caso de que el lenguaje se relacione con algún intérprete) el programa generado teniendo en cuenta los límites de ejecución que pueden ser: de tiempo, memoria, tamaño de programa y de salida generada. Si no existe error de ejecución, el sistema chequea que la salida generada sea la correcta. Luego se envían los resultados de la calificación que pueden ser: • Tamaño de código excedido. • Error de restricción de código. • Error de compilación. • Error de ejecución. • Tiempo excedido. • Memoria excedida. • Tamaño de programa excedido. • Tamaño de salida excedida. • Respuesta incorrecta. • Respuesta aceptada.
Observaciones: El usuario debe estar autenticado.

Tabla 3. HU Calificar solución

Historia de Usuario	
Número: 3	**Nombre:** Gestionar competencia
Usuario: Administrador de competencias	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Medio
Iteración Asignada: 1	
Descripción: El usuario puede adicionar, eliminar o modificar competencias que no hayan comenzado aún. Para adicionar una competencia el usuario debe especificar los siguientes datos: • Habilitado o no. • Nombre. • Descripción. • Fecha de comienzo. • Duración. • Máxima cantidad de concursantes que pueden registrarse. • Usuarios registrados. • Clúster. • Tipo de competencia. • Valores de los parámetros iniciales del tipo de competencia.	

• Ejercicios. • Lenguajes permitidos. • Límites para la calificación de cada ejercicio por cada lenguaje (de tiempo, de memoria, de tamaño del código). Para eliminar o modificar una competencia, el usuario debe poder listar las competencias y seleccionar la que desee eliminar o modificar.
Observaciones: El usuario debe estar autenticado. La competencia no debe haber iniciado.

Tabla 4. HU Gestionar competencia

		Historia de Usuario
Número: 4	**Nombre:** Consultar *ranking*	
Usuario: Todos		
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Bajo	
Iteración Asignada: 1		
Descripción: El usuario tiene la posibilidad de consultar el *ranking* de una competencia si está permitido para esa competencia.		
Observaciones: El usuario debe estar autenticado.		

Tabla 5. HU Consultar ranking

		Historia de Usuario
Número: 5	**Nombre:** Consultar envíos realizados	
Usuario: Todos		
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Bajo	
Iteración Asignada: 1		
Descripción: El usuario tiene la posibilidad de consultar todos los envíos realizados en una competencia si está permitido para esa competencia.		
Observaciones: El usuario debe estar autenticado.		

Tabla 6. HU Consultar envíos realizados

		Historia de Usuario
Número: 6	**Nombre:** Registrar usuario	
Usuario: Todos		
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Bajo	
Iteración Asignada: 1		
Descripción: El usuario tiene la posibilidad de registrarse en el sistema para lo cual debe proveer los datos: • Identificador de usuario. • Contraseña en el sistema.		

Observaciones:

Tabla 7. HU Registrar usuario

Historia de Usuario	
Número: 7	**Nombre:** Modificar perfil
Usuario: Todos	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Bajo
Iteración Asignada: 1	
Descripción: El usuario tiene la posibilidad de modificar los siguientes datos de su perfil: • Identificador de usuario. • Nombre. • Apellidos. • Dirección de correo electrónico. • Contraseña.	
Observaciones:	

Tabla 8. HU Modificar perfil

Historia de Usuario	
Número: 8	**Nombre:** Gestionar usuarios
Usuario: Administrador de usuarios	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Medio
Iteración Asignada: 1	
Descripción: El usuario tiene la posibilidad de adicionar, modificar y eliminar usuarios. Para adicionar un usuario se necesitan los siguientes datos: • Identificador de usuario. • Contraseña. Para eliminar o modificar usuarios, el usuario debe poder listar los usuarios registrados y seleccionar el que desee para eliminar o modificar. Los datos de los usuarios que se han registrado en el sistema que se pueden modificar son: • Identificador de usuario. • Contraseña. • Nombre. • Apellidos. • Dirección de correo electrónico. • Activo o no. • Si es del *staff* (equipo de administración) o no. • Superusuario o no.	

• Grupos a los que pertenece. • Permisos.
Observaciones: El usuario debe estar autenticado.

Tabla 9. HU Gestionar usuarios

		Historia de Usuario
Número: 9	**Nombre:** Gestionar tipos de competencias	
Usuario: Administrador de tipos de competencias		
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Alto	
Iteración Asignada: 1		
Descripción: El usuario tiene la posibilidad de adicionar, eliminar y modificar tipos de competencias. Para adicionar un tipo de competencia el usuario debe introducir los siguientes datos: • Habilitado o no. • Nombre. • Descripción. • Nombre de los parámetros de los tipos de competencia. • Reglas. Para eliminar o modificar un tipo de competencia, el usuario debe poder listar los tipos de competencia y seleccionar el que desee eliminar o modificar.		
Observaciones: El usuario debe estar autenticado.		

Tabla 10. HU Gestionar tipos de competencias

		Historia de Usuario
Número: 10	**Nombre:** Gestionar ejercicios	
Usuario: Administrador de ejercicios		
Prioridad en Negocio: Media	**Riesgo en Desarrollo:** Bajo	
Iteración Asignada: 2		
Descripción: El usuario tiene la posibilidad de adicionar, eliminar y modificar ejercicios dentro del sistema. Para adicionar un ejercicio el usuario debe introducir los siguientes datos: • Habilitado o no. • Nombre. • Descripción del problema. • Descripción de la entrada. • Descripción de la salida. • Ejemplo de entrada. • Ejemplo de salida.		

• Nota o aclaración. • Nombre del autor. • Chequeador, si tiene. • Límites de calificación. • Juegos de datos de entrada y de salida. Para eliminar o modificar un ejercicio, el usuario debe poder listar los ejercicios y seleccionar el que desee eliminar o modificar.
Observaciones: El usuario debe estar autenticado.

Tabla 11. HU Gestionar ejercicios

		Historia de Usuario
Número: 11	**Nombre:** Gestionar lenguajes	
Usuario: Administrador de lenguajes		
Prioridad en Negocio: Media	**Riesgo en Desarrollo:** Medio	
Iteración Asignada: 2		
Descripción: El usuario tiene la posibilidad de adicionar y eliminar los lenguajes permitidos por el sistema y modificar los parámetros de los mismos. Para adicionar un lenguaje el usuario debe introducir los siguientes datos: • Habilitado o no. • Nombre. • Descripción. • Compilador. • Intérprete. • Restricciones de código. Para eliminar o modificar un lenguaje, el usuario debe poder listar los lenguajes soportados por el jurado en línea y seleccionar el que desee eliminar o modificar.		
Observaciones: El usuario debe estar autenticado.		

Tabla 12. HU Gestionar lenguajes

		Historia de Usuario
Número: 12	**Nombre:** Gestionar clústeres de calificación	
Usuario: Administrador de motores		
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Alto	
Iteración Asignada: 2		
Descripción: El usuario tiene la posibilidad de adicionar y eliminar clústeres de calificación en el sistema y modificar los parámetros de los mismos. Para adicionar un clúster de calificación el usuario debe introducir los siguientes datos: • Habilitado o no.		

• Nombre. • Descripción. • Frecuencia del microprocesador. • Motores. Para eliminar o modificar un clúster de calificación, el usuario debe poder listar los clústeres y seleccionar el que desee eliminar o modificar. Un clúster es una agrupación de varios motores de calificación, por lo que el usuario debe tener la posibilidad de adicionar, modificar y eliminar motores de calificación**.** Para adicionar un motor de calificación el usuario debe introducir los siguientes datos: • Habilitado o no. • Nombre. • Descripción. • Cantidad máxima de peticiones. Para eliminar o modificar un motor de calificación, el usuario debe poder listar los motores y seleccionar el que desee eliminar o modificar.
Observaciones: El usuario debe estar autenticado.

Tabla 13. HU Gestionar clústeres de calificación

2.4 Requisitos no funcionales

Requisitos de *software* en el cliente

- Mozilla Firefox 10.0 o superior.

Requisitos de *software* en los servidores

- Sistema operativo Debian 6.0 o superior.
- Gestor de bases de datos PostgreSQL 9.0 o superior.
- Servidor distribuidor de mensajes RabbitMQ 2.8.4 o superior.
- Intérprete de Python 2.7.
- Django 1.5 o superior.
- Apache Web Server 2.0 o superior.

Usabilidad

- El sistema debe contar con una interfaz intuitiva que permita el fácil acceso a cada una de sus funcionalidades.

Rendimiento

- El sistema debe ser capaz de responder a una petición en un tiempo menor de 1 segundo.

Seguridad

- Confiabilidad: el sistema debe garantizar que los datos no se pierdan por causa de inestabilidad del mismo.
- Confidencialidad: los datos manejados por el sistema no deben ser accedidos por personal no autorizado.
- Integridad: los datos almacenados en la base de datos del sistema no pueden presentar incoherencias con respecto a la realidad.

2.5 Metáfora

Una metáfora para un sistema consiste en una sinopsis sobre cómo funciona el mismo. La elección de una metáfora permite mantener la coherencia de todos los componentes a implementar y elimina la selección rígida de una arquitectura inicial.

El jurado en línea propuesto brinda las funcionalidades básicas de una aplicación de su tipo, como lo son la calificación automática de soluciones a ejercicios de programación de máquinas de cómputo y la gestión y configuración de competencias. Permite además la gestión y configuración de diferentes tipos de competencias mediante la configuración de determinadas reglas, lo cual posibilita la realización de competencias de diferentes tipos sin necesidad de modificar o agregar código fuente al sistema.

Una solución a un ejercicio es calificada por un motor de calificación; el sistema cuenta con varios de estos motores que pueden estar agrupados en clústeres de calificación, lo que aumenta la eficiencia. Estos motores recibirán las peticiones del servidor web y escribirán los resultados en la base de datos. Para intercomunicar los componentes del sistema se utiliza un servidor distribuidor de mensajes, garantizando la entrega, velocidad y seguridad de los mismos.

2.6 Arquitectura

La arquitectura de un sistema expresa cuáles son los componentes del mismo y la relación que existe entre ellos. Se encuentra orientada a la satisfacción de los requisitos funcionales y no funcionales del *software*.

El jurado en línea propuesto basa su implementación en la arquitectura n-capas, específicamente cuatro capas: Acceso, Comunicación, Procesamiento y Datos.

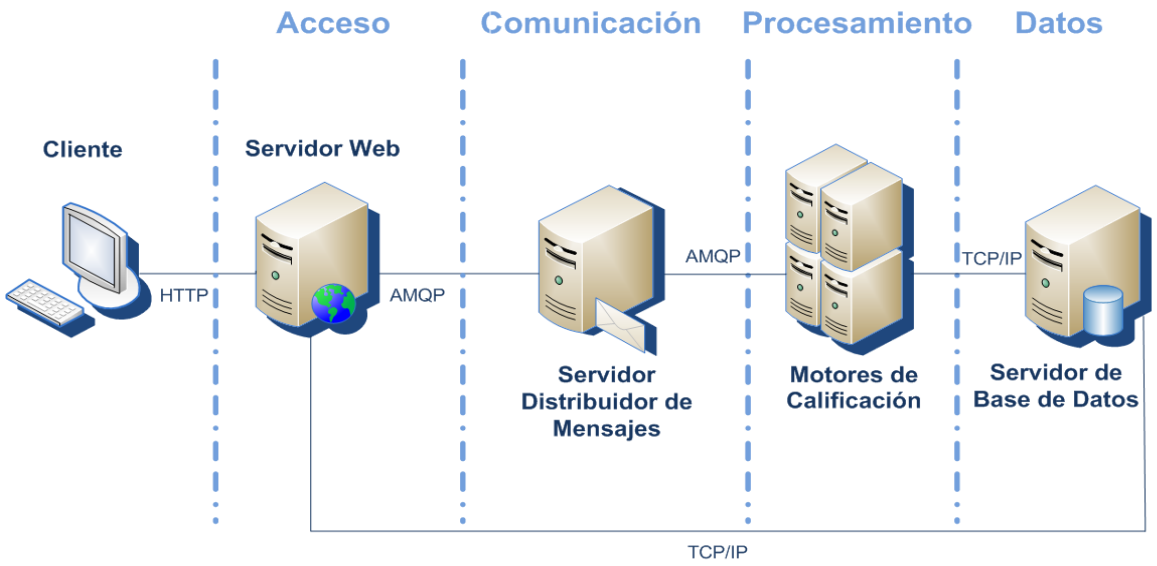

Figura 6. Arquitectura del sistema

2.6.1 Componentes del sistema

El sistema está conformado por cuatro componentes: el Servidor web, el Servidor de bases de datos, el Servidor distribuidor de mensajes y el Motor de calificación.

Servidor web

Se encarga de recibir todas las peticiones de los usuarios. Contiene la interfaz gráfica de acceso al sistema tanto para usuarios comunes como para administradores, por lo que se encarga además de la parte lógica de la gestión de las entidades del sistema, es decir, tiene pleno acceso a la base de datos del mismo. El servidor web también notificará y delegará acciones a los demás componentes del sistema en caso necesario. Para la programación de este componente se cuenta con el *framework* para desarrollo web Django 1.5, se tiene además Apache 2.0 como servidor HTTP y Debian 6.0 como sistema operativo para este servidor.

Servidor de bases de datos

Este servidor contiene la base de datos del sistema en la que persisten los datos de todas las entidades del mismo, tales como usuarios, grupos, competencias, ejercicios y lenguajes de programación permitidos, entre otros. Los datos de configuración de los motores de calificación, las peticiones de calificación de cada competencia, los compiladores e intérpretes de cada lenguaje y la configuración de los tipos de competencias también permanecen en este servidor. El gestor de bases de datos usado es PostgreSQL 9.0 y Debian 6.0 como sistema operativo para este componente.

Servidor distribuidor de mensajes

La comunicación entre el servidor web y los motores de calificación es gestionada por un servidor distribuidor de mensajes. El mismo se encarga de distribuir los mensajes entre dichos componentes de manera segura y eficiente mediante el uso de RabbitMQ 2.8.4 y con Debian 6.0 como sistema operativo.

Motor de calificación

Se encarga de lo referente a la calificación de las soluciones enviadas por los usuarios, es decir, de la obtención de los resultados de la evaluación de las mismas. Los resultados de cada calificación serán almacenados en la base de datos del sistema, por lo que este componente se conectará a la misma de manera regular. El motor también chequeará la realización de los eventos del sistema y ejecutará las acciones relacionadas con los mismos. Un evento es un suceso que provoca la ejecución de un conjunto de acciones que pueden o no alterar el comportamiento de algunas partes específicas del sistema. Estas acciones se definen en dependencia del tipo de competencia y serán configuradas por el usuario que configure dicho tipo. Algunos de los eventos que manejará el motor de calificación son: eventos de tiempo ("al transcurrir determinado período de tiempo"), eventos de inicio y finalización de competencias ("al comenzar determinada competencia" y "al finalizar determinada competencia"), y los eventos de calificación ("cuando se ha juzgado una solución a un ejercicio" y "cuando se ha juzgado una solución para un juego de datos"), entre otros. El sistema está diseñado para que permita el uso de varios motores de calificación, que pueden estar agrupados en clústeres de calificación, aumentando de esta manera la eficiencia del mismo. Para la implementación de este componente se cuenta con el lenguaje de programación Python 2.7 y se tiene Debian 6.0 como sistema operativo del servidor.

2.6.2 Interacción de los componentes de la arquitectura. Petición de calificación

Para comprender cómo se interconectan y colaboran los componentes del sistema a continuación se explica cómo se comporta el jurado en línea ante una petición de calificación.

1. El cliente realiza una petición de calificación.
2. El servidor web recibe la petición, la registra en la base de datos y redirecciona al cliente hacia una página de resultados.
3. El servidor web envía un mensaje con la petición de calificación hacia el servidor distribuidor de mensajes.
4. El servidor distribuidor de mensajes entrega el mensaje a un motor de calificación desocupado.
5. El motor de calificación califica la solución propuesta por el cliente que fue enviada dentro del mensaje.
6. El motor de calificación realiza los eventos asociados y escribe el resultado en la base de datos.
7. El cliente recarga la página de resultados cuando quiere ver el resultado de la calificación.

8. El servidor web consulta la base de datos y muestra los resultados de la calificación si los mismos han sido insertados.

2.7 Plan de iteraciones

El plan de iteraciones (ver anexo 1) determina la duración de cada una de las iteraciones previstas para el desarrollo del sistema, lo que permite la estimación del tiempo requerido hasta la obtención del producto final del proyecto. Con este plan se logra una mayor organización y estructuración del trabajo al mostrar el orden en que serán implementadas cada una de las historias de usuario dentro de cada iteración. Se han definido 12 historias de usuario seccionadas en dos iteraciones para una duración total del proyecto de 16 semanas.

2.8 Plan de entrega

El plan de entrega (ver anexo 2) constituye una herramienta fundamental, pues proporciona una estimación del tiempo de desarrollo que requieren las historias de usuario, fijándose de esta manera el tiempo que tardaría la implementación de cada una de ellas y definiéndose así finalmente las fechas exactas por iteración en que serán liberadas las versiones funcionales del producto.

2.9 Tarjetas CRC

El uso de las tarjetas CRC (Clases, Responsabilidades y Colaboradores) permite un diseño de *software* orientado a objetos. Estas tarjetas proponen una forma de trabajo para encontrar los objetos del dominio del sistema, sus responsabilidades y cómo colaboran con otros para realizar tareas. De esta manera el algoritmo para utilizar las tarjetas CRC es: primero la identificación de clases y asociaciones que participan del diseño del sistema, luego la obtención de las responsabilidades que debe cumplir cada clase y por último el establecimiento de cómo una clase colabora con otras para cumplir con sus responsabilidades.

Aplicación motor de calificación

Esta aplicación consta de cinco módulos: módulo de comunicación con el servidor RabbitMQ, módulo de comunicación con la base de datos, módulo de calificación, módulo de intérprete y módulo de manejo de eventos del sistema. Se realizaron tarjetas CRC por cada módulo (ver anexo 3), donde se establecen las clases principales del motor de calificación, sus responsabilidades y colaboradores.

Aplicación web

Esta aplicación consta de cuatro módulos: módulo de autenticación, módulo de gestión de competencias, módulo de gestión de calificación y módulo de gestión de configuración del sistema. Se realizaron tarjetas CRC por cada módulo (ver anexo 4), donde se establecen las clases principales de la aplicación web, sus responsabilidades y colaboradores.

2.10 Diagrama de clases del diseño

Los diagramas de clases del diseño constituyen una herramienta fundamental para representar gráficamente cada una de las clases del sistema, así como las relaciones existentes entre ellas. En este tipo de diagramas se exponen las clases con sus métodos y atributos, los diferentes tipos de relaciones existentes entre las clases, así como la navegabilidad de dichas relaciones.

