Universidad de las Ciencias Informáticas

Trabajo de diploma

"Análisis, diseño e implementación de un Jurado Online"

Autor (es): Jorge Amado Soria Ramirez
Enrique José Altuna Castillo

Tutor: Lic. Tomás Orlando Junco Vázquez

Ciudad de la Habana, diciembre de 2007

DECLARACIÓN DE AUTORÍA

Declaramos que somos los únicos autores de este trabajo y autorizamos a la Universidad de las Ciencias Informáticas (UCI) a que haga el uso que estimen pertinente de este trabajo.

Para que así conste firmamos la presente a los __ días del mes de Junio del 2008.

Jorge Amado Soria Ramírez

_____

Enrique José Altuna Castillo

_____

A nuestros padres

Agradecimientos

A mis familiares y amigos, por confiar en mí y apoyarme en todo momento.

A todo el que de una forma u otra ha hecho posible la realización de este trabajo.

Enrique José Altuna Castillo

A Reiniel Portelles, Sergio Bermúdez, Leandro Rojas y demás excelentes profesionales y personas del antiguo CEA de la UO, por enseñarme que la programación es ciencia, arte y vida.

A Yandry Pérez Clemente y Dovier Antonio Ripoll, por promover exitosamente nuestro arte en el mundo.

Jorge Amado Soria Ramírez

A José Ernesto Lara Rodríguez, por ayudar incluso sin que se lo pidiesen.

A los programadores de la UCI, por validar nuestro esfuerzo y recordarnos con cada AC que no trabajamos en vano.

Y muy especialmente, a nuestro Comandante Fidel y a la Revolución Cubana por darnos la oportunidad de convertirnos en profesionales.

Resumen

Este trabajo contiene la investigación y desarrollo de una aplicación web de evaluación automática de soluciones de programación, o Jurado Online. Incluye un estudio del estado del arte, las herramientas utilizadas, las funciones implementadas y la estrategia de pruebas llevada a cabo. También se hace mención a distintos anexos y aplicaciones de interés para una mejor explotación del Jurado Online, y un conjunto de conclusiones y recomendaciones que consideramos son de interés para el tema tratado.

Abstract

This paper contains investigation and development information about the implementation of a web application dedicated to the automated evaluation of programming solutions, also named Online Judge. It includes research on state-of-the-art technology available in the world nowadays in our field, tools used, features implemented and the test strategy set in place. Also mentioned are a number of applications and attached documents that help improve the user experience of the Online Judge, and a group of conclusions and recommendations we consider to be of interest about the subject.

Tabla de contenidos

Índice de Imagen

Índice de Tablas

Introducción

Existe la creencia generalizada de que todos los informáticos son programadores, y de que la informática como ciencia depende decisivamente de la programación como disciplina. Como sabemos esto no es cierto, la informática engloba áreas muy interesantes y útiles de trabajo que no dependen de la programación ni siquiera levemente, pero aún así la disciplina y el arte de la programación es una parte fundamental de cualquier proceso de desarrollo informático y un conocimiento que puede probarse útil en cualquier esfera científica relacionada con el procesamiento automatizado de datos.

Hay muchas formas de aprender programación. Los métodos se han perfeccionado tanto como los lenguajes, pero la más segura es como siempre la más sencilla: Se aprende a programar programando. Y es aquí donde puede ser útil un evaluador automático de soluciones.

Hay varios componentes involucrados en la evaluación de soluciones a problemas de programación y solo algunos pueden ser automatizados. En particular los siguientes pueden ser mejorados tremendamente con el uso de un evaluador automático:

- **Correctitud:** Dada una entrada válida la solución produce una salida válida.
- **Robustez:** Dada una entrada ilegal, el programa debe producir un mensaje de error adecuado.
- **Eficiencia:** El programa debe realizar sus tareas sin consumir demasiado tiempo o memoria. Cuanto es demasiado depende de la tarea específica.

Un evaluador automático es una aplicación que alivia el trabajo de calificación de soluciones, moviendo su responsabilidad del humano a la máquina, tratando de lograr mejoras en los tres criterios mencionados anteriormente.

Un concepto más elaborado es el de Jurado Online, que no es exactamente equivalente al de un evaluador automático. La principal diferencia está dada porque este último toma en cuenta otros dos criterios a la hora de automatizar la tarea de evaluación automática:

- **Interface humana:** Un Jurado Online presenta una interface humana más elaborada, principalmente debido al hecho de ser aplicaciones Web y contar con todas las posibilidades de configuración visual correspondientes a este tipo de aplicación.

- **Disponibilidad:** Un Jurado Online es una aplicación que está siempre disponible a cualquier persona, incluso a no interesados, a diferencia de un evaluador automático que solo suele ser accesible a través de protocolos específicos, lugares específicos o ventanas de tiempo determinadas. Esto también es debido a la naturaleza Web de los Jurados Online y al uso general que se le da a los evaluadores como herramientas de apoyo para eventos específicos.

Más específicamente, un Jurado Online es una aplicación de entornos generalmente académicos presentada en la Web (o en cualquier otro soporte igualmente masivo) que provee problemas y sirve para evaluar automáticamente las soluciones algorítmicas que a estos encuentran sus usuarios desarrolladas en uno de entre un número determinado de lenguajes de programación. Un Jurado Online usualmente ofrece además otros servicios de consulta y comunitarios a sus usuarios y suele servir como herramienta para la realización de competencias de programación.

La principal función de un Jurado Online es la de permitir a un usuario auto evaluarse sus conocimientos sobre programación y algoritmos, presentándole varios problemas y permitiendo someter una solución a estos. Estas soluciones son evaluadas y el resultado junto a un conjunto de datos (tradicionalmente longitud de código, tiempo de ejecución y memoria consumida) son presentados al usuario. La implementación y las funcionalidades auxiliares y complementarias de un jurado son variadas, pero existe un conjunto de funcionalidades fijas que son las mínimas para permitir que una aplicación de este tipo cumpla su propósito principal. Esto permite el uso de los Jurados como herramientas efectivas para realizar competencias de programación, ya sean presenciales o distribuidas.

En nuestra Universidad son conocidos desde hace algún tiempo algunos de los jurados internacionales más famosos, utilizados en eventos de categoría mundial. Incluso se llega a utilizar uno de ellos (el Sphere Online Judge, de la Universidad de Gandsk) para una Copa Pascal. Sin embargo hasta el año pasado (2006) no se contaba con una implementación propia de este tipo de aplicación. En este marco surge el Jurado Online de la Facultad 8, el primero de la Universidad y probablemente del país, como una alternativa a la navegación limitada de Internet, que con el tiempo experimenta un crecimiento estable en su base de usuarios. La aceptación es positiva y la aplicación recibe buena crítica y retroalimentación constructiva de sus usuarios.

En el año siguiente y siguiendo este primer lanzamiento se crearon dos Jurados Online adicionales en la Universidad, y comenzó un frenesí de creación de eventos de programación paralelos a las

establecidas Copas Pascal y Void. Todos estos eventos surgen apoyados en uno u otro de los jurados, lo que manifiesta su utilidad como herramientas al servicio de una comunidad docente universitaria.

En la actualidad estas aplicaciones existen pero no están vinculadas de forma oficial al proceso docente, teniendo un aporte significativo pero no reconocido en la formación vocacional de los estudiantes y en la ejercitación de sus habilidades.

Modelo Teórico de Investigación

Situación Problémica

La tarea de facilitar la adquisición y dominio de conocimientos y habilidades en la disciplina de programación entre estudiantes de la carrera de informática en la UCI es difícil y tiene un enfoque teórico casi exclusivo. Esto hace difícil crear motivaciones adecuadas en los estudiantes y provoca una inclinación alarmante hacia disciplinas informáticas con poca o ninguna presencia de habilidades de programación, como diseño gráfico, aseguramiento de la calidad, ingeniería de requisitos, arquitectura de información, etc. y provoca un vacío potencial en roles fundamentales del proceso de desarrollo de software como son diseñador, programador y arquitecto, totalmente dependientes de estos conocimientos.

Problema científico

¿Cómo crear, potenciar y evaluar habilidades, afición y conocimientos en el arte y la ciencia de la programación dentro de la masa estudiantil de la UCI facilitando la ejercitación de dicha disciplina?

Objeto de estudio

Sistemas automáticos de evaluación de soluciones en línea (Jurados Online) y eventos competitivos asociados.

Campo de acción

Aplicación de un sistema automático de evaluación de soluciones en línea como apoyo al proceso de aprendizaje; y la creación de pautas para la creación de un evento competitivo asociado al mismo.

Objetivo general

Implementar y exponer para uso público un jurado online y cualquier otro software, metodología o proceso, necesarios para lograr un funcionamiento correcto y una explotación eficiente de dicha aplicación.

Objetivos específicos

- Diseñar e implementar un jurado online.
- Diseñar un sistema de manejo de contenido para manejar eventos competitivos de programación.
- Diseñar un evento competitivo soportado por la aplicación.
- Diseñar e implementar un sistema bibliográfico de apoyo al uso del jurado.
- Diseñar e implementar un foro para la comunidad del jurado en línea.

Idea a Defender

El uso del Jurado Online potenciara el aumento de conocimientos y habilidades en las disciplinas de Programación y afines, y la motivación de los estudiantes hacia su aprendizaje.

Aportes prácticos esperados

Con la implementación del jurado en línea y sus aplicaciones auxiliares se espera que la afición de los estudiantes a las ramas de la programación crezca, así como su habilidad y práctica en la creación de soluciones de programación a problemas que puedan presentarse en su trabajo cotidiano. El enfoque práctico de la aplicación trae aparejado un efecto didáctico que esperamos pueda ser utilizado en conjunción con las formas actuales de enseñanza para mejorar el proceso de aprendizaje de las asignaturas de programación incluyendo el uso del jurado en línea dentro de los programas temáticos de dichas materias. También se espera que focalizando los esfuerzos de todos los interesados en un solo lugar la aplicación contribuya a la consolidación de la comunidad de programadores de la UCI, primordial para el intercambio de conocimientos dentro de la Universidad.

Tareas investigativas

Tareas generales de investigación

- Analizar referencias bibliográficas en línea sobre el tema de los evaluadores automáticos y la implementación de sus principales funcionalidades.
- Analizar aplicaciones existentes con funcionalidades semejantes a la actual y la posibilidad de adaptarlas a la aplicación final.
- Analizar los eventos competitivos asociados a Jurados Online existentes dentro y fuera del país.
- Analizar y clasificar las funcionalidades necesarias y deseables para ser implementadas en la solución final.

- Investigación y evaluación de las herramientas disponibles para el entorno de desarrollo, el entorno de producción y el desarrollo de los complementos de la aplicación con vistas a utilizar la más adecuada según la situación actual.

Tareas del proceso de ingeniería y desarrollo del software

- Investigar sobre metodología XP de desarrollo de software en su uso para aplicaciones de mediana escala con poco personal y requerimientos cambiantes.
- Definir el proceso de desarrollo a seguir dentro de la filosofía de XP para el caso particular de este trabajo de forma que pueda evaluarse el avance de la aplicación.
- Configurar el entorno de desarrollo y las herramientas necesarias para una estrategia de integración continua compatible con XP.
- Desarrollar un conjunto de reglas y políticas de configuración del entorno de producción a fin de lograr seguridad, compatibilidad y disponibilidad de la aplicación y sus complementos.

Tareas de análisis e implementación de la aplicación

- Desarrollar un conjunto de configuraciones de seguridad de la aplicación de forma que sea lo más invulnerable posible ante ataques externos.
- Analizar e implementar un nuevo módulo de competencia con funcionalidades más adecuadas al panorama competitivo universitario.
- Estudiar la conveniencia y dificultad de la implementación de una solución de integración con el dominio UCI permitiendo acceso más restrictivo a los usuarios del jurado. En conjunción con esto, investigar la posible aplicación de protocolo HTTPS para autenticación segura contra el dominio.
- Investigar posibles soluciones de seguridad, en especial Acegi Security System for Spring para mejorar el manejo de roles.
- Definir del conjunto de lenguajes aceptados por el evaluador, basados en la necesidad y la demanda por parte de los usuarios.
- Implementar un módulo de aseguramiento del código fuente que no permita ataques maliciosos al servidor.
- Investigar la posibilidad de implementar un sistema de puntuación dinámico basado en alternativas existentes en el entorno internacional para permitir una mejor apreciación del desempeño de los usuarios.
- Migrar la capa de acceso a datos a procedimientos almacenados que tienen mayor velocidad y eficiencia.
- Implementar la característica de juzgado especial que permita tipos distintos de respuestas que el modelo actual.
- Cambiar la interfaz gráfica de la aplicación existente logrando aumento de usabilidad mediante la creación de elementos distintivos, como logos o lemas.

- Analizar, diseñar e implementar un módulo de estadísticas para recoger información importante sobre el comportamiento de la aplicación y enfocarla en variables específicas.

Tareas de análisis e implementación de los anexos a la aplicación

- Implementar un instalador para facilitar el despliegue de la aplicación y anexos.
- Investigar una solución para la compartición de archivos que haga accesible documentación a los usuarios de la aplicación.
- Investigar una solución para incorporar un foro de usuarios a la aplicación que permita la creación de una comunidad.
- Diseñar e implementar una aplicación web para mantener información sobre eventos competitivos celebrados en el mismo jurado.
- Definir las reglas y procedimientos de un evento competitivo que pueda ser soportado por la aplicación siguiendo los modelos existentes en la Universidad para tales actividades.

Estado del arte

En lo que al tema de los evaluadores automáticos se refiere se ha escrito y hecho mucho. Sin embargo, no existe una teoría unificada para desarrollar evaluadores, sino más bien las pautas para hacerlo van a depender de las necesidades de la organización que lo haga, si el evaluador cumple una necesidad, o de la visión particular del equipo de desarrollo que toma tal tarea, si el evaluador se pretende utilizar para brindar un servicio general y no para cumplir una necesidad puntual. A esta última categoría corresponden los jurados en línea publicados en Internet, que representan el estado del arte en lo que se refiere a funcionalidades, tipos de eventos ofertados y lenguajes que soportan.

En la definición del estado del arte en relación al contexto que nos ocupa, nos enfocaremos en varias áreas y sub áreas principales:

- Herramientas de desarrollo
  - Lenguaje
  - Gestor de datos
  - Herramientas de automatización del proceso de desarrollo.
- Metodologías de desarrollo
  - RUP
  - XP
- Jurados en línea
  - Funcionalidades principales ofrecidas por las soluciones existentes
  - Tipos de eventos existentes
    - Competencia libre
    - Competencia por invitación
    - Concurso Abierto
  - Lenguajes soportados

Jurados en línea

Los jurados en línea son aplicaciones suficientemente conocidas para no ser exóticas, pero tampoco son muy comunes ni frecuentadas, dado que su público objetivo es muy especializado. Los jurados online se asocian normalmente con entornos académicos, en especial Universidades, debido a

que es en este entorno que la búsqueda de conocimiento teórico se estimula de forma más activa, y a que los objetivos y funcionamiento de un jurado online no son fáciles de convertir en lucrativo excepto por las actividades colaterales de capacitación, prestigio, publicaciones, eventos especiales por invitación, etc.

Las instancias elegidas para representar el estado del arte son un conjunto de jurados en línea bien conocidos por la comunidad especializada en nuestra Universidad, así como los jurados internos de la misma institución:

Internacionales

- Sphere Online Judge (www.spoj.pl)
- ACM-Timus Online Judge (http://acm.timus.ru)
- ACM-PKU Online Judge (http://acm.pku.edu.cn)
- ACM-UVa Online Judge (http://acm.uva.es)

Nacionales

- Jurado CAPV
- UCI Online Judge*

**UCI Online Judge es el nombre oficial del jurado de la facultad 4, aunque a nuestro entender el jurado de la CAPV representa mejor a la Universidad como institución.*

Usaremos estas soluciones junto a un conjunto de funcionalidades que todas ofrecen para definir lo que podemos llamar paquete básico de funciones de un jurado online. Este paquete conformará nuestra propuesta de solución, y estará compuesto de las funciones básicas y de aquellas que se juzguen útiles y/o interesantes, según la experiencia de los usuarios y la dificultad de su implementación.

Sphere Online Judge

El Sphere Online Judge (SPOJ) es un proyecto del Sphere Interest Project, entidad afiliada a la Universidad de Tecnología de Gdansk en Polonia.

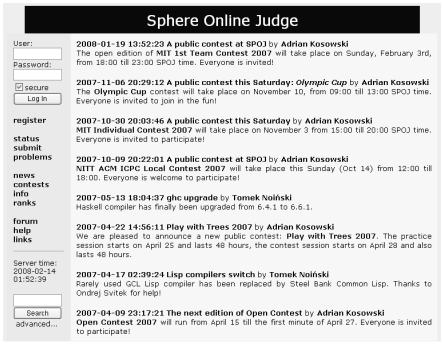


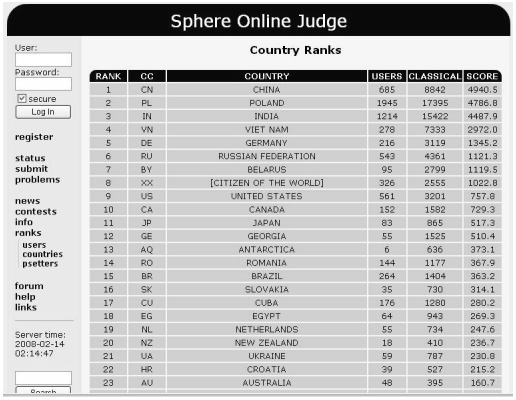
Imagen 1. Página principal de SPOJ

El SPOJ es una solución técnica muy sofisticada. Su sistema de puntuación es el más complicado de todos los jurados analizados, al igual que el sistema de distribución de los problemas, de los cuales reconoce varios tipos distintos. Los problemas aunque no muy numerosos son mantenidos en varios volúmenes, y estos han sido replicados en ruso, portugués y vietnamita. Cubren todos los problemas presentados en las diversas ediciones del ICPC, así como adiciones de los mismos usuarios y administradores del sitio.

El SPOJ cuenta también con varios rankings, en los cuales influye la puntuación de cada usuario, agrupados por países, por categorías (problemsetter, challenger, normal). Estos rankings, a diferencia de los demás sistemas, no cambian solamente cuando el usuario en cuestión realiza una operación, sino cada vez que se realiza una operación sobre uno de los problemas que tributan a la puntuación del usuario.

Esto da como resultado un sistema muy dinámico de puntuación y asegura que los problemas más complejos valgan más, y que los usuarios que pasen cierto tiempo sin realizar postings bajen en el ranking a favor de los usuarios más activos.

También cuenta con un módulo especializado de competencia, muy versátil y eficiente. Ha sido anfitrión de muchas competencias de varios tipos, oficiales y propias de los usuarios.



RANK	CC	COUNTRY	USERS	CLASSICAL	SCORE
1	CN	CHINA	685	8842	4940.5
2	PL	POLAND	1945	17395	4786.8
3	IN	INDIA	1214	15422	4487.9
4	VN	VIET NAM	278	7333	2972.0
5	DE	GERMANY	216	3119	1345.2
6	RU	RUSSIAN FEDERATION	543	4361	1121.3
7	BY	BELARUS	95	2799	1119.5
8	XX	[CITIZEN OF THE WORLD]	326	2555	1022.8
9	US	UNITED STATES	561	3201	757.8
10	CA	CANADA	152	1582	729.3
11	JP	JAPAN	83	865	517.3
12	GE	GEORGIA	55	1525	510.4
13	AQ	ANTARCTICA	6	636	373.1
14	RO	ROMANIA	144	1177	367.9
15	BR	BRAZIL	264	1404	363.2
16	SK	SLOVAKIA	35	730	314.1
17	CU	CUBA	176	1280	280.2
18	EG	EGYPT	64	943	269.3
19	NL	NETHERLANDS	55	734	247.6
20	NZ	NEW ZEALAND	18	410	236.7
21	UA	UKRAINE	59	787	230.8
22	HR	CROATIA	39	527	215.2
23	AU	AUSTRALIA	48	395	160.7

Imagen 2.Ranking por países. Cuba está en el 17

El sistema de jurado especial del SPOJ también es digno de mención, dado que tiene varias implementaciones, a elegir en dependencia del problema. Es el sistema de jurado especial más flexible de todos los jurados analizados, con excepción del UVa.

Soporta más de 30 lenguajes, la mayor cantidad de lenguajes soportados por un jurado en Internet. Sin embargo, es válido decir que la mayoría de estos lenguajes son muy especializados, anticuados o simplemente con una base de usuarios muy pequeña.

En resumen, el SPOJ es una solución tecnológicamente elegante, muy flexible y que soporta todas las funcionalidades básicas de un jurado online maduro. Presenta buena variedad de problemas y una base estable de usuarios y competencias.

UVa

El jurado de la Universidad de Valladolid (UVa) es el jurado que goza de más prestigio en la comunidad universitaria. También es uno de los más viejos y bien surtido en ejercicios, y frecuentemente es la fuente de problemas para los demás jurados.

