Universidad de las Ciencias Informáticas

Facultad 1

“Módulo de agregación de anuncios de competencias para el Juez en Línea Caribeño.”

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

Autor:

Eddy Roberto Morales Pérez

Tutores:

Ing. Jorge Amado Soria Ramírez

Ing. Javier Heredia Ruíz

Ing. Susana González Rodríguez

La Habana

2015

Declaración de autoría

Declaro ser autor de la presente tesis y ofrezco a la Universidad de las Ciencias Informáticas los derechos primordiales de la misma. Para que así conste firmo la presente a los ____ días del mes de _____ del año_____.

_____

Firma del Autor

Eddy Roberto Morales Pérez

_____

Firma del Tutor

Ing. Jorge Amado Soria Ramírez

_____

Firma del Tutor

Ing. Javier Heredia Ruiz

_____

Firma del Tutor

Susana González Rodríguez

Datos de contactos

Autor:

Eddy Roberto Morales Pérez
Universidad de las Ciencias Informáticas, La Habana, Cuba.
Correo electrónico: emoralesp@estudiantes.uci.cu

Tutores:

Ing. Jorge Amado Soria Ramírez
Universidad de las Ciencias Informáticas, La Habana, Cuba.
Correo electrónico: jasoria@uci.cu

Ing. Javier Heredia Ruiz
Universidad de las Ciencias Informáticas, La Habana, Cuba
Correo electrónico: jheredia@uci.cu

Ing. Susana González Rodríguez
Universidad de las Ciencias Informáticas, La Habana, Cuba
Correo electrónico: sgrodriguez@uci.cu

Dedicatoria

Este trabajo está dedicado a mi familia, mis padres que son mi vida y razón de ser. Especialmente mi mamá Reina y mi abuela Eloísa, querida por todos como Merito.

Agradecimientos

Agradezco a mis padres y mi abuela Merito por todo el amor y el apoyo brindado incondicionalmente.

Mi madre Reina, no sé qué sería de mi vida si no estuvieras siempre a mi lado, te agradezco por todo, empezando por mi vida, no ha sido fácil, pero lo has dado todo para hacerlo fácil para mí. Te amo.

Mi papá Eddy, a veces ha sido un poco complicado, te quiero, te agradezco todo la sabiduría que me has inculcado en la vida, y por guiarme siempre por el camino correcto.

Mi abuela Merito, todo esto, todo lo que soy y todo lo que seré, será dedicado a ti, por ser la mejor abuela del mundo, espero que vivas muchas años más a mi lado. Te amo.

Juan, mi segundo papá, te agradezco que hayas estado más de 10 años cuidando a mi mamá y mi abuela, nunca podré pagar esta deuda.

Patricia, mi hermanita querida, aun cuando tienes 7 años ahora, has sabido iluminarme los días cada vez que me encuentro contigo. Te amo.

Mi familia, tengo un especial agradecimiento para mi familia, pues siempre de cierta forma han puesto un granito de arena para armar el castillo, mi abuelo Jesús, mi abuela Felícita, mis tíos Yunior y Alexis. Agradecimientos especiales también para mi familia de Vista Alegre y La Quinta, son muchos, por lo que les llegue un profundo agradecimiento a todos. Un agradecimiento especial a Yoly, la esposa de mi papá, por quererme como un hijo.

Roger y Eiber, mis amigos de toda una vida, les agradezco todo la confianza que han creado en mí.

Gracias a mis tutores, profesores y entrenadores de movimiento por convertirme en el profesional que hoy soy.

En general, siento un profundo agradecimiento por todos, mis amigos, colegas, gente del grupo y compañeros del ACM-ICPC.

Resumen

Las competencias de programación competitiva son una forma de probar y preparar a los profesionales del mundo del *software*. La mayoría de los usuarios que participan en competencias de programación lo hacen en varios Jueces en Línea, lo cual dificulta el seguimiento de las competencias planificadas. Un sistema de publicación de anuncios de competencias permite el seguimiento de los anuncios de las competencias planificadas por los Jueces en Línea. En la presente investigación se identifican Jueces en Línea que serían de interés para los usuarios realizarles seguimiento a sus anuncios de competencias. Se presenta el diseño, la implementación y la validación de la implementación de un módulo para el Juez en Línea Caribeño utilizando el *framework Spring*. El módulo es capaz de gestionar y recopilar automáticamente los anuncios de competencias de programación competitiva en un conjunto de Jueces en Línea. Como resultado se obtuvo un módulo capaz de ayudar a los usuarios en el seguimiento de las competencias de su preferencia y un incremento de los usuarios del Juez el Línea Caribeño debido a la nueva funcionalidad.

Palabras Clave: anuncios; competencias de programación; Juez en Línea; *Spring Framework*.

Índice

Índice de tablas

Índice de figuras

Introducción

El Concurso Internacional Universitario de Programación de la Asociación para Máquinas Computadoras (ACM-ICPC por sus siglas en inglés) es efectuado anualmente con la participación de equipos en representación de diferentes universidades del mundo. Constituye uno de los eventos más importantes y prestigiosos de programación de computadoras donde se fomenta el trabajo en equipo, la resolución de problemas y el desarrollo rápido de *software*. (Lobaina, et al, 2012)

El ACM-ICPC tiene sus raíces en un concurso realizado en el Texas A & M[1] en 1970. Cada año se realiza en distintas sedes por todo el mundo con la participación de equipos conformados por tres estudiantes guiados por un profesor-entrenador. Estos equipos pasan por tres etapas de competición que van desde los concursos locales (efectuados en cada una de las universidades que participan en el evento), hasta los concursos nacionales y regionales. De esta forma los ganadores de cada regional avanzan a la Final Mundial cada año. (Silveira-Romero, et al, 2012) (Vaillant, et al, 2013)

Cuba se ha ido insertando paulatinamente en el movimiento ACM-ICPC, siendo el año 2009 a partir del cual más se ha intensificado luego de celebrarse por primera vez un concurso regional en Cuba. La Universidad de las Ciencias Informáticas (UCI) no ha estado alejada de este movimiento por lo que ha tomado acciones para promover la participación de los estudiantes en este tipo de competencias. Como parte de estas actividades fue creado el Movimiento de Programación Competitiva “Tomás López Jiménez” (MPC-TLJ) el cual agrupa a los estudiantes y profesores involucrados en la programación competitiva. Esto ha posibilitado el fortalecimiento de la institución y una mayor participación de los estudiantes en los eventos locales, nacionales y regionales del ACM-ICPC.

En sus primeros años este concurso contaba con pocos participantes por lo que su gestión se realizaba de forma manual. La evaluación de las soluciones, la confección de la tabla de posiciones, así como otros procesos que se llevan a cabo durante el desarrollo de una competencia eran realizados manualmente. Luego de varios años, este proceso ha ganado complejidad debido al aumento del número de participantes

[1] Originalmente abreviatura de "Agricultura y Mecánica". Se conserva este nombre como un vínculo con el pasado de la universidad.

según *ACM International Collegiate Programming Contest*[1]. Esto provocó el surgimiento de evaluadores automáticos que han ido evolucionando hasta los actuales Jueces en Línea.

El 5 de junio de 2010 se publica en Internet la primera versión del Juez en Línea Caribeño (COJ por sus siglas en inglés) con el objetivo fundamental de brindar más autonomía al movimiento ACM-ICPC en la región del Caribe, proporcionando a sus miembros (concursantes, entrenadores, directores y administradores) un espacio destinado a la auto-preparación y el intercambio de conocimientos. (Lobaina, et al, 2012). Hoy en día, el COJ es una de las principales vías que los concursantes utilizan para su preparación, ofreciendo una variada gama de problemas y concursos, posibilitando que sus usuarios eleven sus habilidades en la programación competitiva.

Los Jueces en Línea tienen una historia documentada como centro y soporte de diversas actividades y aplicaciones. Constituyen un núcleo alrededor del cual se pueden aglutinar personas interesadas en la programación en sus variadas formas. Una de las actividades principales que realizan los usuarios de los Jueces en Línea es participar en competencias de programación abiertas en Internet. Muchos de estos usuarios no se fidelizan con un solo juez en línea sino que participan en disímiles sitios buscando variedad y cantidad para un entrenamiento competitivo más completo. Múltiples sitios web tienen que mantenerse vigilados por los usuarios para construir la planificación de su entrenamiento personal. Esto requiere del acceso a todos los sitios de interés y un seguimiento constante de los nuevos anuncios a medida que estos surgen. Obteniéndose como resultado que el usuario tenga que dedicar más tiempo y esfuerzo para planificar su entrenamiento, al tener que constantemente monitorear los sitios de interés en busca de competencias de programación para realizar su preparación.

No es muy recomendable obtener la información sobre las competencias planificadas mediante una sola vía, consultando los sitios web periódicamente. Los problemas de conectividad de los usuarios dado por el difícil acceso a Internet para los cubanos, los cuales representan el mayor número de usuarios de COJ, no hace posible que el acceso constante a los sitios siempre sea posible. Además, la disponibilidad de los sitios web puede encontrase comprometida, lo cual puede provocar que no se encuentren disponibles en el momento que el usuario realize la consulta.

A raíz de la anterior situación problemática se deriva el siguiente **problema a resolver**:

[1] http://icpc.baylor.edu/

¿Cómo agrupar anuncios de competencias de programación de diferentes Jueces en Línea en un recurso único, facilitando el acceso de los usuarios a los anuncios en las competencias de su interés, reduciendo el esfuerzo necesario para planificar el entrenamiento de los mismos?

Se determina como **objeto de estudio** el proceso de anuncios de competencias en los Jueces en Línea. En este objeto de estudio queda enmarcado el campo de acción. El objeto de estudio es lo suficientemente específico, por lo que no es necesaria la definición de campo de acción.

El **objetivo general** de la investigación es desarrollar un módulo para el COJ que permita agrupar anuncios de competencias de múltiples fuentes en la Red mediante el uso de tecnologías libres.

Para darle solución al problema planteado se han definido los siguientes **objetivos específicos:**

- Valorar la información asociada a los principales Jueces en Línea y su planificación regular de competencias.
- Definir las tecnologías, las herramientas y la metodología para la implementación del módulo de publicación de anuncios de competencias para el COJ.
- Diseñar las funcionalidades del módulo de publicación de anuncios de competencias para el COJ.
- Implementar las funcionalidades del módulo de publicación de anuncios de competencias para el COJ.
- Validar las funcionalidades del módulo de publicación de anuncios de competencias para el COJ.

La **idea a defender** de la presente investigación constituye que el desarrollo de un sistema que agrupe los anuncios de competencias de programación de diferentes Jueces en Línea en un recurso único, reduciría el esfuerzo necesario para planificar el entrenamiento de los usuarios de forma tal que facilite el acceso a los anuncios de las competencias de su interés.

Para darle cumplimiento a los objetivos específicos se plantean las siguientes **tareas de la investigación:**

- Ejecución de un estudio sobre los Jueces en Línea de mayor interés y su planificación regular de competencias.
- Selección de las tecnologías, herramientas, metodología y estándares que se necesitan para implementar la propuesta de solución.
- Definición de los requisitos funcionales y no funcionales de la propuesta de solución.

- Implementación de la propuesta de solución.
- Ejecución de las pruebas unitarias, de integración y de aceptación.

Para guiar el desarrollo de la investigación se usaron los siguientes **métodos científicos:**

Métodos Teóricos:

- Histórico-Lógico: Se utilizó para consultar la bibliografía referente al ACM-ICPC en sus diferentes niveles, su evolución y comportamiento. Permitió un mayor conocimiento sobre el ACM-ICPC así como estudiar la trayectoria histórica y la lógica del desarrollo de los Jueces en Línea.
- Analítico y sintético: Posibilitó analizar la bibliografía y realizar una síntesis de las fuentes para extraer los elementos útiles y de esta forma poder dar solución al problema.

La presente investigación se justifica en la necesidad de mostrar los anuncios de las competencias de otros Jueces en Línea desde un recurso único, el Juez en Línea Caribeño.

El presente trabajo de diploma está compuesto por un resumen, una introducción, tres capítulos, conclusiones, recomendaciones, bibliografía y anexos. A continuación se describen los principales aspectos abordados en cada uno de los capítulos:

- **Capítulo 1: Jueces en líneas y sistemas de publicación de anuncios de competencias, principales características y funcionalidades, tecnologías utilizadas en su desarrollo**. En este capítulo se abordan los Jueces en Línea, principales características y funcionalidades, así como los que son reconocidos tanto a nivel internacional como nacional. Se tratan además el estudio del estado del arte, los sistemas publicación de anuncios de competencias y sus características esenciales. Se abordarán las tecnologías, herramientas y metodología que serán utilizadas.
- **Capítulo 2: Descripción del módulo de publicación de anuncios de competencias para el Juez en Línea Caribeño.** En el presente capítulo se describe la propuesta de solución a la investigación. La propuesta se centra en realizar de forma automática la obtención y visualización de los anuncios de competencias en un conjunto de Jueces en Línea. Se describen las técnicas usadas para la recolección de los datos, se muestran segmentos de código y se generan artefactos según la metodología utilizada.
- **Capítulo 3: Implementación y validación del módulo de anuncios de competencias para el Juez en Línea Caribeño.** En el capítulo se evidencia la confección del ‘’Plan de *release*’’ de la

propuesta de solución, la que debe permitir al cliente y sus desarrolladores trazar una línea imaginaria de tiempo, para la obtención del producto. Por otra parte se elaboran los artefactos, tales como: el diagrama de componentes, el diagrama de despliegue y el diagrama de paquete. Finalmente se plasma el diseño y la ejecución de las pruebas a la solución obtenida.

Capítulo 1: Jueces en Línea y sistemas de publicación de anuncios de competencias, funcionalidades y tecnologías utilizadas en su desarrollo.

En este capítulo se abordan los Jueces en Línea, principales características y funcionalidades, así como los que son reconocidos tanto a nivel internacional como nacional. Se tratan además el estudio del arte, los sistemas publicación de anuncios de competencias y sus características esenciales. Se abordarán las tecnologías, herramientas y metodología que serán utilizadas.

1.1 Principales Jueces en Línea.

Los Jueces en Línea son aplicaciones web disponibles todo el tiempo, que automatizan la evaluación de las respuestas a problemas de programación competitiva. Surgieron para apoyar la preparación de los concursos de programación y potenciar el desarrollo de habilidades en la resolución de problemas de matemática, lógica y algoritmia. (Acosta, et al, 2013)

Existen muchos Jueces en Línea a nivel mundial. Estos son una fuente de entrenamiento, ya sea por la variedad de problemas de entrenamiento que poseen o por los concursos que realizan. Durante los estudios realizados se han identificado Jueces en Línea que tanto por su planificación de competencias o la calidad de las mismas, serían de interés para los usuarios del COJ realizarles su seguimiento. Esos Jueces en Línea son:

- El Juez en Línea de la Universidad de Valladolid (UVA)[1]
- El Juez en Línea Caribeño (COJ)[2]
- El Juez en Línea de la Universidad Federal de los Urales (Timus)[3]
- El Juez en Línea del Instituto Indio de Tecnología de Kanpur (Codechef)[4]
- El Juez en Línea Codeforces[5]
- El Juez en Línea HackerRank[6]

[1] http://uva.onlinejudge.org
[2] http://coj.uci.cu
[3] http://acm.timus.ru
[4] http://www.codechef.com
[5] http://www.codeforces.com
[6] https://www.hackerrank.com/

1.1.1 Caribbean Online Judge (COJ)

El Juez en Línea Caribeño radica en la UCI y representa a la comunidad caribeña de programación competitiva. Luego de realizarse un análisis el 5 de marzo de 2014, se pudo apreciar que cuenta con un archivo de 3 173 problemas de programación competitiva y realiza regularmente competencias. El COJ posee 20 569 usuarios registrados, representando a 929 instituciones de 169 países. Se han realizado 406 competencias y 771 579 envíos de solución. Ha ido ganando seguidores a lo largo de su tiempo de vida y se ha ido convirtiendo en un importante Juez en Línea.