El sistema cuenta con dos aplicaciones: la aplicación del motor de calificación y la aplicación web. La aplicación del motor de calificación está estructurada en cinco módulos: el módulo de comunicación con el servidor RabbitMQ (ver anexo 5), el módulo de comunicación con la base de datos (ver anexo 6), el módulo de calificación (ver anexo 7), el módulo de intérprete (ver anexo 8) y el módulo de manejo de eventos del sistema (ver anexo 9). La segunda aplicación es la web, que sigue el patrón de diseño que impone Django, por ello cada escenario siempre tiene la misma estructura desde el punto de vista lógico. Cuando se genera una petición al servidor, el controlador selecciona la vista adecuada de acuerdo a la dirección de dicha petición, para construir esta vista el servidor consulta la base de datos mediante un modelo especificado previamente durante el mapeo objeto-relacional (más conocido por sus siglas en inglés como ORM)[59]. Debido a que los diagramas de clases que se generarían tendrían siempre la misma estructura, solamente se muestra el correspondiente a las clases que intervienen en una petición de envío de solución (ver anexo 10). Esta aplicación está compuesta por cuatro módulos: el módulo de autenticación, el módulo de gestión de competencias, el módulo de gestión de calificación y el módulo de gestión de configuración del sistema.

2.11 Diagrama de diseño de la base de datos

La base de datos es un elemento clave en el sistema y como tal debe estar correctamente diseñada. Un buen diseño posibilita el acceso a información exacta y actualizada, y que la base de datos pueda modificarse fácilmente y adaptarse a las necesidades del sistema.

Para el diseño de la base de datos del sistema (ver anexo 11), se determinaron las entidades (tablas), se especificaron los atributos (campos) y se relacionaron las tablas.

2.12 Conclusiones parciales

En este capítulo se obtuvieron los artefactos correspondientes a las fases de Planificación y Diseño definidas por la metodología XP. Fueron expuestas la propuesta de solución, las historias de usuario que recogen las principales funcionalidades del *software* y los requisitos no funcionales. Se definieron los planes de iteración y de entrega, necesarios para planificar el proceso de implementación del *software*. Además se realizó cuidadosamente el diseño de la arquitectura del sistema, el diseño de las clases que

[59] El ORM es una técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y el utilizado en una base de datos relacional, utilizando un motor de persistencia.

conforman el mismo, así como la base de datos que contiene la información persistente. En este punto se da paso al próximo capítulo, donde se describen las fases de Codificación y Prueba.

Capítulo 3: Codificación y prueba

3.1 Introducción

En el presente capítulo, luego de concluidas las fases de Planificación y Diseño planteadas por la metodología XP, se codifica y prueba el sistema diseñado. Se muestra la evolución de la solución durante estas fases y además se presentan los diferentes artefactos generados en las mismas, para finalmente obtener un producto de alta calidad.

3.2 Tareas de ingeniería

Las tareas de ingeniería representan el trabajo realizado durante una iteración expresado en tareas de programación. Estas tareas se describen por cada historia de usuario para así especificar las acciones que realizan los programadores en cada una de ellas, lo que facilita en gran medida el trabajo, pues las historias de usuario no brindan los detalles suficientes para enfrentar la etapa de desarrollo del sistema.

A continuación se describen las tareas de programación por cada historia de usuario. En el caso de las historias que impliquen “Gestionar”, solo se exponen las tareas de los escenarios de una sola historia de usuario de este tipo, pues en el resto son muy similares.

Iteración	Historia de usuario	Tareas
1	Autenticar usuario	Definir interfaz de autenticación de usuario. Validar los datos introducidos por el usuario (nombre de usuario y contraseña). Notificar al usuario si los datos son correctos o no.
	Calificar solución	Definir interfaz de calificación de solución. Enviar un mensaje identificado como petición de calificación a los motores de calificación disponibles. Realizar la calificación en el motor de calificación disponible. Notificar al usuario los resultados de la calificación.
	Gestionar competencia	Definir interfaz de adicionar una competencia. Validar los datos de la competencia.

		Notificar al usuario si los datos son correctos o no, en caso de que sean correctos, registrar la competencia en la base de datos. Definir interfaz de listar competencias. Listar las competencias que existen en la base de datos. Definir interfaz de eliminar una competencia. Notificar al usuario que la competencia quedó eliminada. Definir interfaz de modificar una competencia. Validar los nuevos datos introducidos por el usuario, en caso de que sean correctos se modifica en la base de datos.
	Consultar *ranking*	Definir interfaz de *ranking*. Validar que el *ranking* pueda o no mostrarse, de acuerdo al tipo de competencia.
	Consultar envíos realizados	Definir interfaz de envíos. Validar que los envíos puedan o no mostrarse, de acuerdo al tipo de competencia.
	Registrar usuario	Definir interfaz de registrar usuario. Validar los datos de registro del usuario. Notificar al usuario si los datos son correctos o no, en caso de que sean correctos, registrar el usuario en la base de datos.
	Modificar perfil	Definir interfaz de modificar perfil. Validar los datos modificados por el usuario. Notificar al usuario si los datos son correctos o no, en caso de que sean

		correctos, modificarlos en la base de datos.
	Gestionar usuarios	
	Gestionar tipos de competencias	
2	Gestionar ejercicios	
	Gestionar lenguajes	
	Gestionar clústeres de calificación	

Tabla 14. Tareas de ingeniería

3.3 Implementación

3.3.1 Patrones de diseño

Un patrón de diseño constituye una solución a un problema de diseño ya conocido, pues describe la forma esencial del problema y la estructura general de su solución.

Los patrones utilizados en el diseño del jurado en línea fueron: Singleton, Decorador y Proxy.

Singleton se utiliza cuando es necesario el uso de una sola instancia de clase. En el caso de Python este patrón puede implementarse usando los denominados módulos[60], que por sí solos son clases de una sola instancia. El intérprete de Python se encarga automáticamente de devolver siempre la misma instancia. En el sistema las clases de manejo de mensajes y de trabajo con la base de datos son ejemplos de clases que utilizan Singleton.

El patrón Decorador es usado cuando es necesario adicionar, en tiempo de ejecución, más funcionalidades a un objeto de una clase. En el caso de Python se utilizan las denominadas funciones de segundo orden devenidas de la programación funcional, las cuales reciben un objeto por parámetro y devuelven el mismo con las funcionalidades adicionadas.

Por último, el patrón Proxy es usado cuando se necesita el acceso “perezoso” (solo cuando sea necesario) a recursos determinados. En el sistema se utiliza este patrón para acceder a la base de datos solamente cuando es necesario, mediante las clases modelos. Estas clases constituyen objetos intermediarios (*proxy*) que generalmente contienen las mismas funcionalidades y estructura del objeto real, pero solo hacen uso del mismo cuando se requiera.

3.3.2 Diseño del lenguaje de configuración de las reglas de las competencias (WCL)

Para la configuración de las distintas reglas de las competencias se utilizó un minilenguaje creado por los desarrolladores, denominado Wormhole Configuration Language (WCL). Las instrucciones del mismo

[60] Un módulo en Python es un conjunto de funcionalidades definidas en un mismo fichero que pueden utilizar a su vez funcionalidades de otros módulos.

están diseñadas para poder cambiar características de la competencia en el momento en que se activa un evento determinado. Un evento es un suceso que ocurre en el contexto de una competencia. Ejemplos de eventos son: “cuando una solución es juzgada”, “cuando la competencia comienza”, “cuando la competencia termina” y “cuando ha pasado un período de tiempo desde el comienzo de la competencia”. Se identificaron los eventos comunes para todo tipo de competencia. Las acciones que se realicen al momento de la ocurrencia de los eventos son las que diferencian un tipo de competencia de otro.

Si se especifican las instrucciones (denominadas “acciones” en WCL) que deben llevarse a cabo en el momento de activación del evento, se puede de esta manera configurar las reglas de la competencia. El comportamiento del sistema cambia en dependencia de los valores de las variables del sistema que se manejan dentro de una competencia, y una acción puede ser el cambiar el valor de alguna de estas variables. Ejemplos de variables son: “mostrar *ranking*” y “mostrar envíos” (cuyos valores pueden ser verdadero o falso).

Cada vez que el sistema va a realizar una operación, como mostrar el *ranking*, primero verifica el valor de las variables asociadas a esa acción, en este caso la variable “mostrar *ranking*”, y en dependencia de los valores de las mismas será el resultado de la operación, por ejemplo para esta variable si el valor es verdadero se mostrará el *ranking*, en caso contrario no.

Si dos eventos son activados al mismo tiempo, sus acciones podrían estarse ejecutando en paralelo. En este caso para evitar la aparición de datos corruptos en la base de datos, se hace el uso de una transacción diferente en cada activación.

WCL maneja varias acciones además de la asignación de valores a una variable. También pueden configurarse acciones para varios eventos. Para una descripción completa (ver anexo 12), donde se detallan los eventos, acciones permitidas y los nombres de las variables del sistema.

3.4 Diagrama de despliegue

El diagrama de despliegue es una herramienta muy útil que se emplea para exponer los elementos de configuración del procesamiento y las conexiones entre estos elementos (68). Estos diagramas modelan la arquitectura de un sistema en tiempo de ejecución, pues permiten visualizar los diferentes nodos[61] que lo componen, así como las relaciones físicas entre dichos nodos y la distribución de los componentes sobre los mismos.

[61] Un nodo es un objeto físico en tiempo de ejecución, es decir, una máquina que se compone habitualmente de, por lo menos, memoria y capacidad de procesamiento.

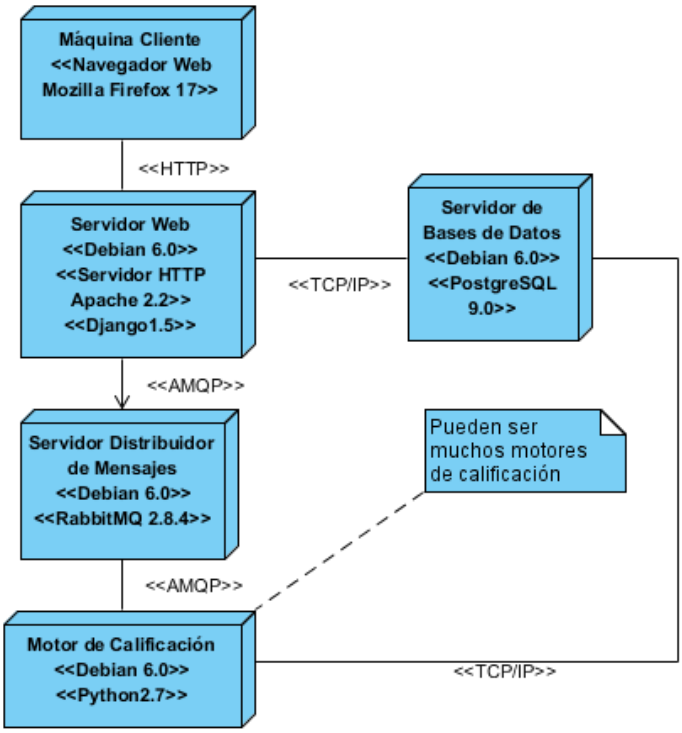

Figura 7. Diagrama de despliegue

3.5 Diagrama de componentes

El diagrama de componentes representa los componentes de *software* (componentes de código fuente, componentes del código binario y componentes ejecutables, entre otros) que integran un sistema, así como las relaciones de dependencia entre dichos componentes. Estas relaciones describen qué componentes utilizan los servicios ofrecidos por otros componentes.