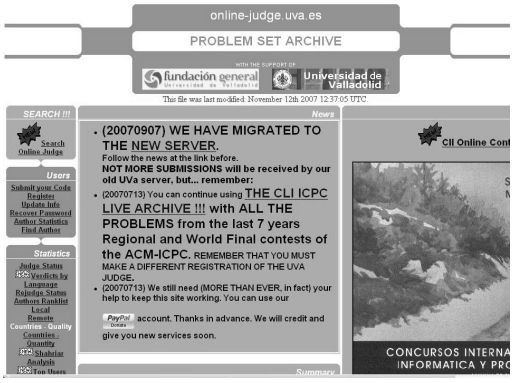


Imagen 3.Página principal

El equipo que atiende el UVa pertenece a la Universidad de Valladolid, y gozan de gran respeto en la comunidad mundial por sus libros de introducción de algoritmos y de programación de competencia. Nombres como Shariar Manzoor y Miguel Revilla son reconocidos como grandes expertos en el tema de la programación de competencia, y han estado envueltos de una forma u otra en las últimas ediciones de las competencias de la ACM a nivel mundial.

El UVa tiene como característica fundamental una gran cantidad de ejercicios organizados en extensos volúmenes. Estos volúmenes se nutren normalmente de las competencias de la ACM y son por tanto casi canónicos, referencia obligada por parte de los interesados en el tema.

Imagen4.Estado del jurado

La aplicación tiene dos principales debilidades: la interfaz gráfica, que deja más que desear de lo que es normal incluso en aplicaciones sin mucho diseño gráfico como son los jurados; y el mecanismo para recibir sumisiones, que es innecesariamente complejo.

Sin embargo, el jurado está en medio de un proceso de migración a un nuevo servidor que también ha servido para mejorar su interfaz gráfica y el acceso a los procesos fundamentales de sumisión, administración de cuentas y consultas de estado y estadísticas.

El UVa tiene la característica de promover productos industriales en su interface, añadiendo un elemento comercial a su propósito tecnológico primario. En la nueva interfaz incluso se incluyen avisos patrocinados en todas las páginas.

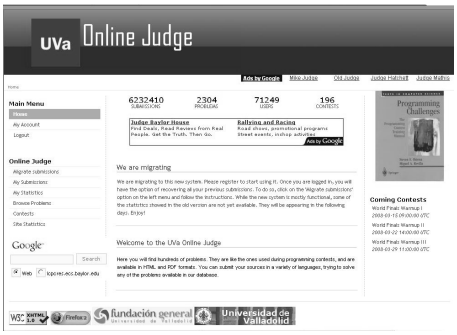


Imagen 5.Nueva interface UVa

En resumen, el UVa es el jurado más grande y prestigioso de todos los analizados. Es host de competencias del ICPC y contiene los volúmenes más grandes de problemas presentes en Internet. En nuestra Universidad, el UVa representa siempre la referencia última sobre problemas en los que existan dudas.

Timus

El jurado Timus, perteneciente a la Universidad de los Urales, en Rusia, es un proyecto estudiantil propio. Este jurado tiene las funcionalidades básicas de su tipo, con una interfaz intuitiva y fácil de usar.

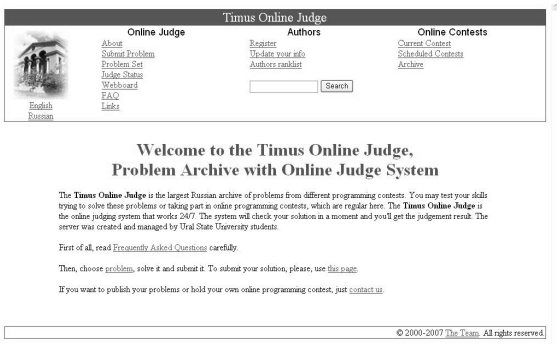


Imagen 6.Página principal

Sus características más distintivas son, aparte de la falta de diseño gráfico estándar, el tener dos versiones del mismo software, en ruso y en inglés, presentar una solución interna de comunicación (los jurados anteriores recaen en soluciones de terceros para implementar un foro de usuarios) y brindar la posibilidad de recibir el reporte de la solución por correo (esta funcionalidad también la brinda SPOJ, pero en Timus puede ser desactivada o activada para cada sumisión).

El diseño de las funcionalidades de este jurado, a pesar de ser muy simple cumple con todas las necesidades de los usuarios. De hecho este conjunto de funcionalidades es virtualmente idéntico al de los jurados chinos de Beijing (que analizaremos a continuación) y Zhejiang, que no vamos a analizar. Hay otras similitudes en la interfaz gráfica que parecen sugerir que se tomaron ideas al menos sobre el diseño web de unos a otros.

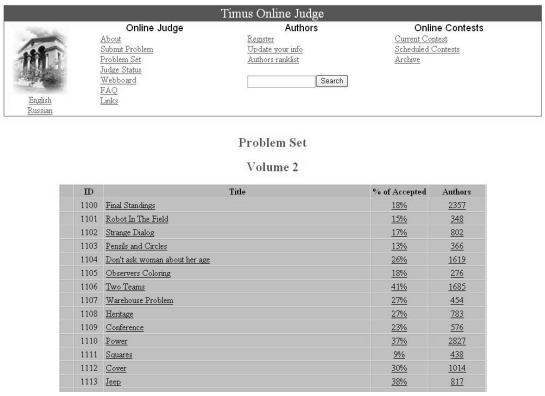


Imagen 7.Volumen de problemas

Las principales características de este jurado son un conjunto sencillo de funcionalidades básicas, un diseño gráfico sobrio, pero funcional, muy intuitivo, buenos tiempos de respuesta y solución de comunicación interna a través de una web board.

PKU Judge Online

Los jurados de Beijing (Peking en el momento de su creación) y Timus tienen ciertas funcionalidades en común aunque las del primero son mucho más numerosas. En adición a las normales, el jurado de Beijing ofrece correo interno como complemento a su web board.

Imagen 8.Página principal con vinculo de descarga

También es de notar el módulo de estadísticas de sumisión, muy interesante y que permite apreciar el devenir del jurado desde que fue creado en 2003. El funcionamiento de este jurado es muy suave, intuitivo y rápido. El trafico del jurado es fuerte, el pico actual, según estadísticas del mismo sitio, es de 1 163 476 sumisiones anuales en el 2007, lo cual junto al análisis de las sumisiones diarias lleva a suponer que este jurado es el más visitado de todos los analizados.

La mayor diferencia entre este y los demás jurados online es la opción de descargar una versión libre de la aplicación. Esta versión libre es mucho más pequeña y restrictiva que el original, además de componerse de ficheros compilados que no proporcionan código fuente. La opción de descarga está diseñada para poner en funcionamiento rápidamente una versión del jurado en cualquier entorno. Esta opción ha sido muy influyente en el desarrollo de nuestro jurado online, lo cual discutiremos en la próxima sección.

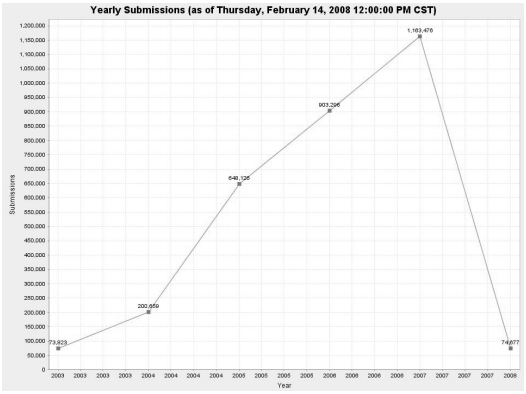


Imagen 9.Estadísticas de sumisiones anuales. El pico es el 2007.

En resumen el PKU es el más visitado de todos los jurados analizados, y el que tiene la interfaz más intuitiva de todos. Alberga muchos servicios adicionales, y se basa en una comunidad activa de usuarios.

Legado

La opción de descargar provee un conjunto de clases en Java, con un script de generación de base de datos para MySQL MyISAM. La tecnología Java utilizada son servlets, sin más dependencias. Los lenguajes soportados son Pascal, C y C++, en contraste con los lenguajes soportados por la versión original que incluyen también Java, Fortran y G++, una versión de C++ en un compilador no estándar.

Esta versión del jurado constituyó la base del primer prototipo del jurado actual. Para poder suplir las carencias de la versión libre, fue necesario embarcarse primeramente en un esfuerzo de ingeniería inversa, comenzando por desensamblar el código compilado de Java para convertirlo en código fuente.

Este esfuerzo fue parcialmente exitoso, dado que no se logro desensamblar la totalidad del código, lo que nos llevó a la segunda parte del proceso de absorción, consistente en reescribir las funcionalidades vitales faltantes. Una vez logrado esto, estuvimos en posición de mejorar todas las funcionalidades del jurado y de añadir nuevas.

Este código se define como código legado, en su calidad de base para un sistema funcional que hereda conceptos e implementaciones del código anterior. Una aclaración pertinente en este punto es decir que el jurado actual de la facultad 8 NO ES una extensión de la versión libre del PKU, sino un súper conjunto de funcionalidades reescritas en base al código legado disponible. Algunas de las funcionalidades añadidas al prototipo actual incluyen la adición de C# mediante un compilador desarrollado específicamente para este propósito, de Java en su versión 5.0, Python en su versión 2.4 y de sumisiones en texto plano; la integración de un foro de usuarios que proporcione una solución más completa que el web board original, algunas opciones de administración, avisos importantes y algunas más. La principal diferencia entre la versión descargable y el prototipo actual es la refactorización extensa que se realizó a las funcionalidades de forma interna, que no afecta la interfaz ni los servicios del usuario final.

El prototipo actual ha sido probado con éxito durante poco más de dos años, desde enero del 2006, lo que ha permitido detectar errores y riesgos que presenta, paliar algunos y diseñar soluciones para los demás. El prototipo actual se convirtió en el primer Jurado Online de la UCI y hasta donde es conocido, del país (aunque en el 2005 se hizo el intento de albergar la Copa Pascal de ese año en un

prototipo temprano de lo que luego sería el UCI Online Judge, este tenía muy pocas funcionalidades y estaba lleno de bugs, y no siguió en funcionamiento al finalizar la competencia). El prototipo ha superado con éxito las pruebas de carga realizadas y ha dado lugar al campeonato XProg, la cuarta competencia más antigua de programación de la UCI, detrás de la Pascal, la Void y la Java.

Debido a circunstancias actuales, la aplicación fue migrada a un servidor no dedicado, lo cual ha degradado severamente su rendimiento y disponibilidad, además de reducir sus funcionalidades lo cual ha mermado su base de usuarios.

En los años siguientes surgieron dos alternativas a nuestro prototipo, siguiendo procesos y objetivos distintos. Como nota interesante, el desarrollo de los Jurados Online en la Universidad ha seguido caminos distintos, se han desarrollado de forma completamente independiente y ha dado resultados tan diversos como los de las aplicaciones en Internet. A continuación analizamos estas alternativas en más detalles.

UCI Online Judge

El UCI Online Judge es el producto del trabajo de programadores de la Facultad 4 bajo la dirección del profesor Iván Alfonso Olamendy. Esta aplicación fue diseñada e implementada entre el 2005 y el 2006, usando tecnología de frameworks de aplicación de Java. Se despliega sobre Linux y soporta 8 lenguajes (C, C++, Pascal, C#, Java, PHP, Perl, Python) y sumisiones en texto plano. Obviamente el diseño visual está inspirado en el SPOJ.

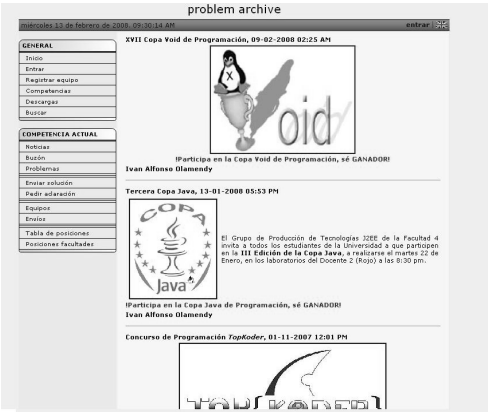


Imagen 10.Página principal

Este jurado presenta todas las funcionalidades esperadas, incluyendo la de comunicación interna. El sistema de autenticación a diferencia del nuestro prototipo es contra el dominio UCI, aunque también permite crear usuarios locales y equipos. El manejo de los usuarios y equipos es un punto fuerte de la aplicación. Se proveen además varios rankings, por facultad, por equipos, total, etc. El jurado también cuenta con un módulo separado de competencia, muy profesional.

La aplicación ha servido y continúa sirviendo como host a diversas competencias de la Universidad, por ejemplo la Copa Java (la competencia nativa del jurado) las Copa Top Koder y Void, y especialmente la 2da edición de la CEIP, el más importante de estos eventos que se haya realizado en Cuba. Actualmente (2007) se utiliza como herramienta de entrenamiento para la Preselección Nacional. En ese mismo año se trató de manejar la Copa Pascal de ese año, pero sin éxito.

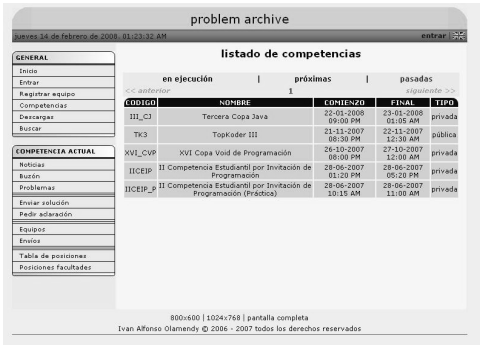


Imagen 11.Competencias pasadas

El uso de frameworks en el desarrollo trajo aparejado flexibilidad en el diseño, lo cual permite en teoría cambiar el diseño y evolucionar el jurado con poco esfuerzo. Sin embargo, el uso de los frameworks también trajo una degradación del rendimiento, al punto de que en versiones anteriores de la aplicación no era posible refrescar continuamente una página sin que la aplicación colapsara. En la versión actual este error parece corregido en cierta medida, aunque aún existe.

Jurado CAPV

El más joven de los jurados de la UCI, el jurado de la cátedra avanzada, rompe con los esquemas de los jurados anteriores, en filosofía, herramientas y objetivos. El CAPV está construido usando un CMS de PHP (más exactamente, el PHP-Fusion) distinto de los desarrollos Java con soluciones particulares que seguimos los demás. Para evaluar las soluciones no utilizan el mismo software, sino un daemon aparte construido en C++.

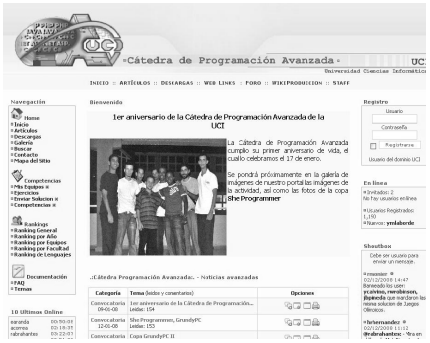


Imagen 12.Página principal

El jurado CAPV ha hosteado varias competencias también, por ejemplo GrundyPC, She Programmer y la Copa Troya. El CAPV ofrece varios tipos de ranking, por año, por facultades, por lenguajes y general. Soporta los lenguajes clásicos de competencia y C#. Recientemente (2008) este jurado hospedó la 3ra edición de la CEIP, renombrada como Copa UCI de Programación, presentando inconvenientes menores en cuanto a su disponibilidad. Está planificado que la Copa Pascal del 2008 también tenga lugar bajo su soporte.

El CAPV ofrece una serie de servicios que derivan de su calidad de CMS, como son la galería, el shoutbox, los anuncios y los últimos usuarios conectados. En contraste, no ofrece la funcionalidad básica de ver el estado actual de las sumisiones fácilmente, lo cual es una deficiencia grave. El sistema de clasificación de los problemas en volúmenes por temas puede ser contraproducente dado que ofrece información sobre los conocimientos necesarios para resolver el problema. El uso de un CMS puede resultar contraproducente a la hora de aumentar las funcionalidades de la aplicación o de evolucionar su interfaz gráfica, dado que no se tiene control completo sobre el código al ser código legado con sus propias funcionalidades y arquitectura.

Imagen 13.Galería

Conclusiones

Después de analizar las soluciones existentes al problema de la creación de una aplicación de Jurado Online, obtenemos como conclusión que debemos ofrecer el siguiente conjunto de funcionalidades:

- Noticias
- Organización de los volúmenes de problemas
- Administración de la cuenta
- Sistema de puntuación
- Jurado especial
- Número suficiente de lenguajes
- Estado actual del jurado
- Competencias nativas
- FAQ
- Comunicación interna
- Estadísticas

Que parecen ser funcionalidades comunes a todas las soluciones existentes. La existencia del prototipo actual del Jurado de la 8 y su funcionamiento durante 2 años ha servido para validar que el modelo de funcionamiento actual es correcto y por tanto podemos darle continuidad.

Herramientas de desarrollo

Lenguaje

Por las propias características del evaluador, este debe desarrollarse como una aplicación Web con soporte de datos. Partiendo de esta base, la selección de herramientas se enfoca en tres vertientes fundamentales, que comprenden un lenguaje de programación (y un estilo de programar en estos lenguajes): PHP, Java y C#. La selección del lenguaje es la más importante porque influye en la decisión sobre el gestor de datos y sobre las herramientas para automatizar el proceso de desarrollo.

A continuación se muestra una comparación entre los tres lenguajes posibles que se evaluaron para la elaboración del software. Los criterios utilizados fueron elegidos arbitrariamente de acuerdo a las necesidades del proceso de desarrollo y a las capacidades de los programadores, y reflejan por tanto un punto de vista subjetivo:

Tabla 1: Comparación entre lenguajes candidatos para el desarrollo

Criterio	PHP	Java	C#
Sencillez del lenguaje	Simple	Complejo	Complejo
Facilidad de uso	Media	Difícil	Fácil
Requerimientos de hardware y software	Bajos	Altos	Altos
Familiaridad con el lenguaje	Media	Alta	Media
Capacidad OO	Baja	Alta	Alta
Adecuación a las necesidades de la aplicación	Alta	Alta	Alta
Dominio de herramientas	Bajo	Alto	Medio
Referencias de desarrollos similares	Abundantes	Abundantes	Numerosas
Legacies	No	Sí	No
Estilos propios de programación	Medianamente desarrollados	Muy desarrollados	Medianamente desarrollados
Curva de aprendizaje*	Media	Ninguna	Alta
Bibliografía y personal capacitado	Acceso medio	Acceso abundante	Acceso medio

**Esta curva no se refiere a la curva real de aprendizaje, sino a la que tendríamos dadas las condiciones en que nos encontramos a la hora de realizar el trabajo.*

En este, como en otros criterios, influyeron factores más ambientales que puramente técnicos.

Basando nuestra decisión en los puntos 4, 5, 7, 9, 10 y 11 se decidió que el lenguaje más adecuado a las circunstancias del momento y las habilidades dominadas era el Java. Esto nos permite compaginar nuestro trabajo principal con el desarrollo de la aplicación en cierta medida, mayor de lo que hubiéramos podido hacer de haber elegido PHP o C#. Nos permite aprovechar el entrenamiento recibido previamente, así como reutilizar conocimiento de herramientas, procesos y estilos de codificación y organización del trabajo, ya establecidos entre nosotros. La ganancia en tiempo y esfuerzo derivada de estas reutilizaciones compensa con creces las desventajas de la plataforma elegida, mayormente la complejidad del modelo programático y la dificultad de manejo de las herramientas conocidas de desarrollo.

Gestor de datos

La elección de Java como lenguaje de desarrollo no condicionó un gestor de datos particular, como hubiera hecho PHP o C#, los cuales tienen gestores recomendados para los cuales tienen soporte especial (MySQL y MS SQL Server, respectivamente). En Java, sin embargo tal decisión todavía queda abierta, haciendo necesario pensar en la elección de uno de los múltiples gestores existentes. De nuevo, se utilizó una serie de criterios de carácter técnico, evaluados subjetivamente, que se adecúan a las condiciones del medio y a las capacidades de los programadores:

Tabla 2: Comparación entre gestores de datos candidatos para el desarrollo

Criterio	MySQL	PostgreSQL	Oracle	MS SQL Server
Requerimiento hardware	Bajo	Medio	Alto	Alto
Facilidad de uso	Fácil	Medio	Difícil	Fácil
Integración con Java	Sí	Sí	Sí	Sí*
Familiaridad con el dialecto	Sí	No	No	Sí
Procedimientos almacenados	Sí**	Sí	Sí	Sí
Adecuación a los requisitos de la aplicación	Sí***	Sí	No	Sí
Dominio de herramientas	Sí	No	No	Sí
Experiencia anterior	Sí	No	No	Sí
Existencia de *legacies*	Sí	No	No	No
Curva de aprendizaje	Baja	Media	Alta	Baja
Bibliografía y personal capacitado	Suficiente	Pobre	Pobre	Suficiente

**La integración de SQL Server con Java constituye un proceso dificultoso que precisa herramientas especiales (es un* hack *más que una funcionalidad).*
***Las primeras versiones de MySQL no soportaban procedimientos almacenados, pero poco antes de comenzar el desarrollo se agregó en la versión 5.x.*
**** En MySQL el nivel de adecuación a los requisitos es afectado por la selección del motor de BD, como se verá más adelante.*

Basados en los puntos 1, 2, 7, 8, 9 y 10 elegimos MySQL como gestor de datos. En atención al punto 6, sobre adecuación a los requisitos de la aplicación, la elección se ramificó debido a la existencia en MySQL de dos motores principales de almacenamiento a elegir: InnoDB y MyISAM.