Ventajas

- La aplicación web está internacionalizada, por lo que soporta español e inglés como idiomas.
- Es visitada frecuentemente por un gran número de usuarios buscando entrenamiento.
- Cuenta con un numeroso archivo de problemas de programación competitiva.
- Soporta 12 lenguajes de programación a usar en la solución de problemas: Bash, C, C#, C++, C++11, Java, Pascal, Perl, PHP, Haskell, Pyhton y Ruby.
- Accesible para usuarios nacionales e internacionales.

Desventajas

- No cuenta con una herramienta para ayudar a sus usuarios en el seguimiento de los anuncios de las competencias programadas en otros Jueces en Línea.

1.2 Sistemas de publicación de anuncios de competencias

Luego del análisis de la bibliografía existente se pudo constatar que no existía una definición precisa sobre los sistemas de publicación de anuncios de competencias. Se puede definir un sistema de publicación de anuncios de competencias como un sistema encargado de mostrar a sus usuarios un conjunto de los anuncios de las competencias programadas en Jueces en Línea en diversas partes del mundo para así facilitar su seguimiento.

Durante el proceso de investigación se identificó un sistema de publicación de anuncios de competencias de Bulgaria ya existente: clist[1].

1.2.1 Sistema de publicación de anuncios de competencias: clist

La aplicación clist presenta un listado de anuncios de competencias de programación planificadas a lo largo del mundo con un diseño simple pero no muy agradable para el usuario pues posee un mal contraste de colores. Se pueden observar tanto competencias en ejecución como las planificadas. Este listado de competencias ofrece como información las fechas de inicio y de fin de las competencias a efectuarse, así como su duración. En el caso de las competencias en ejecución se puede apreciar cuánto falta para su conclusión. Se puede observar el tiempo restante para el inicio de las competencias planificadas. Se muestra el nombre de la competencia y el sitio donde se llevará a cabo.

Este sistema cuenta con un conjunto de preferencias donde se puede elegir el huso horario para mostrar las fechas, para facilitar la realización de la conversión entre husos horarios. Cuenta con filtros para el listado de competencias, así como otras funcionalidades que facilitan la interacción con la información mostrada.

Ventajas

- Muestra las fechas y horarios en todos los husos horarios existentes.
- Muestra competencias que regularmente están planificadas pero que aún no se han publicado en el sitio que las gestionan.

Desventajas

- La aplicación no está internacionalizada.
- No está asociada a ningún Juez en Línea, por lo que resulta difícil para los usuarios encontrar el recurso en Internet.
- Posee un incorrecto contraste de colores.
- Resulta de difícil acceso para la mayoría de los cubanos al tratarse de un sitio en Internet.

[1] La aplicación web no posee un nombre distintivo. Se puede encontrar en la dirección: http://clist.by.

1.3 Tecnologías de desarrollo de *software*

Para el desarrollo del COJ los especialistas desarrollaron un estudio de las herramientas y tecnologías necesarias para su implementación, demostrando su eficiencia en el contexto para el cual fueron utilizadas. La solución en desarrollo, al formar parte modular del COJ deberá basar su selección en la facilidad de integración y evitar que existan inconsistencias en el código. Por lo cual el módulo estará sujeto a las herramientas y tecnologías de las que hace uso el Juez en Línea Caribeño.

1.3.1 Lenguajes de programación

Java

El lenguaje de programación Java es de propósito general, concurrente, basado en clases y orientado a objetos. El lenguaje de programación Java es basado en C y C++, pero es organizado bastante diferente, con un número de aspectos de C y C++ omitidos y algunas ideas de otros lenguajes incluidas. (Oracle, 2014)

Java es un lenguaje de programación caracterizado por ser simple, orientado a objetos, distribuido, interpretado, robusto, seguro, portable, de altas prestaciones, multitarea y multiplataforma.

HTML 5

HTML es la última versión del Lenguaje de Marcado de Hipertexto. El lenguaje que describe la estructura y el contenido semántico de un documento web. (Mozilla, 2015)

CSS 3

CSS es un lenguaje de hojas de estilos creado para controlar el aspecto o presentación de los documentos electrónicos definidos con HTML y XHTML. CSS es la mejor forma de separar los contenidos y su presentación y es imprescindible para crear páginas web complejas. (LibrosWeb, 2015)

JavaScript

Java script es un lenguaje de programación dinámico. Es utilizado como parte de los navegadores web, permitiendo interacción con el usuario, control del navegador, comunicación asincrónica y el cambio del contenido mostrado. JavaScript es clasificado como un lenguaje de *scripts* con tipado dinámico y funciones

de primera clase. Esta mezcla de características hace que soporte varios paradigmas de programación, soportando programación orientada a objetos, imperativa y funcional.

1.3.2 *Framework*

Según (Eguiluz, 2014) *framework* es un conjunto de herramientas, bibliotecas, convenciones y buenas prácticas que pretenden encapsular las tareas repetitivas en módulos genéricos fácilmente reutilizables.

Spring Framework

Es un *framework* para el desarrollo de aplicaciones y contenedor de inversión de control, de código abierto para la plataforma Java. Se ha ganado popularidad en la comunidad de Java como una alternativa, reemplazo o incluso adición a la JEE (*Java Enterprise Edition*).

1.3.3 Sistema Gestor de Bases de Datos

Un sistema gestor de bases de datos (DBMS, por sus siglas en inglés) es un sistema usado para gestionar la organización, almacenamiento, acceso, seguridad e integridad de los datos en una base de datos estructurada. (Altaviser, 2014)

PostgreSQL

Sistema gestor de base de datos relacional orientado a objetos, publicado bajo la licencia PostgreSQL. Es un programa libre y de código abierto, dirigido por una comunidad de desarrolladores llamada PGDG (*PostgreSQL Global Development Group*).

Como servidor de bases de datos, su función primaria es la de almacenar datos de forma segura y usando las mejores prácticas, para ser consultados luego por otras aplicaciones.

Esta herramienta fue seleccionada por los desarrolladores del COJ debido a que es una herramienta multiplataforma, además por las características distintivas que son muy a fines con el sistema desarrollado. Soporta una capacidad de almacenamiento en el orden de los TB[1] y posee buena escalabilidad.

[1] *Tera Bytes*

1.3.4 Servidor web

Servidores web

Un servidor web es un *software* responsable para aceptar solicitudes HTTP de clientes, que son conocidos como navegadores web, y servirles a ellos respuestas HTTP junto con contenidos adicionales, los cuales usualmente son páginas web como documentos HTML y objetos enlazados (imágenes, archivos CSS, entre otros). (Ubuntu, 2014)

Apache-Tomcat

Funciona como un contenedor de servidores desarrollado bajo el proyecto *Jakarta* en la *Apache Software Foundation*. *Tomcat* implementa las especificaciones de los servidores y de *Java Server Pages*.

1.3.5 Entorno Integrado de Desarrollo

Spring Tool Suite

Spring Tool Suite es un ambiente de desarrollo basado en Eclipse que es personalizado para el desarrollo de aplicaciones usando *Spring Framework.* Posee un ambiente para implementar, testear, ejecutar y desplegar aplicaciones desarrolladas usando *Spring*, incluyendo integración con otros servicios.

Este IDE (Entorno Integrado de Desarrollo por su nombre en español) es seleccionado sobre *Netbeans*, que ha sido utilizado anteriormente en el desarrollo del COJ por parte de sus desarrolladores, debido que se ha migrado la estructura del proyecto del COJ a la de *Spring Tool Suite.*

1.3.6 Herramienta de modelado

Herramientas CASE

Se puede definir a las herramientas de Ingeniería de *Software* Asistida por Ordenador (CASE por sus siglas en inglés), como aplicaciones informáticas destinadas a aumentar la productividad y la velocidad en el desarrollo de *software*, reduciendo el costo en términos de tiempo y dinero. Las herramientas CASE son útiles en todos los aspectos del ciclo de vida de desarrollo del *software*, en tareas como el proceso de realizar un diseño del proyecto, cálculo de costos, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores. (Pullés, 2003)

Varias son las herramientas CASE existentes, entre las que se pueden encontrar *Visual Paradigm*, *Erwin* y *Rational Rose.* Se define a *Visual Paradigm como* la herramienta CASE a ser utilizada por ser una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de *software*, es compatible con una amplia gestión de casos de uso y diseño de base de datos y proporciona medidas más eficaces en el análisis y diseño de sistemas. (VisualParadigm, 2014)

Después de analizadas las herramientas y tecnologías a utilizar se puede seleccionar la metodología de desarrollo de *software* a utilizar.

1.4 Metodología de desarrollo de *software*

La metodología de desarrollo es la encargada de guiar todo el proceso del *software*. Para elegir la metodología de desarrollo se deben tener en cuenta diferentes aspectos, como la cantidad de personas que integrarán el proyecto, el tiempo de desarrollo, los requisitos del cliente y el tamaño del proyecto. Existen dos clasificaciones para las metodologías de desarrollo de software, las ágiles y las tradicionales. Las ágiles enfatizan las comunicaciones cara a cara con el cliente en vez de llevar una documentación rígida. Las tradicionales se centran en el control del proceso implantando los artefactos que se deben originar, las actividades implicadas, las herramientas y las notaciones que se usarán.

1.4.1 Matriz de Boehm y Turner

El modelo propuesto por Barry Boehm y Richard Turner en su libro "*Balancing Agility and Discipline: A Guide for the Perplexed*", (Boehm, et al, 2003) es usado para fundamentar el enfoque de la metodología que se aplicará. Este modelo muestra una evaluación de las características del proyecto para determinar la idoneidad de un enfoque ágil o tradicional.

Según (Boehm, et al, 2003) las características medidas son:

Personal: Clasifica a los miembros del proyecto mediante las habilidades que presente cada uno, en tres niveles: Nivel 1 (principiante), Nivel 2 (intermedio) y Nivel 3 (experto).

El nivel Principiante se divide en tres subniveles:

- Nivel -1: Persona que puede tener habilidades técnicas, pero que es incapaz o no está dispuesto a colaborar o seguir los métodos compartidos.
- Nivel 1B: Persona que con el entrenamiento adecuado puede realizar trabajos de baja complejidad. Ejemplo, un método simple, refactorización simple, siguiendo los estándares y procedimientos de codificación, la realización de pruebas, entre otros. Con el tiempo y la experiencia puede dominar algunas habilidades de Nivel 1A.
- Nivel 1A: Persona que con el entrenamiento adecuado puede realizar trabajos de mediana complejidad. Con el tiempo y la experiencia puede dominar algunas habilidades de Nivel 2.

El nivel Intermedio o Nivel 2: Capaz de diseñar un método para adaptarse a una nueva situación teniendo como base algún precedente.

El nivel Experto o Nivel 3: Capaz de revisar un método (romper sus reglas) para adaptarse a una nueva situación sin precedentes.

Para que los proyectos con un enfoque ágil se desarrollen sin problemas, el porcentaje del personal capacitado debe pertenecer en su mayoría a los niveles 2 y 3. Si el equipo tiene un mayor porcentaje de principiantes, entonces un enfoque más tradicional podría ser el más apropiado.

Dinamismo: Evalúa la probabilidad de cambios. ¿Cómo es de cambiante el proyecto?, ¿Qué porcentaje de los requisitos pueden cambiar durante el proyecto?

Si los requisitos que pueden cambiar son cercanos al 50%, se tiende hacia el centro de la gráfica, en la zona ágil. Si es cercano al 1% entonces la tendencia es hacia cerca del exterior, en la zona tradicional, aunque esto no quiere decir que no se puede utilizar un enfoque ágil.

Cultura: Define cuál es el temperamento de la organización. ¿Acepta los cambios que se realizan y hasta se alimentan de ellos, o se apoya en el conocimiento del orden y la tradición?

Los métodos ágiles tienen mejores probabilidades de éxito en un ambiente de trabajo donde cada individuo tiene la libertad de tomar decisiones (grado de libertad) sin afectar el buen funcionamiento del proyecto. Mientras que los métodos más tradicionales se guían mediante el cumplimiento de políticas y procedimientos.

Tamaño: Se refiere a la cantidad de miembros que integren el equipo de desarrollo.

Los métodos ágiles son más fáciles de establecer, ejecutar y gestionar en equipos pequeños. Esto no quiere decir que grandes equipos no pueden trabajar con métodos ágiles, sólo que es mucho más difícil de lograr.

Criticidad: se refiere a la consecuencia de un fallo del sistema.

Boehm y Turner afirman que una metodología ágil es más adecuada para aplicaciones triviales donde la falla de los resultados del sistema no ocasione daños severos, por lo cual es menos aplicable en sistemas de misión crítica o de vida crítica.

Resultados

Personal

El equipo de desarrollo está formado por una persona categorizada según las categorías definidas anteriormente como intermedio o de nivel 2, por lo que su valor tiende al centro.

Dinamismo

Los requisitos funcionales no tendrán mucho cambio, por lo tanto su valor es cercano al extremo de la escala.

Cultura

Se dispone de bastante libertad a la hora de realizar el proyecto, por lo cual su valor en la escala es del 90%.

Tamaño

El equipo de desarrollo dispone de una sola persona que desempeña todos los roles. El valor que toma en la escala correspondiente al tamaño es de 1.

Criticidad

Un fallo del *software* no provocará daños mayores, por tanto su valor en la escala es cercano al centro.

Teniendo presente cada uno de los valores planteados se confeccionó la matriz que se presenta en la Figura 1.

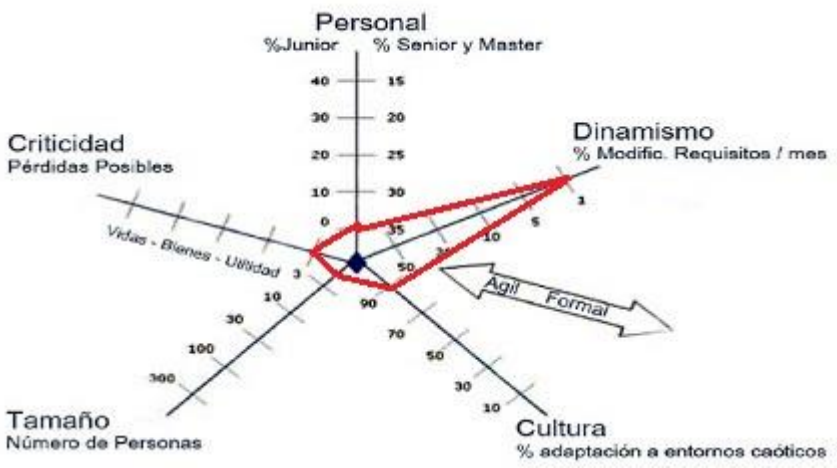

Figura 1: Matriz de Boehm y Turner (Elaboración propia)

En esta figura se representa el resultado del modelo propuesto por Barry Boehm y Richard Turner. Se ilustra la relación entre los factores que influyen en el desarrollo de un proyecto. Una tendencia de estos factores hacia el centro presupone el uso de un enfoque ágil, mientras que hacia el extremo se recomienda el uso de uno tradicional. El análisis de la matriz muestra una tendencia al centro de los resultados obtenidos, donde es sugerida la utilización de una metodología ágil. Dentro de las metodologías ágiles se selecciona SXP.