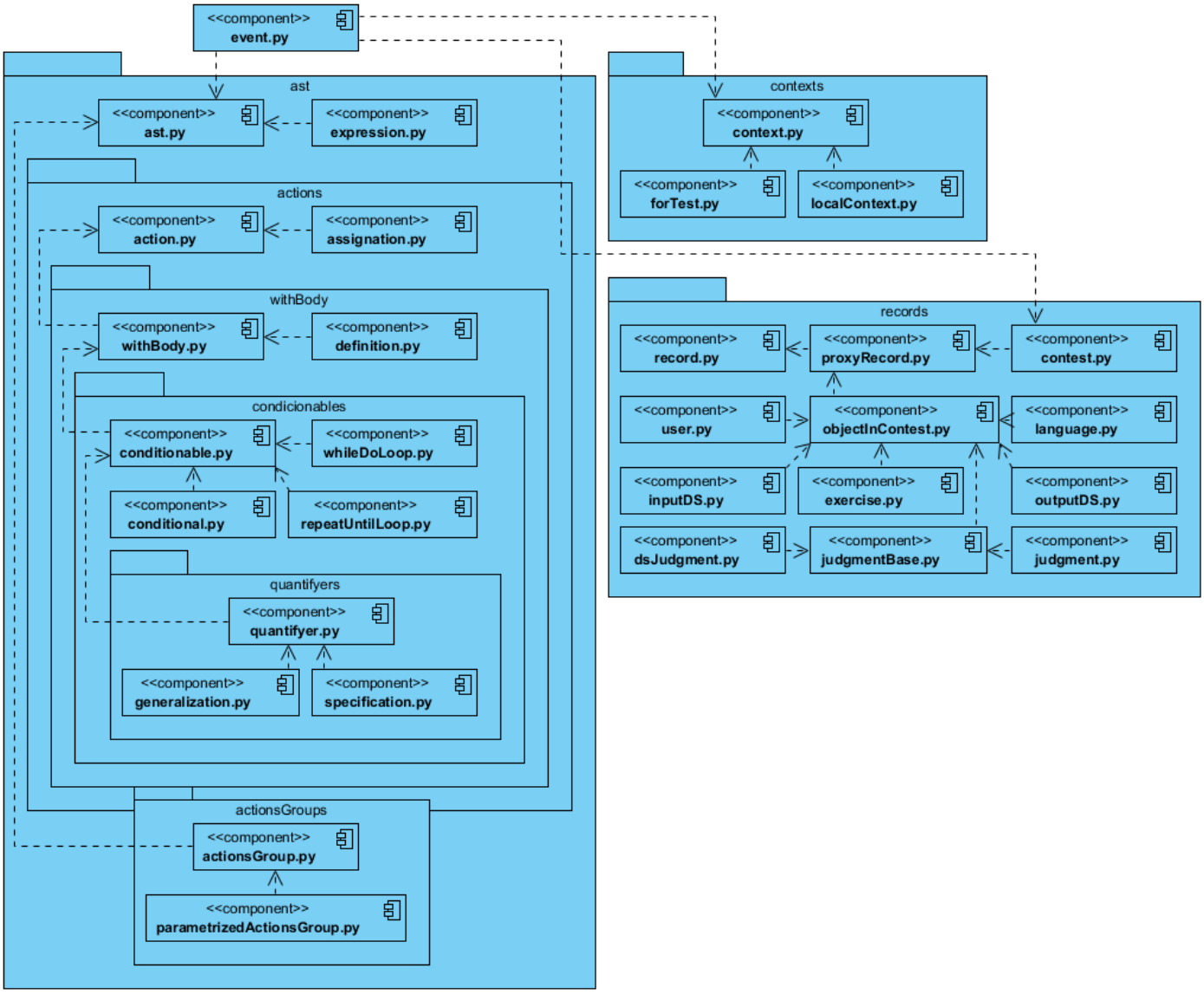

Figura 8. Diagrama de componentes del intérprete de WCL

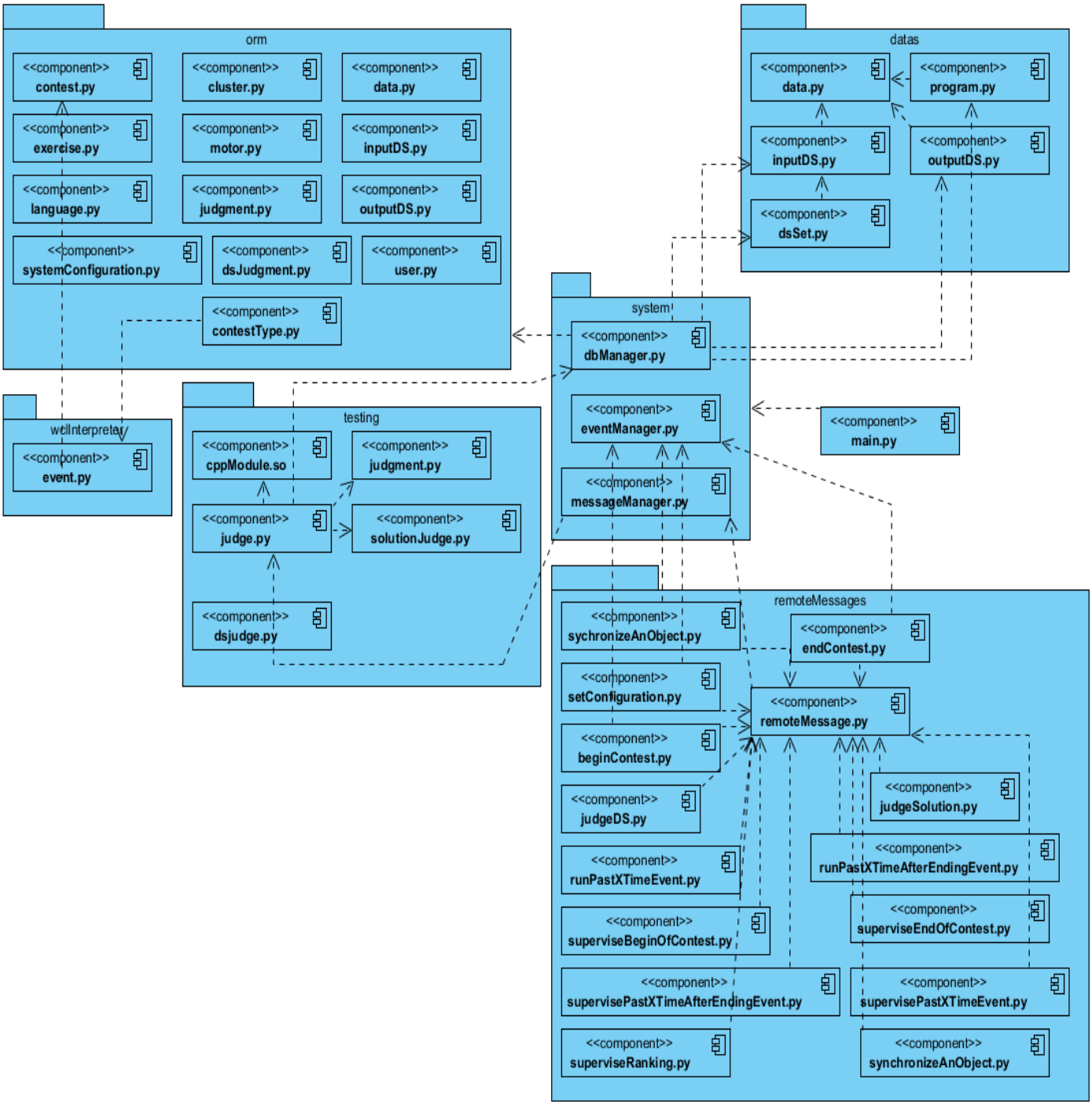

Figura 9. Diagrama de componentes del motor de calificación

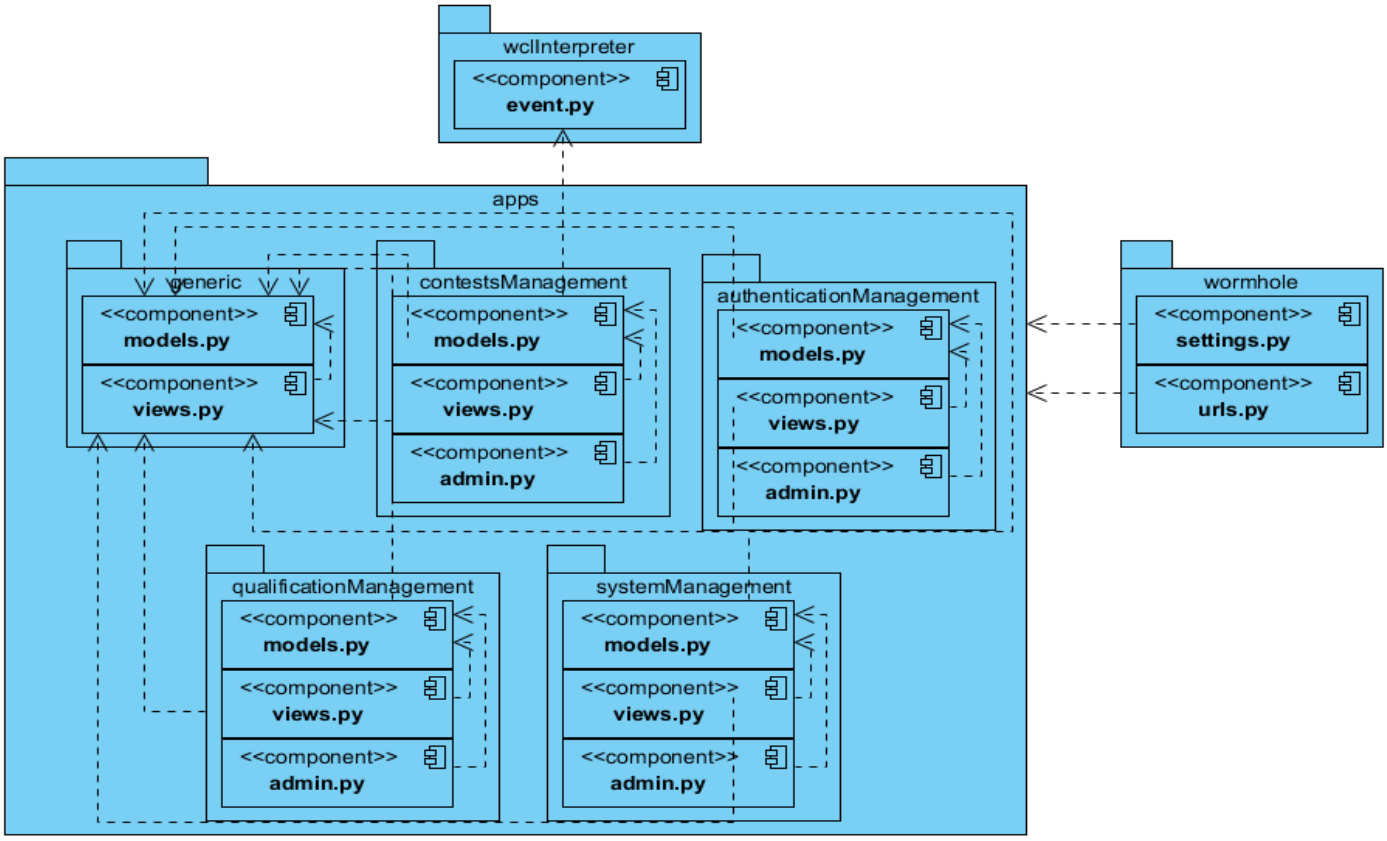

Figura 10. Diagrama de componentes de la aplicación web

3.6 Pruebas

Las pruebas de *software* tienen como objetivo proporcionar información sobre la calidad del *software*, verificando que el mismo logre satisfacer al cliente. La metodología XP propone pruebas al sistema que se dividen en dos grupos: las pruebas unitarias, diseñadas por los programadores para verificar que el código sea correcto y las pruebas de aceptación, que son pruebas funcionales diseñadas por los clientes para evaluar que al final de cada iteración se alcanzó la funcionalidad requerida. Al jurado en línea se le realizaron además pruebas para verificar el cumplimiento de los requisitos no funcionales de rendimiento.

3.6.1 Pruebas unitarias

Las pruebas unitarias son diseñadas por los programadores antes de comenzar la codificación. Se realizan durante todo el desarrollo del *software* para garantizar que los diferentes componentes (clases y funciones, entre otros) del mismo funcionen correctamente de manera individual, lográndose así una mejor integración.

La herramienta seleccionada para la ejecución de pruebas unitarias al sistema desarrollado es Unittest, también conocida como Pyunit. Esta herramienta es un módulo estándar que, desde la versión 2.1, forma parte de la librería de Python, por lo que no necesita ser instalada. Permite escribir pruebas fácilmente, agregarlas y ejecutarlas. El flujo de trabajo con Pyunit básicamente es el siguiente (75):

- Para cada grupo de pruebas crear una clase que herede de unittest.TestCase.
- Añadir métodos que comiencen con test (pruebas a ejecutar dentro de esa batería de pruebas).
- Llamar a la función main() del módulo (se ejecutarán todos los métodos cuyo nombre comience con test).

A continuación se muestran dos pruebas realizadas a dos de las principales funcionalidades del sistema, la funcionalidad “Enviar solución” de la aplicación web y la de “Calificar solución” del motor de calificación:

```
class JudgmentTest(TestCase):

    def test_solution_submitions(self):
        config = System_Configuration(broker_user = 'guest', broker_password = 'guest')
        config.clean()
        config.save()
        user = User.objects.create_user( username = 'test_user', email = '', password = 'test_passw' )
        assert self.client.login(username = 'test_user', password = 'test_passw') == True
        cluster = Cluster(name = "test_cluster", patron_micro_frequency = 2300)
        cluster.clean(); cluster.save()
        cluster.motors.create(name = "test_motor")
        contest_type = Contest_Type(name = "test_contest_type")
        contest_type.clean(); contest_type.save()
        contest = Contest( name= "test_contest", beginning = datetime.now(), cluster = cluster,
                           contest_type = contest_type, began = True )
        contest.clean(); contest.save()
        contest.registered_users.add(user)
        language = Language.objects.create(name= "test_language")
        exercise = Exercise.objects.create( name = "test_exercise",
                                            input_description = "test_input_description",
                                            output_description = "test_output_description" )
        Contest_Languages(contest = contest, language = language).save()
        Contest_Exercises(contest = contest, exercise = exercise).save()
        response = self.client.post( reverse('submit_exercise', kwargs = { 'contest_id' : contest.id,
                                                                            'exercise_id' : exercise.id }),
                                     data = { 'language': language.id, 'user': user.id, 'code': 'test_code',
                                              'exercise': exercise.id, 'contest': contest.id } )
        self.assertRedirects(response, expected_url = reverse( 'list_user_judgments',
                                                                kwargs = { 'contest_id':contest.id,
                                                                           'user_id': user.id,
                                                                           'page': 1,
                                                                           'actual_general_index': 0,
                                                                           'start_index': -49 }))
----------------------------------------------------------------------
Ran 479 tests in 50.817s

OK
```

Figura 11. Prueba unitaria. Enviar solución

```
class JudgmentTest(unittest.TestCase):

    def setUp(self):
        print "Setting judgment..."
        contestId = 102;  self.contest = Contest.get(contestId); self.exercise = self.contest.exercises[0]
        self.language = self.contest.allowedLanguages[0]
        self.judgmentId = Judgment(code = 'test_code', contestID = contestId, exerciseID = self.exercise.id,
                                   userID = 1, languageID = self.language.id, statuses = 'In process\n',
                                   time = 0, memory = 0, programSize = 0, outputSize = 0, codeFileSize = 0,
                                   submitionDate = datetime.now()).id
    def test_judgment(self):
        compiler = Program(dId = self.language.compiler.id, dsHash = self.language.compiler.dataHash,
                           parameters = self.language.compiler.getArguments())
        inputDs = []
        for ds in self.exercise.inputDSs:
            patrons = []
            for patron in ds.outputDSs:
                patrons.append(Data(dataId = patron.id, dsHash = patron.dataHash))
            inputDs.append(InputDS(dataId = ds.id, dsHash = ds.dataHash, patrons = patrons,
                                   testsAmount = ds.testsAmount))
        inputDs = self.exercise.inputDSs
        ok = judgeSolution(solutionJudgedEvent = SolutionJudgedEvent(preCode = None, postCode= None),
                  contestEndedEvent = None, eventsKargs = {}, judgmentId = self.judgmentId,
                  patronMicroFrecuency = 2300, languageId = self.language.id, exerciseId = self.exercise.id,
                  contestId = self.contest.id, userId = 1, code = 'test_code',
                  codeFileSizeLimit = self.exercise.getCodeFileSizeLimit(self.contest.id, self.language),
                  upperCodeFileSize = self.exercise.getUpperCodeFileSize(self.contest.id, self.language),
                  timeLimit = self.exercise.getTimeLimit(self.contest.id, self.language),
                  upperTime = self.exercise.getUpperTime(self.contest.id, self.language),
                  memoryLimit = self.exercise.getMemoryLimit(self.contest.id, self.language),
                  upperMemory = self.exercise.getUpperMemory(self.contest.id, self.language),
                  programSizeLimit = self.exercise.getProgramSizeLimit(self.contest.id, self.language),
                  upperProgramSize = self.exercise.getUpperProgramSize(self.contest.id, self.language),
                  outputSizeLimit = self.exercise.getOutputSizeLimit(self.contest.id, self.language),
                  upperOutputSize = self.exercise.getUpperOutputSize(self.contest.id, self.language),
                  codeCheckRegexs = [], codeCheckers = [], compiler = compiler, runner = None,
                  outputSyntax = None, inputDSs = inputDs, checker = None,
                  dsJudgedEvent = DSJudgedEvent(preCode=None, postCode=None) )
        self.assertTrue(ok)
----------------------------------------------------------------------
Ran 1 test in 0.177s

OK
```

Figura 12. Prueba unitaria. Calificar solución

En las figuras anteriores puede observarse que ambas pruebas resultaron satisfactorias, verificándose de esta manera que con ambas funcionalidades se obtiene en cada caso la respuesta esperada y que se encuentran listas para la integración. De igual forma el resto de las funcionalidades fueron probadas obteniéndose resultados satisfactorios.

Las pruebas unitarias son una herramienta útil en el desarrollo del *software* pues permiten que el mismo cumpla con la codificación esperada para garantizar el cumplimiento de todas las funcionalidades del sistema.

3.6.2 Pruebas de aceptación

Las pruebas de aceptación se realizan para determinar el nivel de satisfacción del cliente final con el producto informático. Para que estas pruebas sean lo más objetivas posible, no deben ser diseñadas por los ingenieros de *software* que desarrollan el producto, sino por el cliente. Las pruebas de aceptación se realizan a partir de las historias de usuario, cada historia de usuario se convierte en un caso de prueba en

el que el cliente especifica los puntos a probar, aunque una historia de usuario pudiera tener más de una prueba de aceptación, las que sean necesarias para garantizar la calidad del producto final con el cumplimiento de los requisitos especificados por el cliente.

Para realizar las pruebas al sistema se diseñaron los casos de prueba (ver anexo 13). En el caso de las historias de usuario que impliquen “Gestionar”, solo se describen los casos de prueba a los escenarios de una sola historia de usuario de este tipo, pues el resto son muy similares.

3.6.3 Pruebas de rendimiento

Para la realización de las pruebas de rendimiento se utilizó la herramienta Apache JMeter, la cual es de código abierto. JMeter permite el diseño de pruebas funcionales de comportamiento para medir de esta manera el rendimiento del sistema, posibilitando la simulación de diferentes cargas en un servidor para probar y analizar su resistencia en diferentes escenarios.

Se añadió el elemento “Reporte resumen”, brindado por la misma herramienta, como receptor de los resultados de las pruebas. El significado de cada uno de sus campos se describe a continuación:

Etiqueta: El nombre de la petición HTTP.

 # Muestras: El número de muestras para cada petición.

Media: El tiempo medio transcurrido para un conjunto de resultados.

Mín: El mínimo tiempo transcurrido para las muestras de la petición dada.

Máx: El máximo tiempo transcurrido para las muestras de la petición dada.

% Error: Porcentaje de las peticiones con errores.

Rendimiento: Rendimiento medido en peticiones por segundo/minuto/hora.

Kb/sec: Rendimiento medido en kilobytes por segundo.

Media de Bytes: Tamaño medio de la respuesta de la petición medido en bytes.

Etiqueta	# Muestras	Media	Mín	Máx	Desv. Estándar	% Error	Rendimiento	Kb/sec	Media de Bytes
Index	100	4762	47	31063	5819,70	0,00%	3,2/sec	32,88	10620,0
Ranking	100	1411	73	12248	2009,69	0,00%	3,2/sec	43,71	14129,0
Calificar solución	100	802	75	15388	1607,20	0,00%	3,2/sec	33,14	10624,0
Total	300	2325	47	31063	4065,56	0,00%	9,5/sec	108,99	11791,0

Figura 13. Reporte de prueba de rendimiento

En la figura se observa un escenario simulado de 100 peticiones por cada etiqueta. Dichas peticiones fueron realizadas en un intervalo, entre una y otra, de 0 segundos. La prueba se realizó sin errores y demostró que en esta situación el sistema presenta un rendimiento de 9,5 peticiones por segundo; ambas conclusiones se interpretan de los campos %Error y Rendimiento respectivamente.

3.7 Conclusiones parciales

En este capítulo se definieron las tareas de ingeniería necesarias para la realización de cada una de las historias de usuario del sistema. Se realizó la codificación del jurado en línea propuesto de acuerdo a diferentes patrones de diseño que reducen en gran medida el trabajo de implementación. Se diseñó e implementó el lenguaje de configuración de las reglas de las competencias (WCL), el cual permite la configuración de las reglas que determinan un tipo de competencia, y se generaron otros artefactos ingenieriles como los diagramas de despliegue y de componentes. Además se ejecutaron las pruebas unitarias, de rendimiento y finalmente las pruebas de aceptación al producto.

Conclusiones

Una vez finalizado el proceso de desarrollo de *software* se demostró que la implementación de un jurado en línea que permite, además de las funcionalidades básicas, la gestión y configuración de nuevas reglas de las competencias de programación, posibilita perfeccionar la gestión y configuración de competencias de programación de diferentes tipos. Lo anterior es útil para promover más habilidades entre los concursantes debido a que existe la posibilidad de crear competencias de diversa índole en interés de cualquier tipo de evento. Este jurado en línea permite también concentrar estos tipos de competencias posibilitando la gestión y reutilización de los mismos. Finalmente, el sistema desarrollado soluciona la problemática que lo originó mediante el cumplimiento de los objetivos trazados. De la investigación realizada se concluye lo siguiente:

1. El análisis de las diferentes metodologías y herramientas posibilitó la selección adecuada de la base tecnológica para el desarrollo del sistema.
2. El establecimiento de las funcionalidades y los requisitos no funcionales permitió definir desde el principio las metas a alcanzar con la implementación del sistema para que el cliente quedara satisfecho.
3. La realización de planes de iteración y de entrega permitió la estructuración y organización del trabajo en aras de lograr la obtención del producto final en tiempo.
4. La definición de una arquitectura de cuatro capas: Acceso, Comunicación, Procesamiento y Datos; permitió la separación de las responsabilidades de cada componente del sistema lográndose una integración eficiente.
5. Definir las tareas de ingeniería y los patrones de diseño para la codificación permitió organizar la etapa de implementación ahorrando tiempo de desarrollo.
6. El uso y escritura de pruebas unitarias sobre partes y funciones sensibles del sistema aseguró una codificación legible y sin errores.
7. La realización de pruebas de aceptación permitió satisfacer las expectativas del cliente, verificando que todas las funcionalidades previstas para el sistema funcionaran.
8. Las pruebas de rendimiento permitieron demostrar que el sistema funciona correctamente en escenarios de carga y estrés.

Recomendaciones

Luego de concluida la investigación y de haberse obtenido finalmente el sistema en su primera versión, se recomienda para futuras versiones:

1. Agregar más funcionalidades a los tipos de datos predefinidos en WCL para facilitar la configuración de las reglas de la competencia.
2. Confeccionar una versión más declarativa de WCL.
3. Mejorar la interfaz de administración del sistema para que pueda ser usada de manera más práctica.
4. Confeccionar un sistema de gestión de trazas para el motor de calificación, que permita registrar los sucesos que ocurren en el mismo.

Bibliografía referenciada

1. **Revilla, Miguel A.** *Competitive Learning in Informatics: The UVA. Olympiads in Informatics.* Vilnius : Institute of Mathematics and Informatics, 2008. págs. 131–148. Vol. 2. 978-959-286-010-0.

2. **SPOJ Team.** Sphere Online Judge (SPOJ). *www.spoj.com.* [En línea] 2013. http://www.spoj.com/contests/.

3. —. Sphere Online Judge (SPOJ)- Info. *www.spoj.com.* [En línea] 2013. http://www.spoj.com/info/.

4. **Sphere Research Labs.** Tutorial for problem-setters (interactive problems). *www.spoj.com.* [En línea] 2013. http://www.spoj.com/tutorials/PSINT/.

5. **SPOJ Team.** Sphere Online Judge (SPOJ). *www.spoj.com.* [En línea] 2013. http://www.spoj.com/clusters/.

6. **Timus Team.** Timus Online Judge. *acm.timus.ru.* [En línea] 2013. http://acm.timus.ru/.

7. —. Timus Online Judge. Past contests. *acm.timus.ru.* [En línea] 2013. http://acm.timus.ru/archive.aspx.

8. **PKU Team.** PKU: Contests. *poj.org.* [En línea] 2013. http://poj.org/pastcontests.

9. **Junco Vázquez, Tomás Orlando.** *Tesis presentada en opción al título de Máster en Informática Aplicada: UN JUEZ EN LÍNEA AJUSTADO A LAS NECESIDADES DE LA DOCENCIA.* La Habana : s.n., 2012.

10. **COJ Team.** COJ:About. *coj.uci.cu.* [En línea] 2013. http://coj.uci.cu/general/about.xhtml.

11. —. COJ: Real contests: Previous. *coj.uci.cu.* [En línea] 2013. http://coj.uci.cu/contest/past.xhtml?page=1.

12. **Monier Jiménez, Reymis y Nogales Cobas, Pedro Manuel.** *Trabajo de Diploma presentado para el titulo de "Ingeniero en Ciencias Informáticas". Propuesta de Jurado Online para la Cátedra de Programación Avanzada (CPAV).* La Habana : s.n., 2008.

13. **Free Software Foundation.** La Definición de Software Libre. *www.gnu.org.* [En línea] 2013. http://www.gnu.org/philosophy/free-sw.es.html.

14. **Joyanes Aguilar, Luis.** *Programación orientada a objetos.* España : McGraw-Hill, 1996. 84-481-0590-7 .

15. **Django Software Foundation.** The Django framework. *www.djangoproject.com.* [En línea] 2013. https://www.djangoproject.com/.

16. **Rodríguez Salas, Jesús Javier.** *Introducción a la programación. Teoría y práctica.* Alicante : Club Universitario, 2003. pág. 4. 84-8454-274-2.

17. **Python community.** About Python. *python.org.* [En línea] 2013. http://python.org/about/.