InnoDB vs MyISAM

Cada uno de estos motores presenta un conjunto de características únicas, pero en general InnoDB es el más moderno y se perfila como el reemplazo de MyISAM. Las respectivas ventajas reconocidas de cada uno son:

- **InnoDB**: transacciones ACID, bloqueo a nivel de fila (*row locking*, que permite a varias transacciones accionar sobre una misma tabla de forma concurrente y mantener la integridad de los datos al mismo tiempo), y mantenimiento general de la integridad referencial mediante el mecanismo de llaves foráneas. InnoDB es recomendada para aplicaciones con elevado número de cambios, que requieren mantener la integridad referencial en la BD, aunque presenta algunos problemas manejando datos de tipo TEXT y BLOB. InnoDB fue adquirido por Oracle.
- **MyISAM**: No mantiene integridad referencial, ni transacciones. MyISAM utiliza el mecanismo de bloqueo a nivel de tabla lo que no permite a dos transacciones concurrentes acceder a los datos en una misma tabla al mismo tiempo. Al no mantener referencias a otros datos, MyISAM es recomendado para aplicaciones con gran cantidad de lecturas. Esto sin embargo, ha ido evolucionando en el tiempo al punto de que hoy en día existe debate sobre la ventaja en velocidad de MyISAM sobre InnoDB.

A pesar de las ventajas de InnoDB en el mantenimiento de integridad referencial y transacciones, los problemas potenciales que podía presentar al manejar campos BLOB (importantes en la aplicación) y la ventaja de velocidad generalmente reconocida en Internet del MyISAM influyeron en su selección como motor de datos para la aplicación. La selección de este motor en la base de datos legada también representó una influencia importante a la hora de tomar una decisión.

Herramientas de automatización del proceso de desarrollo

Las herramientas de desarrollo dependen de la selección del lenguaje, gestor de datos, plataforma y ambiente de trabajo. También dependen del tipo de aplicación a realizar, su alcance, las tecnologías específicas y circunstancias especiales del ambiente de producción o desarrollo que puedan determinar su selección por razones no técnicas (políticas organizacionales, acceso, costo, etc.).

En el caso de las tecnologías elegidas, la decisión se ramifica aún más en los siguientes temas:

- Tecnología de desarrollo Web con Java
- IDE
- Servidor de aplicación
- Herramientas para el proceso de desarrollo
- Cliente del gestor de datos

En estos casos nos vamos a extender sólo en la selección del primer tema. Los demás realmente no fueron objetivo de un proceso de selección de ningún tipo, debido a que el IDE, el servidor de aplicaciones y las herramientas para el proceso de desarrollo ya estaban definidos por el proyecto CICPC, y por tanto contamos con experiencia en su manejo y con una base de personal experimentado en su uso. Las herramientas elegidas conforman lo que podemos llamar un perfil de software bastante popular en los desarrollos de la plataforma Java en la UCI:

- Eclipse como IDE.
- Tomcat como servidor de aplicaciones*
- SVN como control de versiones
- Ant como herramienta script de automatización

**el Tomcat es en realidad un contenedor de servlets, no un servidor de aplicaciones, pero suele competir con estos.*

En el caso de los clientes de la base de datos, existen varias opciones para comunicarse con un servidor MySQL, pero el software más sencillo de usar y más intuitivo es el EMS MySQL Manager, por lo cual esto tampoco constituyó una selección real.

Tecnologías de desarrollo web en Java

En el mundo de la tecnología Java existen varias formas de desarrollar aplicaciones Web. Básicamente el desarrollo se divide en con o sin frameworks. El desarrollo con frameworks, como vimos anteriormente en el estudio de las soluciones existentes, tiene sus ventajas y desventajas, siendo la mayor ventaja el desarrollo rápido usando las capacidades del framework; y la desventaja principal la lentitud de la aplicación debido a las múltiples operaciones que realizan los frameworks internamente. Esta sola desventaja se probó suficiente para desechar el desarrollo de la aplicación usando frameworks, basándonos en experiencias negativas derivadas del rendimiento observado en el UCI Online Judge.

Fuera de los frameworks, la plataforma Java ofrece dos soluciones fundamentales: JSP y Servlets. Siendo JSP una versión de la especificación de servlets mejorada para facilitar el trabajo de diseño, es proporcionalmente más complicado programar en esta, y por tanto se tomó la decisión en este caso de priorizar la facilidad de codificación sobre la facilidad de diseño visual. Para reducir el impacto negativo de esta estrategia en la extensibilidad de la aplicación se aisló toda la lógica relativa a la generación de la interfaz gráfica en un componente especializado. De esta forma se logró un grado suficiente de independencia entre la lógica del negocio y la lógica de la presentación.

Por tanto la solución, principalmente por factores de velocidad, son los servlets de J2EE, en la versión 2.4 de su especificación. Esta es la tecnología Java utilizada, complementada con filtros, sesiones Http y demás componentes adicionales de la especificación.

Debemos hacer notar que esta es la tecnología de Java más primitiva, y que la razón de ser de muchos frameworks de presentación es precisamente abstraer al programador de la necesidad de comunicarse con servlets en su estado puro. Sin embargo los servlets permiten controlar el proceso de la aplicación por completo y son mucho más rápidos que los frameworks de presentación. Además constituyen una tecnología familiar y de bajo riesgo, al ser bastante conocida para nosotros. El tiempo ganado por una curva de aprendizaje inexistente y la facilidad de uso, además del aprovechamiento directo de código legado y la ganancia de velocidad fueron los factores concluyentes para tomar esta decisión.

Perfil tecnológico de desarrollo

Dada la información anterior y las decisiones de desarrollo anteriormente enunciadas, el perfil tecnológico del ambiente de desarrollo y de producción de la aplicación de definió como sigue:

Tabla 3: Listado de herramientas utilizadas

Clasificación	Herramienta	Versión	Ambientes	
			Desarrollo	Producción
Lenguaje (versión JVM)		**6.0**	✓	✓

Complemento IDE	**Exadel Studio Pro**	**4.0.1**	✓	
IDE		**3.3 Europa**	✓	
Gestor de datos		**5.0.18**	✓	✓
Cliente de datos		**3.7.5.1 & 4.1.2.1**	✓	✓
Control de versiones		**1.4.2**	✓	
Script de automatización		**1.7.0**	✓	
JUnit		**4.1**	✓	
Servidor de aplicaciones	**Tomcat**	**6.0.14**	✓	✓
OS		**Professional v2002 SP2**	✓	✓

Metodologías de desarrollo

Para dirigir el proceso de desarrollo de la aplicación, podemos elegir entre las dos metodologías principales manejadas en la Universidad: RUP y XP. Cada una tiene sus cualidades únicas y distintos objetivos, basados en los cuales debemos elegir una para nuestro proceso de desarrollo. La decisión se restringió a estas dos metodologías debido a que son las metodologías en las que se consideró se podía trabajar sin necesidad de una etapa previa de aclimatación y contando con abundante apoyo de bibliografía y expertos.

RUP

RUP es el resultado de varios años de desarrollo y uso práctico en el que se han unificado técnicas de desarrollo, a través del UML, y trabajo de muchas metodologías utilizadas por los clientes. Como proceso define como sus principales elementos:

- **Trabajadores ("quién"):** Define el rol de un individuo, grupo de individuos, sistema automatizado o máquina, que trabajan en conjunto como un equipo. Ellos realizan las actividades y son propietarios de elementos.
- **Actividades ("cómo"):** Es una tarea que tiene un propósito claro, es realizada por un trabajador y manipula elementos
- **Artefactos ("qué"):** Productos tangibles del proyecto que son producidos, modificados y usados por las actividades. Pueden ser modelos, elementos dentro del modelo, código fuente y ejecutables.
- **Flujo de actividades ("Cuándo"):** Secuencia de actividades realizadas por trabajadores y que produce un resultado de valor observable.

En RUP se han agrupado las actividades en grupos lógicos definiéndose 9 flujos de trabajo, los 6 primeros son flujos de ingeniería y los tres últimos de apoyo.

Cada flujo de trabajo cumple con algunas actividades específicas. En el funcionan trabajadores específicos y producen y consumen artefactos también definidos.

Cada fase representa un estado del proyecto, y produce un hito que sirve de entrada a la próxima fase. Todos los flujos se aplican en todas las fases, si bien algunos tienen más carga de trabajo que otros en algunas fases específicas.

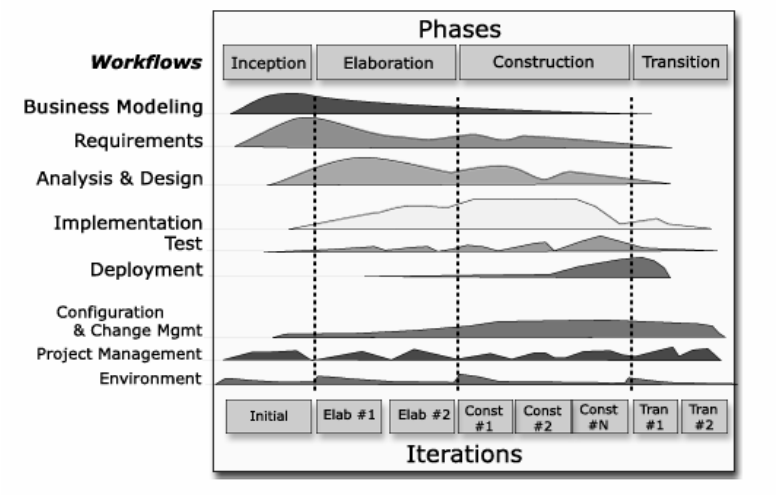


Imagen 14.RUP en Dos Dimensiones

El ciclo de vida de RUP se caracteriza por ser dirigido por casos de uso, que reflejan lo que los usuarios futuros necesitan y desean, guiando el proceso de desarrollo ya que los demás artefactos representan la realización de los casos de uso; centrado en la arquitectura que muestra la visión común del sistema completo y describe los elementos del modelo que son más importantes para su construcción; iterativo e Incremental, lo que significa que cada fase se desarrolla en iteraciones que involucran actividades de todos los flujos de trabajo, aunque desarrolla fundamentalmente algunos más que otros, obteniendo un producto con un determinado nivel que irá creciendo incrementalmente en cada iteración.

XP

La Programación Extrema (XP) es **una perspectiva deliberada y disciplinada al desarrollo de software**. XP tiene éxito porque enfatiza la satisfacción del cliente. Esta metodología está diseñada para entregar el software que los clientes necesitan cuando lo necesitan. XP le da el poder a los desarrolladores para responder con confianza a los requerimientos cambiantes de consumidor final, incluso cuando el proyecto ya está en etapas avanzadas.

La metodología enfatiza el trabajo en equipo. Administradores, clientes y desarrolladores son parte de un equipo dedicado a producir software de calidad. XP implementa una forma simple pero efectiva de permitir desarrollo de software en grupos (groupware).

XP mejora un proyecto de software de cuatro formas esenciales: comunicación, simplicidad, retroalimentación y coraje.

Los programadores de XP se comunican con sus clientes y entre ellos mismos, para mantener su diseño limpio y simple. Obtienen retroalimentación probando su software desde el primer día. Entregan el sistema a los clientes lo más pronto posible, e implementan los cambios a medida que son sugeridos. Sobre estas bases, los programadores pueden afrontar valerosamente requerimientos y tecnología cambiantes.

Los programadores deben respetarse entre sí. Si los miembros del equipo no respetan su trabajo mutuamente, entonces el proyecto no tiene ninguna oportunidad.

XP es diferente. Se parece mucho a un rompecabezas, en que hay muchas piezas pequeñas que individualmente no tienen sentido, pero que cuando se combinan crean una imagen que tiene sentido. Esta es una diferencia fundamental con los métodos tradicionales de desarrollo de software e impulsa un cambio en la forma en la que programamos.

Situación del ambiente de desarrollo

La comparación de estas metodologías harto conocidas en la Universidad se llevó a cabo de forma diferente a como se hizo la selección de las herramientas de desarrollo y de las funcionalidades a implementar.

Primero se definieron algunas condiciones existentes, determinadas por factores ambientales y que por tanto escapan de nuestro control. Estas condiciones determinan la selección de metodología:

- **Poco personal de desarrollo:** Solo se dispone de un equipo de desarrollo.
- **Código legado:** La existencia de código legado hace que la refactorización sea obligatoria
- **Poco tiempo para desarrollo**
- **Aplicación en producción**: El costo de mantenimiento y gestión de la aplicación conspira contra el tiempo total.
- **Requisitos cambiantes:** Los requerimientos cambian a lo largo de todo el ciclo de vida de la aplicación.
- **Pruebas en producción**: La forma más rápida de hacer las pruebas funcionales es manteniendo la aplicación en producción y observando los resultados.

- **Presencia del cliente**: El cliente existe dentro del proceso como ente inconsciente. Todos los usuarios de la aplicación funcionan como clientes, y además están involucrados en el proceso de pruebas y de levantamiento constante de requisitos.

La existencia de estos factores principales determinó la selección natural de XP como metodología de desarrollo. En una situación como la descrita una metodología pesada como RUP se podría aplicar solo con un subconjunto muy restringido de sus prácticas y aun así habría dificultades. Las principales desventajas de RUP en tal ambiente son:

- **Multitud de artefactos**: Para mantener el proceso habría que realizar múltiples artefactos para cada funcionalidad, o modificar los artefactos existentes. Esto consume mucho tiempo.
- **Trazabilidad**: En RUP cada artefacto tiene una traza al artefacto que lo originó. Esto implica que cualquier cambio en un artefacto implica potencialmente cambios en todos los artefactos implicados en su línea de trazabilidad. Esto hace que el mantenimiento de la documentación generada durante el proceso de desarrollo sea dificultoso y lento.
- **Refactorización constante**: Al existir código legado, no desarrollado utilizando RUP (por tanto sin documentación del proceso) es necesario cambiar y ajustar el código constantemente. Esto condicionaría cambios incesantes en toda la documentación del proceso que es necesario mantener si se usa RUP.
- **Poco personal de desarrollo**: Al ser solo dos personas a cargo del desarrollo de la aplicación así como de su mantenimiento y gestión (no olvidar que se está desarrollando con la aplicación en producción) No es sencillo asignar los roles a cumplir por cada uno, dado que tomaríamos varios roles en cada etapa y sería muy complejo cumplir con las actividades de cada uno de ellos.
- **Requisitos cambiantes**: Constantemente durante el proceso de desarrollo de la aplicación aparecen nuevos requisitos. Esto conlleva a que el modelo de fases de RUP se quiebre, debido a que cada nuevo requisito necesitaría reiniciar una iteración para comenzar el ciclo de actividades que den cumplimiento a su funcionalidad.
- **Planificación inexistente**: La planificación de las actividades se lleva a cabo a corto plazo, debido a que una planificación a largo plazo tiene un porciento casi absoluto de seguridad de no cumplirse. Esto se debe a que los requisitos cambian, el tiempo dedicado al desarrollo cambia por factores externos al proceso, los bugs encontrados son imprevistos y de severidad variada, etc. En un proyecto con estas características es muy difícil planificar actividades específicas.

Todas estas condiciones que actúan como inconvenientes de la aplicación del RUP son precisamente las razones por las cuales se sugiere aplicar XP a un proyecto. XP surgió como una metodología rápida y ágil enfocada precisamente en aquellos proyectos en los que las metodologías más

pesadas no eran fáciles de aplicar. Además de las desventajas explicadas anteriormente de una posible aplicación de RUP en nuestro proyecto, existen otros factores que inclinan más la balanza a favor de XP:

- **Conocimiento de las prácticas necesarias**: Uno de los mayores escollos de XP es la necesidad de conocer muchas prácticas de programación y trabajo en equipo para que funcione. La curva de aprendizaje de XP es rápida, pero lleva tiempo acostumbrarse a estas técnicas. Nosotros sin embargo, no tenemos tal desventaja dado que ya teníamos conocimiento de las prácticas básicas de XP, así como cierto grado de experiencia.
- **Herramientas familiares**: Algunas prácticas de la XP como la propiedad colectiva del código y la integración continua son posibles gracias a algunas herramientas, específicamente en nuestro caso el Subversion y el Cruise Control. Gracias a nuestro trabajo regular en el proyecto CICPC, estamos familiarizados con estas herramientas.
- **No hay necesidad de CASEs**: XP hace énfasis en que el código es la documentación, por tanto la generación de documentación para mantener el proceso en un estado comprensible no es reforzada de ninguna forma. Aunque un par de artefactos muy específicos del RUP pueden resultar útiles, mantener la gran cantidad de documentación necesaria, incluso en un proyecto pequeño con un subconjunto de las funcionalidades completas de la metodología resulta una carga demasiado pesada para un proyecto de nuestro alcance.
- **Cliente siempre disponible**: Esta es una ventaja en cualquier metodología, pero en XP es una precondición esencial. En nuestro sistema el rol del cliente está dividido, dado que los clientes son nuestros usuarios asimismo como nosotros mismos. La disponibilidad total del cliente está asegurada, tanto para que suministre requisitos como pruebas funcionales.
- **Estrategia de prueba en producción**: XP se enfoca en tener un sistema en producción lo más pronto posible, para a través de la retroalimentación de los usuarios poder capturar los bugs que escapen de las pruebas de unidad y las pruebas funcionales. Mediante una estrategia de prueba en producción (esto es, probar el sistema con el funcionamiento real) se garantizan las pruebas funcionales en un ambiente real, y se puede mantener la aplicación actualizada haciendo muchas liberaciones pequeñas del software constantemente, otra práctica que RUP no permite y que es fundamental para XP.

Debido a todas estas características, la decisión de metodología no es difícil. Nos inclinamos por XP, una metodología ligera, con menos requerimientos de documentación y planificación a favor de mayor disciplina a la hora de desarrollar la aplicación, y enfocada a proyectos preferentemente de pequeño y mediano tamaño con equipos de trabajo reducidos.

Desarrollo de la solución

Una vez analizado el estado del arte y elegidas las herramientas y metodología a utilizar, estamos en condiciones de plantear una solución a nuestro problema. (Diagrama 1). La solución propuesta, como se verá, no se compone de un sistema sino más bien de un conjunto cooperativo de sistemas donde cada uno proporciona un servicio diferente. Esto permite un desarrollo separado de cada uno y en el caso del foro y el repositorio de bibliografía permite el uso de soluciones de terceros, que pueden ser mucho más útiles que cualquier solución personalizada que se pueda desarrollar.

Para producir la solución propuesta siguiendo la metodología elegida se siguieron los pasos definidos por XP para el proceso de desarrollo. Sin embargo, y utilizando la capacidad de adaptación definida en el mismo XP, el proceso fue adaptado a las condiciones muy especiales de trabajo en las que fue desarrollado el proyecto.

Las fases contempladas dentro del proceso particular de desarrollo fueron: exploración, planificación, primera liberación y mantenimiento. Nótese que las fases de exploración y planificación se repiten en XP en cada iteración de desarrollo, y que las pruebas son una actividad constante desde el comienzo de la implementación de la solución hasta su salida de producción. En nuestro proceso de desarrollo tuvimos características especiales en la planificación de las iteraciones, especialmente las iteraciones iniciales que tributaron a la primera liberación. Esto ocurrió debido a la necesidad de implementar un número significativo de historias de usuario consideradas indispensables para proveer valor de negocio a la aplicación, si bien el sistema podía funcionar con un subconjunto mucho más limitado de estas. El efecto más notable de esta situación fue el retraso inicial en poner el nuevo sistema en producción, lo cual no resultó tan nocivo debido a que las ventajas que normalmente se derivan de mantener el sistema en producción se obtuvieron de todas formas gracias al código legado.

Uno de los objetivos principales de la primera liberación del software fue sustituir el código legado existente por el nuevo código modificado y extendido para soportar las historias de usuario escritas. De esta forma pudimos asegurar la completa extensibilidad de la aplicación, al hacernos definitivamente del control del código base.

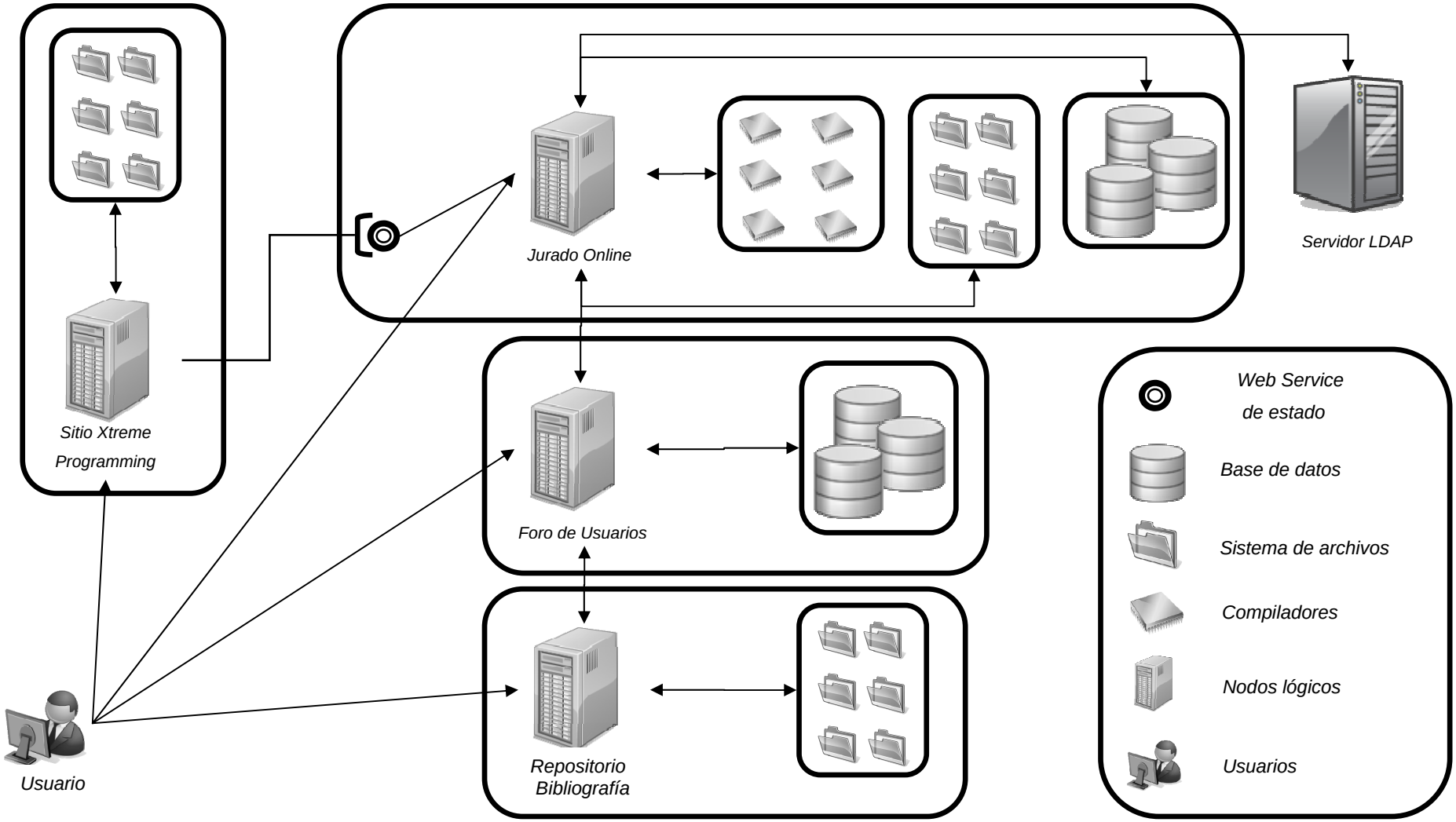


Imagen 15.Diagrama de la arquitectura del prototipo funcional y sus anexos, correspondientes a la solución propuesta.