1.4.2 SXP

SXP es una metodología compuesta por SCRUM y XP que ofrece una estrategia tecnológica que a partir de la introducción de procedimientos ágiles permite actualizar los procesos de *software* fomentando el desarrollo de la creatividad. Se aumenta el nivel de preocupación y responsabilidad de los miembros del equipo, ayudando al líder del proyecto a tener un mejor control del mismo. SCRUM es una forma de gestionar un equipo de manera que trabaje de forma eficiente y de tener siempre medidos los progresos. XP más bien es una metodología encaminada para el desarrollo; consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo, al usuario final, pues es uno de los requisitos para llegar el éxito del proyecto.

SXP está especialmente indicada para proyectos de pequeños equipos de trabajo, rápido cambio de requisitos o requisitos imprecisos, muy cambiantes, donde existe un alto riesgo técnico y se orienta a una

entrega rápida de resultados y una alta flexibilidad. Esta metodología ayuda a que trabajen todos juntos, en la misma dirección, con un objetivo claro, permitiendo además seguir de forma clara el avance de las tareas a realizar, de forma que los jefes puedan ver día a día cómo progresa el trabajo. (Peñalver, 2008)

Se decide mantener la metodología SXP que se utilizó para desarrollar el COJ. Esta se adapta fácilmente a las características del equipo de desarrollo y del módulo de publicación de anuncios de competencias a desarrollar.

1.5 Propuesta de solución

Luego de haber realizado un análisis sobre el COJ se ha concluido que por sí solo no es capaz de dar solución a la problemática planteada inicialmente al no contar con una herramienta para dicho fin. El sistema de publicación de anuncios de competencias de programación clist.by tampoco puede resolver la problemática planteada debido a las desventajas planteadas. Por lo anteriormente expuesto se propone añadirle un módulo al COJ que permita agrupar y mostrar las competencias programadas en Jueces en Línea, aprovechando así las ventajas mostradas por el COJ, de forma que esto constituye la propuesta de solución a la presente investigación.

1.6 Conclusiones parciales

El estudio y diagnóstico realizado permitió identificar que de las soluciones asociadas al objeto de estudio a nivel internacional y nacional, ninguna resolvía el problema planteado. Se propuso el desarrollo de un módulo para el COJ que permita agrupar y mostrar los anuncios de las competencias programadas en un conjunto de Jueces en Línea.

El análisis de las tecnologías y herramientas luego del estudio bibliográfico correspondiente permitió identificar las más acordes para el desarrollo de la solución planteada. Se decidió utilizar las siguientes herramientas y tecnologías: SXP (como metodología para guiar el proceso de desarrollo), *Java* (JDK 1.7) (como lenguaje de programación), *Spring Tool Suite* 3.5.0 (como IDE de desarrollo), *Spring* 3.1.2 (como *framework* para el desarrollo), como servidor web se decidió utilizar *Apache-Tomcat* 7.0.40, como sistema gestor de bases de datos PostgreSQL 9.1 y *Visual Paradigm for UML* 8.0 (como herramienta de modelado).

Capítulo 2: Descripción del módulo de publicación de anuncios de competencias para el Juez en Línea Caribeño.

En el presente capítulo se describe la propuesta de solución a la investigación. La propuesta se centra en realizar de forma automática la obtención y visualización de los anuncios de competencias en un conjunto de Jueces en Línea. Se describen las técnicas usadas para la recolección de los datos, se muestran segmentos de código y se generan artefactos según la metodología utilizada.

2.1 Características del sistema

La solución que se propone a los inconvenientes planteados, es el desarrollo de un módulo al COJ capaz de mostrar a los usuarios los anuncios de competencias de programación competitiva programadas en un conjunto de Jueces en Línea. El módulo se encargará de obtener la información relacionada con los anuncios de competencias para mostrarla de una forma interactiva y de fácil acceso. Realizará también notificaciones de los anuncios y cambios en los horarios por medio del correo, Twitter, Facebook y el micro-blog del COJ.

2.2 Competencias de programación

Al realizar la recopilación de los anuncios de competencias se obtienen una serie de importantes datos relacionados con estos. Estos datos son utilizados para brindarle al usuario una vista más detallada de la competencia. A continuación se muestran los datos obtenidos de cada uno de los anuncios de competencias:

- Nombre
- Juez en línea en el cual se realizará
- Fecha y hora de inicio
- Fecha y hora de fin
- Dirección *URL*

2.3 Recopilación de los anuncios

El módulo realiza una conexión a Internet cada cierto intervalo de tiempo para realizar la recopilación de los anuncios de competencias. Posee un sistema de respuesta para cuando algún recurso donde se realiza la búsqueda no se encuentre disponible. Este sistema consiste en reintentar acceder a los recursos cada dos horas hasta lograr obtener la información buscada.

2.3.1 Páginas *coming* de los Jueces en Línea

Los Jueces en Línea en los cuáles se realizan competencias poseen una página web para mostrar la información relacionada sobre las próximas competencias a realizarse, las que están en curso y las finalizadas. La página de publicación de las próximas competencias a realizarse es la fuente de entrada de información del presente módulo.

Lo ideal sería que todos estos Jueces en Línea brindaran una Interfaz de Programación de Aplicaciones (API, por sus siglas en idioma inglés), como un servicio web, para realizar la extracción de los anuncios de competencias. En la mayoría de los casos no existe dicha API, por lo que se realiza un *parsing* de estas páginas para su obtención. En caso de existir, se consume este servicio. Siempre es más conveniente consumir el servicio web porque este está sujeto a menos cambios. Pequeñas modificaciones en la estructura del HTML de la página a realizarle el *parsing*, pueden provocar el mal funcionamiento del *parser*.

2.3.2 Parsing

El *parsing* de una página consiste en la extracción de la información importante a partir de su código HTML. El módulo para realizar esta operación utiliza la biblioteca jsoup-1.7.3. Esta biblioteca permite un fácil y rápido manejo de textos en lenguaje HTML.

El *parsing* de las páginas es llevado a cabo por clases que heredan de la clase abstracta WbParser.

```
public abstract class WbParser {
        protected WbSite site;
        protected String siteUrl;

        public abstract void init();

        public abstract List<WbContest> parse()  throws ConnectionErrorException;
```

```
        public WbSite getSite() {
            return site;
        }

        public void setSite(WbSite site) {
            this.site = site;
        }

        public String getSiteUrl() {
            return siteUrl;
        }

        public void setSiteUrl(String siteUrl) {
            this.siteUrl = siteUrl;
        }
}
```

A continuación se muestra como ejemplo la clase encargada del *parsing* del Juez en Línea Codechef.

```
@Component("wbCodechefParser")
public class WbCodechefParser extends WbParser {

   private String DATE_FORMAT = "yyyy-MM-dd hh:mm:ss";

        @Resource
        WbSiteDAO wbSiteDAO;

   public WbCodechefParser() {
      this.siteUrl = Config.getProperty("codechef.parsing.url");
   }

   @PostConstruct
   public void init() {
        site = wbSiteDAO.getSiteByUrl(Config.getProperty("codechef.url"));
        if(site == null) {
            site = new WbSite(0, Config.getProperty("codechef.name"),
Config.getProperty("codechef.url"), Config.getProperty("codechef.code"),
                              false, true, Integer.parseInt(Config.getProperty("codechef.timeanddateid")),
Config.getProperty("codechef.timeregion"));
            int sid = wbSiteDAO.insertSite(site);
            site.setSid(sid);
        }
   }
```

```
@Override
public List<WbContest> parse() throws ConnectionErrorException {
    Document doc = null;
    String strName = "";
    String strUrl = "";
    String strBegin = "";
    String strEnd = "";

    Date dateBegin = new Date();
    Date dateEnd = new Date();

    List<WbContest> contests = new LinkedList<WbContest>();
    SimpleDateFormat format = new SimpleDateFormat(DATE_FORMAT);

    try {
        doc = Jsoup.connect(siteUrl).timeout(20 * 1000).get();
    } catch (IOException ex) {
        throw new ConnectionErrorException(site.getSite());
    }

    Elements tables = doc.getElementsByClass("table-questions");

    for(int k = 0;k<2;k++) {
                Elements entries = tables.get(k).getElementsByTag("tr");

                for(int i = 1;i<entries.size();i++) {
                    Element entry = entries.get(i);

                    strName = entry.child(1).getElementsByTag("a").get(0).text();
                    strUrl = siteUrl + entry.child(1).getElementsByTag("a").get(0).attr("href");
                    strBegin = entry.child(2).text();
                    strEnd = entry.child(3).text();

                    try {
                        dateBegin = format.parse(strBegin);
                        dateEnd = format.parse(strEnd);
                    } catch (ParseException ex) {
                        return null;
                    }


                    WbContest contest = new WbContest(strUrl, strName, site.getSid(), dateBegin,
dateEnd);
```

```
                    contests.add(contest);
                }
            }

            return contests;
        }
}
```

2.4 Visualización de los anuncios

El módulo de anuncios de competencias de programación competitiva para el COJ brinda una interfaz web para su visualización y consulta. El usuario puede observar el nombre de la competencia, el Juez en Línea donde se realizará, la fecha y hora de inicio y de fin, el tiempo restante para el comienzo y su estado. En el caso de las fechas y horas de inicio y fin de las competencias, se provee un vínculo hacia *Time and Date*[1]. Esta aplicación web se encargará de realizar la conversión de horas en los diferentes husos horarios existentes hacia la ciudad sobre la cual el usuario está interesado. El tiempo restante para el inicio de una competencia se muestra como un contador regresivo. Al comenzar una competencia, el conteo regresivo cambiará a otro conteo regresivo, pero indicando el fin de esta, además su estado cambiará a "En ejecución" (*Running*).

Esta página de visualización posee dos filtros para mejorar la experiencia del usuario y su funcionalidad. Se puede filtrar su contenido especificando un Juez en Línea en particular sobre el cual se desee consultar sus anuncios o simplemente mostrarlos todos. El usuario autenticado desde su página de perfil o desde esta misma interfaz, puede seguir uno o más Jueces en Línea. Seguir un Juez en Línea permite realizar un filtrado desde esta página de visualización usando este criterio.

A continuación en la Figura 2 se puede apreciar una vista de la página de visualización de anuncios de competencias.

[1] Time and date es una aplicación disponible bajo la URL www.timeanddate.com para realizar conversiones de tiempo entre los diferentes husos de horarios en el mundo.

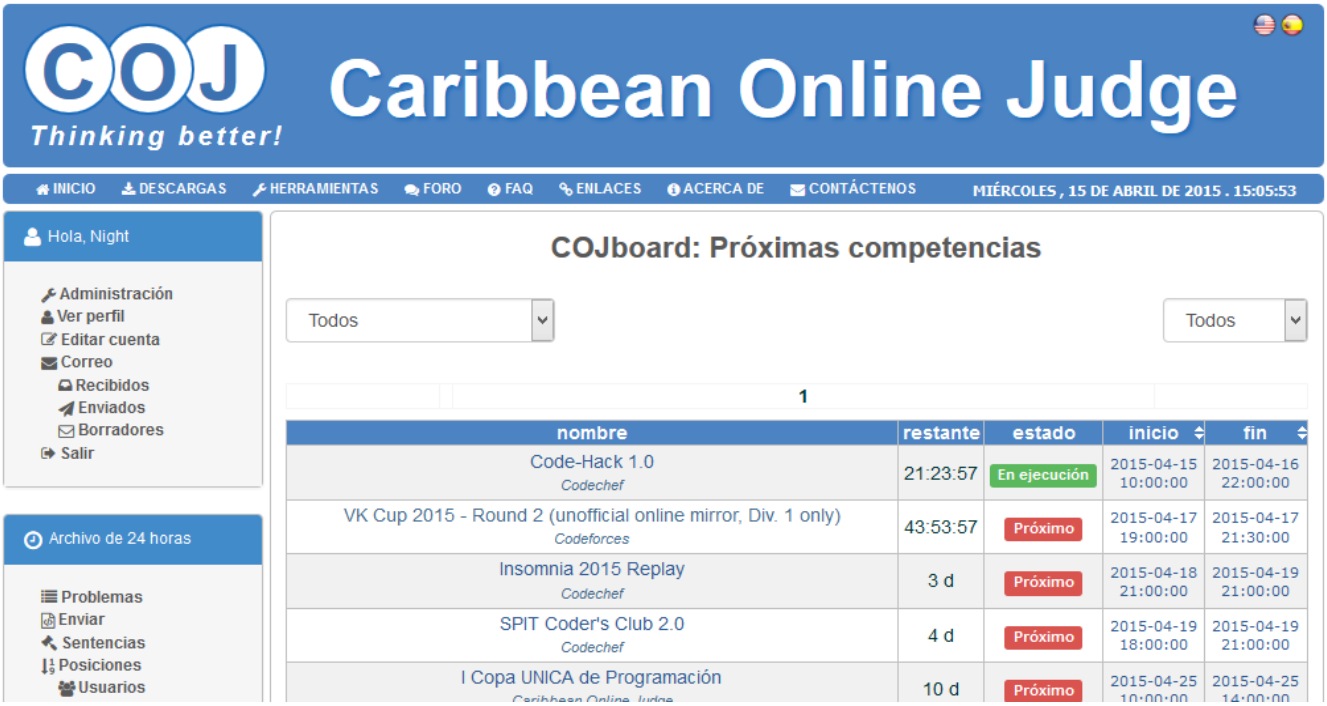

Figura 2: Interfaz web de visualización de los anuncios de competencias.

2.5 Notificaciones

El módulo de anuncios de competencias de programación competitiva para el COJ posee un submódulo encargado del envío y publicación de notificaciones por diferentes vías. Existen tres eventos que generaran notificaciones:

- Publicación de una nueva competencia.
- Cambios en el horario o la fecha de alguna competencia.
- Restan menos de 48 horas para el inicio de alguna competencia.

El módulo genera cada vez que detecta alguno de estos eventos una notificación. Las notificaciones son enviadas periódicamente en intervalo de una vez al día a las 9:00 horas (horario de Cuba) a todos los usuarios que hayan especificado en su perfil su deseo de recibirlas por medio del correo. También estas notificaciones son publicadas en la cuenta del COJ de *Facebook* y de *Twitter*, y en su micro-blog.

Las clases encargadas del envío o publicación de las notificaciones implementan la *interface WbNotificationService*.

```
public interface WbNotificationService {
        public void sendNotifications(List<WbSite> newcontestNotificationSites,
            List<WbSite> changedNotificationSites, List<WbContest> nearContestNotification);
}
```

2.6 Administración

Para brindar funciones de administración el módulo contiene un submódulo para lograr un mejor control. Los administradores pueden adicionar competencias manualmente de Jueces en Línea para los cuales no se está realizando *parsing*. Además, pueden editar, eliminar o simplemente observar todo el trabajo automático que ha realizado el módulo. En caso de algún fallo, también existe una interfaz visual para ejecutar manualmente el *parsing* de algún Juez en Línea en particular o para deshabilitar este en caso de existir algún error.