18. **Garcia de Jalón, Javier, Ignacio Rodriguez, Jose, Mingo, Iñigo, Imaz, Aitor y otros.** Aprenda Java como si estuviera en primero. *www.tecnun.es.* [En línea] 2000. http://www.tecnun.es/asignaturas/Informat1/AyudaInf/aprendainf/Java/Java2.pdf.

19. **Oracle.** ¿Qué es java y por qué lo necesito? *www.java.com.* [En línea] 2013. http://www.java.com/es/download/faq/whatis_java.xml.

20. **gcc team.** GCC, the GNU Compiler Collection. *gcc.gnu.org.* [En línea] 2013. http://gcc.gnu.org/.

21. **Programación en Castellano.** Frameworks para el desarrollo de aplicaciones web y moviles. *www.programacion.com.* [En línea] 2013. http://www.programacion.com/articulo/frameworks_para_el_desarrollo_de_aplicaciones_web_y_moviles_722.

22. **Sánchez Asenjo, Jorge.** Apuntes Completos sobre Sistemas gestores de Bases de Datos. *ubuntuone.com.* [En línea] 2009. http___ubuntuone.pdf---http://ubuntuone.com/p/sqt/.

23. **Martínez, Rafael.** Sobre PostgreSQL. *www.postgresql.org.es.* [En línea] 2010. http://www.postgresql.org.es/sobre_postgresql.

24. **The PostgreSQL Global Development Group.** PostgreSQL: PostgreSQL Featured Users. *www.postgresql.org.* [En línea] 2013. http://www.postgresql.org/about/users/.

25. **Oracle.** About MySQL. *www.mysql.com.* [En línea] 2013. http://www.mysql.com/about/.

26. **Apache Software Foundation.** Apache Qpid: Introduction. *qpid.apache.org.* [En línea] 2013. http://qpid.apache.org/.

27. **Sackman, Matthew.** Messaging In The Cloud. *www.rabbitmq.com.* [En línea] 2010. http://www.rabbitmq.com/resources/RabbitMQ_MessagingInTheCloud_VMworld_2010_MS.pdf.

28. **VMware.** RabbitMQ Features. *www.rabbitmq.com.* [En línea] 2013. http://www.rabbitmq.com/features.html.

29. **Turakhia, Bhavin.** RabbitMQ vs Apache ActiveMQ vs Apache qpid. *bhavin.directi.com.* [En línea] 2010. http://bhavin.directi.com/rabbitmq-vs-apache-activemq-vs-apache-qpid/.

30. **Bryant, Russell.** Features/OpenStack using Qpid. *fedoraproject.org.* [En línea] 2013. http://fedoraproject.org/wiki/Features/OpenStack_using_Qpid.

31. **Apache Software Foundation.** ApacheQpid Faq. *cwiki.apache.org.* [En línea] 2013. https://cwiki.apache.org/qpid/faq.html.

32. **Mateu, Carles.** *Desarrollo de Aplicaciones Web.* Catalunya : Fundació per a la Universitat Oberta de Catalunya, 2004. pág. 2. 84-9788-118-4.

33. **Kabir, Mohammed J.** *Apache Server 2 Bible.* New York : Hungry Minds, Inc, 2002. págs. 4-7. 0-7645-4821-2.

34. **GUTL.** Apache sigue consolidándose como el mejor servidor Web. *gutl.jovenclub.cu.* [En línea] 2013. http://gutl.jovenclub.cu/apache-sigue-consolidandose-como-el-mejor-servidor-web/.

35. **S.Tanembaum, Andrew.** *Sistemas operativos Modernos.* México : Pearson Educación, 2003. pág. 34. 970-26-0315-3.

36. **Debian Project.** About Debian. *www.debian.org.* [En línea] 2013. http://www.debian.org/intro/about.

37. —. Debian Releases. *www.debian.org.* [En línea] 2013. http://www.debian.org/releases/.

38. **Perens, Bruce, Rudolph, Sven, Grobman, Igor, Treacy, James y Di Carlo, Adam.** Guía de instalación de Debian. *www.debian.org.* [En línea] 2004. http://www.debian.org/releases/stable/i386/install.pdf.es.

39. **Debian Project.** Who's using Debian? *www.debian.org.* [En línea] 2013. http://www.debian.org/users/.

40. **Comunidad Fedora.** Acerca de Fedora. *fedoraproject.org.* [En línea] 2013. http://fedoraproject.org/es/about-fedora.

41. —. Proyecto Fedora: Caracteristicas. *fedoraproject.org.* [En línea] 2013. http://fedoraproject.org/es/features/.

42. **Ubuntu Project.** The Ubuntu story. *www.ubuntu.com.* [En línea] 2013. http://www.ubuntu.com/project/about-ubuntu.

43. **Ubuntu Documentation Project.** *Ubuntu Server Guide.* s.l. : Fultus Corporation, 2011. pág. 1. 978-1-59682-260-3.

44. **Ramos Salavert, Isidro y Lozano Pérez, María Dolores.** *Ingeniería Del Software Y Bases de Datos. Tendencias Actuales.* Castilla-La Mancha : Univ. Castilla-La Mancha, 2000. pág. 78. 84-8427-077-7.

45. **Ubuntu Documentation Project.** Mi primera hora con Eclipse. *ubuntulife.files.wordpress.com.* [En línea] 2008. http://ubuntulife.files.wordpress.com/2008/03/intro_eclipse_espanol.pdf.

46. **Eclipse Foundation.** Eclipse Public License - v 1.0. *www.eclipse.org.* [En línea] 2013. http://www.eclipse.org/legal/epl-v10.html.

47. **Oracle.** An Introduction to Netbeans. *netbeans.org.* [En línea] 2013. http://netbeans.org/about/index.html.

48. **Administrador.** Netbeans.org detalla el nuevo mapa del marco de herramientas Java. *www.faq-mac.com.* [En línea] 2003. http://www.faq-mac.com/noticias/netbeansorg-detalla-nuevo-mapa-marco-herramientas-java/5152.

49. **Debian Project.** Computer Language Benchmarks Game, x64 Ubuntu : Intel® Q6600® quad-core. *benchmarksgame.alioth.debian.org.* [En línea] 2012. http://benchmarksgame.alioth.debian.org/u64q/which-programs-are-fastest.php.

50. —. Computer Language Benchmarks Game, Ubuntu : Intel® Q6600® one core. *benchmarksgame.alioth.debian.org.* [En línea] 2012. http://benchmarksgame.alioth.debian.org/u32/which-programs-are-fastest.php.

51. —. Computer Language Benchmarks Game, x64 Ubuntu : Intel® Q6600® one core. *benchmarksgame.alioth.debian.org.* [En línea] 2012. http://benchmarksgame.alioth.debian.org/u64/which-programs-are-fastest.php.

52. —. Computer Language Benchmarks Game, Ubuntu : Intel® Q6600® quad-core. *benchmarksgame.alioth.debian.org.* [En línea] 2012. http://benchmarksgame.alioth.debian.org/u32q/which-programs-are-fastest.php.

53. **Python Software Fundation.** Extending and Embedding the Python Interpreter. *docs.python.org.* [En línea] 2013. http://docs.python.org/2/extending/.

54. **rigaux.org Team.** Syntax Across Languages. *rigaux.org.* [En línea] 2008. http://rigaux.org/language-study/syntax-across-languages.html.

55. **Python Software Foundation.** The Python Language Reference. *docs.python.org.* [En línea] 2013. http://docs.python.org/2/reference/.

56. **cplusplus.com Team.** C++ Documentation. *www.cplusplus.com.* [En línea] 2013. http://www.cplusplus.com/doc/.

57. **Oracle.** The Java Language Specification, Java SE 7 Edition. *docs.oracle.com.* [En línea] 2012. http://docs.oracle.com/javase/specs/jls/se7/jls7.pdf.

58. **Canonical Ltd.** Ubuntu and Debian. *www.ubuntu.com.* [En línea] 2013. http://www.ubuntu.com/project/about-ubuntu/ubuntu-and-debian.

59. **The Apache Software Foundation.** ApacheQpid Downloads. *qpid.apache.org.* [En línea] 2013. http://qpid.apache.org/download.html.

60. **VMware.** Downloading and Installing RabbitMQ. *www.rabbitmq.com.* [En línea] 2013. http://www.rabbitmq.com/download.html.

61. **Lighttpd Developers.** Overview. *redmine.lighttpd.net.* [En línea] 2013. http://redmine.lighttpd.net/projects/lighttpd.

62. —. Benchmark. *www.lighttpd.net.* [En línea] 2007. http://www.lighttpd.net/benchmark/.

63. —. Lighttpd Developers. *redmine.lighttpd.net.* [En línea] 2013. http://redmine.lighttpd.net/projects/lighttpd/wiki/DevelopersList.

64. —. Documentation Overview. *redmine.lighttpd.ne.* [En línea] 2013. http://redmine.lighttpd.net/projects/lighttpd/wiki/Docs.

65. **Lopez Ortega, Alvaro.** Cherokee Web Server. *www.cherokee-project.com.* [En línea] 2013. http://www.cherokee-project.com/.

66. **Oracle.** NetBeans Plugin Portal. *plugins.netbeans.org.* [En línea] 2013. http://plugins.netbeans.org/PluginPortal/.

67. **Appcelerator Inc.** What is PyDev. *pydev.org.* [En línea] 2013. http://pydev.org/.

68. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James.** *El proceso unificado de desarrollo de software.* Madrid : Addison Wesley, 2000. págs. 1-6. 84-7829-036-2.

69. **Letelier, Patricio.** Rational Unified Process (RUP). *eva.uci.cu.* [En línea] 2010. http://eva.uci.cu/mod/resource/view.php?id=9303&subdir=/Metodologias/RUP.

70. **Fraile, Luis.** Microsoft_Solutions_Framework_Agile_MSF_Agil. *eva.uci.cu.* [En línea] http://eva.uci.cu/file.php/161/Documentos/Materiales_complementarios/UD_1_Procesos/Metodologias/MSF/Microsoft_Solutions_Framework_Agile_MSF_Agil_.pdf.

71. **Scalzone, Patricia.** Microsoft Solution Framework v.4 Agile. *eva.uci.cu.* [En línea] 2011. http://eva.uci.cu/file.php/161/Documentos/Materiales_complementarios/UD_1_Procesos/Metodologias/MSF/Microsoft_Solution_Framework_v.4_Agile_MSF_.ppt.

72. **Letelier, Patricio y Penadés, Mª Carmen.** Métodologías ágiles para el desarrollo de software. *eva.uci.cu.* [En línea] 2010. http://eva.uci.cu/file.php/161/Documentos/Materiales_complementarios/UD_1_Procesos/Metodologias/XP/XP_y_otras_Metodologias_Agiles.pdf.

73. **Palacio, Juan.** El Modelo Scrum. *eva.uci.cu.* [En línea] 2006. http://eva.uci.cu/file.php/161/Documentos/Materiales_complementarios/UD_1_Procesos/Metodologias/SCRUM/El_modelo_SCRUM.pdf.

74. **Fernández Escribano, Gerardo.** Introducción a Extreme Programming. *eva.uci.cu.* [En línea] 2002. http://eva.uci.cu/file.php/161/Documentos/Materiales_complementarios/UD_1_Procesos/Metodologias/XP/Presentacion-XP.pdf.

75. **González Duque, Raúl.** Python para todos. *edge.launchpad.net.* [En línea] 2010. http://edge.launchpad.net/improve-python-spanish-doc/0.4/0.4.0/+download/Python%20para%20todos.pdf.

Bibliografía consultada

AMQP Working Group. AMQP Working Group 0-8. *www.amqp.org.* [En línea] 2006. http://www.amqp.org/specification/0-8/amqp-org-download.

Cabral, Juan B. PET: Python entre todos. *revista.python.org.ar.* [En línea] 2011. http://revista.python.org.ar/5/es/html/pet-python-entre-todos.html#zen-vs-zen.

Comer, James. ICPC Wiki: Regional Rules. *icpc.baylor.edu.* [En línea] 2012. http://icpc.baylor.edu/info/Regional+Rules.

Fisanotti, Juan Pedro. Introducción a Django. *revista.python.org.ar.* [En línea] 2011. http://revista.python.org.ar/2/es/html/intro_django.html.

Gamma, Eric, Helm, Richard, Johnson, Ralph y Vlissides, John. *Design Patterns.Elements of Reusable Software.* New York : Addison-Wesley, 1995. págs. 12-14. 978-0201634983.

gcc team. GCC Development Mission Statement (1999-04-22). *gcc.gnu.org.* [En línea] 2013. http://gcc.gnu.org/gccmission.html.

Grigoras, Dan. *Programming Models for Cluster Computing.* Cork : Springer-Verlag, 2002. pág. 1. 3-540-43672-3 .

Guarino, Joseph. Ubuntu Linux Server Edition. *www.evolutionaryit.com.* [En línea] 2007. http://www.evolutionaryit.com/presentations/Ubuntu_Linux_Server_Overview.pdf.

Larson, James A. Interactive software tools for building interactive user interfaces. New Jersey : Yourdon Press, 1992. 9780139240447 .

Open Source Initiative. The Open Source Definition. *opensource.org.* [En línea] 2013. http://opensource.org/docs/osd.

Riggs, Simon y Krosing, Hannu. *PostgreSQL 9 Administration Cookbook.* Birmingham : Packt Publishing Ltd., 2010. pág. 8. 978-1-849510-28-8.

Sphere Research Labs. Tutorial for contest-setters. *www.spoj.com.* [En línea] 2013. http://www.spoj.com/tutorials/CS/.

Glosario de términos

Calificación automática: Proceso que consiste en evaluar una solución a un ejercicio enviada por el usuario y emitir el resultado de la calificación. Para la evaluación, se verifican las restricciones establecidas para dicho ejercicio, se ejecuta el código de la solución y luego se chequea la salida del programa.

Chequeador: Se encarga de determinar si la salida de una respuesta a un ejercicio es o no correcta sintáctica y semánticamente, verificando generalmente que la misma sea exactamente igual al juego de datos de salida que se posee de antemano.

Clúster de calificación: En este documento se refiere a una agrupación lógica de motores de calificación que se comportan con características similares.

Competencia de programación: Evento que tiene como objetivo potenciar el aprendizaje y motivación hacia la programación de computadoras. Se centra en el desarrollo de algoritmos y su codificación.

Evento del sistema: Suceso que provoca la ejecución de un conjunto de acciones que pueden o no alterar el comportamiento de algunas partes específicas del sistema.

Juego de datos: Colección de los datos de entrada y de salida esperada para un ejercicio y que es usada para calificarlo.

Jurado en línea: Aplicación web que incluye la descripción de varios problemas que pueden ser solucionados mediante técnicas de programación. Evalúa las soluciones enviadas por los usuarios de manera automática y emite una calificación.

Motor de calificación: *Software* encargado de realizar la calificación automática. Chequea la realización de los eventos del sistema y ejecuta las acciones relacionadas con los mismos.

Reglas de la competencia: Definen el comportamiento de una competencia dentro del sistema y las acciones permitidas a los concursantes de la misma en el sistema. No se refieren solamente a los métodos o fórmulas que se usen para asignar o sustraer puntos a determinado usuario, sino también a las condiciones de comienzo y finalización de una competencia, la forma en que se muestra el *ranking* y los parámetros por los que se ordena, la forma en que son mostrados los resultados, las restricciones de calificación manejadas por cada lenguaje permitido para cada ejercicio, la cantidad de veces que un usuario puede emitir una posible solución a un problema determinado, entre otras características.

Restricción de calificación: Representa los límites de tiempo, memoria y tamaño de código, entre otros, con los que una solución a determinado ejercicio debe cumplir para que sea válida.

Wormhole Configuration Language (WCL): Minilenguaje creado por los desarrolladores del sistema que permite la configuración de las reglas de la competencia.

Anexos

Anexo 1: Plan de iteraciones

Iteración	No. HU	Historia de Usuario	Semanas estimadas
Iteración 1	1	Autenticar usuario	9
	2	Calificar solución	
	3	Gestionar Competencia	
	4	Consultar *ranking*	
	5	Consultar envíos realizados	
	6	Registrar usuario	
	7	Modificar perfil	
	8	Gestionar usuarios	
	9	Gestionar tipos de competencias	
Iteración 2	10	Gestionar ejercicios	7
	11	Gestionar lenguajes	
	12	Gestionar clústeres de calificación	

Tabla 15. Plan de iteraciones

Anexo 2: Plan de entrega

Entregable	Fin Iteración 1	Fin Iteración 2
WormHole Online Judge	Marzo/2013	Mayo/ 2013

Tabla 16. Plan de entrega

Anexo 3: Tarjetas CRC de la aplicación motor de calificación

Módulo de comunicación con el servidor RabbitMQ

Clase Administrador de mensajes	
Responsabilidades	**Colaboradores**
Iniciar el recibo de mensajes	
Recibir mensajes	
Crear cola personal	
Crear cola de calificación	
Enviar mensaje	

Tabla 17. CRC. Administrador de mensajes

Módulo de comunicación con la base de datos

Clase Administrador de base de datos	
Responsabilidades	**Colaboradores**
Obtener configuración del motor	Clase persistente de motor Clase persistente de configuración del sistema
Añadir calificación de juego de datos	Clase persistente de calificación de juego de datos
Refrescar calificación	Clase persistente de calificación
Sincronizar caché	Clase persistente de competencia Clase persistente de ejercicio Clase persistente de lenguaje Clase persistente de tipo de competencia

Tabla 18. CRC. Administrador de base de datos

Módulo de calificación

Clase Juez	
Responsabilidades	**Colaboradores**
Chequear el tamaño del código	
Chequear restricciones de código	
Compilar	
Ejecutar fichero compilado	Clase Supervisor
Chequear la salida del programa	
Ejecutar acciones de evento de calificación	Clase Evento

Tabla 19. CRC. Juez

Clase Supervisor	
Responsabilidades	**Colaboradores**
Supervisar ejecución restringida	

Tabla 20. CRC. Supervisor

Módulo de intérprete

Clase Evento	
Responsabilidades	**Colaboradores**
Interpretar código WCL	Clase Contexto Clase persistente de competencia

Tabla 21. CRC. Evento

Clase Contexto	
Responsabilidades	**Colaboradores**
Crear variable	Clase Registro de competencia
Obtener valor de variable	Clase Registro de competencia
Asignar valor de variable	Clase Registro de competencia

Tabla 22. CRC. Contexto

Clase Registro de competencia	
Responsabilidades	**Colaboradores**
Crear variable de registro	Clase persistente de competencia
Obtener valor de variable de registro	Clase persistente de competencia
Asignar valor de variable de registro	Clase persistente de competencia
Acceder a funciones permitidas	Clase persistente de competencia

Tabla 23. CRC. Registro de competencia

Módulo de manejo de eventos del sistema

Clase Administrador de eventos	
Responsabilidades	**Colaboradores**
Ejecutar eventos de competencia	Clase Evento
Ejecutar eventos de calificación	Clase Evento

Tabla 24. CRC. Administrador de eventos

Anexo 4: Tarjetas CRC de la aplicación web

Módulo de autenticación

Clase Gestor de usuarios	
Responsabilidades	**Colaboradores**
Agregar usuario	Clase persistente de usuario Clase persistente de grupo
Modificar usuario	Clase persistente de usuario Clase persistente de grupo
Eliminar usuario	Clase persistente de usuario
Mostrar datos de usuario	Clase persistente de usuario Clase persistente de grupo