Exploración

En esta fase se exploran las herramientas y tecnologías a usar, las diferentes formas de resolver problemas concretos de implementación que puedan presentarse y sobre todo se definen historias de usuario, que son la forma de definir los requisitos del sistema a implementar.

Estas historias de usuario a su vez producen tareas de programación, las cuales se corresponden más exactamente con objetivos puntuales que un programador (o par de programadores) puede manejar para dar respuesta a la historia de usuario correspondiente. Una vez que se cumple una tarea, se realizan un conjunto de pruebas de unidad para asegurarse que los componentes desarrollados funcionan y una prueba de integración para probar el nuevo desarrollo en su interacción con los demás componentes previos.

Cada vez que se cumple una historia de usuario (o conjunto de historias de usuario) se presenta el software debidamente probado al cliente, para realizar las pruebas de aceptación de la funcionalidad implementada. Cuando se reúnen un número suficiente de funcionalidades que representan una versión útil y parcialmente completa de la aplicación (según lo definido en el juego de planificación de liberaciones) se produce una liberación, lo cual es una versión funcional de la aplicación que aporta valor al negocio y que debe ser mantenida a la par que se desarrollan las siguientes funcionalidades.

Las historias de usuarios definidas en la fase de exploración inicial del proyecto normalmente tributan a la primera iteración, y sirven para conformar la arquitectura de la aplicación, así como para proveer el primer lote de valor de negocio al cliente. En nuestro caso, las primeras historias de usuario fueron elegidas de forma que conformaran un sistema completamente funcional, si bien mejorable.

Las historias de usuario fueron estimadas tomando en cuenta su importancia en el negocio y la dificultad de su implementación. Ningún otro criterio fue tenido en cuenta, debido a que se considero que solo estos eran suficientes para determinar la factibilidad y la importancia de una historia de usuario. Utilizando estos datos se pudieron asignar las historias de usuario a cada una de las iteraciones de desarrollo que se definieron a continuación.

Plantilla de historia de usuario

Para definir las historias de usuario utilizamos la siguiente planilla, que contiene todos los datos necesarios para desarrollar la funcionalidad descrita.

Historia de Usuario	
Número: *Número de la 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: *Que tan importante es para el cliente*	**Riesgo en Desarrollo:** *Que tan difícil es para el desarrollador*
Iteración Asignada: *La iteración (liberación en nuestro proceso) a la que corresponde*	
Descripción: *La descripción de la historia, detallando las operaciones del usuario y opcionalmente las respuestas del sistema*	
Observaciones: *Algunas observaciones de interés, como glosario, información sobre usuarios, etc.*	

Cada historia de usuario fue asignada a una iteración, proceso el cual se discutirá con más detalle en la próxima sección. En total se llegaron a definir 28 historias de usuario en las fases de exploración de las 2 primeras iteraciones.

Historia de Usuario	
Número: 1	**Nombre:** Someter Solución.
Usuario: Todos	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Medio
Iteración Asignada: 1	
Descripción: El usuario codifica la solución al problema seleccionado en alguno de los lenguajes de programación soportados por el Jurado escoge el identificador del problema, inserta el código fuente y escoge el lenguaje de programación en el que será juzgado el problema. El sistema toma el código fuente y lo compila con el compilador correspondiente. En caso de que exista un error de compilación el sistema lo informará a través de la página de estado. Una vez compilado el código y generado el ejecutable se comprueba en dependencia de si el problema es de juzgado simple o múltiple. En caso de que el problema sea de juzgado simple se le aplicará al ejecutable un conjunto de archivos de entrada y se tomará la salida y se comprobará con la salida esperada. En caso de que sea de juzgado múltiple se le pasarán al archivo ejecutable un conjunto archivos de entrada, se tomará la salida y en conjunto con la entrada que la generó se le alimentarán a un programa al que se le delegará la responsabilidad de comprobar la correctitud de la solución. En caso de que durante la comprobación de la solución se detecte sobre la misma un error de correctitud o de cualquier otra clase se informará de esto al usuario.	

Observaciones:

Historia de Usuario	
Número: 2	**Nombre:** Administrar Competencia.
Usuario: Administrador	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Bajo
Iteración Asignada: 1	
Descripción: El usuario tiene la posibilidad de crear una nueva competencia y eliminar alguna de las competencias que estén por realizarse.	
Observaciones:	

Historia de Usuario	
Número: 3	**Nombre:** Consultar Ranking en Competencia.
Usuario: Todos	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Medio
Iteración Asignada: 1	
Descripción: El usuario puede ver el ranking de cualquier competencia pasada o en curso en el cual se mostrarán los equipos organizados por la posición que ese momento ocupaban en la competencia. De cada equipo se mostrará la cantidad de problemas que ha resuelto, la cantidad de tiempo que ha usado para darle solución a los mismos y para cada problema la cantidad de envíos que ha realizado sobre el mismo, si lo ha solucionado o no.	
Observaciones:	

Historia de Usuario	
Número: 4	**Nombre:** Consultar Estado de Sumisiones en Competencias.
Usuario: Todos	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Bajo
Iteración Asignada: 1	
Descripción: El usuario podrá ver los envíos que se han hecho durante la competencia actual organizados por fecha y de cada envió se mostrará el problema al que pertenece, el resultado de este envió, la memoria y el tiempo que se consumió en la ejecución de la solución así como el lenguaje de programación usado.	

Observaciones:

Historia de Usuario	
Número: 5	**Nombre:** Crear Usuario Local.
Usuario: Todos	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Bajo
Iteración Asignada: 1	
Descripción: El usuario puede crear un nuevo usuario local del sistema para lo cual deberá proveerle al mismo el usuario que desea usar, el *nick* por el que se le conocerá en el sistema, la contraseña que usará para acceder al sistema, así como la escuela a la que pertenece y su correo.	
Observaciones:	

Historia de Usuario	
Número: 6	**Nombre:** Autenticar Usuario.
Usuario: Todos	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Medio
Iteración Asignada: 1	
Descripción: El usuario presentara ante el sistema su id de usuario y su contraseña y el mismo comprobará la correctitud de los mismos teniendo en cuenta si es un usuario local o de dominio. En caso de que el usuario sea local la comprobación se realizara contra la Base de Datos y en caso de que sea un usuario del dominio la comprobación se realizara directamente al controlador de dominio.	
Observaciones:	

Historia de Usuario	
Número: 7	**Nombre:** Administrar Información Usuario.
Usuario: Administrador	
Prioridad en Negocio: Media	**Riesgo en Desarrollo:** Bajo
Iteración Asignada: 1	
Descripción: El usuario tiene la posibilidad de modificar cualquier información referente a cualquiera de los usuarios, tal como puede ser datos de información del mismo y otros referidos a la interacción del usuario con el sistema como pudiese ser la cantidad de problemas resueltos.	
Observaciones:	

Historia de Usuario	
Número: 8	**Nombre:** Ver información de Usuario.

Usuario: Todos	
Prioridad en Negocio: Media	**Riesgo en Desarrollo:** Bajo
Iteración Asignada: 1	
Descripción: El usuario podrá ver toda la información relacionada con su registro en el sistema, tal como id de usuario, nick, contraseña, escuela a la que pertenece y el correo electrónico del mismo.	
Observaciones:	

Historia de Usuario	
Número: 9	**Nombre:** Modificar Información de Registro de Usuario.
Usuario: Todos	
Prioridad en Negocio: Media	**Riesgo en Desarrollo:** Bajo
Iteración Asignada: 1	
Descripción: El usuario podrá modificar cualquier información relacionada con su registro en el sistema, tal como id de usuario, nick, contraseña, escuela a la que pertenece y el correo electrónico del mismo.	
Observaciones:	

Historia de Usuario	
Número: 10	**Nombre:** Comparar Usuarios.
Usuario: Todos	
Prioridad en Negocio: Media	**Riesgo en Desarrollo:** Medio
Iteración Asignada: 2	
Descripción: El usuario podrá comparar los resultados de dos usuarios y obtendrá los problemas que ha hecho solo cada uno de estos usuarios, los problemas que ambos han hecho, los problemas que cada uno de ellos ha intentado resolver y ha fallado y los problemas donde han fallado ambos.	
Observaciones:	

Historia de Usuario	
Número: 11	**Nombre:** Ver Competencias Pasadas.
Usuario: Todos	
Prioridad en Negocio: Media	**Riesgo en Desarrollo:** Medio
Iteración Asignada: 2	
Descripción: El usuario puede ver las competencias que se han efectuado en el Jurado y de cualquiera de ellas podrá ver toda la información: ranking de la competencia, problemas que componían la misma, envíos que se realizaron así como las estadísticas para cada uno de los problemas.	
Observaciones:	

Historia de Usuario	
Número: 12	**Nombre:** Ver Estadísticas de Competencias.
Usuario: Todos	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Bajo
Iteración Asignada: 1	
Descripción: El usuario podrá ver para cada una de las competencias en curso o terminadas las estadísticas totales y de cada uno de los problemas. Lo cual consistirá en los cantidades de cada uno de los tipos de resultados posibles para cada problema, así como las cantidades de envíos en cada uno de los lenguajes de programación para cada uno de los ejercicios.	
Observaciones:	

Historia de Usuario	
Número: 13	**Nombre:** Revisar Estado Sumisiones.
Usuario: Todos	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Medio
Iteración Asignada: 1	
Descripción: El usuario podrá ver los envíos que se han hecho hasta ese momento, organizados por fecha, de cada envío se mostrará el problema al que pertenece, el resultado de este envió, la memoria y el tiempo que se consumió en la ejecución de la solución así como el lenguaje de programación usado.	
Observaciones:	

Historia de Usuario	
Número: 14	**Nombre:** Consultar FAQ.
Usuario: Todos	
Prioridad en Negocio: Baja	**Riesgo en Desarrollo:** Bajo
Iteración Asignada: 2	
Descripción: El usuario podrá consultar en cualquier momento la FAQ obteniendo de esta cualquier dato que necesite sobre el funcionamiento del jurado o de los compiladores usados por este. Así como podrá obtener de la misma soluciones ejemplo para cada uno de los lenguajes de programación soportados por el jurado.	
Observaciones:	

Historia de Usuario	
Número: 15	**Nombre:** Administrar FAQ.
Usuario: Administrador	
Prioridad en Negocio: Media	**Riesgo en Desarrollo:** Bajo
Iteración Asignada: 2	
Descripción: El usuario podrá modificar los datos que se exponen en la FAQ con	

cualquier motivo, el que pudiese ser la inclusión o eliminación de un leguaje de programación al jurado.
Observaciones:

Historia de Usuario	
Número: 16	**Nombre:** Consultar Ranking de Autores.
Usuario: Todos	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Medio
Iteración Asignada: 1	
Descripción: El usuario podrá ver los autores ordenados por la cantidad de problemas que tienen resueltos, y de cada uno de ellos podrá ver la cantidad de problemas resueltos, la cantidad de envíos que tiene y la razón entre problemas resueltos y envíos realizados por cada usuario.	
Observaciones:	

Historia de Usuario	
Número: 17	**Nombre:** Consultar Problema.
Usuario: Todos	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Medio
Iteración Asignada: 1	
Descripción: El usuario podrá ver el texto correspondiente al problema en el que se expone la descripción del mismo, el formato de entrada y el de salida así como un ejemplo de cada uno de ellos. Adicionalmente se mostrara las cantidades máximas de memoria, tiempo y tamaño de código a usar por la solución.	
Observaciones:	

Historia de Usuario	
Número: 18	**Nombre:** Consultar Estado de un Problema.
Usuario: Todos	
Prioridad en Negocio: Media	**Riesgo en Desarrollo:** Medio
Iteración Asignada: 1	
Descripción: El usuario podrá ver para un problema determinado los usuarios que lo han resuelto de forma satisfactoria, ordenados el tiempo usado por su solución. Además se mostrara un grafico de pastel con los porcientos de cada uno de los tipos de resultados que han recibido los usuarios para este problema.	
Observaciones:	

Historia de Usuario	
Número: 19	**Nombre:** Consultar Ranking Reciente de Autores.

Usuario: Todos	
Prioridad en Negocio: Media	**Riesgo en Desarrollo:** Alto
Iteración Asignada: 2	
Descripción: El usuario podrá ver el listado de usuarios que han enviado soluciones entre las ultimas N, siendo N un número introducido por el usuario. Estos autores aparecerán ordenados por la cantidad de problemas que tienen resueltos, y de cada uno de ellos podrá ver la cantidad de problemas resueltos, la cantidad de envíos que tiene y la razón entre problemas resueltos y envíos realizados por cada usuario.	
Observaciones:	

Historia de Usuario	
Número: 20	**Nombre:** Administrar Problemas.
Usuario: Administrador	
Prioridad en Negocio: Media	**Riesgo en Desarrollo:** Bajo
Iteración Asignada: 1	
Descripción: El usuario podrá adicionar y eliminar problemas al sistema pudiendo escoger si estos pertenecen a una competencia que está planificada o si simplemente pertenece al volumen general de problemas.	
Observaciones:	

Historia de Usuario	
Número: 21	**Nombre:** Consultar Volúmenes de Problemas.
Usuario: Todos	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Bajo
Iteración Asignada: 1	
Descripción: El usuario podrá ver el listado de los volúmenes de problemas con los que cuenta el sistema y de cada uno de estos podrá listar los problemas que lo componen de los que además del nombre del mismo podrá ver la cantidad de usuarios que han aceptado dicho problema, la cantidad de envíos que se han realizado sobre el mismo, así como su dificultad.	
Observaciones: La dificultad de un problema se calculara mediante la razón entre la cantidad de	

usuarios que ha aceptado el ejercicio con la cantidad de envíos que ha tenido este.

Historia de Usuario	
Número: 22	**Nombre:** Administrar anuncios importantes.
Usuario: Administrador	
Prioridad en Negocio: Baja	**Riesgo en Desarrollo:** Bajo
Iteración Asignada: 2	
Descripción: El usuario podrá activar y desactivar los anuncios importantes, además podrá cambiar el texto de dicho anuncio.	
Observaciones:	

Historia de Usuario	
Número: 23	**Nombre:** Administrar Compiladores.
Usuario: Administrador	
Prioridad en Negocio: Media	**Riesgo en Desarrollo:** Alto
Iteración Asignada: 1	
Descripción: El usuario podrá adicionar o eliminar lenguajes de programación al jurado adicionando o eliminado compiladores.	
Observaciones:	

Historia de Usuario	
Número: 24	**Nombre:** Gestionar Correo Interno.
Usuario: Todos	
Prioridad en Negocio: Media	**Riesgo en Desarrollo:** Medio
Iteración Asignada: 1	
Descripción: El usuario podrá enviar correos a otros usuarios, leer los correo que otros usuarios le han escrito o eliminarlos.	
Observaciones:	

Historia de Usuario	
Número: 25	**Nombre:** Utilizar Foro.
Usuario: Todos	
Prioridad en Negocio: Medio	**Riesgo en Desarrollo:** Bajo
Iteración Asignada: 2	
Descripción: El usuario podrá usar un foro de discusión para crear temas o responder en temas existentes, relacionados con el Jurado, las competencias de programación, los algoritmos, o cualquier otro tema que resulte de interés para la	

comunidad.
Observaciones:

Historia de Usuario	
Número: 26	**Nombre:** Autenticación contra dominio LDAP.
Usuario: Todos	
Prioridad en Negocio: Bajo	**Riesgo en Desarrollo:** Alto
Iteración Asignada: 2	
Descripción: El sistema debe permitir elegir entre dos sistemas de autenticación, local o utilizando un dominio LDAP existente. De esta forma es más fácil usarlo en intranets empresariales. El cambio entre estas configuraciones debe ser externo y fácil, sin necesidad de modificar la aplicación.	
Observaciones: Esta opción se desarrolló debido a las circunstancias de encontrarnos con un dominio LDAP. Su principal ventaja en una intranet cerrada es la posibilidad de evitar duplicación de usuarios y proliferación de los mismos.	

Historia de Usuario	
Número: 27	**Nombre:** Juzgado Especial.
Usuario: Todos	
Prioridad en Negocio: Alto	**Riesgo en Desarrollo:** Alto
Iteración Asignada: 2	
Descripción: El sistema debe permitir además de la forma normal de juzgar los problemas, una opción de juzgado especial, consistente en un programa que interactúe con la solución del usuario y lo someta a una prueba particular de correctitud. Esta opción permite problemas que pueden tener múltiples soluciones y sienta las bases para los problemas de tipo *challenge*.	
Observaciones: El juzgado especial es una opción importante de los jurados modernos. Abre el camino a los problemas *challenge* y a una nueva dinámica de competencias en el futuro.	

Historia de Usuario	
Número: 28	**Nombre:** Sistema de puntuación dinámico.
Usuario: Administrador	

Prioridad en Negocio: Medio	**Riesgo en Desarrollo:** Alto
Iteración Asignada: 2	
Descripción: El sistema debe permitir de forma opcional, un sistema de puntuación que funcione de forma dinámica, variando los puntos ganados por cada problema en función de la cantidad de personas que lo resuelva. Esto permite un listado de ranking más adecuado, dado que las personas con más puntos no los logran por tener más problemas, sino por tener los problemas más difíciles: $$puntuacion\_problema = \frac{200}{40 + usuarios\_resueltos}$$ Lo cual trae como resultado una puntuación base de 5 puntos por problema, la cual va bajando paulatinamente cada vez que un usuario nuevo resuelve el problema. La puntuación baja para todos, asegurando que lo que se mide sea la dificultad y no el momento en que se hizo el problema.	
Observaciones: Este sistema garantiza que el ranking sea más justo para todos. Como efectos laterales tiene dos, ambos negativos: Puede tener un efecto de aumento de la competitividad y por tanto una disminución en el volumen de información intercambiado entre los usuarios. Por otro lado, el costo asociado con las operaciones necesarias para actualizar el ranking de puntuación puede ser alto para ciertos problemas particulares. Estos problemas se calificaron como compromisos aceptables, debido a que la competitividad es un problema que existía de todas formas, y la carga de cálculo del ranking se decidió realizarla una vez al día para todos los problemas.	

Estimación y planificación

En XP, las métricas son libres, pudiendo utilizarse cualquier criterio para medir el desarrollo del proyecto. Una métrica popular es la razón entre tiempo ideal de trabajo y tiempo real de trabajo, que sirve para determinar la velocidad del proyecto. Se recomiendan de 3 a 4 métricas a la vez, las cuales se pueden sustituir cada vez que sea necesario.

Sin embargo, en nuestro proyecto decidimos no mantener ninguna métrica, asumiendo conscientemente el riesgo que implica el no poder definir la velocidad con la cual se está desarrollando. La razón para asumir tal riesgo es que la métrica de tiempo no representaría utilidad, dado que al realizar la estimación de tiempo de cada historia de usuario la velocidad del proyecto resultante fluctuaría en todo momento, dando como resultado que la métrica sería inútil para establecer el rendimiento real del equipo de trabajo. En este pobre resultado de la medición influyen algunos de los factores ambientales que determinaron anteriormente las demás decisiones tecnológicas y metodológicas.

La falta de métricas conlleva un riesgo grande de no cumplir con los plazos de entrega del software (en XP, plan de liberaciones). Asumimos el riesgo, y lo mitigamos lo más posible trabajando en las tareas de programación definidas por orden de importancia para el negocio, y manteniendo el sistema en producción todo el tiempo, lo cual reduce efectivamente el tiempo de las pruebas de aceptación (las cuales se hacen al mismo tiempo que se implementan otras funcionalidades). Otras medidas incluyen:

- **Mantenimiento del control de versiones**: Lo cual permite regresar rápidamente a versiones anteriores de la aplicación en producción.
- **Monitoreo constante del rendimiento de la aplicación**: Lo cual permite descubrir bugs en el comportamiento de la aplicación de forma rápida, preferentemente incluso antes de que los usuarios sean afectados.
- **Uso regular de todas las funcionalidades de la aplicación**: Lo cual permite detectar pudriciones potenciales del software debidas a la adición de funcionalidades nuevas o la refactorización constante del código existente.
- **Simplificación constante del código y externalización de la configuración**: Lo cual permite aislar el cambio en la menor cantidad de lugares de forma que sea más sencillo de controlar.