2.7 Seguridad

Para garantizar la confiabilidad e integridad de la información ante posibles accesos no autorizados, el módulo utiliza el mismo mecanismo que el sistema donde será integrado, *Spring Security* en su versión 3.0.

2.8 Requerimientos

Después de analizadas las características, se identificaron los requisitos funcionales y no funcionales que tendrá el módulo que se desarrolla. Los requisitos funcionales son capacidades o condiciones que el sistema debe cumplir, mientras que los no funcionales son propiedades o cualidades que el producto debe tener. Los requisitos forman parte del modelo del sistema que se desarrolla (Pressman, 2005). A continuación se presentan los requisitos funcionales y no funcionales extraídos.

2.8.1 Requisitos funcionales

RF1 - Mostrar listado de próximas competencias.

RF2 - Obtener los anuncios de competencias del COJ.

RF3 - Obtener los anuncios de competencias de Codeforces.

RF4 - Obtener los anuncios de competencias de Codechef.

RF5 - Obtener los anuncios de competencias de Timus.

RF6 - Obtener los anuncios de competencias de HackerRank.

RF7 - Obtener los anuncios de competencias del UVa.

RF8 - Administrar *parsers*.

RF9 - Gestionar anuncios de competencias.

RF10 - Gestionar sitios.

RF11 - Enviar notificaciones por correo.

RF12 - Publicar notificaciones en Twitter.

RF13 - Publicar notificaciones en Facebook.

RF14 - Publicar notificaciones en el micro-blog del COJ.

2.8.2 Requisitos no funcionales

Usabilidad

RNF1 - El módulo brindará facilidades para su utilización por usuarios de otros husos horarios en cuanto a la conversión de las horas.

Confiabilidad

RNF2 - En caso de que el sistema presente alguna falla, los errores se deben mostrar sin detalles de información que pueda comprometer la seguridad e integridad del sistema, solo podrán mostrarse detalles de la información para los usuarios con privilegios de administración dentro del sistema.

Seguridad

RNF3 - El módulo establecerá mecanismos para garantizar la confiabilidad e integridad de la información ante posibles accesos no autorizados.

Accesibilidad

RNF4 - Se podrá acceder a la aplicación desde *Mozilla Firefox* (v3.0 o superior), *Chrome* (v5.0 o superior), *Opera* (v9.0 o superior) y *Safari* (v4.0 o superior).

Soporte

RNF5 - El sistema debe dar la posibilidad de ser mejorado, así como, de incorporarle nuevas técnicas en caso de ser necesarias.

Interfaz

RNF6 - El diseño debe cumplir con los estándares internacionales de desarrollo web aplicados al COJ, con una interfaz sencilla e intuitiva que garantice un mayor nivel de usabilidad.

Interfaz de Hardware

RNF7 - Requisitos mínimos de *hardware* para el servidor: *Core* 2 DUO 2.0 GHZ y 2GB RAM.

Interfaz de Software

RNF8 - PC Servidor web. Sistema Operativo: *Ubuntu* 13.04 o superior. Servidor web: apache *Tomcat* 7.x o superior Lenguaje de programación: JDK 1.7.

RNF9 - PC Servidor de BD. Sistema Operativo: *Ubuntu* 13.04 o superior. Sistema Gestor de Base de Datos PostgreSQL v8.4 o superior.

RNF10 - Utilizar el entorno de ejecución java "OpenJDK-JRE" en su versión 7 o superior.

Requisitos legales, de derecho de autor y otros.

RNF11- Las tecnologías y herramientas utilizadas deben estar bajo licencia GPL[1].

RNF12- El sistema posee una sección donde se publican los términos y condiciones de uso que reglamentan el acceso y uso del COJ y de todos sus subdominios o dominios secundarios incluyendo sus contenidos y servicios, puestos a disposición que deben ser de estricto cumplimiento por parte de los usuarios.

RNF13- Los derechos de autor sobre la información de terceros que se publica en el COJ pertenecen a sus respectivos autores, su publicación en el COJ se realiza sin ánimos de lucro con propósito informativo.

2.9 Lista de reserva del producto

La lista de reserva del producto es una lista priorizada de trabajo para el equipo de desarrollo que se deriva de la hoja de ruta y sus requisitos. Los elementos más importantes se muestran en la parte superior por lo que el equipo conoce lo que debe realizar primero. (Radigan, 2014)

Asignado a	Elemento	Descripción	Estimación	Estimado por
Requisitos funcionales				

[1] *General Public Licence*, por su nombre en inglés.

Prioridad Alta				
Programador Eddy Roberto Morales Pérez	1	Mostrar listado de próximas competencias.	1	Programador
Programador Eddy Roberto Morales Pérez	2	Obtener los anuncios de competencias del COJ.	0.5	Programador
Programador Eddy Roberto Morales Pérez	3	Obtener los anuncios de competencias de Codeforces.	0.5	Programador
Programador Eddy Roberto Morales Pérez	4	Obtener los anuncios de competencias de Codechef.	0.5	Programador
Programador Eddy Roberto Morales Pérez	5	Obtener los anuncios de competencias de Timus.	0.5	Programador
Programador Eddy Roberto Morales Pérez	6	Obtener los anuncios de competencias de UVa.	0.5	Programador
Programador Eddy Roberto Morales Pérez	7	Obtener los anuncios de competencias de HackerRank.	0.5	Programador
Programador Eddy Roberto Morales Pérez	8	Enviar notificaciones por correo.	0.5	Programador
Prioridad Normal				

Programador Eddy Roberto Morales Pérez	9	Administrar parsers.	1	Programador
Programador Eddy Roberto Morales Pérez	10	Gestionar anuncios de competencias.	2	Programador
Programador Eddy Roberto Morales Pérez	11	Gestionar sitios.	2	Programador
Programador Eddy Roberto Morales Pérez	12	Publicar notificaciones en Twitter.	0.5	Programador
Programador Eddy Roberto Morales Pérez	13	Publicar notificaciones en Facebook.	0.5	Programador
Programador Eddy Roberto Morales Pérez	14	Publicar notificaciones en el micro blog del COJ.	0.5	Programador
Requisitos no funcionales				
Usabilidad				
	1	El módulo brindará facilidades para su utilización por usuarios de otros husos horarios en cuanto a la conversión de las horas.		
Confiabilidad				

	2	En caso de que el sistema presente alguna falla, los errores se deben mostrar sin detalles de información que pueda comprometer la seguridad e integridad del sistema, solo podrá mostrarse detalles de la información para los usuarios con privilegios de administración dentro del sistema.		
Seguridad				
	3	El módulo establecerá mecanismos para garantizar la confiabilidad e integridad de la información ante posibles accesos no autorizados.		
Accesibilidad				
	4	Se podrá acceder a la aplicación desde *Mozilla Firefox* (v3.0 o superior), *Chrome* (v5.0 o superior), *Opera* (v9.0 o superior) y *Safari* (v4.0 o superior).		
Soporte				
	5	El sistema debe dar la posibilidad de ser mejorado, así como de incorporarle nuevas técnicas en caso de ser necesarias.		
Interfaz				
	6	El diseño debe cumplir con los estándares internacionales de desarrollo web aplicados al COJ, con una interfaz sencilla e intuitiva que garantice un mayor nivel de usabilidad.		
Interfaz de *Hardware*				
	7	Requisitos mínimos de *hardware* para el servidor: *Core* 2 DUO 2.0 GHZ y 2GB RAM.		
Interfaz de *Software*				

	8	PC Servidor web. Sistema Operativo: *Ubuntu* 13.04 o superior. Servidor web: apache *Tomcat* 7.x o superior Lenguaje de programación: JDK 1.7.		
	9	PC Servidor de BD. Sistema Operativo: Ubuntu 13.04 o superior. Sistema Gestor de Base de Datos PostgreSQL v8.4 o superior.		
	10	Utilizar el entorno de ejecución java “OpenJDK-JRE’’ en su versión 7 o superior.		
Requisitos legales, de derecho de autor y otros				
	11	Las tecnologías y herramientas utilizadas deben estar bajo licencia GPL.		
	12	El sistema posee una sección donde se publican los términos y condiciones de uso que reglamentan el acceso y uso del COJ y de todos sus subdominios o dominios secundarios incluyendo sus contenidos y servicios, puestos a disposición que deben ser de estricto cumplimiento por parte de los usuarios.		
	13	Los derechos de autor sobre la información de terceros que se publica en el COJ pertenecen a sus respectivos autores, su publicación en el COJ se realiza sin ánimos de lucro con propósito informativo.		

Tabla 1: Lista de reserva del producto.

2.10 Historias de usuario

Las historias de usuario sirven para los mismos propósitos que los casos de uso, pero no son lo mismo. Son usadas para crear estimados de tiempo. Las historias de usuario son escritas por los clientes como

requerimientos que el sistema debe cumplir. Están en un formato de aproximadamente tres oraciones escritas en la terminología del cliente en lenguaje común. (Wells, 2013)

A continuación se muestra un conjunto de las historias de usuarios más importantes. El resto se pueden consultar en el Anexo 1.

Historia de usuario	
Número: 1	**Nombre Historia de Usuario:** Mostrar listado de próximas competencias.
Usuario: Usuario	**Iteración asignada:** 1
Prioridad en Negocio: Alta	**Puntos Estimados:** 1
Riesgo en Desarrollo: Alto	**Puntos Reales:** 1
Descripción: Como usuario, quiero que el usuario no autenticado o autenticado pueda obtener el listado de las próximas competencias a efectuarse. El usuario autenticado podrá filtrar por su fuente y por la lista de Jueces en Línea a los que sigue para mejorar la usabilidad.	
Observaciones:	

Tabla 2: Historia de usuario: Mostrar listado de próximas competencias.

Historia de usuario	
Número: 3	**Nombre Historia de Usuario:** Obtener los anuncios de competencias de Codeforces.
Usuario: Tiempo	**Iteración asignada:** 1
Prioridad en Negocio: Alta	**Puntos Estimados:** 0.5
Riesgo en Desarrollo: Alto	**Puntos Reales:** 0.5
Descripción: Como usuario, quiero que el sistema realice el *parsing* de la página de competencias del Juez en Línea Codeforces para conocer los datos de las competencias existentes.	
Observaciones: Se generan y almacenan las notificaciones correspondientes.	

Tabla 3: Historia de usuario: Obtener los anuncios de competencias de Codeforces.

Historia de usuario	
Número: 8	**Nombre Historia de Usuario:** Enviar notificaciones por correo.

Usuario: Tiempo	**Iteración asignada:** 1
Prioridad en Negocio: Alta	**Puntos Estimados:** 0.5
Riesgo en Desarrollo: Medio	**Puntos Reales:** 0.5
Descripción: Como usuario, quiero que el sistema envíe por medio del correo las notificaciones pendientes recolectadas en las últimas 24 horas a todos los usuarios que lo hayan solicitado en su perfil de usuario para lograr una mejor difusión de la información.	
Observaciones:	

Tabla 4: Historia de usuario: Enviar notificaciones por correo.

Historia de usuario	
Número: 18	**Nombre Historia de Usuario:** Publicar notificaciones en Twitter.
Usuario: Tiempo	**Iteración asignada:** 2
Prioridad en Negocio: Normal	**Puntos Estimados:** 0.5
Riesgo en Desarrollo: Bajo	**Puntos Reales:** 0.5
Descripción: Como usuario, quiero que el sistema publique en Twitter las notificaciones pendientes recolectadas en las últimas 24 horas usuario para lograr una mejor difusión de la información.	
Observaciones:	

Tabla 5: Historia de usuario: Publicar notificaciones en Twitter.

2.11 Tareas de ingeniería.

Según (Cunningham, 2014), las tareas de ingeniería son una herramienta que permite asignar las tareas relacionadas con cada historia de usuario a los involucrados del proyecto. Son la representación gráfica de las responsabilidades asignadas a cada miembro del equipo de desarrollo.

A continuación se muestra un conjunto de las tareas de ingeniería más importantes. El resto se pueden consultar en el Anexo 2.

Tarea de Ingeniería	
Número Tarea: 1	**Número Historia de Usuario:** 1

Nombre Tarea: Programación de la interfaz visual para mostrar el listado de los anuncios de las competencias.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 0.5
Fecha Inicio: 12/01/2015	**Fecha Fin:** 14/01/2015
Programador Responsable: Eddy Roberto Morales Pérez	
Descripción: Se lleva a cabo la programación de la interfaz que muestra los anuncios de las próximas competencias a efectuarse.	

Tabla 6: Tarea de ingeniería 1.

Tarea de Ingeniería	
Número Tarea: 3	**Número Historia de Usuario:** 3
Nombre Tarea: Programación del *parser* para obtener los anuncios de las competencias de Codeforces.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 0.5
Fecha Inicio: 19/01/2015	**Fecha Fin:** 21/01/2015
Programador Responsable: Eddy Roberto Morales Pérez	
Descripción: Se lleva a cabo la programación del *parser* para obtener los anuncios de las competencias de Codeforces.	

Tabla 7: Tarea de ingeniería 3.

Tarea de Ingeniería	
Número Tarea: 8	**Número Historia de Usuario:** 8
Nombre Tarea: Programación de la clase para enviar notificaciones por correo.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 0.5
Fecha Inicio: 05/02/2015	**Fecha Fin:** 07/02/2015
Programador Responsable: Eddy Roberto Morales Pérez	
Descripción: Se lleva a cabo la implementación de la *interface WbNotificationService* para el envío de notificaciones a los usuarios por medio del correo.	

Tabla 8: Tarea de ingeniería 8.

Tarea de Ingeniería	
Número Tarea: 18	**Número Historia de Usuario:** 18
Nombre Tarea: Programación de la clase para publicar notificaciones en Twitter.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 0.5
Fecha Inicio: 16/03/2015	**Fecha Fin:** 18/03/2015
Programador Responsable: Eddy Roberto Morales Pérez	
Descripción: Se lleva a cabo la implementación de la *interface WbNotificationService* para publicar notificaciones en Twitter.	

Tabla 9: Tarea de ingeniería 18.

2.12 Diagrama de clases de diseño

El diagrama de clases del diseño es la representación gráfica de las clases que serán implementadas en el sistema. Este diagrama muestra las especificaciones y detalles más concretos de cada clase del sistema, así como su relación de asociación, composición o agregación existentes entre ellas (Ambler, 2003).