Tabla 25. CRC. Gestor de usuarios

Clase Gestor de grupos	
Responsabilidades	**Colaboradores**

Agregar grupo	Clase persistente de grupo
Modificar grupo	Clase persistente de grupo
Eliminar grupo	Clase persistente de grupo
Mostrar datos de grupo	Clase persistente de grupo

Tabla 26. CRC. Gestor de grupos

Módulo de gestión de competencias

Clase Gestor de competencias	
Responsabilidades	**Colaboradores**
Agregar competencia	Clase persistente de competencia Clase persistente de tipo de competencia Clase persistente de ejercicio Clase persistente de lenguaje
Modificar competencia	Clase persistente de competencia Clase persistente de tipo de competencia Clase persistente de ejercicio Clase persistente de lenguaje
Eliminar competencia	Clase persistente de competencia
Mostrar datos de competencia	Clase persistente de competencia Clase persistente de tipo de competencia Clase persistente de ejercicio Clase persistente de lenguaje

Tabla 27. CRC. Gestor de competencias

Clase Gestor de lenguaje	
Responsabilidades	**Colaboradores**
Agregar lenguaje	Clase persistente de lenguaje Clase persistente de programa
Modificar lenguaje	Clase persistente de lenguaje Clase persistente de programa
Eliminar lenguaje	Clase persistente de lenguaje
Mostrar datos de lenguaje	Clase persistente de lenguaje Clase persistente de programa

Tabla 28. CRC. Gestor de lenguaje

Clase Gestor de ejercicio	
Responsabilidades	**Colaboradores**
Agregar ejercicio	Clase persistente de ejercicio

Modificar ejercicio	Clase persistente de ejercicio
Eliminar ejercicio	Clase persistente de ejercicio
Mostrar datos de ejercicio	Clase persistente de ejercicio

Tabla 29. CRC. Gestor de ejercicio

Clase Gestor de calificación	
Responsabilidades	**Colaboradores**
Mostrar datos de calificación	Clase persistente de calificación

Tabla 30. CRC. Gestor de calificación

Módulo de gestión de calificación

Clase Gestor de motor	
Responsabilidades	**Colaboradores**
Agregar motor	Clase persistente de motor
Modificar motor	Clase persistente de motor
Eliminar motor	Clase persistente de motor
Mostrar datos de motor	Clase persistente de motor

Tabla 31. CRC. Gestor de motor

Clase Gestor de clúster	
Responsabilidades	**Colaboradores**
Agregar clúster	Clase persistente de clúster Clase persistente de motor
Modificar clúster	Clase persistente de clúster Clase persistente de motor
Eliminar clúster	Clase persistente de clúster
Mostrar datos de clúster	Clase persistente de clúster Clase persistente de motor

Tabla 32. CRC. Gestor de clúster

Módulo de gestión de configuración del sistema

Clase Gestor de configuración del sistema	
Responsabilidades	**Colaboradores**
Modificar configuración	Clase persistente de configuración del sistema

Tabla 33. CRC. Gestor de configuración del sistema

Anexo 5: Diagrama de clases del módulo de comunicación del motor de calificación con el RabbitMQ

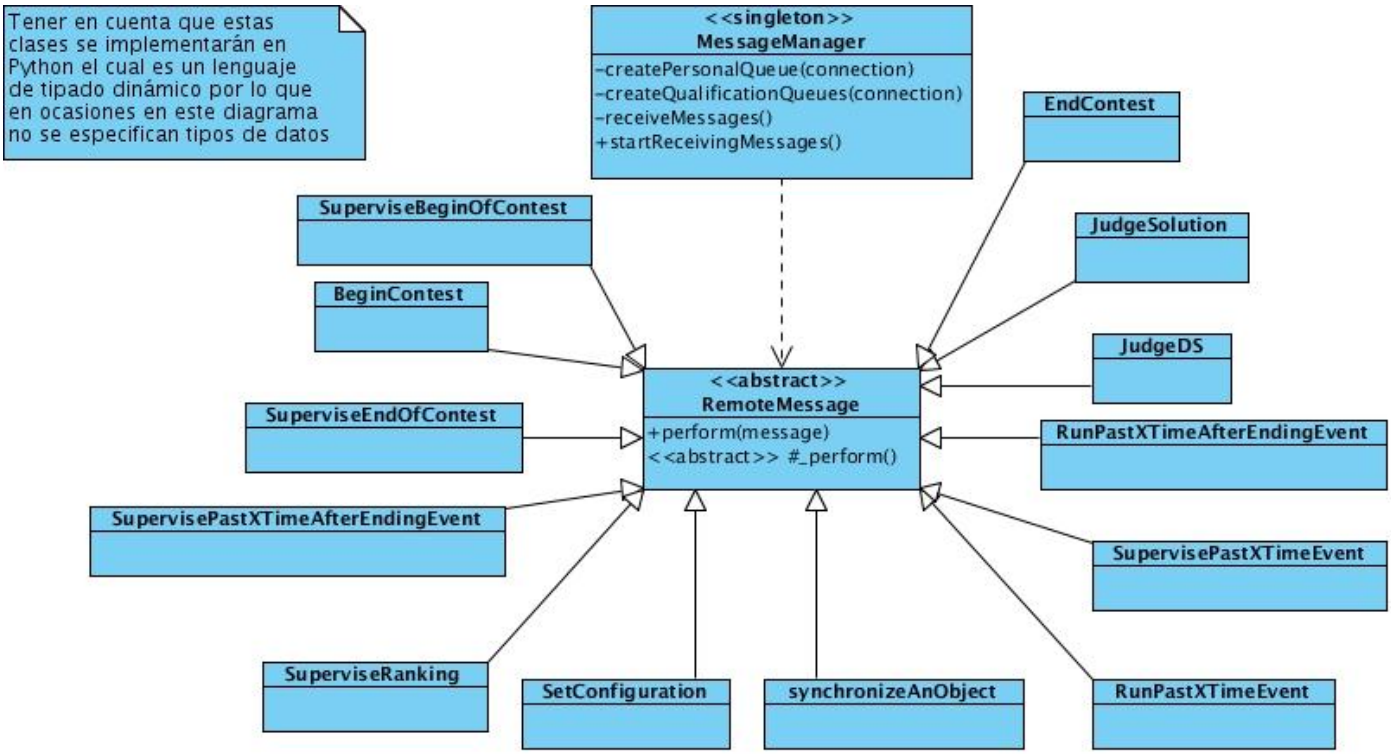

Figura 14. Módulo de comunicación del motor de calificación con el RabbitMQ

Anexo 6: Diagrama de clases del módulo de comunicación del motor de calificación con la base de datos

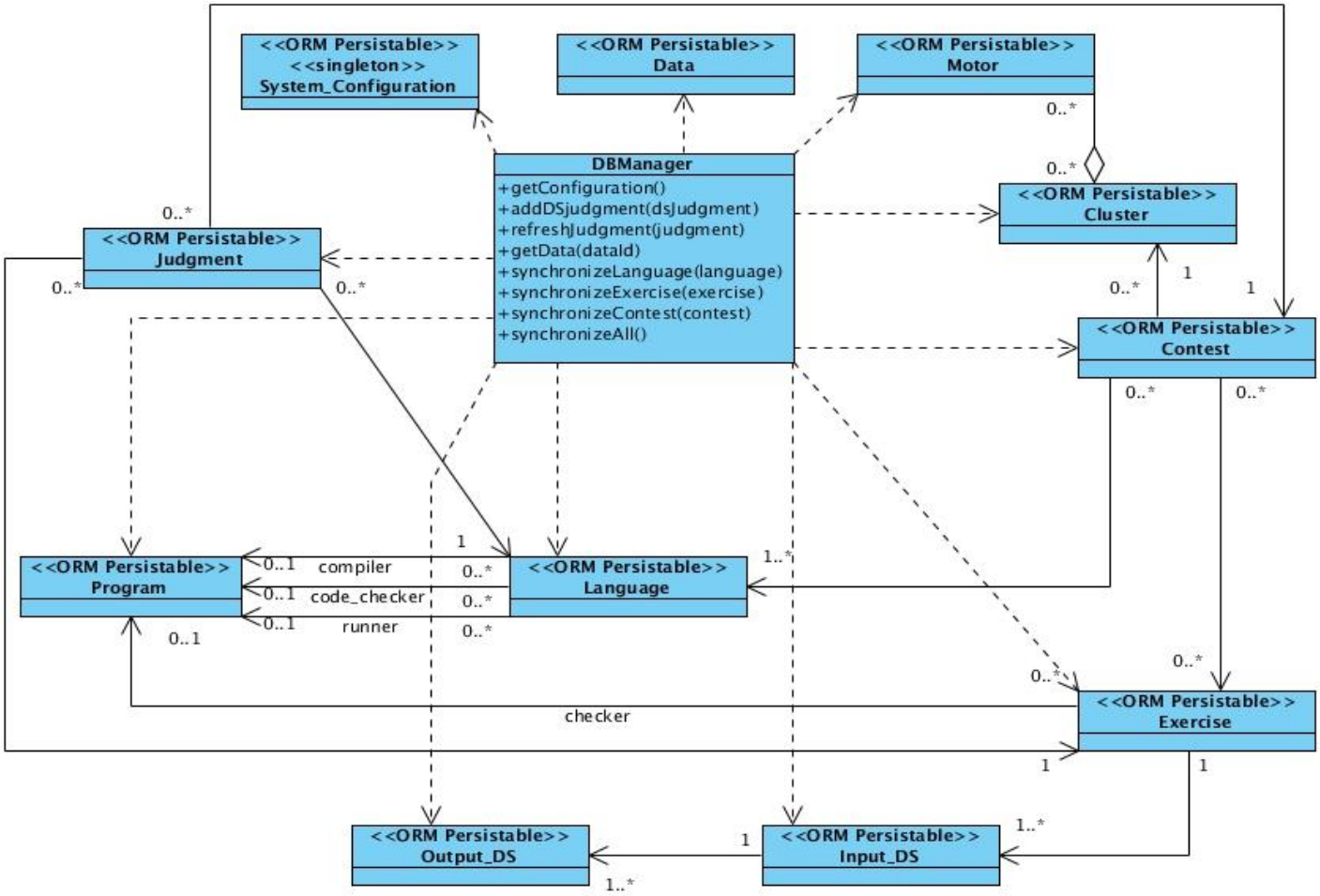

Figura 15. Módulo de comunicación del motor de calificación con la base de datos

Anexo 7: Diagrama de clases del módulo de calificación del motor de calificación

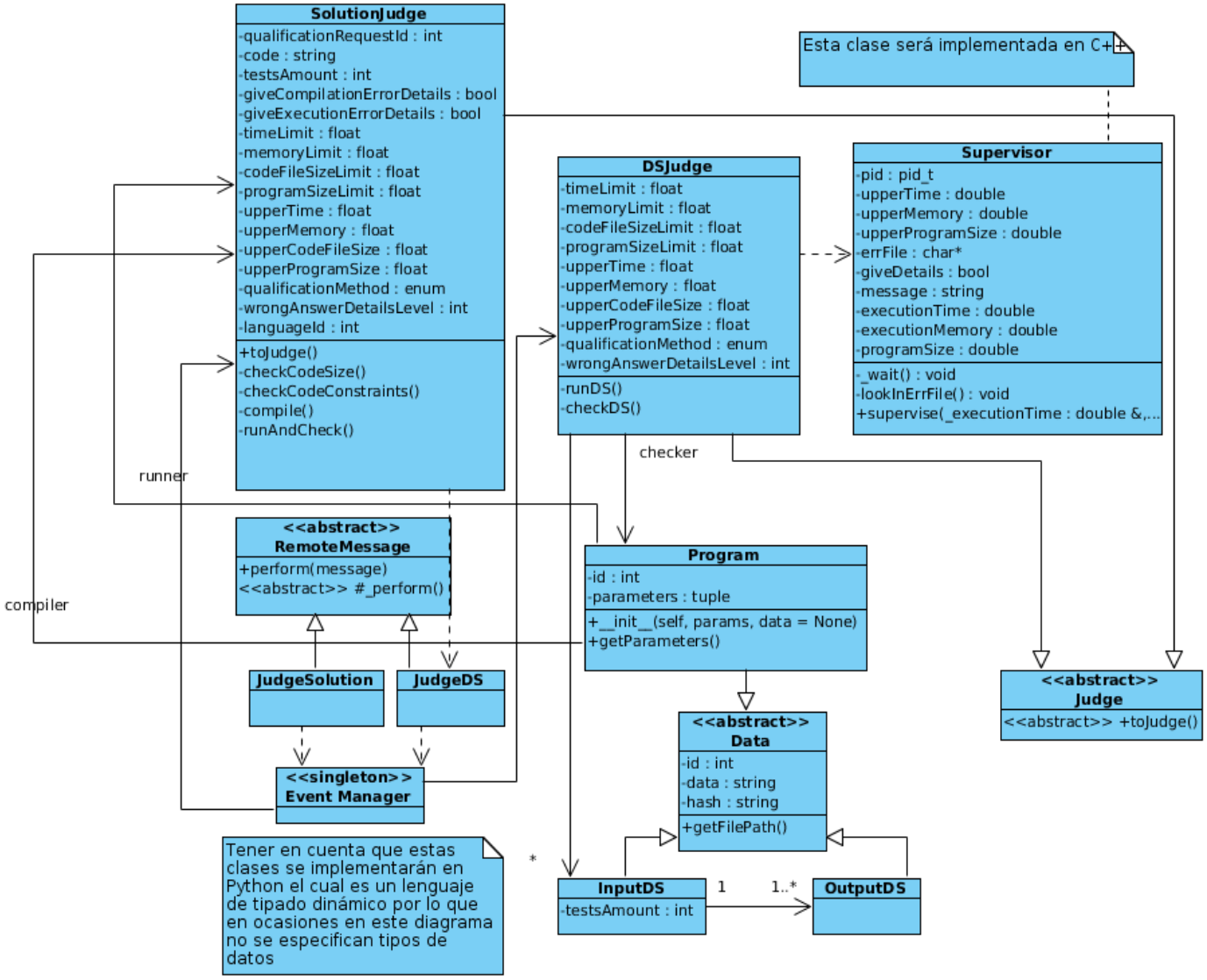

Figura 16. Módulo de calificación del motor de calificación

Anexo 8: Diagrama de clases del módulo de intérprete del motor de calificación

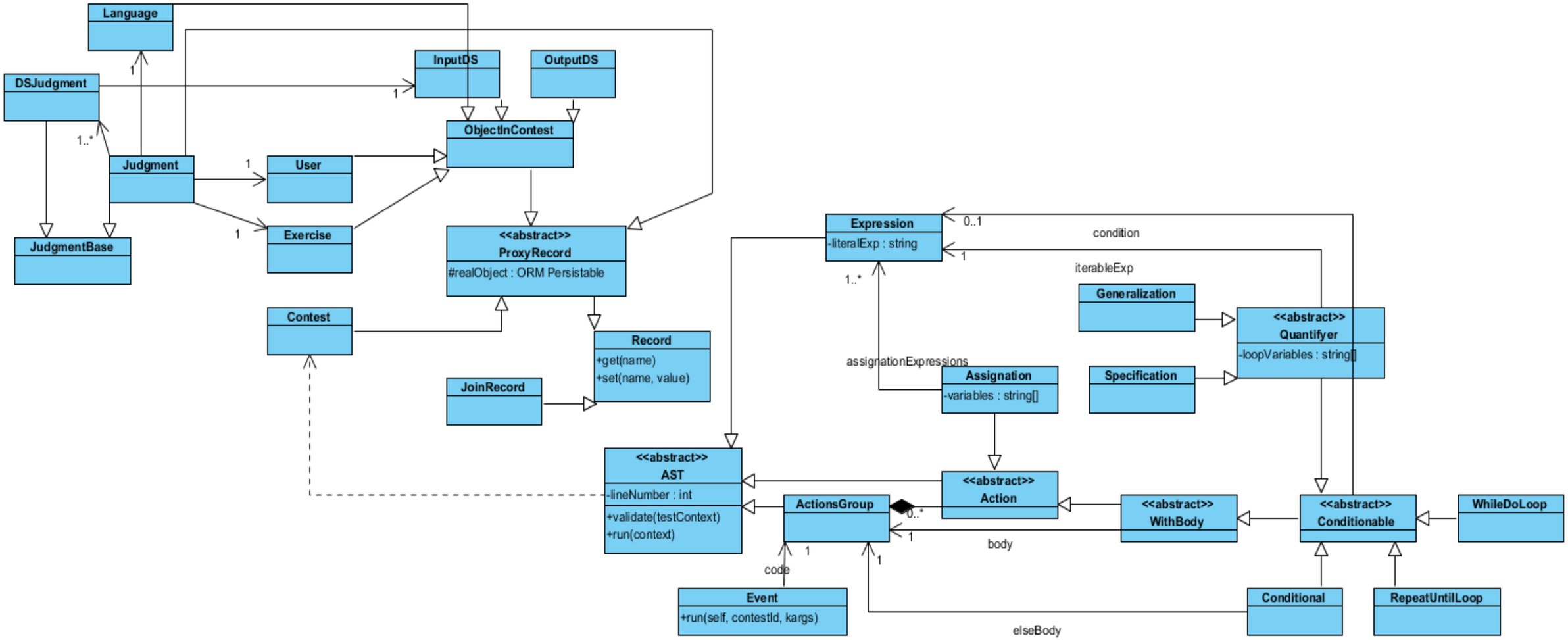

Figura 17. Módulo de intérprete del motor de calificación

Anexo 9: Diagrama de clases del módulo de manejo de eventos del sistema del motor de calificación

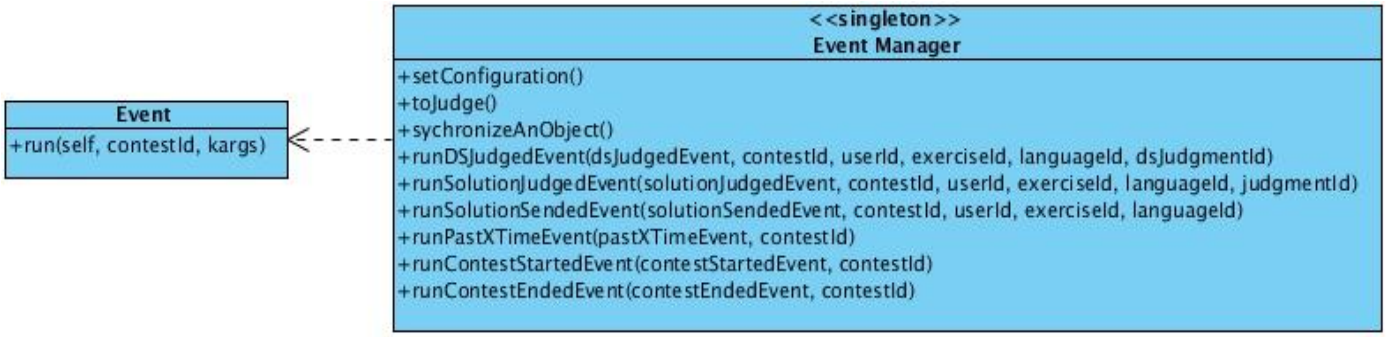

Figura 18. Módulo de manejo de eventos del sistema del motor de calificación

Anexo 10: Diagrama de clases de la aplicación web para el escenario de envío de una solución