A pesar de que parezca que la estimación ha sido ignorada, tal no es el caso. Lo único de lo que se prescinde es de las métricas y su función de controlar la velocidad del proceso de desarrollo. Sin embargo otros factores de interés para el proceso de planificación como son la importancia de las funcionalidades para el negocio, la dificultad técnica de las historias de usuario, su impacto en el estado de producción actual, etc. sí son tenidos en cuenta a la hora de ordenar las historias de usuario y de decidir que funcionalidades implementar. De hecho, las historias implementadas son solo las que se consideraron posibles de implementar dadas las condiciones previsibles del desarrollo, las demás no se consideraron para inclusión en este documento.

En XP se realizan dos actividades de planificación fundamentales, la planificación de liberaciones, donde participa el cliente de forma decisiva, y la planificación de la iteración, donde participan los desarrolladores definiendo las tareas, estimándolas, comprometiéndose e implementándolas.

Planificación de liberaciones

El plan de liberaciones de nuestro desarrollo se estructuró en torno a 3 hitos fundamentales, donde cada liberación se correspondió con una iteración de desarrollo:

Tabla 3: Plan de liberaciones de la aplicación

Entregables	Liberaciones		
	1/5/2008	1/6/2008	1/7/2008
Jurado Online	v2.0	v2.1	v2.5 final
Foro	v1.0 final	finalizado	finalizado
Repositorio de Bibliografía		v1.0	v2.0 final
Instalador			v1.0 final
CMS Xtreme Programming		v1.0 final	finalizado

Cada liberación ocurre normalmente luego de varias iteraciones, en el transcurso de las cuales se cumplen las historias de usuario correspondientes a cada liberación. A su vez, cada historia de usuario se compone de varias tareas, además de existir tareas que no soportan ninguna historia en particular, sino que tienen como objetivo un mejor funcionamiento de la aplicación como un todo.

En nuestro proceso cada liberación está constituida por una iteración solamente, dado que más no son necesarias debido a la planificación de grano grueso utilizada de forma general. Además, la particularidad de que tanto el negocio como el desarrollo estén representados por las mismas personas hace que no sea necesario una negociación constante de ámbitos ni estimaciones en la planeación de iteraciones, dado que Negocio y Desarrollo están completamente de acuerdo y completamente informados todo el tiempo. Planificar y seguir el desarrollo es mucho más fácil y por tanto no es necesario hacer planificaciones de grano muy fino, que de todas formas no serian útiles debido al ambiente caótico en que se desarrolla el proyecto.

Especialmente la primera iteración es atípica según lo que dicta la metodología XP en lo referente a la duración de las iteraciones. Como se ve en la planificación, tanto la iteración 2 como la 3 constan de 3 semanas de desarrollo precedidas de una semana de exploración y conceptualización para cada una. Sin embargo la primera iteración se prolongó por más de 3 meses de forma deliberada, hasta que todas las historias de usuario que conformasen la aplicación con las funcionalidades necesarias estuviesen implementadas. Se tomó esta decisión dado que no tenía sentido liberar software que no estaba completo funcionalmente hablando. Sin embargo cada historia de usuario fue probada a medida que se terminaba sustituyendo el código correspondiente a ella en la aplicación prototipo, lo cual consideramos mitiga el riesgo corrido al retardar la entrada del sistema en producción. Una vez lograda la primera liberación (1/5/2008) se hizo con la absoluta certeza de que todos los componentes desarrollados funcionaban correctamente, gracias a la estrategia de pruebas constantes permitida por el prototipo.

La planificación de las liberaciones también se realizó atendiendo a un criterio particular: La completitud de la versión. Debido a las condiciones ambientales particulares de desarrollo de la aplicación, se hizo necesario definir estados en los cuales la aplicación pudiera congelarse un tiempo indefinido de ser necesario. Cada uno de estos estados o versiones puede mantenerse en producción de forma autosuficiente por un intervalo de tiempo ilimitado, lo cual sirve de mecanismo de contingencia por si es necesario detener el desarrollo durante semanas o incluso meses.

Tabla 4: Planificación de historias de usuario por iteración (iteración 1)

Historias de usuario	
Someter Solución	Consultar Estado de Sumisiones en

	Competencias
Administrar Competencia	Crear Usuario Local
Consultar Ranking en Competencia	Autenticar Usuario
Modificar Información de Registro de Usuario	Ver información de Usuario
Administrar Información Usuario	Consultar Ranking de Autores
Revisar Estado Sumisiones	Ver Estadísticas de Competencias
Consultar Problema	Consultar Estado de un Problema
Administrar Problemas	Consultar Volúmenes de Problemas
Administrar Compiladores	Gestionar Correo Interno

Tabla 5: Planificación de historias de usuario por iteración (iteración 2)

Historias de usuario	
Consultar Ranking Reciente de Autores	Comparar Usuarios
Ver Competencias Pasadas	Consultar FAQ
Administrar FAQ	Sistema de puntuación dinámico
Administrar anuncios importantes	Juzgado Especial
Utilizar Foro	Autenticación contra dominio LDAP

Nótese que existen historias de usuario planificadas solo para las dos primeras iteraciones. Esto es debido a que el plazo de entrega del documento se cumplió antes de lo previsto, y no fue posible reflejar en él la planificación de la 3ra iteración original.

Artefactos de implementación

A pesar de que la metodología utilizada no define ningún artefacto del proceso de desarrollo para visualizar el progreso del sistema, algunos autores encuentran útil mantener algunos artefactos, siempre y cuando el tiempo dedicado a mantenerlos sea mucho menor que el tiempo dedicado al desarrollo y que la utilidad que brindan. Bajo estos criterios hemos decidido mantener tres diagramas que pueden ser útiles para una mejor compresión del funcionamiento de la aplicación y que representan artefactos cuya probabilidad de cambio es pequeña:

- Modelo de datos
- Sistema de archivos
- Compiladores

Los artefactos así representados no han sufrido cambios significativos durante todo el tiempo de desarrollo del software, las historias de usuario contempladas para un futuro próximo no prevén que sea necesario un cambio radical de la estructura de ninguno, ni siquiera del modelo de datos.

Modelo de datos

Modelo de datos de la versión 2.1. Este modelo está basado en un motor MyISAM de MySQL, para lograr mayor velocidad a costa de no mantener la integridad referencial en la base de datos, sino desde la aplicación.

Las tablas más pesadas en cuanto al número de tuplas son: *users*, *problem*, *solution* y *source_code*, en ese orden.

Muchos de los campos de las tablas son en realidad valores calculados. Estos se actualizan para optimizar las lecturas, dado que hay una frecuencia mucho mayor de lecturas que de escrituras. Esta es otra ventaja de mantener la integridad referencial desde la aplicación.

El modelo de datos ha demostrado ser muy estable, dado que luego de la primera refactorización necesaria para adaptar el legado a los cambios en el código de la aplicación solo ha sufrido pequeños cambios, casi todos adiciones.

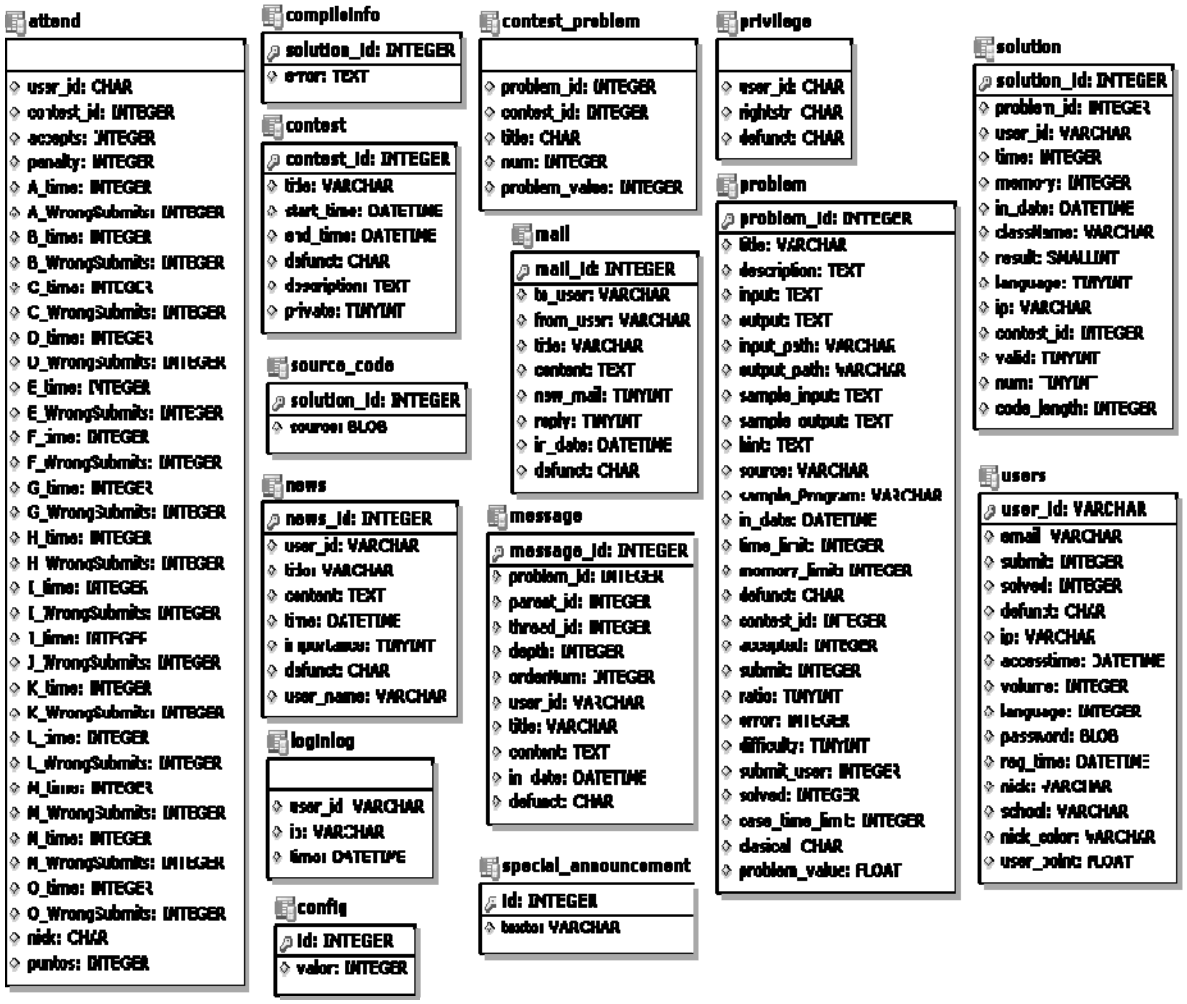


Imagen 16.Modelo de datos. Nótese los campos calculados, embebidos en las tablas para lograr velocidad.

Sistema de archivos

El sistema de archivos es un componente importante de la aplicación. La estructura es determinada y específica, con cada componente cumpliendo una función única.

Nótese que esta estructura de carpetas es propia de Windows, que el software actual en el servidor. Sin embargo debe ser válida también en Linux, debido al hecho de que toda la estructura esté dentro de una carpeta raíz, lo que facilita la migración.

Imagen 17.Estructura del sistema de archivos del jurado online.

Entre los archivos existen 3 secciones determinadas cada una con una función propia, las cuales son las siguientes:

- **bin:** En esta carpeta se colocan los ejecutables de los compiladores que soportan los lenguajes del jurado. Nótese que estos ejecutables se corresponden con los lenguajes compilados. Para los lenguajes que producen código intermedio existe solo una configuración en el archivo `server.property`, por ejemplo para Python y Java. Una excepción notable a esta regla lo constituye el compilador de C#, dado que el C# produce archivos ejecutables. De todas formas es necesario que esté instalado el framework .NET en versión 2.0 o superior para poder ejecutar las soluciones compiladas. Para Python y Java también es necesario que estén instalados el compilador de Python y la máquina virtual de Java. En la versión actual de la aplicación se utilizan el Python 2.4 y la JDK 1.5.

- **data:** En esta carpeta se colocan todos los problemas del jurado numerados a partir del 1000 en adelante. Cada carpeta corresponde a un problema, y el número que la identifica es el número que identifica al problema en el volumen. Dentro de cada una de estas carpetas se encuentran los juegos de datos correspondientes al problema. Tradicionalmente también puede encontrarse la solución de ejemplo del problema con la que se generaron los juegos de datos para probar las demás. Esta solución es importante porque representa la garantía de que la solución es correcta y que las variables ajustables de tiempo y memoria máximas son realistas. La aplicación ignora cualquier archivo que se encuentre en estas carpetas que no tenga la convención de nombre de "`dataX. (in|out)`" donde X es un número. Para cada archivo `in` debe existir un archivo `out` correspondiente.
- **temp:** En esta carpeta se encuentran las últimas sumisiones hechas por los usuarios, cada una en una carpeta aparte identificada con el número de la sumisión. Dentro de esta carpeta se encuentra el código fuente, el producto compilado o interpretado según el lenguaje y las salidas producto de la ejecución de la solución. Estas carpetas son útiles para re juzgar soluciones específicas a mano, probar las restricciones temporales para lenguajes costosos (como Java y C#), etc. La desventaja es que esta carpeta puede crecer mucho en poco tiempo, y debe ser vaciada cada cierto tiempo. La aplicación cuenta con una configuración para mantener solo las N últimas soluciones. El valor de esta configuración depende de la confianza que se tenga en las restricciones de los problemas y de la cantidad de espacio en disco que es posible utilizar sin comprometer el rendimiento del servidor.

server.property

El sistema de archivos cuenta también con el archivo `server.property`, la fuente principal de las configuraciones de la aplicación. Este es un archivo `Properties` estándar de Java que se utiliza para especificar los parámetros de inicio de la aplicación. Las principales configuraciones que maneja son:

- **Acceso a datos:** la configuración necesaria para conectarse a la base de datos (usuario, contraseña, dirección del host y nombre de la BD). También incluye otras configuraciones útiles como el tamaño de la piscina de conexiones de la aplicación.
- **Acceso a los shells de compilación y ejecución:** Las direcciones absolutas en el sistema de archivos de los programas utilizados para compilar, ejecutar y evaluar las soluciones de los usuarios. Estos programas son externos al jurado y pueden ser sustituidos en cualquier momento.
- **Información de los compiladores:** La información de los compiladores incluye su dirección en el sistema de archivo, el tipo de archivo que producen como salida, la extensión de dicho archivo y una configuración especial llamada factor de velocidad, que determina si un lenguaje determinado tiene que ser juzgado según criterios especiales a la hora de calcular el tiempo que se toma para resolver un problema. Esto puede ser útil para equiparar la competencia en lenguajes altamente eficientes como C++ o Pascal contra lenguajes lentos como Java o C#.

- **Información de administración:** incluye el correo del administrador, el nombre del servidor de la aplicación, la dirección del foro de usuarios, etc.

Compiladores

Los compiladores proveen el soporte para los lenguajes en la aplicación. No todos los lenguajes constan de un compilador, como por ejemplo Java o Python. Estos lenguajes tienen una configuración diferente por completo que también abordaremos.

Compiladores actuales

Para la versión 2.1 del jurado se cuenta con 7 lenguajes, 5 de ellos compilados:

Tabla 6: Compiladores disponibles para la versión 2.1

Lenguaje	Compilador	Compilado
C	GCC	✓
C++	G++	✓
Pascal	Free Pascal Compiler	✓
C#	C Sharp Compiler*	✓
Java	JDK 1.5.05	
Python	Python 2.4	
Texto	TextCompiler**	✓

** Este compilador fue desarrollado especialmente para el jurado y usa .Net 2.0 para la compilación.*
*** Este compilador también fue diseñado específicamente para el jurado usando C#. Actualmente estamos en medio de un estudio para sustituirlo por una implementación más adecuada, debido a que existe un inconveniente en lo que respecta al tiempo calculado para las sumisiones.*

Los próximos lenguajes contemplados para inclusión son Perl, Ruby y Prolog.

Pruebas

Pruebas de caja blanca

XP se precia de ser un proceso afín al TDD (*Test Driven Development*), y hace mucho énfasis en las pruebas tanto de unidad, como de integración y aceptación, aunque permite cualquier otro tipo. Un programador extremo debe siempre probar su código de forma unitaria, integrar continuamente y hacer prueba de aceptación con cierta regularidad, cada vez que una funcionalidad esta lista.

En nuestro proceso se cumplen satisfactoriamente estas tres exigencias de XP. Todo el código se prueba primero en aislamiento, se integra mediante el control de versiones en cada máquina de desarrollo y una vez que pasan las pruebas de integración se añade cuidadosamente al prototipo funcional en producción, para que los usuarios realicen las pruebas de aceptación, manteniendo siempre una copia de la versión anterior en reserva, por los problemas que puedan presentarse. En caso de detectarse algún error o queja, se vuelve a la versión anterior y se arregla la no conformidad. Dada la idiosincrasia del público objetivo de este tipo de aplicaciones, los usuarios tienden a buscar funcionalidades nuevas y rendimiento, más que a criticar diseño o accesibilidad. Esto es favorable al proceso porque hace que las pruebas de aceptación tengas más probabilidades de aceptarse que de rechazarse.

La estrategia de pruebas está a tono con las condiciones de trabajo, en el sentido de que tanto las pruebas de aceptación como de integración son casi indoloras gracias al SVN y a la base de usuarios de la aplicación en producción. Para hacer que el éxito de las pruebas de integración sea aún más probable nunca se trabajan dos funcionalidades relacionadas al mismo tiempo, de forma que se eviten lo más posible los conflictos en el código base.

Para las pruebas unitarias ayuda mucho otra regla de XP referente a “la cosa más simple que funcione” (*The simplest thing that could possibly work*). La aplicación directa de este principio se traduce en que el código puede ser probado, o sea que las funcionalidades se implementan a nivel de funciones, no de estructuras de objetos, y lo más independientes posible (si bien en general no es posible llevar este precepto muy adelante). Esto hace que cada funcionalidad sea fácil de probar tanto por JUnit como por casos de prueba escritos a mano, métodos ambos usados para las pruebas unitarias. El hecho de que la aplicación sea muy ligera también posibilita pruebas de la aplicación en su totalidad durante el proceso de

desarrollo, lo que puede ser útil en el caso de funcionalidades polimórficas (que afectan a varias secciones del desarrollo de forma diferente, por ejemplo, la seguridad o la interface) que no son fácilmente verificables con JUnit.

Pruebas de caja negra

Además de probar el código fuente mediante pruebas de unidad, y pruebas de integración, también fue necesario realizar otras pruebas, nominalmente pruebas de carga y de estrés.

Los objetivos de tales pruebas son claros:

- Estimar la concurrencia máxima esperada de la aplicación.
- Estimar la concurrencia máxima de las transacciones sobre la base de datos.
- Determinar un conjunto de configuraciones válidas del servidor de datos y el servidor de aplicaciones, que maximice dentro de los límites del hardware la respuesta de la aplicación.
- Localizar e identificar errores que la aplicación pueda tener.

En los casos en los cuales el dilema queda entre memoria y tiempo de respuesta, siempre se favorece el tiempo de respuesta, dado que según la arquitectura de la aplicación no se utiliza mucha memoria pero si es necesario tener un tiempo de respuesta rápido para soportar concurrencia alta de usuarios conectados o anónimos..

JMeter

La herramienta utilizada para estas pruebas es el Apache JMeter, un software especializado en hacer pruebas de carga para aplicaciones, especialmente Web. El JMeter ofrece una gran variedad de pruebas para aplicaciones, incluyendo peticiones HTTP, SOAP y JDBC. La arquitectura del software es jerárquica en forma de árbol, donde cada elemento hereda los valores de los elementos similares en niveles superiores. Existen muchos tipos de elementos de prueba, y cada uno tiene un rol específico en el plan de pruebas.

Imagen 18.Logo del JMeter

Para nuestro proyecto utilizamos el JMeter para probar la estabilidad y concurrencia soportada por la aplicación, antes que la velocidad de respuesta. Se consideró que la estabilidad de la aplicación ante

grandes cantidades de peticiones es más importante que la velocidad con que se responden dichas peticiones, debido a que la fiabilidad de la aplicación no puede comprometerse en casos especiales, tales como durante la ejecución de un proceso de compilación o la realización de una competencia, mientras que la velocidad es un factor secundario en este punto. Usar el JMeter de esta forma tiene implicaciones en la forma en que se diseñaron los planes de prueba.