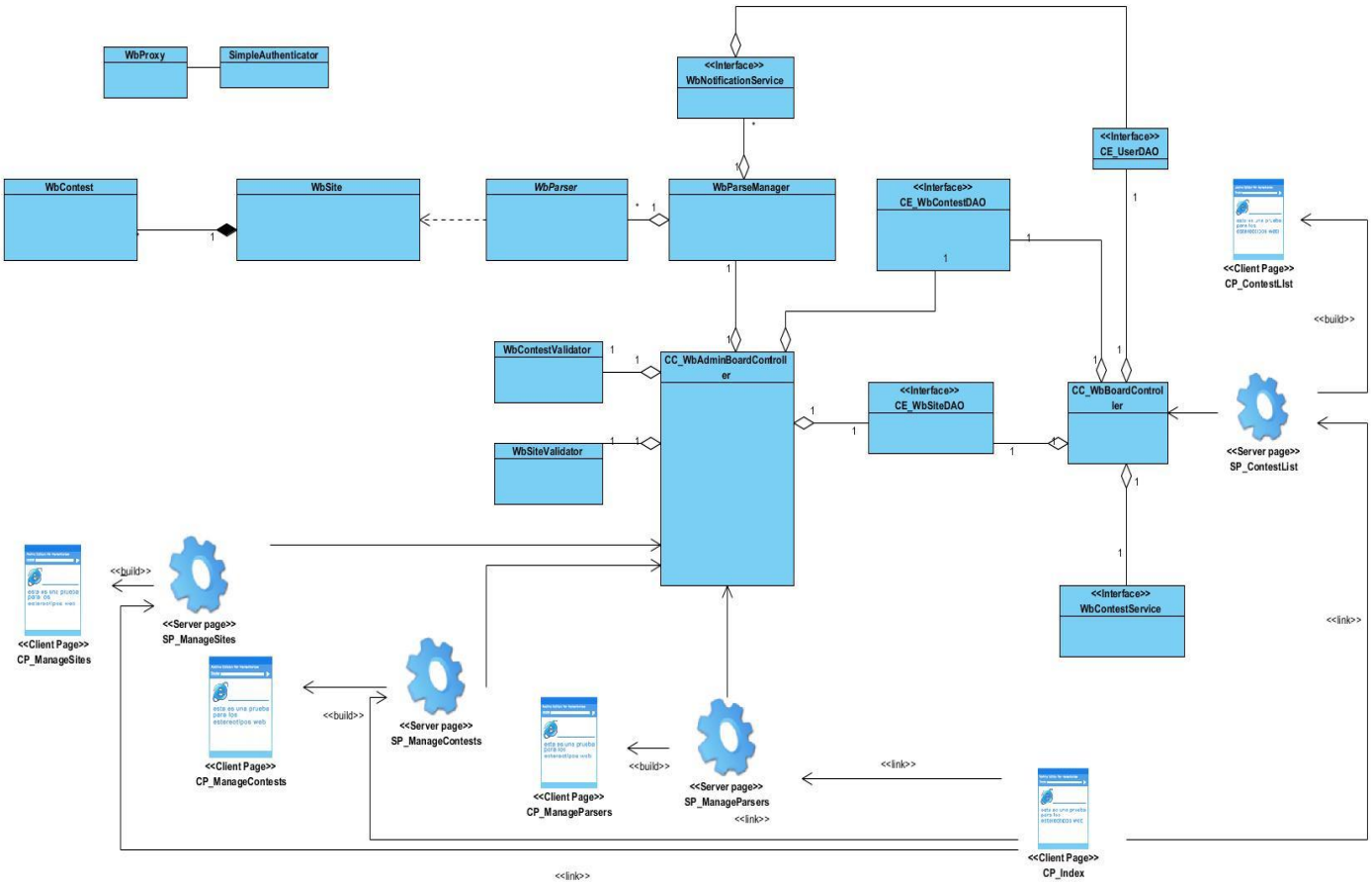

Figura 3: Diagrama de clases del diseño.

Este diagrama de clases del diseño (Figura 3) muestra completamente la estructura del módulo. La vista está representada por las *Server Page* y las *Client Page.* Las clases controladoras son *WbBoardController* y *WbAdminBoardController*. El modelo lo conforman las clases *WbContest*, *WbSite*, *WbSiteDAO* y *WbContestDAO*. Las demás clases son las encargadas de la realización del *parsing*, del envío y publicación de las notificaciones y configuración de la conexión a Internet.

La figura (Figura 4) describe las clases encargadas de la obtención de la información, el *parsing*. Todas heredan e implementan la clase abstracta *WbParser*. En el caso de *WbRestCojParser,* esta consume un servicio web del COJ.

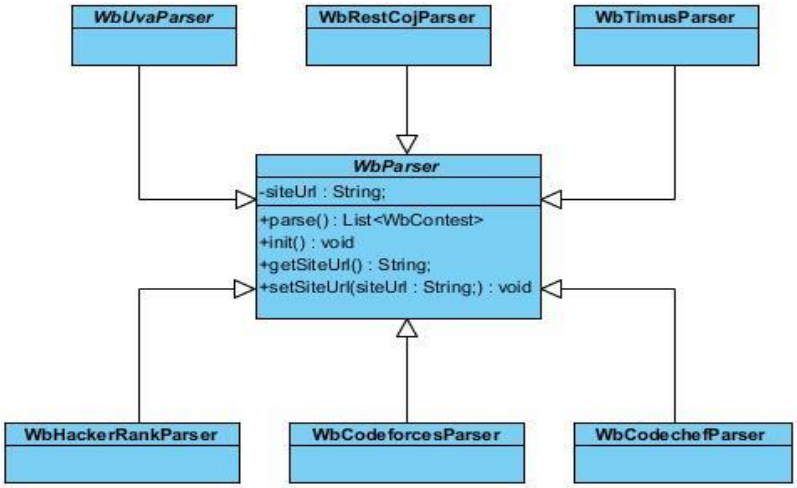

Figura 4: Diagrama de clases de los parsers.

El siguiente diagrama (Figura 5) describe la estructura para el envío y publicación de notificaciones. Todas las clases encargadas de esta función implementan la *interface WbNotificacionService*.

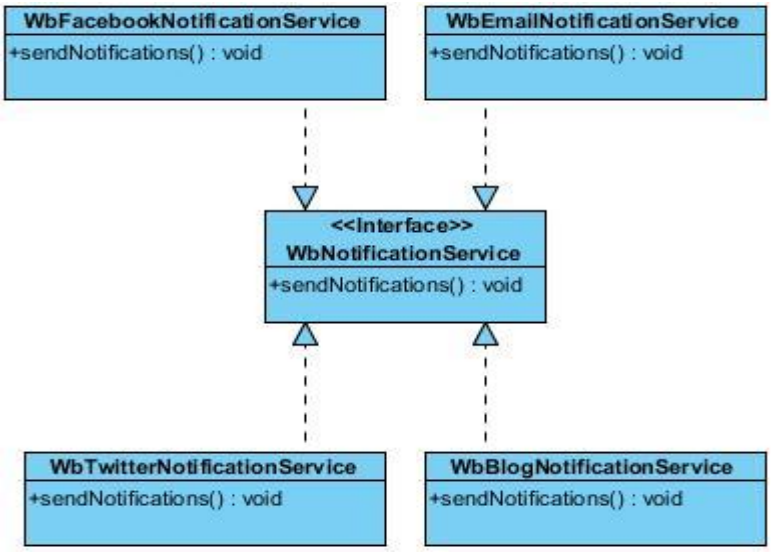

Figura 5: Diagrama de clases del servicio de notificaciones.

2.13 Diagrama del modelo de datos

Un modelo de datos es la representación de un fenómeno de la realidad objetiva a través de los objetos, sus propiedades y las relaciones que se establecen entre ellos (Matos, 1999). El Modelo de datos es el lenguaje orientado a describir la Base de Datos, permite además describir los elementos de la realidad que intervienen en el problema dado y la forma en que se relacionan entre sí.

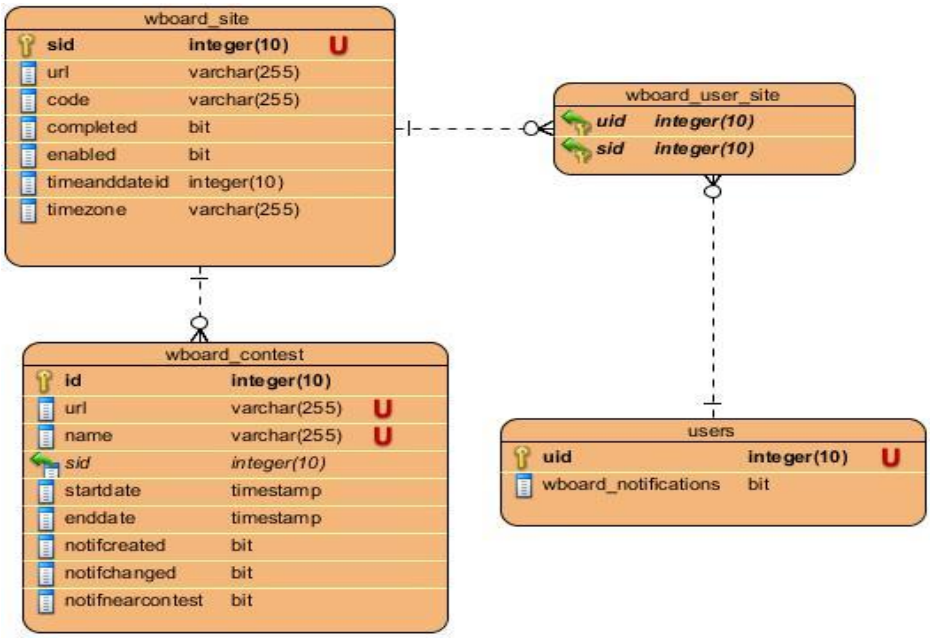

Figura 6: Diagrama del modelo de datos.

La base de datos está formada por cuatro tablas. Todas las tablas son nuevas a excepción de users. La tabla wboard_site contiene la información relacionada con los Jueces en Línea. La tabla wboard_contest almacena la información relacionada con las competencias. La tabla wboard_user_site es creada por la relación mucho a mucho entre wboard_site y users para representar los Jueces en Línea que los usuarios siguen.

2.14 Arquitectura

El estilo arquitectónico o el patrón de arquitectura de *software* empleado para el desarrollo de la aplicación es la arquitectura Modelo Vista Controlador (MVC) (Figura 7). Se basa en separar los datos de la

aplicación, la interfaz de usuario, y la lógica del negocio en tres capas diferentes, agrupa el código según su función. El modelo define el comportamiento, los datos de la aplicación y la lógica del negocio; la base de datos pertenece a esta capa. La vista muestra la información que emplean los usuarios para interactuar con la aplicación. La capa controlador es la que descifra las acciones realizadas por los usuarios, realiza las llamadas al modelo para obtener la información y se las envía a la vista para que las muestre a los usuarios. La vista y el controlador dependen del modelo, el cual no depende de las otras clases. Esta separación permite construir y probar el modelo independientemente de la representación visual (Bascón, 2011).

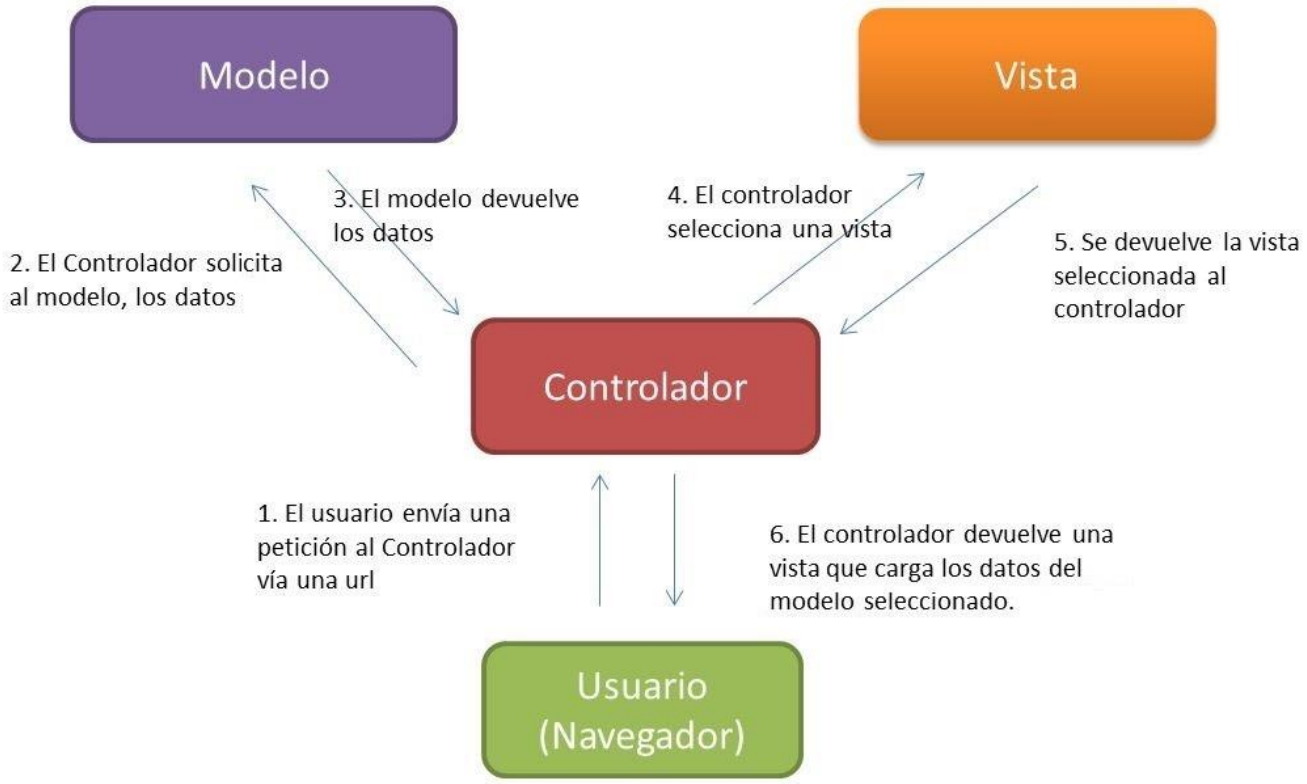

Figura 7: Modelo- Vista-Controlador

El framework *Spring* utilizado para el desarrollo de la solución basa su funcionamiento en la implementación del patrón MVC. Propone una estructura basada en modelos (entidades y clases DAO), vistas (ficheros ".jsp") y controladores (ficheros ".java") que interactúan entre sí del mismo modo que define

el patrón. Es seleccionado este estilo por las ventajas que brida su estructura, además de ser el que utiliza el COJ, plataforma donde será integrado el módulo de publicación de anuncios de competencias de programación.

2.15 Patrones de diseño

En ingeniería del *software*, un patrón es una solución ya probada y aplicable a un problema que se presenta una y otra vez en el desarrollo de distintas aplicaciones y en distintos contextos (*Pattern Definitions*, 2012). Un patrón no es una solución en forma de código directamente, sino una descripción de cómo resolver el problema y ante qué circunstancias es aplicable.

Patrones GRASP

En el diseño de la aplicación se hace uso de los patrones **GRASP**; acrónimo de *General Responsibility Assignment Software Patterns,* patrones generales de software para asignar responsabilidades.

Experto

Se basa en que la responsabilidad de realizar una tarea es de la clase que posee los datos involucrados; una clase contiene toda la información necesaria para realizar la tarea que tiene encomendada. Este patrón se evidencia en las clases que heredan de la clase *WbParser*. Estas clases cuentan con la información necesaria para cumplir la responsabilidad de la extracción de los anuncios de los Jueces en Línea. También se puede observar su utilización en las implementaciones de la *interface WbNotificationService*, encargadas del envío y publicación de las notificaciones.