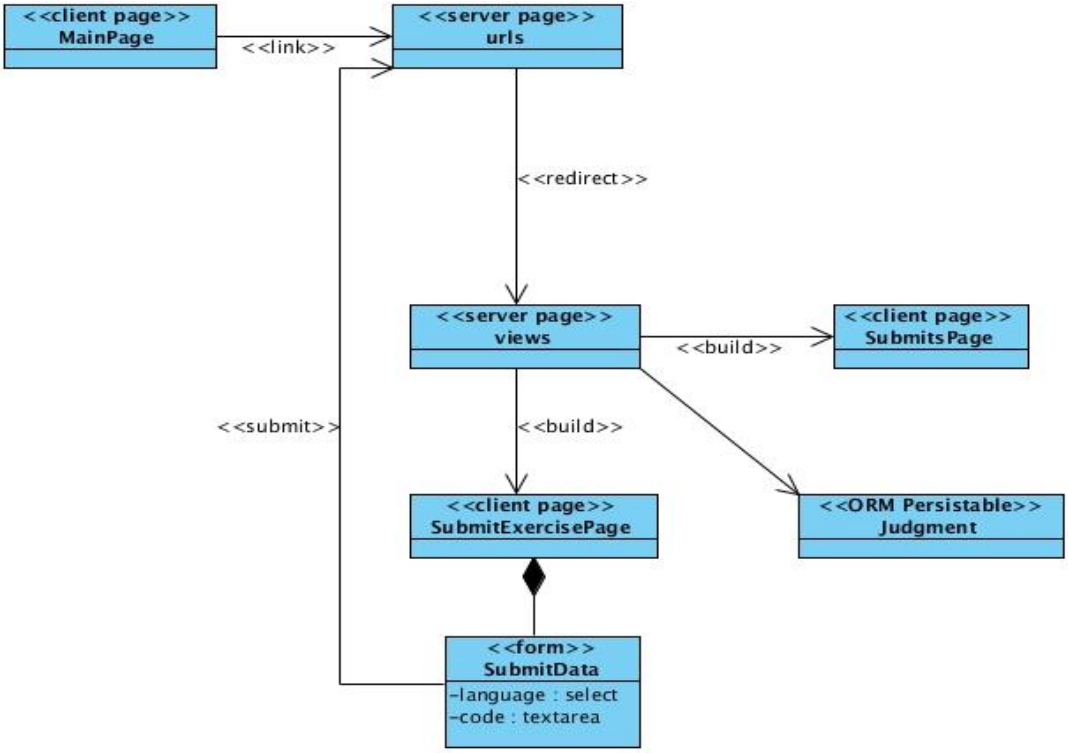

Figura 19. Diagrama de clases de la aplicación web para el escenario de envío de una solución

Anexo 11: Diagrama de clases persistentes

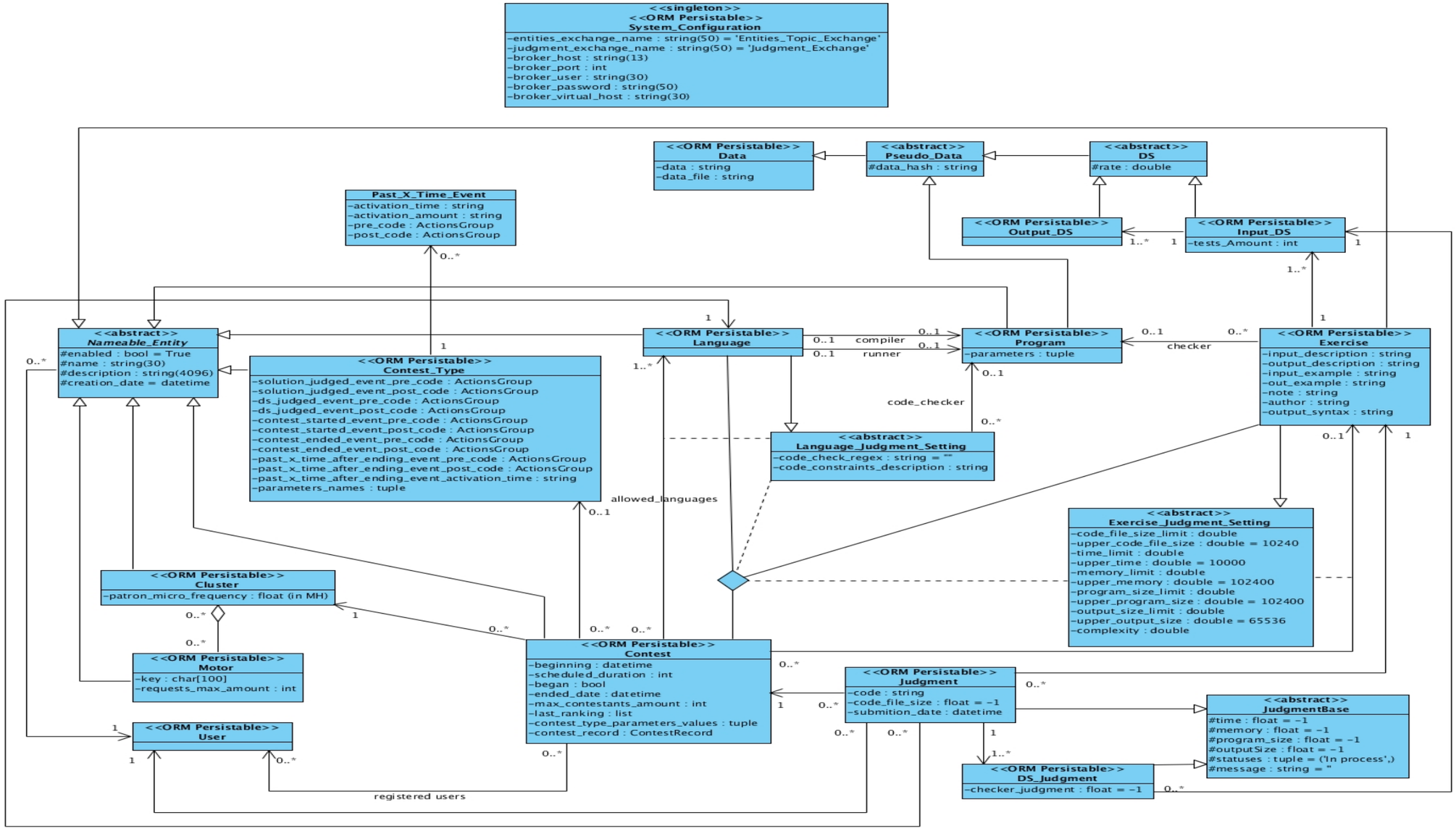

Figura 20. Diagrama de clases persistentes

Anexo 12: Configuración de las reglas de la competencia

Para la configuración de las distintas reglas de las competencias se utiliza un minilenguaje denominado Wormhole Configuration Language (WCL).
WCL es un lenguaje que especifica las acciones que el sistema debe realizar al momento de la ocurrencia de un evento. Estas acciones son interpretadas por los motores de calificación y configuradas en la interfaz web.

Eventos

Un evento representa una situación determinada que está relacionada con un proceso interno de la competencia. Los eventos se activan en el momento en que se cumplen las condiciones de la situación a la que están asociados. Cuando un evento es activado el sistema ejecuta un conjunto de acciones por defecto que definen el evento en sí. Por ejemplo, cuando un usuario envía una solución para ser calificada, el sistema reconoce que se ha activado el evento de calificación de solución y ejecuta las acciones de dicha situación, que en este caso son, entre otras, calificar la solución y guardar los resultados en la base de datos. Las reglas de las competencias son configuradas especificando las acciones que el sistema debe realizar en el momento justo antes o después de la ocurrencia de cada evento. Al momento de la activación de un evento se ejecutan en orden:

- Acciones configuradas para ser ejecutadas antes de la ejecución del evento.
- Acciones por defecto.
- Acciones configuradas para ser ejecutadas después de la ejecución del evento.

A continuación se presentan los eventos reconocidos por el sistema y sus significados.

- **Evento de comienzo de competencia:** Este evento es activado justamente al inicio de la competencia. Las acciones por defecto del evento son:
  - Marcar como comenzada la competencia en la base de datos.
- **Evento de calificación de solución:** Este evento se activa cuando una solución es enviada por un usuario. Dentro de este evento se crea automáticamente una variable llamada $\_judgment$ de tipo $Judgment$[62] que representa el resultado de calificación actual. Las acciones por defecto del evento son:
  - Chequear restricciones de código.
  - Compilar código.
  - Por cada juego de datos de entrada, activar el evento de calificación de juegos de datos de manera concurrente.
  - Esperar por la finalización de la calificación de juegos de datos.

[62] Ver tipos de datos para más información.

- Actualizar datos en la base de datos.

- **Evento de calificación de juego de datos:** Este evento es activado cuando un juego de datos es calificado. Dentro de este evento se crea automáticamente una variable llamada $\_dsJudgment$ de tipo $DSJudgment$[63] que representa el resultado de calificación actual. Las acciones por defecto del evento son:
  - Ejecutar solución chequeando límites de calificación.
  - Chequear la salida de la solución.
  - Actualizar datos en la base de datos.
- **Evento de ronda de tiempo:** Este evento se activa cada vez que ha trascurrido una cantidad de tiempo determinado durante una competencia que aún no ha terminado. Toma como parámetro dos enteros que representan el tiempo en segundos en que debe activarse y la cantidad de veces que debe hacerlo respectivamente. Si la cantidad de veces es igual o menor que 0 , significa que el evento no tiene un límite de números de veces de activación. Las acciones por defecto de este evento son:
  - Actualizar la cantidad de veces que se ha activado y la fecha de la próxima vez que se activará.
- **Competencia finalizada:** Este evento es activado justamente al finalizar la competencia. Las acciones por defecto de este evento son:
  - Actualizar la fecha en que terminó la competencia y desmarcarla como comenzada en la base de datos.
- **Evento de ronda de tiempo luego de finalizada la competencia:** Este evento es activado al transcurrir un tiempo luego de terminada la competencia. Toma como parámetro el tiempo (en segundos) de activación. No tiene acciones por defecto.

Variables

En WCL se hace uso de variables, tanto para guardar valores temporales, como para especificar el valor de opciones dentro de la competencia.

El identificador de una variable constituye una cadena de caracteres que sigue el mismo patrón regular de los identificadores de las variables usadas por el lenguaje Python, la única diferencia es que dicho identificador no puede comenzar con dos *underscore* ("_") seguidos uno de otro. Existe una variable llamada $contest$, la cual es un registro de competencia. A través de ella se puede acceder a todas las variables y definiciones del sistema, sin embargo, para abreviar, se pueden usar directamente los identificadores de los atributos de esta variable. Estos atributos sirven para acceder o modificar algunas

[63] Ver tipos de datos para más información.

características de la competencia, como los ejercicios y usuarios que están registrados dentro de la misma.

También es posible definir variables nuevas que sirvan para guardar algún valor temporal.

Las variables definidas por el sistema pueden clasificarse en dos tipos según su nivel de accesibilidad:

- Variables de solo lectura.
- Variables de lectura y escritura.

Las variables de solo lectura son aquellas cuyos valores no pueden ser modificados. Existen solo para brindar alguna información que podría ser valiosa para tomar decisiones. Sin embargo, a las variables de lectura y escritura, se les puede modificar su valor para de esta forma alterar el desarrollo de la competencia, la asignación de puntos a los usuarios en la misma u otros aspectos. Las variables del sistema siempre contienen información referente solo dentro del contexto de la competencia a la que pertenezcan.

Las variables definidas por el usuario son siempre de lectura y escritura y una vez creadas pueden ser usadas por la misma instrucción WCL o cualquier otra a lo largo del tiempo de ejecución de la competencia. Sin embargo, los valores de las variables que comiencen con "_" no se almacenan en la base de datos, por lo que no estarán disponibles en otros eventos que no sea aquel en la que fueron creadas.

Algunas variables pueden comportarse como registros que contienen atributos ($contest$ es una de ellas). Un atributo es una variable dentro de otra que contiene un valor y tiene asociado un identificador. A su vez, un atributo puede ser un registro. Los registros también poseen métodos predefinidos en dependencia del tipo de registro. Para referirse a estos métodos en WCL se usa el signo de punto ("."), o sea, el identificador completo de un atributo son los identificadores de las variables contenedoras y del atributo separados por puntos. Puede definirse un atributo nuevo dentro de una variable cualquiera (sea del sistema o no) si se hace referencia a un identificador que no exista.

Expresiones

Las variables pueden ser usadas dentro de expresiones aritméticas. Las expresiones deben tener la misma estructura sintáctica y semántica que las expresiones de Python. Lo mismo sucede con la estructura léxica de las constantes y operadores. Si dentro de la expresión se hace referencia a un atributo que no ha sido creado aún, el sistema, por omisión, devolverá $not\_defined$, el cual es un valor especial para dichos casos; $not\_defined$ evalúa a verdadero si es usado como una condición.

Acciones

Una acción describe lo que debe hacerse cuando un evento es activado. A continuación se describen las acciones permitidas en WCL.

- **Asignación:** Esta acción es usada para asignar un valor a las variables y tiene la siguiente estructura sintáctica:

$$< \boldsymbol{a\_assignation} > ::= < \boldsymbol{variable} > \text{ “ = ” } < \boldsymbol{expression} >$$

Donde $< \boldsymbol{expresion} >$ puede ser una constante o una expresión y $< \boldsymbol{variable} >$ es el identificador de una variable. Si algún identificador no es reconocido por el sistema, se creará una nueva variable con dicho identificador y con el valor de $< \boldsymbol{expression} >$. Si el identificador es reconocido por el sistema, no puede referirse a una variable de solo lectura.

- **Condicional**

La acción de condicional es usada para elegir entre flujos alternativos de ejecución. Su funcionamiento es similar a una instrucción condicional de Python. Su estructura sintáctica es la siguiente:

$$< \boldsymbol{a\_conditional} > ::= \boldsymbol{if} < \boldsymbol{condition} > \boldsymbol{then} < \boldsymbol{actions} > < \boldsymbol{optional\_else} >$$

$$< \boldsymbol{optional\_else} > ::= \boldsymbol{else} < \boldsymbol{actions} > \mid \boldsymbol{e}$$

- **Ciclo condicionado “while do”**

Esta acción funciona igual que un ciclo $while\ do$ del lenguaje Python y su estructura sintáctica es similar.

$$< \boldsymbol{a\_whileDoLoop} > ::= \boldsymbol{while} < \boldsymbol{condition} > \boldsymbol{do} < \boldsymbol{actions} >$$

Las acciones $< actions >$ se ejecutan mientras la condición $< condition >$ sea verdadera.

- **Ciclo condicionado “repeat until”**

Esta acción funciona igual que un ciclo $repeat - until$ del lenguaje Pascal y su estructura sintáctica es similar.

$$< \boldsymbol{a\_repeatUntilLoop} > ::= \boldsymbol{repeat} < \boldsymbol{actions} > \boldsymbol{until} < \boldsymbol{condition} >$$

Las acciones $< actions >$ se ejecutarán hasta que la condición $< condition >$ devuelva un valor verdadero.

- **Generalización**

La acción de generalización es usada para aplicar un mismo conjunto de acciones a varios elementos con iguales características dentro de la competencia. Su estructura sintáctica es la siguiente:

$$< \boldsymbol{a\_generalization} > ::= \boldsymbol{foreach} < \boldsymbol{expression} > \boldsymbol{in} < \boldsymbol{expression} >$$

$$< \boldsymbol{optional\_condition} > \boldsymbol{do} < \boldsymbol{actions} >$$

$$< \boldsymbol{optional\_condition} > ::= \boldsymbol{that} < \boldsymbol{condition} > \mid \boldsymbol{e}$$

Esta acción funciona de forma similar a una instrucción for del lenguaje Python, la diferencia es que tiene incluida una condición. En palabras comunes, la acción de generalización significa “a cada x en X que cumpla la condición Y aplicar las acciones Z”.

- **Especificación**

La acción de especificación es usada para aplicar un mismo conjunto de acciones a un elemento con determinada característica dentro de la competencia. Si existen varios elementos, se aplicarán las acciones al primero que se encuentre. Su estructura sintáctica es la siguiente:

$$< \boldsymbol{a\_specification} >::= \boldsymbol{forfirst} < \boldsymbol{expression} > \boldsymbol{in} < \boldsymbol{expression} >< \boldsymbol{optional\_condition} > \boldsymbol{do} < \boldsymbol{actions} >$$

$$< \boldsymbol{optional\_condition} >::= \boldsymbol{that} < \boldsymbol{condition} > |\boldsymbol{e}$$

Funciona similar a la generalización, con la diferencia de que solo se aplicarán las acciones para el primer elemento que cumpla las condiciones.

Esta acción está diseñada para usarla cuando se sabe que existe solamente un elemento que cumple la condición y no se quiere perder tiempo buscando más elementos.

Resumen de tipos de datos manejados por el sistema

Todos los tipos de datos básicos de Python son permitidos en WCL. Para el caso de las variables de tipo *bool*, se usan las constantes $true$ y $false$, que corresponden a las constantes $True$ y $False$ de Python. Análogamente la constante $none$ corresponde a la constante $None$.

A continuación se describen los tipos de datos específicos de WCL:

$\boldsymbol{Iterator}$: Usado para iterar sobre listas de objetos. Esta clase realiza el *casting* implícito de un objeto ORM a un registro.

$\boldsymbol{Record}$: Una variable de este tipo contiene una serie de atributos a los que se puede acceder a través del operador de punto (“.”). Esta es la clase base de todos los registros.

$\boldsymbol{JoinRecord}$: Es un registro usado para representar una unión entre dos registros. Es útil cuando se requiere la definición de valores de variables para un par de entidades de la base de datos. Su constructor toma como parámetros dos registros para hacer la unión.

$\boldsymbol{ProxyRecord}$: Se trata de un registro que actúa como *proxy* a una tupla en la base de datos. Contiene un atributo entero de solo lectura llamado id que se refiere al identificador de la tupla que representa.

$\boldsymbol{ObjectInContest}$: Se trata de un registro que representa un objeto dentro de una competencia tales como un usuario o un ejercicio. Hereda de $ProxyRecord$.