Al ser demasiados elementos y no constituir el JMeter el objetivo de este trabajo, nos limitaremos a explicar al paso los elementos utilizados en el plan de pruebas:

- **Plan de pruebas:** Un plan de pruebas consiste de uno o más elementos de configuración organizados en una estructura jerárquica de forma que simule un grupo de usuarios navegando la aplicación en cuestión.
- **Proxy HTTP:** Elemento de prueba que permite grabar una prueba usando un navegador web estándar. Mediante esta prueba un usuario puede navegar un sitio o aplicación web, y grabar todas sus peticiones .De esta forma luego se puede reproducir todo este flujo de navegación de forma automática, lo cual produce una prueba lo más cercana posible a la realidad.
- **Grupo de hilos:** Elemento que simula un grupo de usuarios, uno por cada hilo. Permite también configurar otras propiedades de la prueba como intervalo de inicialización, peticiones por usuario y momento en que va a comenzar la prueba.
- **Controladores:** Los controladores son los elementos primarios cuya función es contener los elementos de prueba y agruparlos en unidades lógicas. Los controladores sirven para controlar una prueba a un vínculo particular, así como para definir comportamientos especiales según el tipo de prueba que quiera hacerse. Una de las mayores fortalezas del JMeter, su flexibilidad a la hora de definir las pruebas, viene dada precisamente por la capacidad de los controladores de contenerse unos a otros y de afectar el comportamiento de todos los elementos dentro de sí mismos.
- **Petición HTTP:** El componente básico de una prueba Web, la petición HTTP es la piedra básica con la que se construye una prueba. Constituye un *request* HTTP específico, y puede contener encabezados del navegador (otro elemento de configuración del JMeter, que se genera automáticamente en las pruebas grabadas) y heredar propiedades de otros elementos de configuración especializados que se encuentren por encima en la jerarquía del plan de pruebas. Permiten especificar muchas cosas, desde el protocolo hasta la información que se pasa por parámetros en la petición.
- **Elemento de configuración HTTP:** Estos elementos permiten compartir la configuración común a muchas peticiones HTTP, como pueden ser nombre de host, puerto, protocolo, etc. Si alguna petición en particular quiere beneficiarse solo de algunos de estos datos, puede utilizar la configuración y sobrescribir los datos necesarios.

- **Aserción de respuesta:** Este elemento considera una prueba correcta o fallida según un criterio definido. Normalmente se definen código de respuesta HTTP o tiempos mínimos de respuesta. En nuestro caso se definió como criterio de evaluación de las aserciones el código 200, lo cual significa que la página pedida se retornó correctamente.

Un punto importante a tener en cuenta a la hora de interpretar los resultados de una prueba de estrés es la configuración del hardware y el software utilizados. Para las pruebas realizadas se utilizaron los mismos hardware y software de producción actuales consistentes en Windows Server 2003 para el sistema operativo, la máquina virtual: JDK 1.5 y un servidor de aplicaciones Tomcat 5.5 para el software, corriendo sobre una PC Pentium 4 con CPU a 3.00 GHz con 256 Mb de memoria RAM. Nótese que este hardware es insuficiente para hospedar satisfactoriamente la aplicación con total seguridad, sobre todo porque el servidor no es dedicado, sino que es utilizado en el trabajo normal del departamento de programación a la vez que como huésped de la aplicación web y el servidor de datos.

Al principio de un plan de pruebas se identifican junto a los criterios que definen la prueba, los vínculos que serán probados. En nuestro proyecto se identificaron tres páginas fundamentales que concentran el grueso de la concurrencia pesada (o sea, con procesamiento serio) de la aplicación. En JMeter se realizaron dos planes de pruebas (configuraciones de las pruebas) que utilizan estas páginas, cada uno con un objetivo en particular. Los planes sencillos de JMeter fueron puestos deliberadamente muy por encima de las observaciones que representan el estado real de la aplicación:

- **Plan de pruebas de concurrencia:** Un plan de pruebas consistente de 200 hilos, con 5 peticiones cada una en un intervalo de subida de un segundo. La prueba consistió en un controlador simple con una request pesada, consistente en una sumisión de código fuente al Jurado Online de un ejercicio aceptado. Esta prueba constituyó la base para estimar la cantidad de sumisiones simultáneas que puede aguantar el jurado. Nótese en este caso que debemos tener en cuenta que un mismo usuario, por restricciones de seguridad, no puede hacer sumisiones en intervalos menores de 3 segundos. Esta prueba dio resultados satisfactorios en cuanto a estabilidad, aunque arrojó también como resultado que el jurado se demora un tiempo relativo a la cantidad de sumisiones en procesar los códigos sometidos. Este comportamiento es natural y no se consideró erróneo.
- **Plan de pruebas de carga:** Para este plan se utilizaron las páginas de ranking total de usuarios y de estado del jurado, las cuales realizan gran número de operaciones para cada petición. Estas páginas contienen un volumen de procesamiento respetable, además de que son importantes para todos los usuarios y se espera tenga mucha concurrencia lo mismo de usuarios registrados que anónimos. Para

este plan se utilizó un grupo de hilos de 20 usuarios, cada uno con 100 peticiones, a iniciar en un intervalo de 1 segundo. La prueba consistió en un controlador aleatorio que elige al azar una de las dos páginas para navegar en cada petición (por tanto totalizan 2000 peticiones entre las dos, pero no necesariamente 1000 y 1000). El plan de prueba se enfocó en que la aplicación pudiera servir la página pedida de forma satisfactoria, sin tomar en cuenta el tiempo que se tomara. La aplicación respondió satisfactoriamente a este requerimiento, si bien requirió configurar el umbral de conexiones del servidor MySQL para que aceptara más conexiones. Posteriormente se realizó un segundo plan de pruebas sobre la aplicación con una configuración diferente, el cual detallaremos más adelante.

En resumen, las pruebas dieron como resultado que la aplicación soporta una concurrencia suficiente para el tráfico observado (promedio 12 sumisiones al día, 5 usuarios, pico 65 usuarios, más de 100 conexiones en competencia). Es razonable suponer que la estabilidad de la aplicación no está comprometida por el tráfico que puede encontrarse en la Universidad, y por extensión en cualquier intranet cubana.

Para lograr dicha estabilidad fue necesario configurar algunos parámetros del servidor de aplicaciones y de datos. Estas configuraciones son consistieron básicamente en aumentar el número de conexiones permitidas y el número de hilos mantenidos por el servidor de MySQL. Estas configuraciones permiten una ganancia en velocidad de respuesta, confiabilidad y estabilidad. Otras configuraciones, aunque no probadas son también posibles con el objetivo de aumentar la respuesta del servidor a altos volúmenes de datos, como el cache de respuesta, la memoria temporal y la cantidad de consultas abiertas por conexión. Sin embargo el volumen de datos manejados en el presente no justifica el peso de estas configuraciones en la memoria del servidor, que es un recurso crítico.

Con esta configuración se asegura la respuesta y estabilidad de la aplicación en el contexto universitario con un margen de confianza casi absoluto.

Ejemplo de plan de pruebas de carga

Además de los dos planes de pruebas mencionados anteriormente, también se implementó un segundo plan de pruebas de carga, con el objetivo de probar la estabilidad de la aplicación con un comportamiento menos cercano al comportamiento de un humano, y por tanto más directo a la hora de estresar los recursos de la aplicación.

El plan se compone de un grupo de hilos con 100 usuarios, cada uno iniciando 50 peticiones con un período de inicio de 1 segundo. Cada petición consiste en 4 controladores, cada uno con una o varias peticiones HTTP para un total de 6 peticiones sobre las páginas de **listado de usuario**, **estado actual del jurado**, **volumen de problemas**, **descripción del problema** y **estado del problema**. Estas cinco páginas son consideradas las más visitadas de toda la aplicación y también agrupan una parte importante de todo el procesamiento de información. En total se ejecutaron 18000 peticiones sucesivas sobre la aplicación.

En la imagen siguiente puede verse el resultado de la prueba, utilizando un componente *Aggregate Graph* de JMeter:

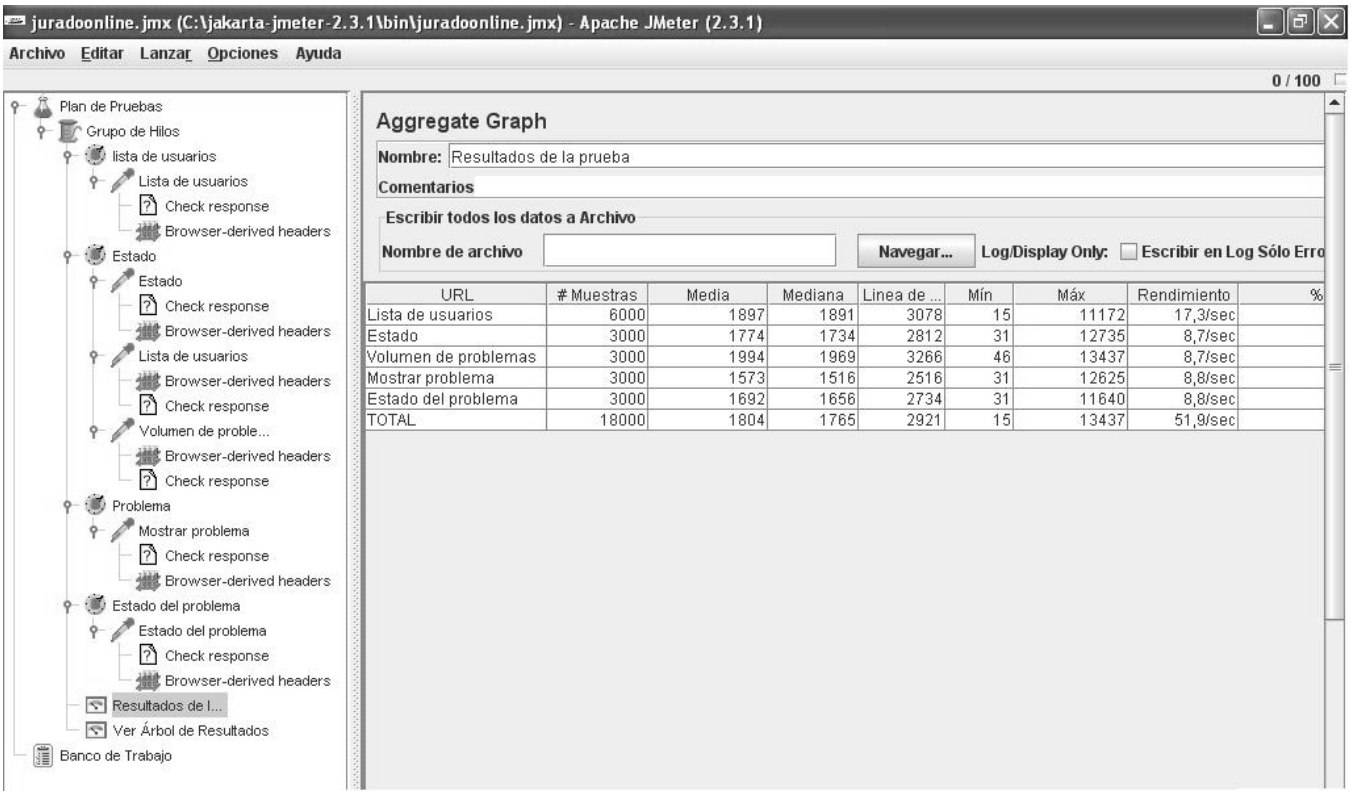


Imagen 19.Plan de pruebas de carga

El componente *Aggregate Graph* también produce una salida gráfica, que puede resultar más informativa en muchos casos. Estas gráficas representan información sobre distintos criterios, brindando cada uno información valiosa. Los criterios elegidos para generar las gráficas fueron: Tiempo mínimo de respuesta, tiempo máximo de respuesta, tiempo medio de respuesta y línea del 90%. Este último representa el mayor tiempo consumido por una petición que se encuentra en el 90% más rápido del conjunto total de peticiones, es decir, una buena aproximación al tiempo que es posible que un usuario real deba esperar cuando esté utilizando el sitio.

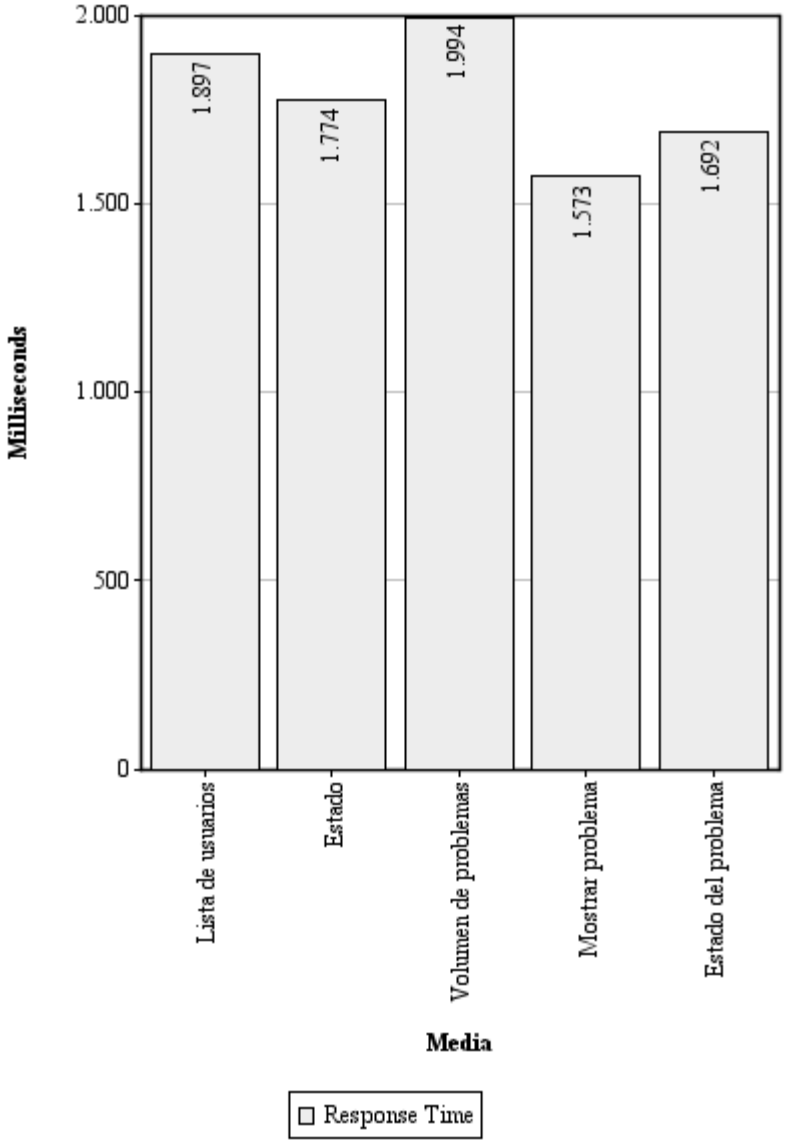


Imagen 20.Tiempo promedio de respuesta.

La gráfica representa el tiempo promedio de respuesta de la aplicación para cada una de las peticiones. Nótese que el tiempo promedio siempre está por debajo de los 2 segundos. La página más lenta en promedio resulta ser el volumen de problemas con 1994 milisegundos.

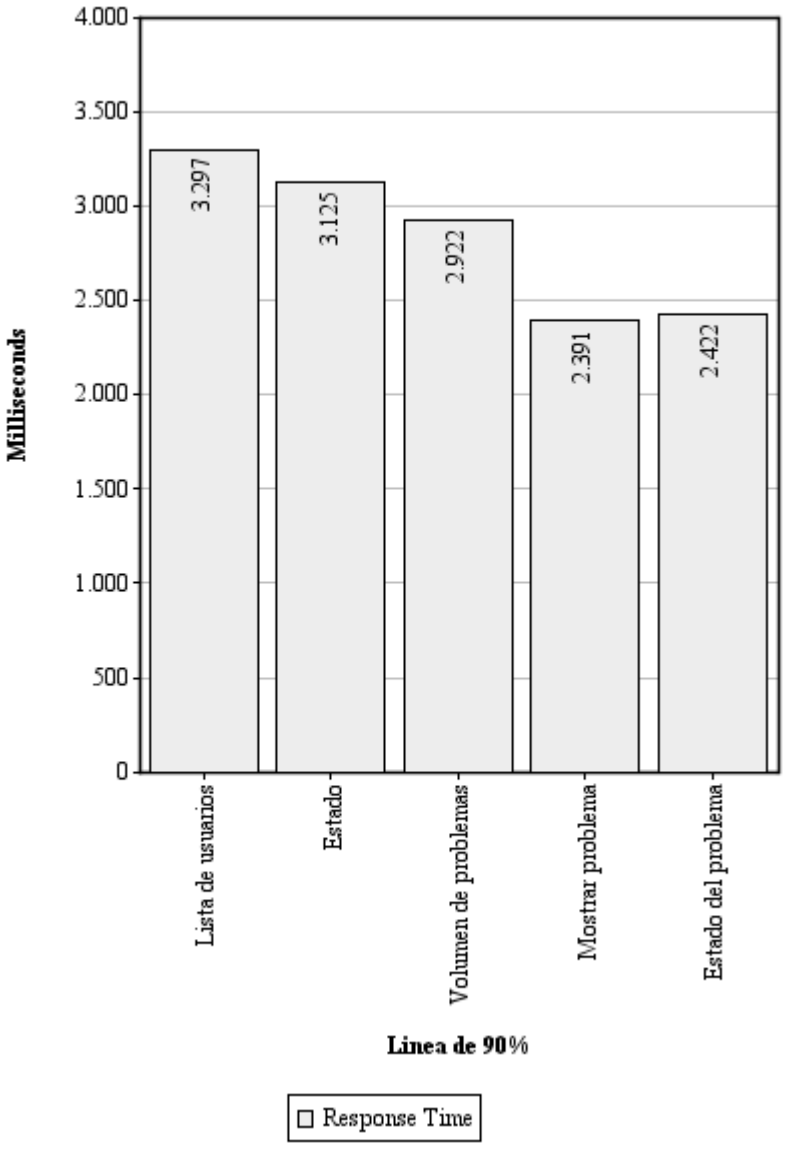


Imagen 21. Línea del 90% de las peticiones

La gráfica de línea de 90% indica que el 90% de las peticiones realizadas terminan en menos de 4 segundos. La página de respuesta más lenta resulta ser la lista de usuarios.

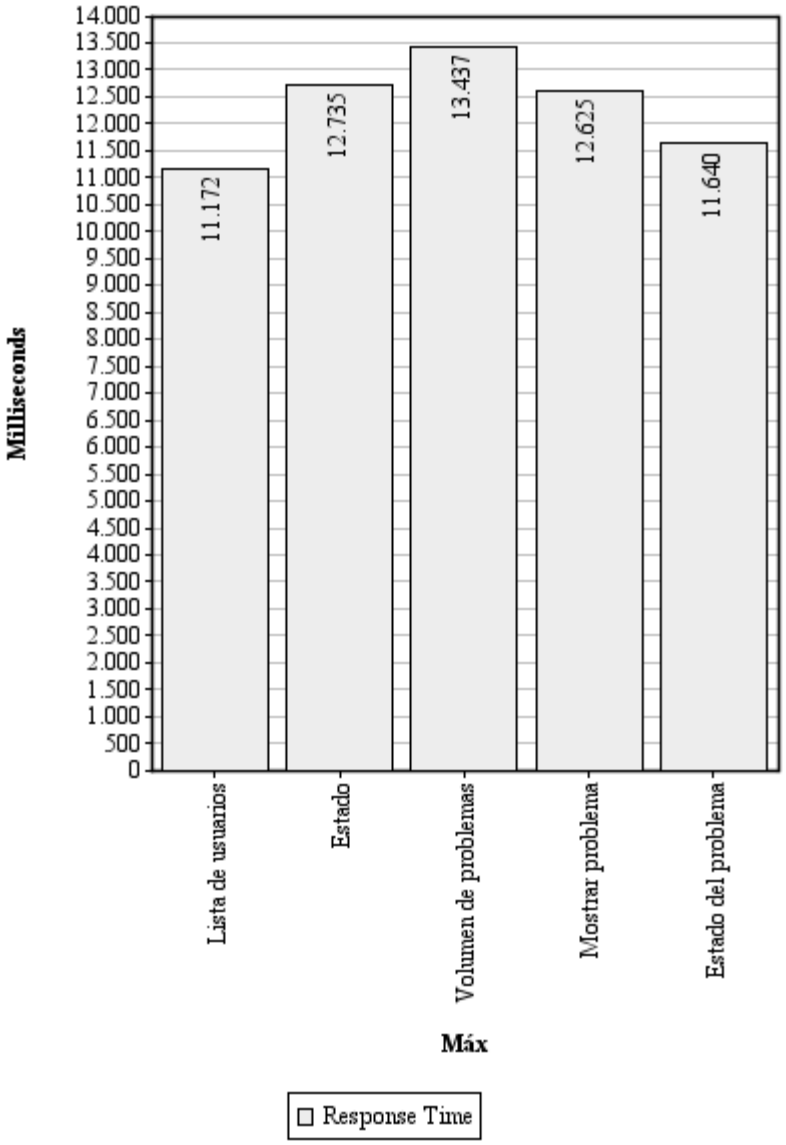


Imagen 22. Tiempo máximo de respuesta de una petición.

La gráfica indica el tiempo máximo que se toma una petición cualquier en dar respuesta. La petición más lenta demora 13 segundos, un tiempo inesperadamente alto. Dado que esta medición es

muy lejana con respecto a las mediciones de promedio (2 s) y línea de 90 (4 s) significa que sólo un pequeño número de peticiones caen en estos valores extremos.