Creador

Este patrón como su nombre lo indica es el que crea, el que guía la asignación de responsabilidades relacionadas con la creación de objetos, se asigna la responsabilidad de que una clase B cree un objeto de la clase A. Este patrón se evidencia en las clases que heredan de la clase *WbParser*, encargadas de crear instancias de los objetos *WbSite* y *WbContest*.

Controlador

Se asocia con operaciones del sistema y respuestas a sus eventos, tal como se relacionan los mensajes y los métodos. El controlador delega en otros objetos el trabajo que se necesita hacer pero coordina o

controla la actividad. Este patrón se evidencia en las clases controladoras *WbBoardController* y *WbParserManager* que delega las funcionalidades a otras clases.

Alta Cohesión

Se manifiesta en la medida en que se relacionan las clases y el grado de focalización de las responsabilidades de un elemento. Cada elemento del diseño debe realizar una labor única dentro del sistema, no desempeñada por el resto de los elementos, una clase con baja cohesión hace muchas cosas no relacionadas o hace demasiado trabajo. En la solución este patrón es el resultado de asignar responsabilidades únicas a cada uno de los componentes (controladores *WbBoardController.java, WbBoardAdminController.java* y modelos: *WbSiteDAO,* Wb*ContestDAO*).

Bajo Acoplamiento

Asigna la responsabilidad de controlar el flujo de eventos del sistema a clases específicas. Esto facilita la centralización de actividades como las validaciones y la seguridad de la aplicación.

Patrones GoF

Además se aplican algunos patrones *Gang of Four* (GoF), también conocidos como los patrones de la pandilla de los cuatro (abstracción, encapsulamiento, herencia y polimorfismo).

Singleton

El patrón *Singleton* (instancia única) está diseñado para restringir la creación de objetos pertenecientes a una clase o el valor de un tipo a un único objeto. Su intención consiste en garantizar que una clase sólo tenga una instancia y proporcionar un punto de acceso global a ella. El patrón *Singleton* se implementa en todos los *beans* del módulo, pues este es su comportamiento por defecto.

Objeto de Acceso a Datos

El patrón *Data Access Object* (DAO), como se conoce en inglés, crea una interfaz para encapsular los accesos a las fuentes de datos. Es utilizado para brindar una abstracción entre el uso y la implementación del acceso a la base de datos. Especial atención requiere el uso de dicho patrón en los objetos DAO que se conectan a la base de datos, pues evita la creación de múltiples conexiones por los diferentes usuarios.

En la aplicación se utiliza otro patrón de diseño que a pesar de no incluirse en ninguna categoría específica, juega un papel clave dentro de las nuevas tendencias del desarrollo de este tipo de proyectos, la **Inyección de dependencias,** base del funcionamiento del *Framework Spring***.** Se utiliza para inyectar en los controladores cada uno de los modelos requeridos para su correcto funcionamiento.

2.16 Estilo de código

Los estilos son normas usadas para escribir código y que incluyen una gran gama de aspectos dentro de proceso de codificación. Un buen estilo de programación debe aportar a la eficiencia del proceso de desarrollo, logrando que los programas sean robustos y comprensibles (Muñoz Caro, et al, 2002).

Independientemente de que las plataformas de desarrollo influyan en cierto modo a la formación de estilos de codificación, es responsabilidad de cada equipo determinar el suyo. A la hora de confeccionar un estilo de código, hay aspectos fundamentales que se deben tener en cuenta: notación, comentarios, indentación, espacios y líneas en blanco, longitud máxima de las líneas de caracteres, declaración de variables, declaración de constantes y parámetros de los métodos (Muñoz Caro, et al, 2002).

Para el desarrollo del módulo se usa el estilo definido para el lenguaje de programación Java (López Gaona, 2011). A continuación se presenta un breve ejemplo de su uso.

```
package cu.uci.coj.board.util;

import java.net.Authenticator;
import java.net.PasswordAuthentication;

public class SimpleAuthenticator extends Authenticator {
	private String username;
	private String password;

	public SimpleAuthenticator(String username, String password) {
		this.username = username;
		this.password = password;
	}

	protected PasswordAuthentication getPasswordAuthentication() {
		return new PasswordAuthentication(username, password.toCharArray());
	}
}
```

2.17 Conclusiones parciales

La realización el diseño permitió obtener una propuesta de solución que representa el módulo de una manera fácil de interpretar para su implementación. Quedó definido como será realizado el *parsing*, que datos serán extraídos y como se mostrarán al usuario, como se gestionará el envío y publicación de las notificaciones generadas y de qué manera será realizada la administración del módulo.

Los requisitos funcionales y no funcionales obtenidos a partir del proceso de identificación de los requisitos, sirvieron de guía para desarrollar las distintas funcionalidades y de este modo satisfacer las necesidades detectadas. Los artefactos generados según la metodología de desarrollo utilizada y los patrones de arquitectura y diseño descritos, constituyeron una guía fundamental para la construcción de la propuesta de solución.

Capítulo 3: Implementación y validación del módulo de competencias para el Juez en Línea Caribeño.

En el capítulo se evidencia la confección del ‘’Plan de *release*’’ de la propuesta de solución, la que debe permitir al cliente y sus desarrolladores trazar una línea imaginaria de tiempo, para la obtención del producto. Por otra parte se elaboran los artefactos, tales como: el diagrama de componentes, el diagrama de despliegue y el diagrama de paquete. Finalmente se plasma el diseño y la ejecución de las pruebas a la solución obtenida.

3.1 Plan de *release*

De acuerdo a las funcionalidades descritas en las Historias de Usuarios (HU) y la prioridad asignada a cada una, se planificaron dos iteraciones para obtener la solución propuesta.

Release	Descripción de la iteración	Orden de las HU a implementar	Duración total
1	Implementar las UH de prioridad alta	1 – 8	4.5 semanas
2	Implementar las UH de prioridad normal	9 – 20	5.5 semanas

3.2 Diagrama de componentes

El diagrama de componentes modela el empaquetado físico del sistema en unidades reutilizables llamadas componentes y sus relaciones. Un componente es una unidad física de implementación que encapsula una o más clases del diseño. Este diagrama describe la descomposición del *software* en capas y subsistemas de implementación al igual que sus dependencias (Ambler, 2003).

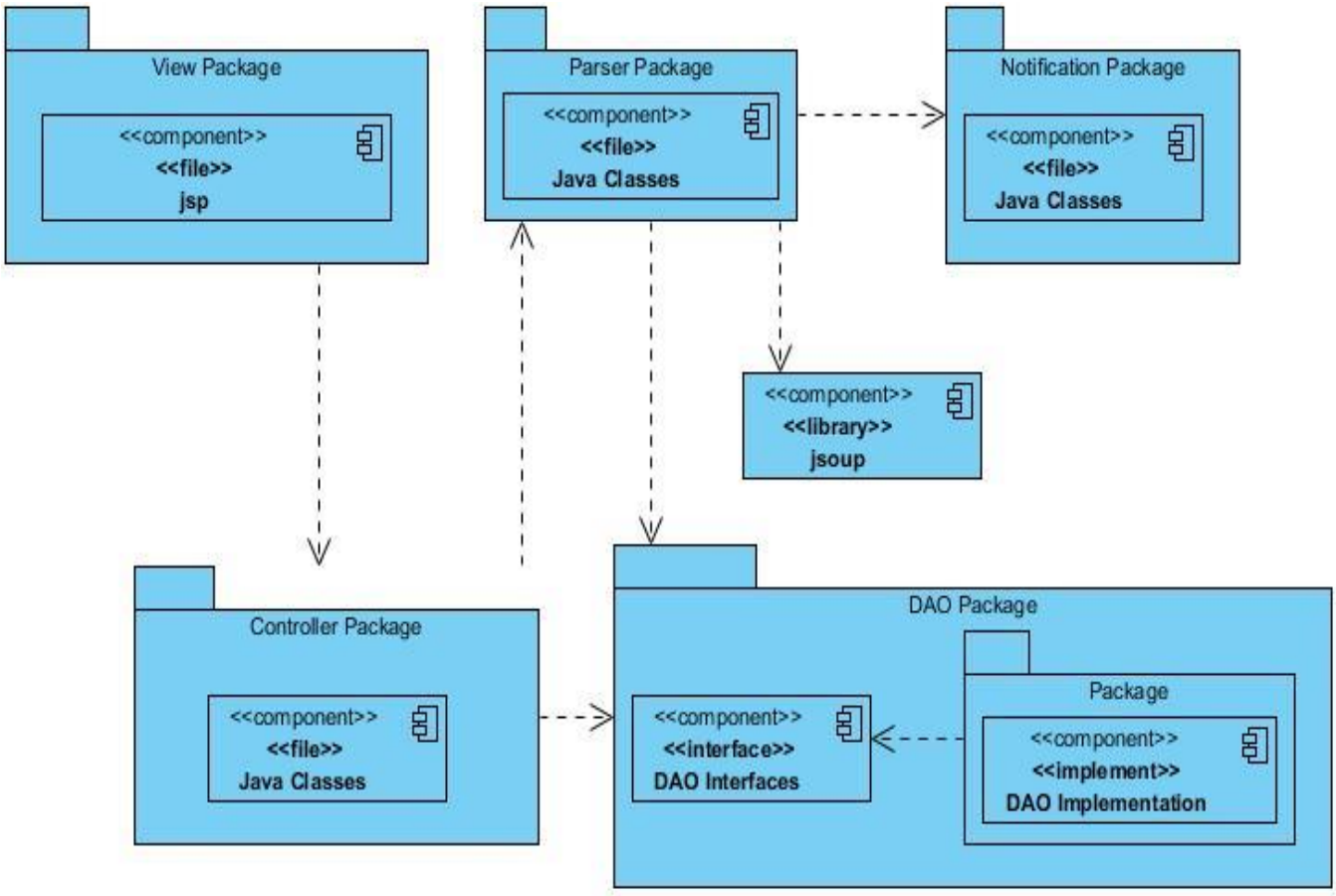

Figura 8: Diagrama de componentes

3.3 Diagrama de paquetes

Permite organizar los elementos modelados con UML, facilitando de ésta forma el manejo de los modelos de un sistema complejo. Además permite dividir un modelo para agrupar y encapsular sus elementos en unidades lógicas individuales. Se puede utilizar para plantear la arquitectura del sistema a nivel macro. El diagrama muestra como está estructurado el sistema. Cada paquete puede contener otros paquetes o clases, que tienen *interfaces* y realizan cierta funcionalidad. También se pueden mostrar algunas clases dentro de los paquetes, así como las relaciones de dependencia de estas clases con otras clases o paquetes (Tutorial - UML *Package Diagram*, 2011).

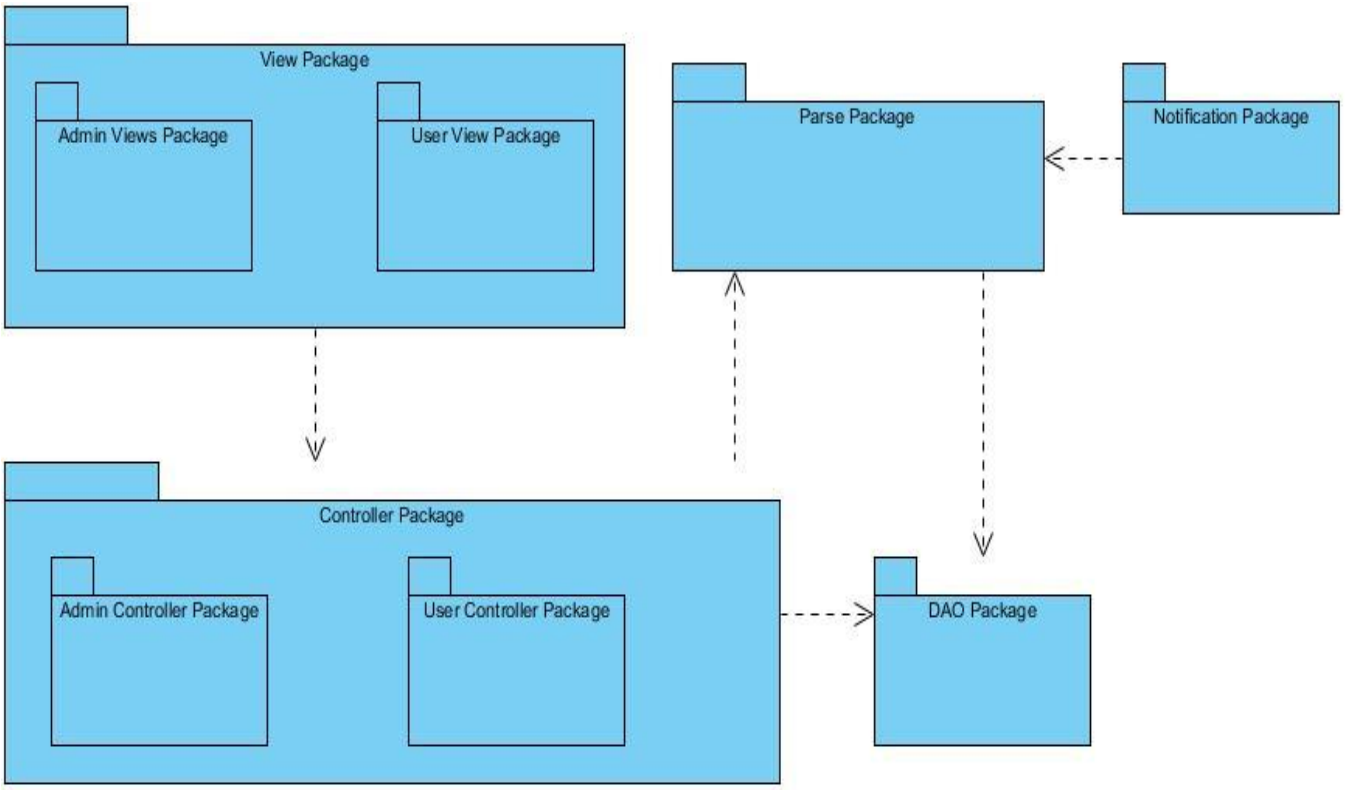

Figura 9: Diagrama de paquetes

3.4 Diagrama de despliegue

El diagrama de despliegue muestra la configuración física sobre la que será desplegado el *software*. Este presenta los nodos computacionales que intervienen en el funcionamiento del sistema, las conexiones entre estos y los protocolos de comunicación que serán utilizados, estableciendo posibles configuraciones que se ilustran mediante los diagramas de despliegue (Ambler, 2003).

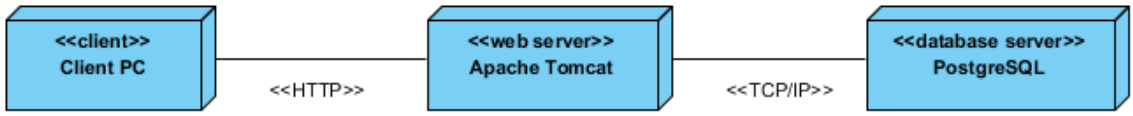

Figura 10: Diagrama de despliegue

La conexión entre el cliente y el servidor web es realizada mediante el protocolo HTTP, al no contarse con un certificado digital firmado por una entidad certificadora.