$\boldsymbol{Contest}$: Representa una competencia en el sistema. Hereda de $ProxyRecord$ y posee los siguientes atributos y métodos:

Identificador en WCL	Tipo	Nivel de acceso	Descripción
$contest$	$Contest$	Solo lectura	Una referencia a sí misma.
$users$	$Iterator$	Solo lectura	Registros de usuarios de la competencia.
$exercises$	$Iterator$	Solo lectura	Registros de ejercicios de la

			competencia.
$languages$	$Iterator$	Solo lectura	Registros de lenguajes de la competencia.
$judgments$	$Iterator$	Solo lectura	Registros de calificaciones de la competencia.
$beginning$	$datetime$	Solo lectura	La fecha de inicio de la competencia.
$scheduledDuration$	$timedelta$	Solo lectura	El tiempo programado que debe durar la competencia.
$endedDate$	$datetime$	Solo lectura	La fecha en que terminó la competencia.
$rankingRefreshDelay$	int	Lectura y escritura	El tiempo en minutos en que debe refrescarse el *ranking*. Su valor inicial es 1.
$rankingSortingParameters$	$tuple$	Lectura y escritura	Usado para el ordenamiento del *ranking*. Es una tupla de cadenas, donde cada una representa un atributo de usuario. El *ranking* será ordenado atendiendo a los valores de estos atributos. Si se quiere que el ordenamiento se haga descendente para algún atributo, se debe especificar el nombre del mismo con un signo de menos ("-") delante. El valor por defecto de esta variable es $()$.
$Record$	$Record$	Solo lectura	Una referencia a la clase $Record$.
$JoinRecord$	$JoinRecord$	Solo lectura	Una referencia a la clase $JoinRecord$.
$not\_defined$	_____	Solo lectura	Una referencia al valor $not\_defined$.
datetime, math, regex, itertools, abs, all, any, bin, bool, bytearray, chr, cmp, dict, divmod, enumerate, filter, float, format, frozenset, hash, hex, objId, int, isinstance, iter, len, list, long, map, max, memoryview, min, next, oct, ord, pow, range, reduce, repr,	_____	Solo lectura	Son referencias a funciones, constantes y módulos de Python y por tanto funcionan de la misma manera que en este lenguaje.

reversed, round, set, slice, sorted, str, sum, tuple, unichr, Unicode, xrange, zip, apply, buffer, coerce, intern, false, true, none, Ellipsis			

Atributos de Contest

Identificador en WCL	Parámetros	Descripción
now	Ninguno	Devuelve la fecha y hora actual del servidor de base de datos.
$toEndContest$	Ninguno	Se usa para terminar una competencia de forma abrupta aunque todavía no sea el momento, de acuerdo al tiempo en que la misma fue programada, para concluir.

Métodos de Contest

$\boldsymbol{Exercise}$: Representa un ejercicio en el sistema. Hereda de $ProxyRecord$ y posee los siguientes atributos y definiciones:

Identificador en WCL	Tipo	Nivel de acceso	Descripción
$name$	str	Solo lectura	Nombre del ejercicio.
$users$	$Iterator$	Solo Lectura	Registros de usuarios que han realizado envíos al ejercicio.
$judgments$	$Iterator$	Solo lectura	Registros de calificaciones hechas al ejercicio.
$inputDS$	$Iterator$	Solo lectura	Registros de juegos de datos de entrada del ejercicio.

Atributos de Exercise

Identificador en WCL	Parámetros	Descripción
$getCodeFileSizeLimit$	$language$	Para obtener el tamaño límite de calificación del código. El parámetro debe ser un registro tipo $Language$.
$getUpperCodeFileSize$	$language$	Para obtener el tamaño máximo del código. El parámetro debe ser un registro tipo $Language$.
$getTimeLimit$	$language$	Para obtener el tiempo de ejecución límite de calificación. El parámetro debe ser un registro tipo $Language$.
$getUpperTime$	$language$	Para obtener el tiempo máximo de ejecución. El

		parámetro debe ser un registro tipo $Language$.
$getMemoryLimit$	$language$	Para obtener la memoria de ejecución límite de calificación. El parámetro debe ser un registro tipo $Language$.
$getUpperMemory$	$language$	Para obtener la memoria máxima de ejecución. El parámetro debe ser un registro tipo $Language$.
$getProgramSizeLimit$	$language$	Para obtener el tamaño del programa límite de calificación. El parámetro debe ser un registro tipo $Language$.
$getUpperProgramSize$	$language$	Para obtener el tamaño del programa máximo de ejecución. El parámetro debe ser un registro tipo $Language$.
$getComplexity$	$language$	Para obtener la complejidad asignada al ejercicio. El parámetro debe ser un registro tipo $Language$.

Métodos de Exercise

$\boldsymbol{User}$: Representa un usuario en la competencia. Hereda de $ProxyRecord$ y posee los siguientes atributos:

Identificador en WCL	Tipo	Nivel de acceso	Descripción
$username$	str	Solo lectura	El identificador del usuario en el sistema.
$groups$	$Iterator$	Solo lectura	Registros de grupos a los que pertenece el usuario.
$exercises$	$Iterator$	Solo lectura	Registros de ejercicios a los que el usuario ha realizado envíos.
$rankingPositionColor$	str	Lectura y escritura	Se refiere al color que debe tener la fila correspondiente a este usuario, dentro del *ranking*.

Atributos de User

Identificador en WCL	Parámetros	Descripción
$setFrozenRanking$	Ninguno	Luego de llamado este método, el *ranking* será mostrado para este usuario tal como aparecía en el momento en que se llamó al mismo.
$isFrozenRanking$	Ninguno	Retorna $true$ si se llamó anteriormente a $setFrozenRanking$. Retorna $false$ en caso contrario.
$clearFrozenRanking$	Ninguno	Realiza la acción opuesta a $setFrozenRanking$. Luego de llamado este método el usuario podrá

		observar el *ranking* actualizado.

Métodos de User

$\boldsymbol{Group}$: Representa un grupo en el sistema. Hereda de $ProxyRecord$ y posee los siguientes atributos:

Identificador en WCL	Tipo	Nivel de acceso	Descripción
$name$	str	Solo lectura	El nombre del grupo.
$users$	$Iterator$	Solo lectura	Registros de usuarios que pertenecen al grupo.

Atributos de Group

$\boldsymbol{Language}$: Representa un lenguaje en el sistema. Hereda de $ProxyRecord$ y posee los siguientes atributos:

Identificador en WCL	Tipo	Nivel de acceso	Descripción
$name$	str	Solo lectura	El nombre del lenguaje.

Atributos de Language

$\boldsymbol{InputDS}$: Representa un juego de datos de entrada. Hereda de $ProxyRecord$ y posee los siguientes atributos:

Identificador en WCL	Tipo	Nivel de acceso	Descripción
$rate$	int	Solo lectura	La importancia del juego de datos.
$outputDSs$	$Iterator$	Solo lectura	Los juegos de datos de salida asociados a este juego de datos de entrada.

Atributos de InputDS

$\boldsymbol{OutputDS}$: Representa un juego de datos de salida. Hereda de $ProxyRecord$ y posee los siguientes atributos:

Identificador en WCL	Tipo	Nivel de acceso	Descripción
$rate$	int	Solo lectura	La importancia del juego de datos.

Atributos de OutputDS

$\boldsymbol{JudgmentBase}$: Es una clase abstracta y representa un resultado de calificación cualquiera. Hereda de $ProxyRecord$ y posee los siguientes atributos:

Identificador en WCL	Tipo	Nivel de acceso	Descripción
$statuses$	str	Solo lectura	Los resultados de la calificación.
$message$	str	Solo lectura	Una cadena para brindar detalles del resultado de la calificación.
$outputSize$	$float$	Solo lectura	El tamaño de la salida dada.
$programSize$	$float$	Solo lectura	El tamaño del programa.
$time$	$float$	Solo lectura	El tiempo de ejecución.
$memory$	$float$	Solo lectura	La memoria de ejecución.
$wasAccepted$	$bool$	Solo lectura	$true$ si la respuesta fue aceptada.

Atributos de JudgmentBase

Identificador en WCL	Parámetros	Descripción
$emitNotJudged$	$message$	Para emitir un resultado de no juzgado. $message$ es un mensaje opcional para brindar más detalles del resultado.

Métodos de JudgmentBase

$\boldsymbol{Judgment}$: Representa un resultado de calificación de un ejercicio. Hereda de $JudgmentBase$ y posee los siguientes atributos:

Identificador en WCL	Tipo	Nivel de acceso	Descripción
$user$	$User$	Solo lectura	El usuario que envió la solución.
$exercise$	$Exercise$	Solo lectura	El ejercicio de la solución.
$language$	$Language$	Solo lectura	El lenguaje en que se implementó la solución.
$dsJudgments$	$Iterator$	Solo lectura	Los resultados de calificación de cada juego de datos.
$codeFileSize$	$float$	Solo lectura	El tamaño del archivo de código fuente.
$submitionDate$	$datetime$	Solo lectura	La fecha en que se realizó el envío.
$wasCodeSizeExceeded$	$bool$	Solo lectura	$true$ si la respuesta fue “tamaño de código excedido”.
$wasCodeConstraintsError$	$bool$	Solo lectura	$true$ si la respuesta fue “error de restricción de código”.
$wasCompilationError$	$bool$	Solo lectura	$true$ si la respuesta fue “error de compilación”.

Atributos de Judgment

$\boldsymbol{DSJudgment}$: Representa un resultado de calificación de un juego de datos. Hereda de $JudgmentBase$ y posee los siguientes atributos:

Identificador en WCL	Tipo	Nivel de acceso	Descripción
$judgment$	$Judgment$	Solo lectura	El resultado de calificación del ejercicio al que pertenece este resultado de calificación de juego de datos.
$inputDSs$	$InputDS$	Solo lectura	El juego de datos de entrada asociado.
$checkerJudgment$	$float$	Solo lectura	Un número que emite el chequeador y que representa qué tan incorrecta es la respuesta. Un número mayor representa mayor error. Si la respuesta es aceptada y se usó el chequeador por defecto, la variable contiene el identificador del juego de datos de salida patrón que coincidió con la salida dada por el usuario.
$matchedOutput$	$OutputDS$	Solo lectura	Es la salida patrón que coincidió con la respuesta dada.
$wasExecutionError$	$bool$	Solo lectura	$true$ si el resultado fue “error de ejecución”.
$wasTimeLimitExceeded$	$bool$	Solo lectura	$true$ si la respuesta fue “tiempo excedido”.
$wasMemoryLimitExceeded$	$bool$	Solo lectura	$true$ si la respuesta fue “memoria excedida”.
$wasProgramSizeExceeded$	$bool$	Solo lectura	$true$ si la respuesta fue “tamaño de programa excedido”.
$wasOutputSizeExceeded$	$bool$	Solo lectura	$true$ si la respuesta fue “tamaño de salida excedida”.
$wasWrongAnswer$	$bool$	Solo lectura	$true$ si la respuesta fue “respuesta incorrecta”.

Atributos de DSJudgment

Variables de nombre especial

Para controlar la forma en que se mostrarán los datos en la interfaz gráfica se han definido variables con nombres especiales. A continuación se explica cada una de ellas.

Variable $show$

Si su valor se define para un usuario en específico (a través del uso de un $JoinRecord$) que esté registrado en la competencia, entonces se usará este valor para controlar cuándo el objeto debe ser mostrado o no en la interfaz gráfica para ese usuario.

Por ejemplo, si se define esta variable con valor $false$ dentro de una instancia de $JoinRecord$ para el ejercicio 1 y el usuario 2, entonces ese ejercicio no podrá ser visto dentro de la competencia por el usuario 2.

Si su valor se define para un registro tipo $ProxyRecord$, entonces se usará su valor para determinar si se puede mostrar o no este objeto a los usuarios que no están registrados en la competencia.

Variable $showRanking$

Si su valor se define para la competencia en general, se tomará para determinar si se muestra o no el *ranking* para los usuarios no registrados en la competencia.

Si se define para un usuario registrado en la competencia, su valor se tomará para determinar si se muestra o no el *ranking* para ese usuario en específico.

Variable para mostrar el valor de un campo

Esta variable tiene la forma general $show\_[attribute\_name]\_value$, donde $[attribute\_name]$ se reemplaza por el nombre del campo[64]. Esta variable se usa para determinar si se muestra o no el valor del campo especificado al momento de listar los objetos.

Si se define dentro de una instancia de $ProxyRecord$, el valor es tomado para los usuarios que no están registrados en la competencia. Por el contrario, si se define para un usuario registrado en la competencia, usando un $JoinRecord$, su valor será tomado para determinar si se muestra o no el valor del objeto para ese usuario.

Variable para mostrar los detalles de un campo

Esta variable tiene la forma general $show\_[attribute\_name]\_details$, donde $[attribute\_name]$ se reemplaza por el nombre del campo.

Su uso es el mismo que el de la variable anteriormente descrita, sin embargo su valor se toma para determinar si se muestran o no los detalles del campo, en caso de que el mismo tenga.

[64] El nombre del campo se refiere al nombre de la columna (en minúsculas y reemplazando espacios por *underscore*) de la tabla donde se listan los objetos dentro de la interfaz gráfica.

Ejemplos de configuración de distintos tipos de competencias usando WCL

Para especificar el conjunto de acciones que deben ejecutarse antes o después de activado algún evento se ha utilizado la siguiente sintaxis:

$$\boldsymbol{when} < \boldsymbol{eventName} > \ \boldsymbol{do} < \boldsymbol{precode} >< \boldsymbol{postcode} >$$

$$< \boldsymbol{precode} >::= \boldsymbol{precode} : < \boldsymbol{actions} > |\boldsymbol{e}$$

$$< \boldsymbol{postcode} >::= \boldsymbol{postcode}: \ < \boldsymbol{actions} > |\boldsymbol{e}$$

Donde <$eventName$> es el nombre del evento. Las acciones del $precode$ son las que deben ejecutarse antes de las acciones por defecto del evento y las acciones del $postcode$ son las que deben ejecutarse después.

Los nombres de los eventos pueden ser: $\boldsymbol{beginOfContest}$ para el evento de inicio de competencia, $\boldsymbol{endOfContest}$ para el evento de fin de competencia, $\boldsymbol{judgmentOfSolution}$ para el evento de calificación de solución, $\boldsymbol{judgmentOfDS}$ para el evento de calificación de juego de datos, $\boldsymbol{pastXTime}(\boldsymbol{time}, \boldsymbol{amount})$ para el evento de ronda de tiempo y $\boldsymbol{pastXTimeAfterEnd}(\boldsymbol{time})$ para el evento de ronda de tiempo luego de finalizada la competencia, donde $time$ es el tiempo de activación en segundos y $\boldsymbol{amount}$ es la cantidad de veces que puede activarse el evento.

Los comentarios en WCL se realizan de igual manera que en el lenguaje Python, o sea, comienzan con "#".

Ejemplo 1: Una competencia similar al tipo ACM-ICPC

```
when beginOfContest do
  precode:
    # ^^ Poniendo las condiciones iniciales de la competencia
        rankingSortingParameters = ('-score', 'failedSubmissions', 'acceptedSubmitDate')
    # ^^ El ordenamiento se hará por puntos, luego por el que menos respuestas no aceptadas tenga
    #   y luego por la fecha del último envío correcto.
    foreach _exercise in exercises do
      _exercise.score = 1000
      _exercise.acceptedSubmissions = 0
      foreach _user in users do
        _user.failedSubmissions = 0
        _x = JoinRecord(_exercise, _user)
        _x.partialFailedSubmissions = 0
        _x. acceptedBefore = false
        _user.score = 0
        _user.acceptedSubmitDate = 'No submissions yet'
    showJudgments = true

when judgmentOfSolution do #Cuando un ejercicio es calificado.
  precode:
    foreach _user in users do
      _x = JoinRecord(_judgment, _user)
      if not showJudgments then
```

```
            _x.show = false
            _judgment.show = false
        _x.show_date_details = false
        _judgment.show_date_details = false

  postcode:
     _user_exercise = JoinRecord(_judgment.user, _judgment.exercise)
   if not _user_exercise.acceptedBefore then
     # Si el usuario no había aceptado el ejercicio antes.
       if _judgment.wasAccepted then # Si el ejercicio fue aceptado.
            _judgment.exercise.acceptedSubmissions = _judgment.exercise.acceptedSubmissions + 1
            _newScore = 1000/_judgment.exercise.acceptedSubmissions
            # ^^ Disminuir la importancia del ejercicio en dependencia de la cantidad de veces que
            #    ha sido aceptado
            _delta = _judgment.exercise.score - _newScore
            _judgment.exercise.score = _newScore
            _judgment.user.score = _judgment.user.score + _newScore
            foreach _user in users that JoinRecord(_user, _judgment.exercise).acceptedBefore do
                _user.score = _user.score - _delta
                 # ^^ Para cada usuario que haya hecho envíos correctos al ejercicio, actualizar sus puntos.
            _judgment.user.failedSubmissions = _judgment.user.failedSubmissions + _user_exercise.partialFailedSubmissions
            # ^^ Actualizar la cantidad de envíos incorrectos a este ejercicio.
            _judgment.user.acceptedSubmitDate = _judgment.submitionDate
            # ^^ Actualizar la fecha del último envío aceptado para este usuario.
            _user_exercise.acceptedBefore = true
   else
       _user_exercise.partialFailedSubmissions = _user_exercise.partialFailedSubmissions + 1
        # ^^ Actualizar la cantidad de envíos incorrectos, para una posible actualización
        #    de user.failedSubmissions, ya que los envíos incorrectos a un ejercicio solo se
        #    cuentan si el ejercicio fue finalmente aceptado.

when pastXTime( scheduledDuration - timedelta(hours=1), 1 ) do
  postcode:
     foreach _user in users do
        _x=_user.setFrozenRanking()
  # ^^ Cuando quede una hora, se congela el ranking y empieza el tiempo muerto.

when pastXTime( scheduledDuration - timedelta(minutes=30), 1 ) do
  postcode:
     showJudgments = false
     foreach _user in users do
        foreach _judgment in judgments do
           _x = JoinRecord(_judgment, _user)
           _x.show = false
           _judgment.show = false
  # ^^ Cuando quede media hora, no se brindarán resultados de calificación a los usuarios

when endOfContest do
  postcode:
       foreach _user in users do
          _x=_user.clearFrozenRanking()
```

```
        foreach _judgment in judgments do
          _x = JoinRecord(_judgment, _user)
          _x.show = true
          _judgment.show = true
  # ^^ Cuando la competencia termine, actualizar el ranking y mostrar los resultados de los
  #    envíos que no se podían ver.
```

Ejemplo 2: Una competencia similar a las usadas en los entrenamientos de la preselección nacional del nivel de enseñanza preuniversitario

```
when beginOfContest  do
   precode:
      # ^^ Poniendo las condiciones iniciales de la competencia por si acaso no fueron
      #    puestas en su creación.
      #    El ranking y los resultados de los envíos no son mostrados inicialmente.
      showRanking = false
      foreach _user in users do
         _user.showRanking= false
         _user.score = 0
      rankingSortingParameters = ('-score',)
      # ^^ El ordenamiento se hará por los puntos.

when judgmentOfSolution do
   precode:
      foreach _user in users do
         _x = JoinRecord(_judgment, _user)
         _x.show = false
         _judgment.show = false
   postcode:
      JoinRecord(_judgment.user, _judgment.exercise).acceptedBefore = true
      # ^^ Una vez que un usuario ha hecho un envío, no se permitirá
      #    que vuelva a hacer más envíos al mismo ejercicio.

when judgmentOfDS do #Cuando un juego de datos es chequeado
   postcode:
      if not JoinRecord(_dsJudgment.judgment.user, _dsJudgment.judgment.exercise).acceptedBefore and  _dsJudgment.wasAccepted then
         _dsJudgment.judgment.user.score = _dsJudgment.judgment.user.score + _dsJudgment.rate

when endOfContest do
     postcode:
# ^^ Cuando la competencia acaba es que se permite ver el ranking
#    y el resultado de todos los envíos.
         showRanking = true
         foreach _user in users do
            _user.showRanking = true
            foreach _judgment in judgments do
               _x = JoinRecord(_judgment, _user)
               _x.show = true
               _judgment.show = true
```