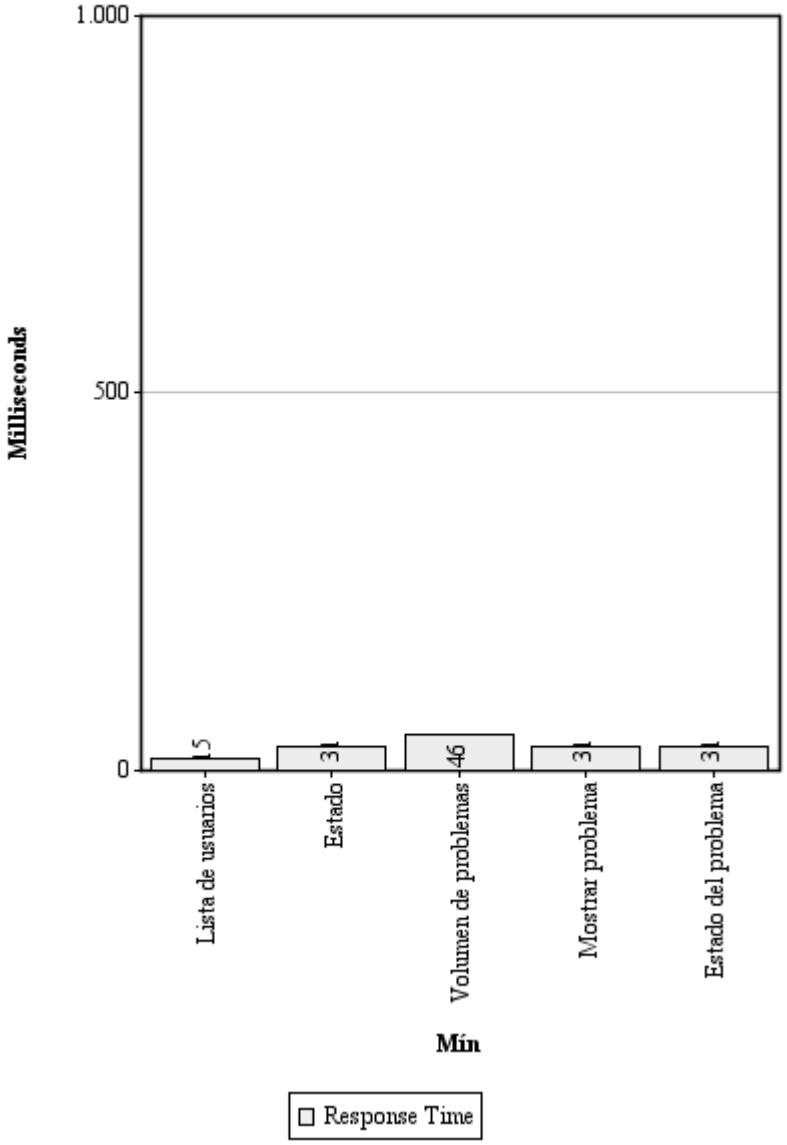


Imagen 23. Tiempo mínimo de respuesta de una petición.

La gráfica indica el tiempo mínimo que se toma una petición cualquier en dar respuesta. La petición más rápida demora 15 milisegundos, un tiempo muy bajo. Aplicando el mismo razonamiento que se aplicó respecto a la medición anterior, llegamos a la conclusión similar de que sólo un pequeño número de peticiones caen en estos valores extremos.

Complementos de la solución

Además de la solución principal consistente en el jurado online, existen otros elementos que componen un ecosistema más completo que mejora la experiencia del usuario y complementan la función de la aplicación principal.

Estos complementos constituyen parte de la Iniciativa Xtreme, que explicaremos más adelante. Contando con estas aplicaciones auxiliares, el ecosistema total de la aplicación quedaría representado en la figura siguiente:

Imagen 24. El instalador, así como la documentación referente al jurado, no son contemplados con más detalles en este documento debido a que se terminó el presente en un plazo más corto del originalmente planificado.

Iniciativa Xtreme

La Iniciativa Xtreme es un movimiento de estudiantes y profesores de la facultad 8 que apunta a crear un sistema compuesto por varias aplicaciones web con propósitos diferentes dentro de un objetivo principal: potenciar la programación de competencia y el estudio de la algoritmia en nuestra Universidad.

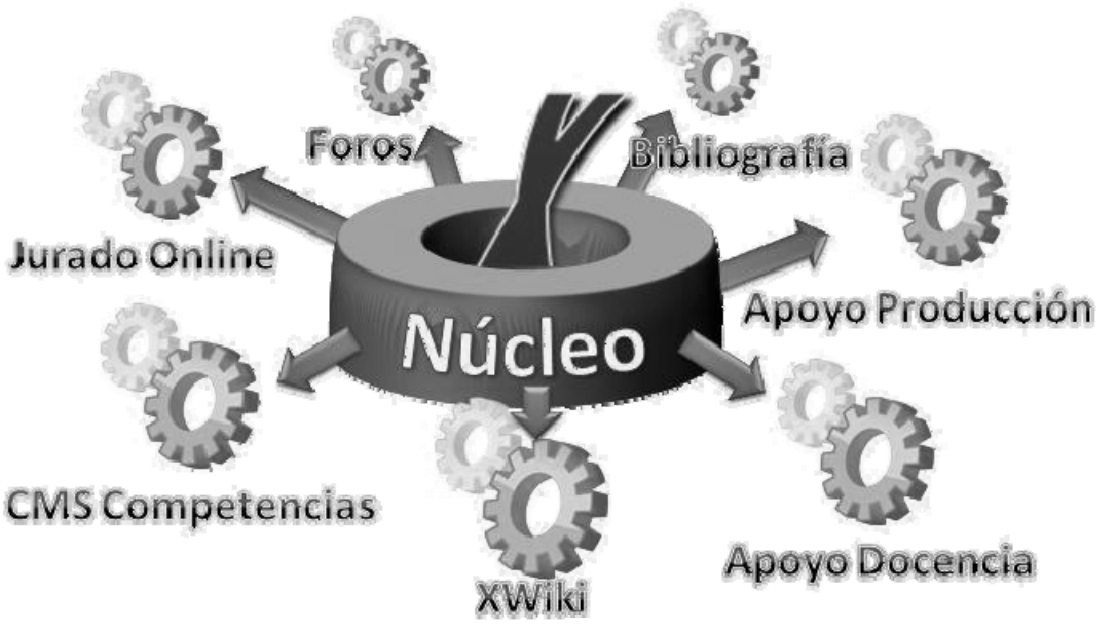


Imagen 25.Esbozo de la visión de la Iniciativa Xtreme

La misión del sistema Xtreme es la de ser la Meca de la programación en la UCI, brindando respuestas y servicios a proyectos, docencia y comunidades, así como también servir para el entretenimiento constructivo y el intercambio de conocimientos entre todos aquellos que lo necesiten. La estructura está dividida en varias aplicaciones separadas completamente, y un sistema central que permite navegar entre ellas y mantiene información importante de todas las aplicaciones, como pueden ser estadísticas de uso, información administrativa, vínculos y noticias y un sistema de autenticación *single sign on*, etc.

El desarrollo de la Iniciativa Xtreme está previsto que sea lento y progresivo, afianzando primero cada uno de sus componentes para integrarlos luego en el sistema total. Esta estrategia está dictada por

la poca disponibilidad de recursos y personal dedicado a este trabajo, pero tiene las ventajas de permitir desarrollos de terceros y la familiarización de los usuarios con cada software en particular a medida que sean creados.

Para implementar estas aplicaciones se siguió la filosofía de utilizar la mayor cantidad de software comercial posible, para disminuir el esfuerzo de implementación en pro de brindar las funcionalidades necesarias, y poder volcar el esfuerzo mayor en la aplicación central y las integraciones con sus auxiliares. A continuación veremos una breve descripción de cada uno de estos sistemas, su estado actual y su función.

CMS Xtreme para competencias de programación

Resumen

La construcción de un sistema de manejo de contenidos especializado en competencias de programación requiere una serie de definiciones iniciales acerca de arquitectura, metodología, diseño visual y tecnología utilizada. Para desarrollar este trabajo se empleó una arquitectura modular basada en componentes, con una metodología XP modificada para el desarrollo y tecnologías de frameworks Java, aunque la selección de tecnología facilita el desarrollo a costa de disminuir la facilidad de extensión del CMS. El análisis del problema y el diseño preliminar de los componentes arquitectónicos de alto nivel demostraron que es posible el desarrollo del software en las funcionalidades necesarias.

Introducción

En la UCI se lleva a cabo numerosas copas de programación, que constituyen un marco para medir conocimientos de programación entre los estudiantes y profesores de la Universidad. Algunas han llegado a ser prestigiosas y conocidas entre el personal de la institución. Las mismas generan gran atención entre la comunidad de programadores, y mucha documentación asociada a ellas: convocatorias, conjuntos de ejercicios, informaciones, documentaciones, etc. Pero actualmente la comunidad universitaria no cuenta con un sitio para encontrar información acerca de su copa favorita: fuera de las notas en la Intranet, las copas no tienen prácticamente ninguna presencia en la red. Ya que todas las copas tienen muchos puntos en común, se tomó la decisión de crear un sistema de contenido modular, desarrollado alrededor de un núcleo sólido y que puede ser extendido mediante módulos creados por terceras personas, para montar un sitio informativo de una copa de programación.

Desarrollo

La solución a desarrollarse consiste en un CMS para competencias, con funcionalidades adecuadas al tema, fácil de administrar y que aligere el trabajo de los organizadores de las competencias con respecto a la documentación que generan las mismas, como las inscripciones y acreditaciones de los equipos, la generación del reporte con los resultados de la competencia y otros mediante la integración con el jurado online que se utilice para desarrollarla.

Para la creación del CMS se seleccionó una colección de frameworks existentes para el lenguaje Java que facilitan el trabajo, como se detallará posteriormente. La aplicación está concebida utilizando varios patrones arquitectónicos y de diseño contemplados dentro de las buenas prácticas para las aplicaciones Web.

Tabla 7: Frameworks utilizados en el CMS

Nombre	Capa
JSF 1.2 Sun RI **Apache Tomahawk** **JBoss RichFaces**	Presentación
Spring framework 2.0.6	Negocio
Hibernate 3.2	Persistencia
JUnit 3.8	Prueba
Jasper Report	Reportes

Descripción de la Arquitectura

Se utilizó una arquitectura en tres capas y modular. El objetivo es lograr que se activen o desactiven sólo los módulos que sean necesarios, sin afectar para ello al resto de la aplicación. También que sea posible el desarrollo y adición de nuevos módulos, y que cualquier cambio en algún módulo no provoque afectaciones en los demás. Para las necesidades iniciales, se identificaron los módulos siguientes:

- **Texto:** Sirve para mostrar información estática, como la misión y visión de la competencia, las bases, el reglamento, etc. El formato de la información puede ser en texto plano o HTML.
- **FAQ:** Para publicar las respuestas a las preguntas más frecuentes. Aunque esta funcionalidad pudiera ser cubierta por el módulo de texto, se decidió hacerla en un módulo aparte para facilitar su administración.
- **Noticias:** Para publicar noticias, ya sea referentes a la competencia, como convocatorias, resultados o cualquier otra noticia, referente al mundo de la programación y la algoritmia.
- **Descargas:** Para ofrecer descargas como documentación, juegos de datos, compiladores u otros. Nótese que la funcionalidad ofrecida por este módulo puede ser aprovechada de varias formas: documentación, recursos, problemas… en dependencia de lo que se quiera ofertar en la descarga.
- **Competencias:** El módulo principal objetivo de nuestro trabajo, gestiona las diferentes ediciones de las competencias, opcionalmente mediante la integración con el jurado online, se encarga de gestionar las inscripciones en el evento, y de obtener y publicar automáticamente los resultados, así como los juegos

de datos y las soluciones de ejemplo del jurado. También se encarga de la generación automática de los reportes de las competencias. Este módulo es el más complejo y el que constituye además la razón de ser de toda la aplicación. Está formado por partes que pueden constituir a su vez módulos por derecho propio, si bien no tienen sentido fuera del contexto de una edición de una competencia.

El core (núcleo) se encarga de proveer las funcionalidades básicas, así como las configuraciones centrales del acceso a datos, integración con el jurado online o la interacción con la interfaz gráfica. El core constituye el módulo que une a todos los demás y que puede en un momento determinado servir de puente para manejar dependencias entre los módulos. El core se comunica con cada módulo mediante una interfaz que todos estos deben implementar.

La comunicación entre módulos debe ser nula o mínima. En caso de ser necesaria, esta será gestionada por el core. Para intercambiar datos entre ellos, se usarán los tipos de datos primitivos de Java, tipos de datos predefinidos en el core, mapas de cadenas (clave=valor) o XML.

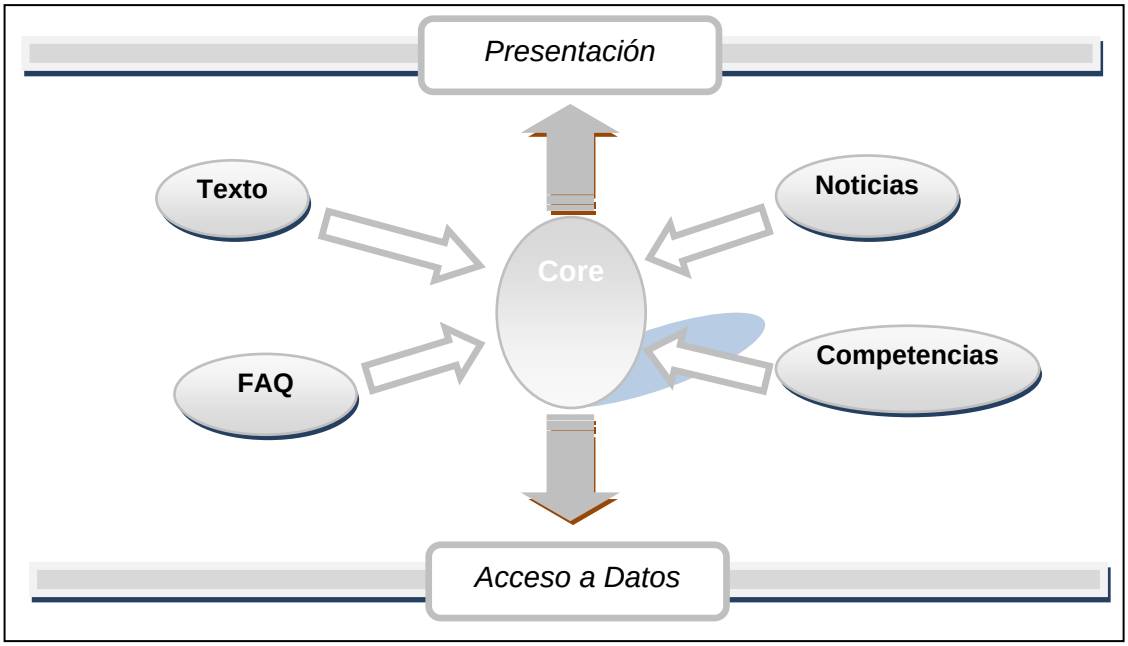


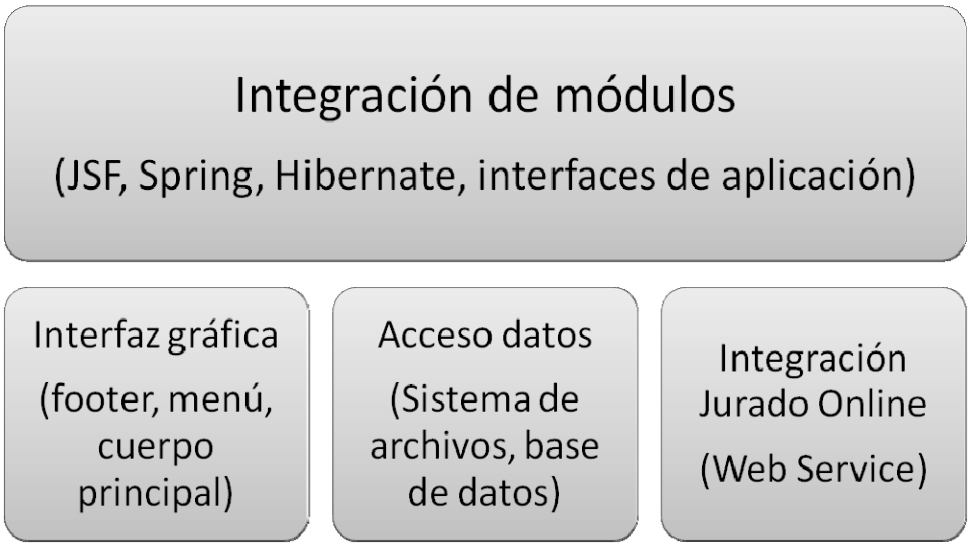
Imagen 26.La arquitectura a grandes rasgos

Componentes del core

Dentro del core, podemos enumerar los siguientes cinco grandes componentes, que componen los servicios clave que él brinda:

- **Exploración del sistema de archivos:** El core se encarga de proveer datos a los módulos que lo requieran a partir del sistema de archivos. De hecho, se piensa que esta sea la fuente principal de datos de los módulos que están planeados para la versión inicial, de tal manera que para publicar una noticia en el sitio, sea tan sencillo como teclearla con un editor de texto y copiarla en una carpeta del jurado.
- **Acceso a la base de datos:** Para los módulos que lo requieran, se provee acceso a bases de datos, utilizando Hibernate. Cada módulo debe proveer al core las configuraciones de Hibernate que dictan cómo se obtienen las entidades que necesitan de la base de datos (mapeos de Hibernate).
- **Integración con el Jurado Online:** El servicio que se encarga de comunicar con el jurado online y proveer de datos a la aplicación también es parte del core. Basado en este servicio se obtiene automáticamente la información referente a las competencias que se han efectuado para ser utilizada en la generación de reportes, por ejemplo.
- **Integración de los módulos:** Cada módulo debe proveer información acerca de sí mismo, y se comunica con el core mediante una interfaz que publica el mismo core, y cada módulo debe implementar. La misma sirve como punto de entrada a toda la comunicación con el módulo, ya sea desde el core o desde otro módulo.
- **Estructura de la interfaz y menú:** El core se encarga de proveer la estructura de la interfaz, los estilos que se aplican a la misma, los servicios centrales, y provee una sección donde se mostrará la interfaz de cada uno de los módulos. También se gestiona el menú principal, con entradas que serán dictadas por los módulos y se encargan de la navegación hacia ellos.

Imagen 27.Componentes arquitectónicos básicos (core)



Estado actual

El trabajo se encuentra actualmente en fase de exploración y conceptualización. Las principales dificultades con el desarrollo de esta aplicación es la falta de tiempo y recursos, y personal dedicado completamente a él.

Se ha desarrollado a la par con el diseño de la arquitectura un prototipo no funcional para tomar ideas sobre el diseño gráfico de la aplicación y su alcance.

Imagen 28.Prototipo no funcional del CMS, mostrando el menú desplegado y la pagina de noticias

El CMS será distribuido como un paquete instalable de forma libre para permitir que sea adaptado a cualquier competencia particular. El objetivo final es que todas las competencias de la UCI cuenten con su propio sitio web interno donde poder mantener sus registros históricos y evitar que su pasado se olvide entre la gente cada año.

Foro de usuarios

Para el foro de usuarios se utiliza una solución de terceros, el phpBB3. Este se distribuye *out-of-the-shelf* para sitios web, desarrollado en PHP por el phpBB Group. El acrónimo phpBB significa PHP Bulletin Board, y se distribuye bajo la GNU General Public License. Por lo tanto, phpBB es software libre.

Imagen 29.Logo phpBB

El foro phpBB cuenta con muchas funcionalidades útiles de administración, como *banning* de usuarios, distintos niveles de autorización, personalización de estilos visuales, etc. La versión 3 es mucho más configurable que la 2, principalmente el soporte predeterminado para la integración con un dominio LDAP.

El objetivo principal del foro de usuarios es crear el intercambio sencillo dentro de la comunidad del jurado. Aparte de este, el foro cumple otros objetivos, como dar soporte para cada problema o evento separadamente, aplicar encuestas, y servir como repositorio de quejas y sugerencias de los usuarios sobre el funcionamiento en general del jurado y de cualquier otro tema relacionado con los algoritmos o la programación. También la interface mejorada de administración permite un manejo más fino de los usuarios, su estado, permisos y limitaciones de los archivos.

Estado actual

El foro de usuarios está actualmente en producción e integrado con el jurado. Las tareas pendientes comprenden la integración con el dominio UCI y la configuración de los hilos predeterminados de discusión, todo lo cual está pendiente de la designación de un administrador a tiempo completo.

Solución de descargas

El PHCDownload es una de las soluciones más poderosas y útiles para administración de archivos existentes en el mercado, de acuerdo con el sitio de PHPCredo, el grupo que lo mantiene.

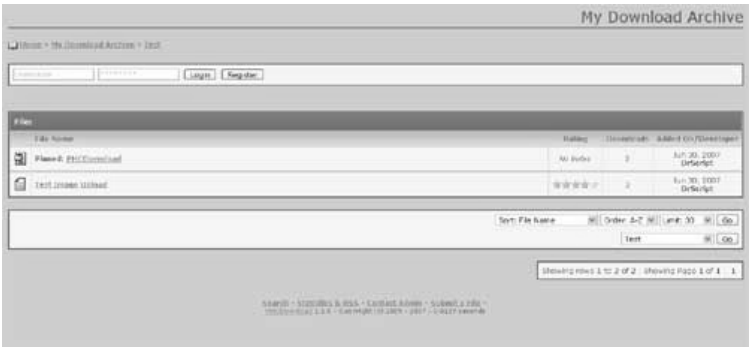


Imagen 30.Interface del PHC Download

El PHC proporciona características muy útiles como la separación de los archivos disponibles en categorías, una descripción de los archivos que se muestra como vista previa a la descarga, estadísticas sobre archivos más descargados, estado actual de las conexiones del servidor, etc.

Estado actual

La solución de descargas se encuentra actualmente funcional, pero aún consideramos necesario aumentar el número de documentación disponible antes de integrarla con el jurado.