3.5 Pruebas de validación

Las pruebas de validación en la ingeniería de *software* son el proceso de revisión que verifica que el sistema de *software* producido cumple con las especificaciones y logra su cometido. La validación es el proceso de comprobar que lo que se ha especificado es lo que el usuario realmente desea.

3.5.1 Pruebas unitarias

Una prueba unitaria es un fragmento de código (usualmente un método) que invoca otro fragmento de código y comprueba la exactitud de algunos supuestos. Si las suposiciones resultan ser equivocadas, la prueba falla. Una prueba unitaria es un método o una función (Osherove, 2014).

Las pruebas unitarias fueron desarrolladas constantemente cada vez que se terminaba de implementar alguna funcionalidad. Dichas pruebas fueron realizadas usando jUnit y Mockito. A continuación se pueden observar una muestra de ellas.

```
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration(locations = {"file:src/main/webapp/WEB-INF/applicationContext.xml",
"file:src/main/webapp/WEB-INF/dispatcher-servlet.xml"})
@WebAppConfiguration
public class WbCodeforcesParserTest {

	final static String CONTEST_NAME = "testname";
	final static String CONTEST_URL = "http://testurl.com";

	@Resource
	WbCodeforcesParser wbCodeforcesParser;

	@Resource
	WbContestDAO wbContestDAO;

	@Resource
	WbSiteDAO siteDAO;


	@Test
	public void init() {
		Integer site = siteDAO.getSiteId( Config.getProperty("codeforces.name") );
		assertTrue(site != null);
```

```
        }

        @Test
        public void checkParsing() {
                try {
                        Integer site = siteDAO.getSiteId( Config.getProperty("codeforces.name") );
                        List<WbContest> list = wbCodeforcesParser.parse();
                        assertTrue(list != null);


                        for(int i = 0;i<list.size();i++) {
                                assert( list.get(i).getSid() == site );
                        }
                } catch (ConnectionErrorException e) {
                        assertTrue(false);
                }
        }
}
```

Este código recién mostrado representa dos pruebas unitarias realizadas a la clase encargada del *parsing* de Juez en Línea Codeforces. A continuación se muestra un fragmento de código utilizado para realizar una prueba unitaria a un controlador.

```
@Test
public void tablesManageParsers() throws Exception {
        when( wbSiteDAOMock.getSiteList() ).thenReturn( Arrays.asList(site1, site2) );

        mockMvc.perform(get ("/admin/tables/manageparsers.xhtml"))
                        .andExpect(status().isOk())
                        .andExpect(view().name("/admin/tables/manageparsers"))
                        .andExpect(model().attribute("sites", hasItem(


                        .andExpect(model().attribute("sites", hasSize(2)))
                           allOf(
                                hasProperty("sid", is(1)),
                                hasProperty("site", is("site1")),
```

```
                        hasProperty("url", is("url1")),
                        hasProperty("code", is("code1")),
                        hasProperty("completed", is(false)),
                        hasProperty("enabled", is(false)),

                        hasProperty("timeanddateid", is(1)),
                        hasProperty("timezone", is("zone1"))
                    )
                )))
                .andExpect(model().attribute("sites", hasItem(
                    allOf(
                        hasProperty("sid", is(2)),
                        hasProperty("site", is("site2")),
                        hasProperty("url", is("url2")),
                        hasProperty("code", is("code2")),
                        hasProperty("completed", is(false)),
                        hasProperty("enabled", is(false)),
                        hasProperty("timeanddateid", is(2)),
                        hasProperty("timezone", is("zone2"))
                    )
                )));
        verify(wbSiteDAOMock, times(1)).getSiteList();
        verifyNoMoreInteractions(wbSiteDAOMock);
}
```

Al aplicarse las 41 pruebas unitarias realizadas se obtuvieron los siguientes resultados:

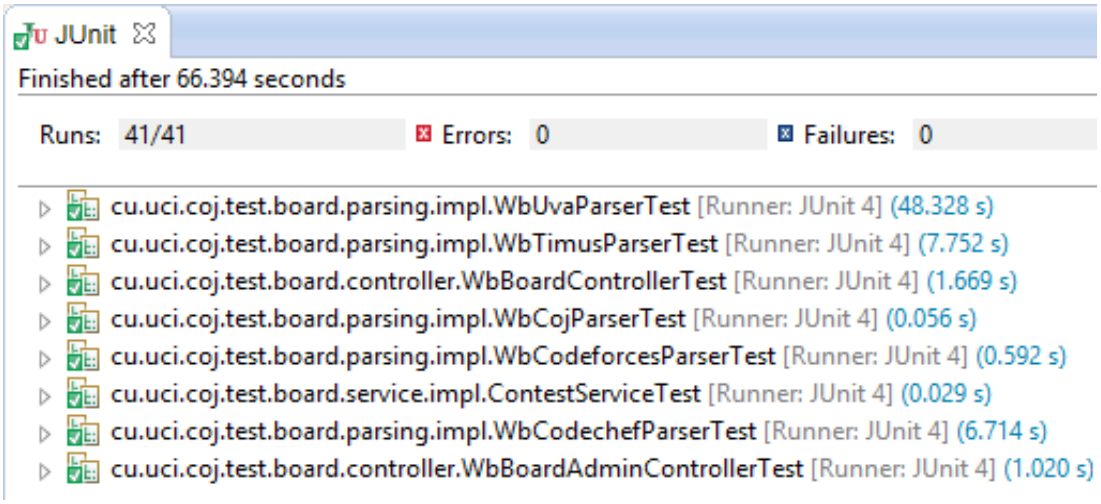

Figura 11: Resultado de las pruebas unitarias.

En la figura 11 se puede apreciar el reporte generado por Eclipse al correr las pruebas unitarias. Se puede observar como las 41 pruebas unitarias aplicadas fueron exitosas y no se produjo ningún fallo o error.

3.5.2 Pruebas de integración

Una prueba de integración es una extensión lógica de las pruebas unitarias. En su forma más simple, dos unidades que ya han sido probadas son combinadas en un componente y la interfaz entre ellos es probada (Microsoft, 2015).

Las siguientes imágenes muestran el módulo de publicación de anuncios de competencias integrado al COJ. En transcurso del desarrollo de las pruebas no fue detectada ninguna situación que atentara contra el funcionamiento del COJ, lo cual permite que tanto el sistema y como el módulo desarrollado trabajan concurrentemente de forma correcta.

La figura 12 muestra la vista que permite observar los anuncios de las próximas competencias y las que están en ejecución. Se accede a ella a través de un vínculo en el sistema.

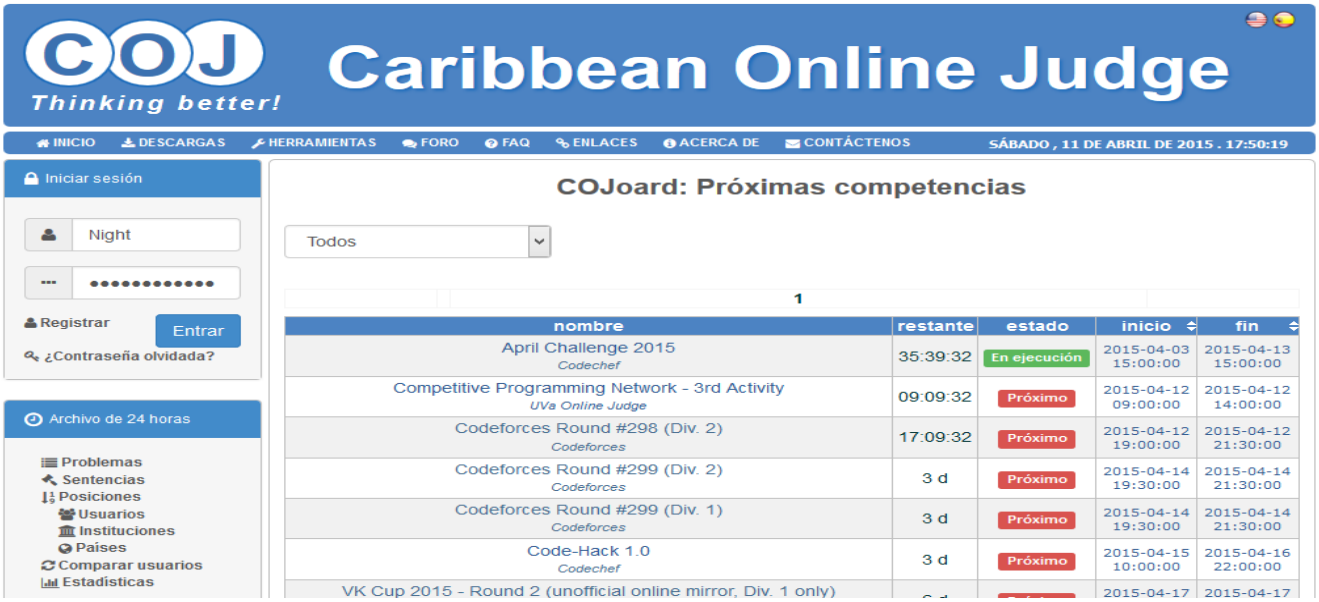

Figura 12: Integración: Próximas competencias

La figura 13 muestra la vista desde la cual los administradores del sistema pueden habilitar o deshabilitar un determinado Juez en Línea y ejecutar manualmente el *parsing*.

Figura 13: Integración: Administrar parsers

La figura 14 muestra la vista sobre la cual los administradores pueden agregar un anuncio de competencia manualmente sobre un Juez en Línea de su interés que aún no se realiza *parsing* automático.

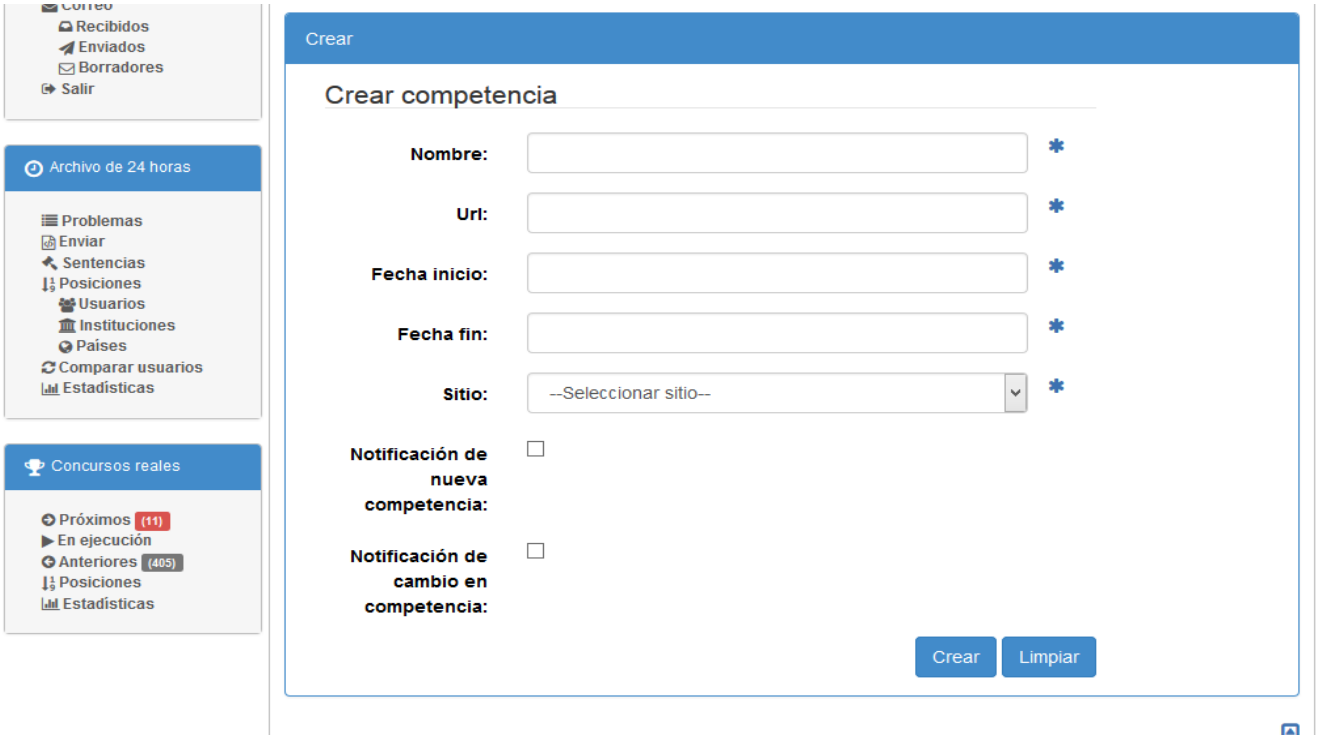

Figura 14: Integración: Crear competencia

La realización de las pruebas de integración arrojó resultados satisfactorios sobre el trabajo coherente entre el COJ y el módulo de publicación de anuncios de competencias adicionado. Actualmente se encuentra desplegado siendo visible para todos los usuarios de Internet.

3.5.3 Pruebas de aceptación

Las pruebas de aceptación son creadas a partir de las historias de usuario. El cliente especifica los escenarios donde las historias de usuario han sido correctamente implementadas. Una historia de usuario puede tener una o más pruebas de aceptación, las necesarias para comprobar el correcto funcionamiento de la implementación de la historia de usuario (Wells, 2013).

Las pruebas de aceptación propuestas a realizarse se encuentran divididas en las siguientes secciones para una mayor organización:

- Clases Válidas: describe cada uno de los pasos seguidos durante el desarrollo de la prueba, se tiene en cuenta cada una de las entradas válidas que hace el usuario con el objetivo de verificar si se obtiene el resultado esperado.
- Clases Inválidas: describe cada uno de los pasos seguidos durante el desarrollo de la prueba, se tiene en cuenta cada una de las posibles entradas inválidas que hace el usuario con el objetivo de verificar si se obtiene el resultado esperado y cómo responde el sistema.
- Resultado Esperado: se describe el resultado que se espera ya sea para entradas válidas o inválidas.
- Resultado de la Prueba: se describe el resultado que se obtiene.
- Observaciones: algún señalamiento o advertencia que sea necesario hacerle a la sección que se está probando.

A continuación se muestra un conjunto de las pruebas realizadas. El resto puede consultarse en el Anexo 3.

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la prueba	Observaciones
El administrador ejecuta la acción del botón Parsear todos.		Se obtiene el resultado de Bien para el *parsing* de todos los sitios.	Satisfactorio.	
El administrador ejecuta la acción del botón Si/No en la columna habilitado.		El botón cambia hacia No/Si, y se realiza el cambio del estado de habilitación del sitio en el módulo y del botón Parsear correspondiente al sitio.	Satisfactorio.	
El administrador ejecuta la acción del		Se muestra una imagen de cargando	Satisfactorio.	

botón Parsear de un sitio.		para el sitio correspondiente. Luego se obtiene una respuesta de Bien al *parsing* del sitio.		

Tabla 10: Prueba de aceptación: Administrar parsers.

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la prueba	Observaciones
El administrador ejecuta la acción del botón Crear competencia.		El sistema muestra la vista de Crear competencia.	Satisfactorio.	
El administrador ejecuta click en el paginador sobre la página deseada.		El sistema carga la página solicitada sin recargar toda la vista.	Satisfactorio.	
El administrador ejecuta click sobre la cabecera de la columna nombre.		El sistema muestra las competencias ordenadas por su nombre.	Satisfactorio.	
El administrador ejecuta click sobre la cabecera de la columna estado.		El sistema muestra las competencias ordenadas por su estado.	Satisfactorio.	
El administrador ejecuta click sobre la cabecera de la columna sitio.		El sistema muestra las competencias ordenadas por su sitio.	Satisfactorio.	