Ejemplo 3: Una competencia de desafío

En esta competencia gana quien brinde las soluciones que consuman menos recursos del sistema (tiempo, memoria de ejecución y tamaño de programa). Se ordenan los usuarios de acuerdo a la mayor cantidad de ejercicios que resuelvan, que consuman menos recursos que las demás soluciones a dicho ejercicio. Para esto se tienen en cuenta los recursos en el siguiente orden:

1. Tiempo de ejecución. Dos cantidades que se diferencien en 500 ms serán consideradas como si fueran iguales.
2. Memoria de ejecución. Dos cantidades que se diferencien en 512 bytes serán consideradas como si fueran iguales.
3. Tamaño del programa. Dos cantidades que se diferencien en 512 bytes serán consideradas como si fueran iguales.

```
when beginOfContest do
  precode:
    # ^^ Poniendo las condiciones iniciales de la competencia por si acaso no fueron
    #    puestas en su creación.
    foreach _exercise in exercises do
      _exercise.minTime = 60000 # ms. 1 minuto
      _exercise.minMemory = 51200 # kb. 50 mb
      _exercise.minProgram = 100000 # kb. 100 mb

    foreach _user in users do
      _user.minTimeCount = 0
      _user.minMemoryCount = 0
      _user.minProgramCount = 0
    rankingSortingParameters = ('-minTimeCount', '-minMemoryCount', '-minProgramCount')

when judgmentOfSolution do # Cuando un ejercicio es calificado
  postcode:
    if _judgment.wasAccepted then
      if _judgment.exercise.minTime > _judgment.time + 500 then # Si esta solución fue mejor en tiempo
        _judgment.exercise.minTime = _judgment.time # Actualizar el tiempo mínimo para el ejercicio
        # Actualizar los valores de los usuarios implicados
        if _judgment.exercise.minTimeUser != not_defined then
          foreach _user in _judgment.exercise.minTimeUser do
            _user.minTimeCount = _user.minTimeCount - 1
        _judgment.exercise.minTimeUser.append(_judgment.user)
        _judgment.user.minTimeCount = _judgment.user.minTimeCount + 1
      if _judgment.exercise.minMemory > _judgment.memory + 0.5 then # Lo mismo para los demás recursos
        _judgment.exercise.minMemory = _judgment.memory
        if _judgment.exercise.minMemoryUser != not_defined then
          foreach _user in _judgment.exercise.minMemoryUser do
            _user.minMemoryCount = _user.minMemoryCount - 1
        _judgment.exercise.minMemoryUser.append(_judgment.user)
        _judgment.user.minMemoryCount = _judgment.user.minMemoryCount + 1
```

```
        if _judgment.exercise.minProgram > _judgment.programSize + 0.5 then
          _judgment.exercise.minProgram = _judgment.programSize
          if _judgment.exercise.minProgramUser != not_defined then
            foreach _user in _judgment.exercise.minProgramUser do
                _user.minProgramCount = _user.minProgramCount - 1
          _judgment.exercise.minProgramUser.append(_judgment.user)
          _judgment.user.minProgramCount = _judgment.user.minProgramCount + 1
```

Ejemplo 4: Una competencia de optimización

En esta competencia gana quien brinde la mejor solución a problemas de optimización con cantidad finita de salidas válidas a un mismo juego de datos de entrada. Cada salida de un mismo juego de datos tiene una importancia asociada. Como existe una cantidad finita de salidas válidas, se asigna un valor de importancia más grande a determinada salida si la misma está más próxima al valor óptimo.

```
when beginOfContest do
  precode:
    # ^^ Poniendo las condiciones iniciales de la competencia por si acaso no fueron
    #    puestas en su creación.
    foreach _exercise in exercises do
      _exercise.maxScore = 0
      foreach _user in users do
        _x = JoinRecord(_exercise, _user)
        _x.score = 0
        _user.optimusSolutionsCount = 0;
    rankingSortingParameters = ('-optimusSolutionsCount',)

when judgmentOfDS do # Cuando un juego de datos es chequeado
  postcode:
    if _dsJudgment.wasAccepted then
      _x = JoinRecord(_dsJudgment.judgment.user, _dsJudgment.judgment.exercise)
      _x.score = _x.score + _dsJudgment.matchedOutput.rate

when judgmentOfSolution do # Cuando un ejercicio es calificado.
  postcode:
    if _judgment.exercise.wasAccepted then
      _x = JoinRecord(_judgment.user, _judgment.exercise)
      if _x.score > _judgment.exercise.maxScore then
        if _x.userMaxScore != not_defined then
          foreach _user in _x.userMaxScore do
            _user.optimusSolutionsCount = _user.optimusSolutionsCount - 1
            _x.userMaxScore.remove(_user)
        _x.userMaxScore.append(_judgment.user)
        _judgment.user.optimusSolutionsCount = _judgment.user.optimusSolutionsCount + 1
        _judgment.exercise.maxScore = _x.score
```

Ejemplo 5: Una competencia inusual

Competencia en la que los problemas son publicados cada 30 min, habiendo solamente un problema disponible en cada momento. Cada vez que un concursante resuelve un ejercicio se muestra otro y se

oculta el anterior que estaba publicado. La competencia se acabará cuando llegue el tiempo de culminación de la misma, o se acaben los ejercicios, o cuando nadie logre aceptar el ejercicio actual.

```
when beginOfContest do
   precode:
      # ^^ Poniendo las condiciones iniciales de la competencia por si acaso no fueron
      #    puestas en su creación.
      foreach _user in users do
         _user.score = 0
      foreach _exercise in exercises do
         _exercise.show = false
         foreach _user in users do
            _x = JoinRecord(_exercise, _user)
            _x.show = false
            _x.acceptedBefore = false
      forfirst _exercise in exercises do
         _exercise.show = true
         foreach _user in users do
            _x = JoinRecord(_exercise, _user)
            _x.show = true
      enabledExercisePos = 0
      rankingSortingParameters = ("-score",)

when pastXTime( timedelta(minutes=30), 0 ) do
   postcode:
   # ^^ Mientras dure la competencia y cada 30 minutes.
      forfirst _judgment in judgments that _judgment.wasAccepted and _judgment.exercise == exercises[enabledExercisePos] do
         _judgment = _judgment
      if _judgment == not_defined or not _judgment.wasAccepted or _judgment.exercise != exercises[enabledExercisePos] then
         toEndContest()
      else
         exercises[enabledExercisePos].show = false
         foreach _user in users do
            _x = JoinRecord(_exercise, _user)
            _x.show = false

         enabledExercisePos = enabledExercisePos + 1
         if enabledExercisePos >= len(exercises) then
            toEndContest()
         else
            exercises[enabledExercisePos].show = true
            foreach _user in users do
               _x = JoinRecord(_exercise, _user)
               _x.show = true

when judgmentOfSolution do # Cuando un ejercicio es calificado.
   postcode:
      if _judgment.wasAccepted then
         _x = JoinRecord(_judgment.user, _judgment.exercise)
         if not _x.acceptedBefore then
            _judgment.user.score = _judgment.user.score + 1
            _x.acceptedBefore = true
            exercises[enabledExercisePos].show = false
            foreach _user in users do
               _x = JoinRecord(_exercise, _user)
               _x.show = false

            enabledExercisePos = enabledExercisePos + 1
```

```
        if enabledExercisePos >= len(exercises) then
          toEndContest()
        else
          exercises[enabledExercisePos].show = true
          foreach _user in users do
            _x = JoinRecord(_exercise, _user)
            _x.show = true
```

Ejemplo 6: Una competencia de entrenamiento en la que, en los resultados de cada envío se muestran todos los detalles posibles

Para esta competencia no es necesario configurar ningún evento pues el comportamiento por defecto del sistema es el mismo que en este tipo de competencia.

Anexo 13: Casos de prueba

	Caso de prueba de aceptación
Código de caso de prueba: CP1_ HU3	**Nombre de la historia de usuario:** Gestionar competencia
Responsable de la prueba: Ernesto Soto Gómez	
Descripción de la prueba: Prueba de funcionalidad para adicionar una competencia.	
Condiciones de ejecución: El usuario que va a añadir la competencia debe estar autenticado en el sistema y tener permisos para adicionar competencias.	
Entrada/Pasos de ejecución: • El usuario selecciona “Adicionar competencia”. • El sistema muestra un formulario que espera los siguientes datos de entrada. ➢ Nombre (campo obligatorio que espera una cadena de texto de una longitud de 50 como máximo que admite cualquier caracter imprimible). ➢ Descripción (cadena de texto que admite cualquier caracter imprimible). ➢ Fecha de comienzo (campo obligatorio que espera una cadena de texto con el formato: año-mes-día hora:minuto:segundo). ➢ Duración (entero positivo). ➢ Máxima cantidad de concursantes que pueden registrarse (campo obligatorio que espera un entero positivo). ➢ Valores de los parámetros iniciales del tipo de competencia (cadena de texto que admite cualquier caracter imprimible). ➢ Habilitado o no (campo obligatorio de selección). ➢ Usuarios registrados (campo obligatorio de selección). ➢ Clúster (campo obligatorio de selección). ➢ Tipo de competencia (campo obligatorio de selección). ➢ Ejercicios (campo obligatorio de selección).	

➢ Lenguajes permitidos (campo obligatorio de selección). • El usuario escribe los datos y los envía. • El sistema valida los datos. • Si los datos son válidos el sistema muestra un mensaje de que se ha añadido satisfactoriamente la competencia, en caso contrario el sistema muestra un mensaje de que los datos son incorrectos.
Resultado esperado: Los datos de la competencia quedan almacenados en la base de datos.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 34. CP1_HU3. Adicionar competencia

	Caso de prueba de aceptación
Código de caso de prueba: CP2_ HU3	**Nombre de la historia de usuario:** Gestionar competencia
Responsable de la prueba: Beatriz Lazo Tamayo	
Descripción de la prueba: Prueba de funcionalidad para listar competencias.	
Condiciones de ejecución: El usuario que va a listar las competencias debe estar autenticado en el sistema y tener permisos para modificar competencias.	
Entrada/Pasos de ejecución: • El usuario selecciona “Listar competencia”. • El sistema muestra una lista de todas las competencias que existen.	
Resultado esperado: Mostrar satisfactoriamente el listado de las competencias.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 35. CP2_HU3. Listar competencias

	Caso de prueba de aceptación
Código de caso de prueba: CP3_ HU3	**Nombre de la historia de usuario:** Gestionar competencia
Responsable de la prueba: Ernesto Soto Gómez	
Descripción de la prueba: Prueba de funcionalidad para eliminar una competencia.	
Condiciones de ejecución: El usuario que va a eliminar la competencia debe estar autenticado en el sistema y tener permisos para eliminar competencias.	
Entrada/Pasos de ejecución: • El usuario selecciona la competencia a eliminar. • El usuario elimina la competencia. • Si la competencia comenzó el sistema muestra un mensaje de que no puede eliminarse la competencia, en caso contrario se muestra un mensaje de que la competencia ha quedado eliminada.	
Resultado esperado: Los datos de la competencia quedan eliminados de la base de datos del	

sistema.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 36. CP3_HU3. Eliminar competencia

	Caso de prueba de aceptación
Código de caso de prueba: CP4_ HU3	**Nombre de la historia de usuario:** Gestionar competencia
Responsable de la prueba: Beatriz Lazo Tamayo	
Descripción de la prueba: Prueba de funcionalidad para modificar una competencia.	
Condiciones de ejecución: El usuario que va a modificar la competencia debe estar autenticado en el sistema y tener permisos para modificar competencias.	
Entrada/Pasos de ejecución: • El usuario selecciona la competencia a modificar. • El sistema muestra los siguientes datos modificables de la competencia. ➢ Nombre (cadena de texto no vacía de una longitud de 50 como máximo que admite cualquier caracter imprimible). ➢ Descripción (cadena de texto que admite cualquier caracter imprimible). ➢ Fecha de comienzo (cadena de texto no vacía con el formato: año-mes-día hora:minuto:segundo). ➢ Duración (entero positivo). ➢ Máxima cantidad de concursantes que pueden registrarse (campo no vacío que espera un entero positivo). ➢ Valores de los parámetros iniciales del tipo de competencia (cadena de texto que admite cualquier caracter imprimible). ➢ Habilitado o no (campo no vacío de selección). ➢ Usuarios registrados (campo no vacío de selección). ➢ Clúster (campo no vacío de selección). ➢ Tipo de competencia (campo no vacío de selección). ➢ Ejercicios (campo no vacío de selección). ➢ Lenguajes permitidos (campo no vacío de selección). • El usuario inserta los nuevos datos y los envía al sistema. • El sistema valida los datos. • Si la competencia no ha comenzado y los datos son válidos, el sistema muestra un mensaje de que se ha modificado la competencia, en caso contrario, si la competencia comenzó el sistema muestra un mensaje de que la competencia no se puede modificar, si no, si los datos son incorrectos el sistema muestra un mensaje de que los datos son incorrectos.	
Resultado esperado: Los nuevos datos de la competencia quedan registrados en la base de datos del sistema.	

Evaluación de la prueba: Prueba satisfactoria.

Tabla 37. CP4_HU3. Modificar competencia

	Caso de prueba de aceptación
Código de caso de prueba: CP5_ HU1	**Nombre de la historia de usuario:** Autenticar usuario
Responsable de la prueba: Ernesto Soto Gómez	
Descripción de la prueba: Prueba de funcionalidad para autenticar un usuario.	
Condiciones de ejecución: El usuario debe estar registrado en el sistema.	
Entrada/Pasos de ejecución: • El sistema muestra un formulario que espera los siguientes datos. ➢ Identificador de usuario (cadena de texto no vacía de 30 caracteres como máximo que admite solo letras, números y los caracteres especiales "@ . + - _"). ➢ Contraseña (cadena de texto no vacía de 128 caracteres como máximo). • Si el identificador de usuario existe y la contraseña es correcta, el sistema muestra una interfaz de bienvenida, si no, se muestra un mensaje de error de identificador de usuario o contraseña incorrecta.	
Resultado esperado: El usuario queda autenticado en el sistema.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 38. CP5_HU1. Autenticar usuario

	Caso de prueba de aceptación
Código de caso de prueba: CP6_ HU2	**Nombre de la historia de usuario:** Calificar solución
Responsable de la prueba: Beatriz Lazo Tamayo	
Descripción de la prueba: Prueba de funcionalidad para calificar una solución.	
Condiciones de ejecución: El usuario que va a realizar el envío de la solución debe estar autenticado en el sistema.	
Entrada/Pasos de ejecución: • El usuario selecciona un ejercicio y selecciona "Enviar solución". • El sistema muestra una interfaz para que el usuario inserte el código de la solución al ejercicio (cadena de texto). • El usuario inserta el código, selecciona el lenguaje en que fue resuelto el ejercicio y envía los datos. • El sistema redirecciona hacia una página de resultados de calificación.	
Resultado esperado: El sistema muestra correctamente los resultados de calificación.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 39. CP6_HU2. Calificar solución

	Caso de prueba de aceptación
Código de caso de prueba: CP7_ HU4	**Nombre de la historia de usuario:** Consultar *ranking*
Responsable de la prueba: Ernesto Soto Gómez	
Descripción de la prueba: Prueba de funcionalidad para consultar *ranking*.	
Condiciones de ejecución:	
Entrada/Pasos de ejecución: • El usuario selecciona "Mostrar *ranking*" dentro de una competencia. • El sistema muestra el *ranking* si está permitido para ese tipo de competencia, en caso contrario el sistema muestra un mensaje de *ranking* vacío.	
Resultado esperado: El sistema muestra correctamente los datos del *ranking*.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 40. CP7_HU4. Consultar ranking

	Caso de prueba de aceptación
Código de caso de prueba: CP8_ HU5	**Nombre de la historia de usuario:** Consultar envíos realizados
Responsable de la prueba: Beatriz Lazo Tamayo	
Descripción de la prueba: Prueba de funcionalidad para consultar envíos realizados.	
Condiciones de ejecución:	
Entrada/Pasos de ejecución: • El usuario selecciona "Mostrar envíos" realizados dentro de una competencia. • El sistema muestra los envíos realizados si está permitido para ese tipo de competencia, en caso contrario el sistema muestra un mensaje de que no existen envíos.	
Resultado esperado: El sistema muestra correctamente los datos de los envíos realizados.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 41. CP8_HU5. Consultar envíos realizados

	Caso de prueba de aceptación
Código de caso de prueba: CP9_ HU6	**Nombre de la historia de usuario:** Registrar usuario
Responsable de la prueba: Ernesto Soto Gómez	
Descripción de la prueba: Prueba de funcionalidad para registrar un usuario.	
Condiciones de ejecución:	
Entrada/Pasos de ejecución: • El usuario selecciona "Registrar usuario". • El sistema muestra un formulario que espera los siguientes datos. ➢ Identificador de usuario (cadena de texto no vacía de 30 caracteres como máximo que admite solo letras, números y los caracteres especiales "@ . + - _"). ➢ Contraseña en el sistema (cadena de texto no vacía de 128 caracteres como máximo).	

• El usuario escribe los datos y los envía. • El sistema valida los datos. • Si los datos son válidos el sistema muestra un mensaje de que se ha registrado satisfactoriamente el usuario, si no, el sistema muestra un mensaje de que los datos son incorrectos.
Resultado esperado: Los datos del usuario quedan almacenados en la base de datos.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 42. CP9_HU6. Registrar usuario

	Caso de prueba de aceptación
Código de caso de prueba: CP10_ HU7	**Nombre de la historia de usuario:** Modificar perfil
Responsable de la prueba: Beatriz Lazo Tamayo	
Descripción de la prueba: Prueba de funcionalidad para modificar el perfil de un usuario.	
Condiciones de ejecución: El usuario debe estar autenticado en el sistema.	
Entrada/Pasos de ejecución: • El usuario selecciona “Modificar perfil”. • El sistema muestra un formulario que espera los siguientes datos. ➢ Identificador de usuario (cadena de texto no vacía de 30 caracteres como máximo que admite solo letras, números y los caracteres especiales “@ . + - _”). ➢ Nombre (cadena de texto de una longitud de 30 como máximo que admite cualquier caracter imprimible). ➢ Apellidos (cadena de texto de una longitud de 30 como máximo que admite cualquier caracter imprimible). ➢ Dirección de correo electrónico (cadena de texto de una longitud de 75 como máximo y que represente una dirección de correo válida). • El usuario modifica los datos y los envía. • El sistema valida los datos. • Si los datos son válidos el sistema muestra un mensaje de que se ha modificado satisfactoriamente el perfil del usuario, en caso contrario el sistema muestra un mensaje de que los datos son incorrectos.	
Resultado esperado: Los datos del perfil del usuario quedan modificados en la base de datos.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 43. CP10_HU7. Modificar perfil