Conclusiones

El jurado online es una herramienta web de múltiples aplicaciones en entornos académicos, con precedentes sólidos internacionalmente sobre sus características y .funcionamiento. La universidad cubana, particularmente la informática, no contaba con una herramienta de producción nacional como esta para su desarrollo.

En los últimos dos años de aplicación del software notamos varios resultados interesantes:

- La consolidación de la comunidad universitaria de algoritmia.
- La realización exitosa de varias ediciones de la Copa Pascal de varias facultades, hospedadas en nuestro jurado online.
- La creación de un nuevo evento competitivo de programación, el campeonato Xtreme Programming. Este evento ya cuenta con dos años en funcionamiento y varias ediciones a nivel de facultad y universitario.
- El aumento de la conciencia colectiva en lo referente a los temas de algoritmos y programación en general, las personas que se ocupan de estos y las actividades que se realizan en la Universidad sobre estos temas.
- El uso experimental de la aplicación como apoyo al proceso docente, especialmente en las asignaturas más cercanas a la algoritmia.
- La familiarización de los más jóvenes con la programación a partir del uso del jurado y su participación en los eventos competitivos de programación.

Hemos presenciado un aumento considerable del interés por la algoritmia, tanto en su práctica como en su estudio por parte de estudiantes y profesores. Igualmente la creación de un número de aplicaciones semejantes a la nuestra y de eventos de programación acompañantes también forma parte de este movimiento de redescubrimiento de la programación como rama fundamental de nuestra disciplina.

También podemos afirmar que esta aplicación tiene aún mucho más potencial para aumentar el conocimiento teórico-práctico de sus usuarios, la unión entre ellos y el intercambio de conocimientos. Mediante la adición de nuevas funcionalidades, la integración con otras aplicaciones de software, su uso en eventos competitivos y en el proceso docente podemos lograr que el jurado se convierta en aquello

que está llamado a ser: un fuerte propulsor de la programación en la UCI, un lugar donde la algoritmia sea el tema principal y cualquier persona interesada o necesitada de conocimiento pueda encontrarlo, compartirlo y discutirlo con sus semejantes.

Recomendaciones

Basándonos en el impacto que ha tenido en nuestra Universidad la aplicación del jurado online y toda la actividad que este provoca, así como la situación internacional en lo referente a temas de algoritmia y programación de competencia, consideramos necesario hacer una serie de recomendaciones:

- **Desarrollo:** El jurado es, como toda aplicación en producción, un trabajo vivo y en constante cambio. Es necesario destinar recursos más cuantiosos al desarrollo, mantenimiento y mejoramiento tanto del código base como del ambiente de producción del mismo. Las formas de hacer esto pueden ser disímiles pero la más directa, tomando en cuenta las características del ambiente productivo universitario, es crear un proyecto productivo con el objetivo de dar seguimiento al desarrollo alcanzado. El propósito final de dicho proyecto deberá ser el lograr implementar no solo el jurado, sino todo un ambiente de carácter docente productivo, compuesto por varios servicios sobre plataforma web y que pueda ser migrado fácilmente entre sistemas operativos y entidades. El objetivo y alcance de este ambiente que hemos dado en llamar Iniciativa Xtreme escapa los límites de este documento por lo cual nos limitaremos a proveer una breve descripción [Anexo J].
- **Producción:** Que la aplicación, en su estado actual pero sin prejuicio de desarrollos futuros sea provista de un hardware más adecuado a sus necesidades y objetivos, dígase por ejemplo un servidor web dedicado visible a toda la intranet universitaria. De esta forma se podría llevar el desarrollo de la aplicación a un nivel superior en lo que respecta a las pruebas, corrección de errores, concurrencia y tolerancia al error.
- **Distribución:** Que la aplicación, empaquetada debidamente, sea distribuida a cualquier casa de altos estudios, empresas, centros de investigación o intranets en general donde pueda ser útil de la misma forma en que lo ha sido en la UCI. La aplicación más directa de esta recomendación es lograr la instalación del Jurado Online en todas las demás universidades, sedes de entrenamientos de preselección, institutos pre universitarios de ciencias exactas e institutos preuniversitarios de informática. En estas localizaciones la aplicación puede cumplir el rol de apoyo docente de manera muy efectiva, preparando a los ingenieros del futuro de forma integral aun si la algoritmia y la programación no son precisamente el fuerte de su especialidad. Especialmente en los preuniversitarios de informática y las sedes de entrenamiento de preselecciones, la tarea formativa de la aplicación tributaría al desempeño del país en los concursos internacionales de programación y matemática. Vale decir que los mejores desempeños de Cuba en dichas competencias (ACM, IOI) se deben precisamente a personas dedicadas al estudio de la algoritmia y la programación de competencia.
- **Documentación y retroalimentación:** Que para aumentar la eficiencia y utilidad de la aplicación se realicen encuestas constantes entre los usuarios, para determinar la popularidad, impacto y soporte de

cada funcionalidad, sus quejas y sugerencias. Dichas encuestas deben soportarse en alguna herramienta informática, para poder cumplir su objetivo de retroalimentación constante. Dentro de este tópico también entra la necesidad de generar documentación sobre los jurados online en general, su funcionamiento, forma de uso, utilidad, los eventos de competencia, su papel en el mundo y los principales eventos a nivel nacional e internacional, todos los cuales son accesibles a los estudiantes de la UCI. Una serie de artículos, un libro o incluso una publicación periódica aun en formato digital sería la solución ideal a esta situación, siguiendo los precedentes internacionales en esta materia.

- **Logística:** Como en cualquier otra acción humana, la participación de las personas en nuestras actividades debe ser correctamente estimulada y/o recompensada, de forma tangible e intangible. La no existencia de premios específicos ni de artículos personalizados alegóricos a la aplicación conspira en contra de su conocimiento por el público general, de su participación frente a otras actividades más rentables o más importantes debido a los beneficios que pueden brindar, etc. Artículos simples como llaveros, agendas, pulóveres o bolígrafos temáticos constituyen un punto de partida sencillo para estimular la participación de las personas en las actividades relacionadas con la aplicación. Esto es especialmente importante en cuanto al soporte material a las competencias, que es más complejo por cuanto es más diverso. En una competencia idealmente debe proveerse algunos artículos a los participantes y ganadores, de forma individual y colectiva. Merienda, trofeos, pulóveres, iniciativas propias del evento (los globos de la ACM son el ejemplo más claro) y cualquier premio de carácter individual que pueda ofertarse, son un muy buen aliciente para la participación de los programadores en los eventos y para la captación de nuevos interesados.

Referencias

1. **Stephens, Matt y Rosenberg, Doug.** *Extreme Programming Refactored: The Case Against XP.* : Apress, 2003.

2. **Shore, James y Warden, Shane.** *The Art of Agile Development.* : O´Reilly, 2007.

3. **Moodie, Matthew.** *Pro Apache Tomcat 6.* : Apress, 2007.

4. **Kniberg, Henrik.** *Scrum and XP from the Trenches.* : InfoQ, 2007.

5. **Jeffries, Ron, Anderson, Ann y Hendrickson, Chet.** *Extreme Programming Installed.* : Addison Wesley, 2000.

6. **Hightower, Richard, Onstine, Warner y Visan, Paul.** *Professional Java Tools for Extreme Programming.* : Wrox, 2004.

7. **Duvall, Paul, Matyas, Steve y Glover, Andrew.** *Continuous Integration.* : Addison Wesley, 2007.

8. **Crispin, Lisa y House, Tip.** *Testing Extreme Programming.* : Addison Wesley, 2002.

9. **Cohen, Mike.** *Agile Estimating and Planning.* : Prentice Hall, 2005.

10. **Beck, Kent y Fowler, Martin.** *Planning Extreme Programming.* : Addison Wesley, 2000.

11. **Beck, Kent.** *Extreme Programming Explained.* : Addison Wesley, 2000.

12. **Ashraful Anam, Mohammad.** Error 139 from storage engine. *MySQL Bugs.* [En línea] 8 de Agosto de 2007. [Citado el: 25 de Enero de 2008.] http://bugs.mysql.com/bug.php?id=30295.

13. **Larman, Craig.** UML y Patrones. Introducción al análisis y diseño orientado a objetos. : Prentice Hall, 2003.

14. **Couch, Justin y Steinberg, Daniel.** *Java 2 Entreprise Edition Bible.* : Hungry Minds, Inc., 2002.

15. **Falkner, Jayson y Jones, Kevin.** *Servlets and JavaServer Pages.* : Addison Wesley, 2004.

16. MyISAM vs InnoDB. *túfuncion.* [En línea] 11 de Diciembre de 2007. [Citado el: 25 de Enero de 2008.] http://www.tufuncion.com/myisam-vs-innodb.

17. InnoDB vs MyISAM vs Falcon benchmarks. *MySQL Performance Blog.* [En línea] 8 de Enero de 2007. [Citado el: 25 de Enero de 2008.] http://www.mysqlperformanceblog.com/2007/01/08/innodb-vs-myisam-vs-falcon-benchmarks.

18. Extreme Programming: A gentle introduction. *Extreme Programming.* [En línea] 17 de Febrero de 2006. [Citado el: 26 de Enero de 2008.] http://www.extremeprogramming.org/.

19. The Algorithmist. *The Algorithmist.* [En línea] 18 de Mayo de 2006. [Citado el: 27 de Enero de 2008.] http://www.algorithmist.com/index.php.

20. Daily Submissions. *Peking University Online Judge.* [En línea] 28 de Enero de 2008. [Citado el: 28 de Enero de 2008.] http://acm.pku.edu.cn/JudgeOnline/chart.

21. **Cohn, Mike.** Advantages of User Stories for Requirements. *Mountain Goat Software.* [En línea] 1 de Octubre de 2004. [Citado el: 10 de Febrero de 2008.] http://www.mountaingoatsoftware.com/article_view/27-advantages-of-user-stories-for-requirements.

22. **Fowler, Martin.** Planning and Running an XP Iteration . *Martin Fowler´s Blog.* [En línea] 12 de Enero de 2001. [Citado el: 15 de Febrero de 2008.] http://martinfowler.com/articles/planningXpIteration.html.

23. **Ambler, Scott.** *Agile Database Techniques: Effective Strategies for the Agile Software Developer.* : John Wiley & Sons , 2003.

24. —. Agile Modeling and eXtreme Programming (XP). *Agile Modeling.* [En línea] 3 de Marzo de 2007. [Citado el: 20 de Febrero de 2008.] http://www.agilemodeling.com/essays/agileModelingXP.htm.

Glosario

AC: Código de respuesta aceptada. Estado de una sumisión exitosa.

ACM: Acrónimo de **Association for Computing Machinery**, organización fundada en 1947 como la primera sociedad científica y educativa acerca de la Computación. Publica varias revistas y periódicos científicos relacionados con la computación; patrocina conferencias y otros eventos relacionados con las ciencias de la computación como por ejemplo el internacional ACM **International Collegiate Programming Contest** (ICPC).

Autor: Ver **Submitter.**

Bug: Error, falla o funcionalidad no documentada que impide que el software funcione de la forma deseada, por ejemplo, produciendo resultados incorrectos.

Campeonato Xtreme Programming: Por este nombre se conoce al Campeonato de Programación que organiza la Facultad 8, una idea gestada en el seno de la FEU y los estudiantes de mayores resultados en eventos competitivos de la Universidad, y hecha posible por la dirección de la Facultad, la FEU y el trabajo de muchas personas. Hasta el momento se han convocado tres ediciones del evento principal (2008) y varias de otros eventos accesorios.

CEIP: Competencia estudiantil por invitación de programación, competencia de programación realizada con carácter anual en la que participan estudiantes de varias universidades así los estudiantes preseleccionados a nivel universitario para participar en los eventos de conocimientos representando a nuestro país.

CMS: Un sistema de manejo de contenido o CMS es un software usado para crear, editar, administrar y publicar contenido de forma consistente. Los CMSs son utilizados frecuentemente para almacenar, controlar, versionar y publicar documentación específica como artículos de noticias, manuales, guías de venta y propaganda de marketing. El contenido que puede manejarse incluye ficheros, imágenes, audio, documentos electrónicos y contenidos Web.

Código legado: Dentro de un proyecto de software dícese del código anterior al desarrollo que coexiste en producción junto al código de la aplicación que está siendo escrito. El código legado comporta problemas de compatibilidad con el resto de la aplicación, pero puede ser necesario mantenerlo debido a razones diversas de funcionalidades, facilidad de uso del cliente, velocidad, etc.

Copa Pascal: Copa de programación que se realiza todos los años e la UCI con el objetivo de incentivar el aprendizaje y de demostrar los conocimientos de programación. Se compite en equipos mixtos de 3 estudiantes (normalmente dos hombres y una mujer), los que podrán utilizar como máximo 2 computadoras.

Copa Void: Copa de programación realizada en la UCI por la Facultad 10, donde se miden conocimientos sobre el análisis y diseño de algoritmos. La Copa Void de Programación provee, a estudiantes y profesores, la oportunidad de intercambiar experiencias y conocimientos. Promueve el aprendizaje del idioma inglés, metodologías para la resolución de problemas, uso de herramientas Software Libre y habilidades de trabajo en equipo. La competencia propicia centrar la atención en la generación futura de profesionales de la Informática, mientras persiguen la excelencia. La Copa Void de Programación es organizada por la Facultad X, acorde a sus políticas y procedimientos. El Comité Organizador, dirigido por el Director Organizador, fija las Reglas Oficiales para la conducta de la competencia.

CPAV: Cátedra de programación avanzada, nombre con el que surge un proyecto en la UCI en el curso 2006-2007, con el objetivo de agrupar a los estudiantes con interés o conocimiento para formar un grupo que promueva el estudio de la algoritmia así como su aplicación en la producción.

Daemon: Programa de computadora que se ejecuta en el trasfondo y no bajo el control de un usuario. Normalmente se usan para tareas de mantenimiento constante que no requieren de la presencia de un usuario.

Envío: Ver **sumisión.**

Estado: El estado de un problema determinado se compone del conjunto de soluciones que se le ha dado. El estado aporta mucha información al usuario experto sobre el problema, información que se puede inferir a partir de los datos públicos de consumo de memoria, longitud de código, *submitter* y tiempo.

Debido a esto el estado usualmente no es mostrado durante competencias, para evitar brindar información adicional sobre los problemas.

Estado actual: El estado actual de la aplicación es el subconjunto más actual de sumisiones que se han hecho, sin tomar en cuenta problema o usuario. El estado actual es una herramienta útil para evaluar la actividad actual de la aplicación, los ejercicios que despiertan más interés y los lenguajes más utilizados, entre otros datos de interés. El tamaño del subconjunto que se considera estado actual varía de una aplicación a otra.

Evaluador automático: Aplicación que alivia el trabajo de calificación de soluciones programáticas, moviendo su responsabilidad del humano a la máquina y tratando de lograr mejoras en cuanto al chequeo de correctitud, robustez y eficiencia.

ICPC: El ACM **International Collegiate Programming Contest** (abreviado como ACM-ICPC o solo ICPC) es una competición anual de programación en varios niveles que se celebra entre numerosas Universidades del mundo. Este concurso es patrocinado por IBM, y tiene su sede en la Universidad de Baylor. El ICPC define regiones autónomas en seis continentes y opera bajo los auspicios de la ACM.

IOI: La **International Olympiad in Informatics** (IOI) u Olimpiada Internacional de Informática es una competencia anual de informática para estudiantes de educación avanzada. La primera IOI tuvo lugar en Pravetz, Bulgaria, en 1989. La competencia consiste en dos días de programación, resolviendo problemas de naturaleza algorítmica. Los estudiantes compiten individualmente, con un máximo de 4 estudiantes compitiendo por un país determinado. Los estudiantes se seleccionan a partir de los concursos nacionales de computación.

Jurado en línea: Evaluador automático desplegado en entornos generalmente académicos presentado en la Web (o en cualquier otro soporte igualmente masivo) que provee problemas y sirve para evaluar automáticamente las soluciones algorítmicas que a estos encuentran sus usuarios desarrolladas en uno de entre un número determinado de lenguajes de programación. Un Jurado Online usualmente ofrece además otros servicios de consulta y comunitarios a sus usuarios y suele servir como herramienta para la realización de competencias de programación.

Juego de datos: Conjunto formado por uno o varios archivos que contienen las entradas que se le aplicarán a la solución propuesta por el usuario, así como las soluciones que se consideran correctas para las mismas.

Juzgado simple: Forma de juzgado más convencional en la cual a la solución propuesta por el usuario se le aplica un conjunto de entradas, se toma la salida que este devuelve como respuesta y se compara con la salida que se esperaba, en caso de que estas coincidan se considera la respuesta como correcta e incorrecta en caso contrario.

Juzgado múltiple: Forma de juzgado que se aplica a problemas muy específicos en los cuales para una misma salida se pueden aceptar varias salidas. En este tipo de ejercicios se toma la salida que devuelve como respuesta la solución del usuario y se le alimenta como entrada a un programa evaluador que es capaz de decidir si la solución es aceptada o no, y que varía para cada problema en específico.

Kent Beck: Es el creador de la Extreme Programming, desarrollada cuando servía como líder de proyecto en el **Chrysler Comprehensive Compensation Project (C3)**, un proyecto a largo plazo para la implementación de una nómina de pago para la transnacional Chrysler. Beck fue uno de los 17 signatarios originales del Manifiesto Ágil en el 2001. Beck ha sido pionero en el trabajo con patrones de diseño, el desarrollo orientado a pruebas y la aplicación comercial de Smalltalk. Beck popularizó las tarjetas CRC junto a Ward Cunningham, y creó junto a Erich Gamma el framework de pruebas unitarias JUnit. Tiene una maestría en Ciencias por la Universidad de Oregón.

Martin Fowler: Es un famoso autor y conferencista internacional en temas de arquitectura de software, especializado en análisis y diseño orientado a objetos, UML, patrones y metodologías agiles, incluida XP. Martin Fowler comenzó a trabajar en el software en los 80 y ha escrito 5 libros muy populares sobre el tema del desarrollo de software. En el 200 se convirtió en Científico Jefe en ThoughtWorks, una empresa consultora y de integración de sistemas. Fowler es miembro de la Alianza Ágil y uno de los propulsores del Manifiesto para el Desarrollo Ágil de Software o Manifiesto Ágil. Invento el término Inyección de dependencias para describir el patrón de software de Inversión de Control. Martin Fowler nació en Walsall, Inglaterra, y vivió en Londres durante una década antes de mudarse a los Estados Unidos en 1994. Vive cerca de Boston, Massachusetts.

Problema: Enunciado preciso de una situación que presenta un problema a solucionar mediante la aplicación de técnicas algorítmicas, lógica, geometría o cualquier otra rama matemática. Los problemas generalmente se dividen en secciones muy específicas, siendo la división más común: descripción, especificación de entrada, especificación de salida, ejemplo de entrada, ejemplo de salida y fuente del problema. Tradicionalmente el enunciado del problema debe proveer cualquier conocimiento que caiga fuera de las áreas algorítmicas o matemáticas y que sean necesarios para resolverlo, siguiendo el razonamiento de que lo que se mide no es el nivel de conocimiento o cultura general del usuario, sino su conocimiento y habilidad en la rama de programación de competencias.

Problem set: Ver **Volumen de problemas.**

Problemsetter: Persona con privilegios para añadir problemas en el Jurado Online. Usualmente el problema setter posee la solución de ejemplo por la cual se crean los juegos de datos, e incluso a veces puede proveer la demostración matemática o algorítmica de correctitud de la solución. El problemsetter es la última instancia para aclarar las dudas sobre correctitud y precisión del problema.

Programación de competencia: Se refiere la programación centrada en el análisis y la implementación de programas con el objetivo de medir conocimientos teóricos en aspectos particulares de algoritmia, matemática o lógica. Esta rama de programación presenta características diferentes de la programación empresarial o educativa, en que no presta atención a los principios de orientación a objetos, chequeo de errores, patrones de diseño, etc.; concentrándose exclusivamente en aspectos de eficiencia y correctitud de los programas. Una aplicación para competencias suele ser de código ofuscado, sin diseño claro y con poco mantenimiento, con poca o ninguna extensibilidad ni modularidad debido a que el interés principal esta puesto en la eficiencia en el uso de memoria y tiempo y en la implementación correcta del algoritmo correspondiente. Opcionalmente algunas personas tienden a desarrollar sus propios estilos de programación, buscando elegancia en sus soluciones lo cual puede contrarrestar las propiedades negativas anteriormente expuesta.

Pudrición de software: También conocida como pudrición de código o decadencia del software. Este concepto describe el deterioro lento y perceptible del software en el tiempo, la cual inevitablemente lleva a fallos, falta de usabilidad o necesidad de mantenimiento. Este no es un fenómeno físico, dado que el software no puede realmente decaer, sino que sufre de una falta de actualización con respecto al ambiente cambiante en que reside.

Submit: Ver **sumisión.**

Submitter: Usuario que realiza sumisiones con cierta regularidad.

Solución: Sumisión exitosa.

Sumisión: Intento de solución de un problema. Una sumisión generalmente se identifica con un número secuencial, y corresponde a un intento realizado por un usuario para resolver un problema determinado. Constituye la unidad básica de interacción del usuario con la aplicación, y puede tener asociados distintos estados en dependencia de si el intento es exitoso o no, y si no, del error que provocó el fracaso.

Volumen de problemas: Conjunto de problemas que se encuentran en un jurado online o en un evento competitivo. Los problem sets pueden ser creados siguiendo distintos criterios, como complejidad, asunto, tamaño o simplemente ordenados cronológicamente, dependiendo el criterio del jurado o evento donde se encuentre.

WA: Código de respuesta incorrecta. Estado de una sumisión desacertada.