El administrador ejecuta click sobre la cabecera de la columna fecha inicio.		El sistema muestra las competencias ordenadas por su fecha de inicio.	Satisfactorio.	
El administrador ejecuta click sobre la cabecera de la columna fecha fin.		El sistema muestra las competencias ordenadas por su fecha de fin.	Satisfactorio.	
El administrador ejecuta click sobre el nombre de una competencia.		El sistema muestra la vista de Detalles para la competencia solicitada.	Satisfactorio.	
El administrador ejecuta click sobre el sitio de una competencia.		El sistema direcciona al usuario hacia la página de inicio del Juez en Línea seleccionado.	Satisfactorio.	
El administrador ejecuta click sobre la fecha de inicio de una competencia.		El sistema direcciona al usuario hacia la aplicación web: *Time and Date*[1].	Satisfactorio.	
El administrador ejecuta click sobre la fecha de fin de una competencia.		El sistema direcciona al usuario hacia la aplicación web: *Time and Date*.	Satisfactorio.	

[1] Time and date es una aplicación disponible bajo la URL www.timeanddate.com para realizar conversiones de tiempo entre los diferentes husos de horarios en el mundo.

El administrador ejecuta click sobre Detalles en la columna acciones.		El sistema muestra la vista de Detalles para la competencia solicitada.	Satisfactorio.	
El administrador ejecuta click sobre Editar en la columna acciones.		El sistema muestra la vista de Editar para la competencia solicitada.	Satisfactorio.	
El administrador ejecuta click sobre Eliminar en la columna acciones.		El sistema muestra un formulario pidiendo confirmación. Si se ejecuta click en Si, la competencia es eliminada y el usuario es redireccionado a Lista de competencias. En caso contrario se restaura la vista.	Satisfactorio.	

Tabla 11: Prueba de aceptación: Listar anuncios de competencias.

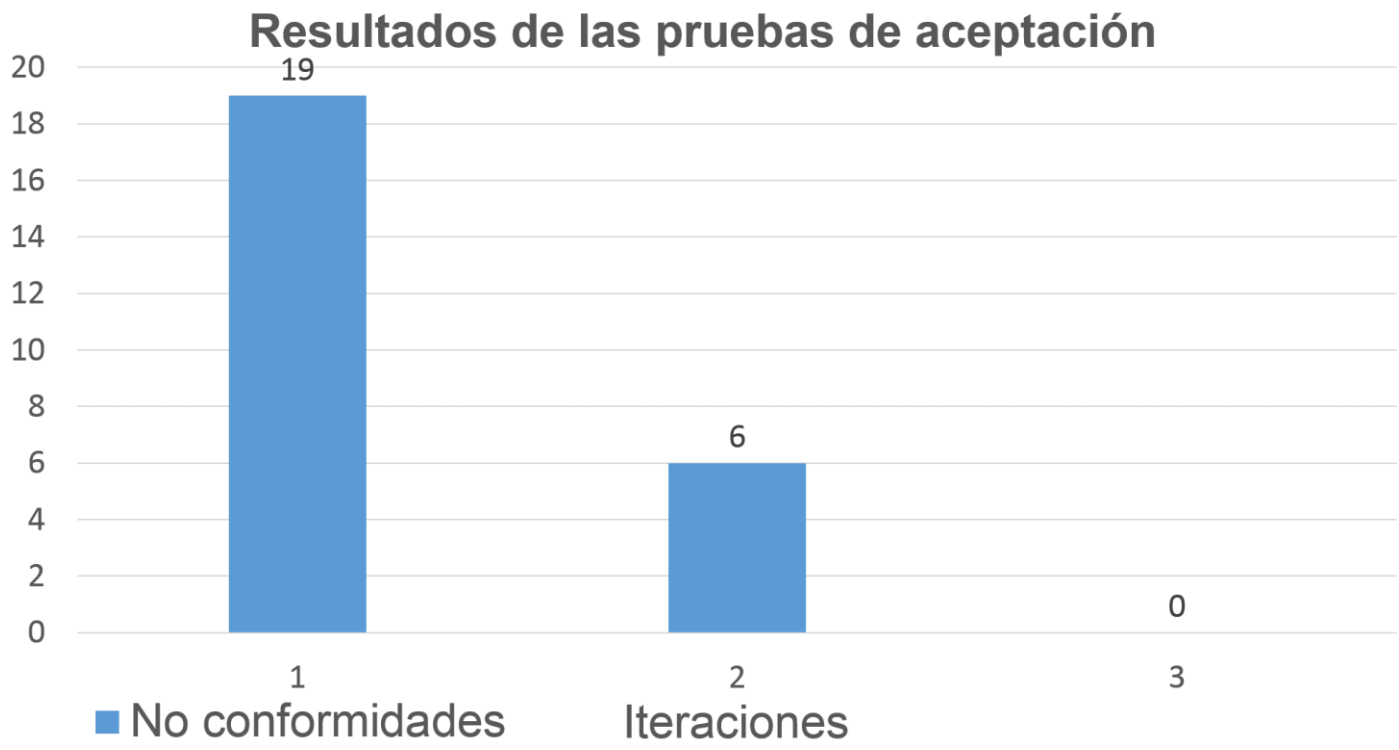

Figura 15: Resultado de las pruebas de aceptación.

Fueron realizadas tres iteraciones como se puede apreciar en la Figura 15 para realizar la corrección de las no conformidades. Luego de llevadas a cabo las pruebas de aceptación a cada uno de los escenarios del módulo, se llega a la conclusión que la solución se encuentra libre de no conformidades. Se obtuvo un resultado satisfactorio para los 50 escenarios probados sobre las ocho vistas que el usuario puede tener interacción.

3.6 Conclusiones parciales

Se obtuvo un plan *release* para realizar la implementación de la propuesta de solución. El diagrama de componentes definido permitió apreciar de forma visual la organización y las dependencias lógicas entre componentes *software* del sistema. El diagrama de paquete facilitó la obtención de un mayor entendimiento de los paquetes que intervienen en la propuesta de solución y su relación entre ellos. La realización de las pruebas unitarias, de integración y de aceptación según los resultados arrojados

evidencia que la solución obtenida está libre de errores que invaliden su correcto funcionamiento, quedando de esta manera validada.

Conclusiones

El estudio y diagnóstico realizado permitió identificar que de las soluciones asociadas al objeto de estudio a nivel internacional y nacional, ninguna resolvía el problema planteado. Se propuso el desarrollo de un módulo para el COJ que permita agrupar y mostrar los anuncios de las competencias programadas en un conjunto de Jueces en Línea.

El análisis de las tecnologías y herramientas luego del estudio bibliográfico correspondiente permitió identificar las más acordes para el desarrollo de la solución planteada como es el caso de la metodología SXP. Esta metodología permitió aligerar la modelación del sistema, evitar la elaboración de documentación innecesaria, disminuir el tiempo de desarrollo de la solución y mantener una constante retroalimentación con el cliente.

La realización el diseño permitió obtener una propuesta de solución que representa el módulo de una manera fácil de interpretar para su implementación. Los requisitos funcionales y no funcionales obtenidos a partir del proceso de identificación de los requisitos, sirvieron de guía para desarrollar las distintas funcionalidades y de este modo satisfacer las necesidades detectadas. Los artefactos generados según la metodología de desarrollo utilizada y los patrones de arquitectura y diseño descritos, constituyeron una guía fundamental para la construcción de la propuesta de solución.

La implementación se llevó a cabo siguiendo el diseño y el plan *release* definidos. La realización de las pruebas unitarias, de integración y de aceptación arrojó resultados satisfactorios validando la solución.

Recomendaciones

- Incorporación progresiva de otros Jueces en Línea para brindarle al usuario una mayor información sobre los anuncios de competencias de programación competitiva.
- Contactar con los desarrolladores de los Jueces en Línea para que provean la información de los anuncios de sus competencias mediante un servicio web. Esto permitiría la eliminación del *parsing*, traduciéndose a consumir un servicio web.
- Realizar la implementación de una API REST[1] para proveer la información recolectada por el módulo a los desarrolladores interesados.

[1] *Representational State Transfer* (**REST**) por sus siglas en el idioma inglés, es un estilo arquitectónico de *software* que consiste de las mejores guías y prácticas para desarrollar servicios web escalables.

Referencias bibliográficas

Acosta, N. S, Soria, J. A y Reyes, A. 2013. Alternativa de comunicación para el Juez en línea Caribeño. [En línea] 2013. [Citado el: 3 de Diciembre de 2014.] http://publicaciones.uci.cu/index.php/SC/article/view/1202/659.

Altaviser. 2014. A Guide to Database Management Systems - DBMS. [En línea] 2014. [Citado el: 22 de Marzo de 2015.] http://dbms.ca/concepts/introduction.html.

Ambler, S.W. 2003. UML 2 Class Diagrams: An Introduction. [En línea] 2003. [Citado el: 15 de 02 de 2015.] http://www.agilemodeling.com/artifacts/classDiagram.htm.

Bascón, E. 2011. El patrón de diseño Modelo Vista Controlador (MVC) y su implementación en Java Swing. [En línea] 2011. [Citado el: 18 de Febrero de 2014.] http://ucbconocimiento.ucbcba.edu.bo/index.php/ran/article/download/84/81.

Bernot, Y. M. 2011. Sistema Gestor de Base de Datos. [En línea] 2011. [Citado el: 3 de Diciembre de 2014.] http://www.ecured.cu/index.php/Sistema_Gestor_de_Base_de_Datos.

Boehm, Barry y Turner, Richard. 2003. *Balancing Agility and Discipline: A Guide for the Perplexed.* 2003.

Cunningham. 2014. Portland Pattern Repository. [En línea] 2014. [Citado el: 23 de Marzo de 2015.] http://c2.com/cgi/wiki?EngineeringTask.

Eguiluz, J. 2014. CSS avanzado. [En línea] 2014. [Citado el: 4 de Diciembre de 2014.] http://librosweb.es/css_avanzado/capitulo_5.html.

Herlocker, J, et al. 2004. Evaluating Collaborative Filtering Recommender Systems. [En línea] 2004. [Citado el: 3 de Diciembre de 2014.] http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.97.5270&rep=rep1&type=pdf.

LibrosWeb. 2015. LibrosWeb. [En línea] 2015. [Citado el: 22 de Marzo de 2015.] http://librosweb.es/libro/css/capitulo_1.html.

Lobaina, J. C y Roque, J. L. 2012. Desarrollo de la versión 2 del Juez en línea Caribeño. [En línea] 2012. [Citado el: 3 de Diciembre de 2014.] http://bibliodoc.uci.cu/RDigitales/2012/octubre/31/TD_05129_12.pdf.

López Gaona, A. 2011. Facultad de Ciencias, UNAM. [En línea] 2011. [Citado el: 19 de Febrero de 2015.] http://hp.fciencias.unam.mx/~alg/normas/estilo.html.

Matos, R. M. 1999. *Diseño de Base de Datos.* 1999.

Microsoft. 2015. MSDN - The microsoft development network. [En línea] 2015. [Citado el: 6 de Abril de 2015.] https://msdn.microsoft.com/en-us/library/aa292128%28v=vs.71%29.aspx.

Mozilla. 2015. Mozilla Developer Network. [En línea] 2015. [Citado el: 22 de Marzo de 2015.] https://developer.mozilla.org/en-US/docs/Web/HTML.

Muñoz Caro, C, Niño, A y Vizcaino Barceló, A. 2002. *Introdución a la programación con orientación a objetos.* s.l. : Prentice-Hall, 2002.

Oracle. 2014. Oracle. [En línea] 2014. [Citado el: 11 de Noviembre de 2014.] https://docs.oracle.com/javase/specs/jls/se8/html/jls-1.html.

Osherove, Roy. 2014. *The Art of Unit Testing.* New York : Manning Publications Co., 2014. ISBN: 9781617290893.

2012. Pattern Definitions. [En línea] 2012. [Citado el: 18 de Febrero de 2015.] http://st-www.cs.illinois.edu/patterns/definition.html.

Peñalver, G. M. 2008. MA-GMPR-UR2 Metodología ágil para proyectos de software libre. [En línea] 2008. [Citado el: 3 de Diciembre de 2014.] http://repositorio_institucional.uci.cu/jspui/bitstream/ident/TD_1309_08/1/TD_1309_08.pdf.

Pressman, R. S. 2002. Ingeniería del Software Un enfoque práctico: Quinta edición. [En línea] 2002. [Citado el: 3 de Diciembre de 2014.] http://unpocodejava.wordpress.com/2013/05/09/tecnicas-para-la-captura-de-requisitos.

Pullés, Y. D. 2003. Herramienta CASE. [En línea] 2003. [Citado el: 3 de Diciembre de 2014.] http://www.sisman.utm.edu.ec/libros/FACULTAD%20DE%20CIENCIAS%20INFORM%C3%81TICAS/CARRERA%20DE%20INGENIER%C3%8DA%20DE%20SISTEMAS%20INFORMATICOS/09/TECNICAS%20DE%20IV%20GENERACION/Libro.pdf.

Radigan, Dan. 2014. Atlassian. [En línea] 2014. [Citado el: 15 de April de 2014.] https://www.atlassian.com/agile/backlogs.

Silveira-Romero, D y Hernández-Legrá, L. 2012. Convocatoria de participación. [En línea] 2012. [Citado el: 3 de Diciembre de 2014.] http://www.instec.cu/acm-icpc/.

Tutorial - UML Package Diagram. 2011. Visual Paradigm. [En línea] 2011. [Citado el: 2 de Abril de 2015.] http://www.visual-paradigm.com/product/vpuml/tutorials/packagediagram.jsp.

Ubuntu. 2014. Official Ubuntu Documentation. [En línea] 2014. [Citado el: 22 de Marzo de 2015.] https://help.ubuntu.com/lts/serverguide/web-servers.html.

Vaillant, M y Labaut, R. 2013. Portal web de la subsede cubana de la final caribeña del ACM-ICPC. [En línea] 2013. [Citado el: 3 de Diciembre de 2014.] http://bibliodoc.uci.cu/RDigitales/2013/septiembre/24/TD_06531_13.pdf.

VisualParadigm. 2014. Visual Paradigm. [En línea] 2014. [Citado el: 11 de Noviembre de 2014.] http://www.visual-paradigm.com.

Walls, C. 2011. *Spring in Action Third Edition.* New York : Manning Publications Co, 2011. ISBN 9781935182351.

Wells, Don. 2013. Extreme Programming. [En línea] 2013. [Citado el: 6 de Abril de 2015.] http://www.extremeprogramming.org/rules/functionaltests.html.

—. 2013. Extreme Programming:. [En línea] 2013. [Citado el: 23 de Marzo de 2015.] http://www.extremeprogramming.org/rules/userstories.html.

Bibliografía

Acosta, N. S, Soria, J. A y Reyes, A. 2013. Alternativa de comunicación para el Juez en línea Caribeño. [En línea] 2013. [Citado el: 3 de Diciembre de 2014.] http://publicaciones.uci.cu/index.php/SC/article/view/1202/659.

Altaviser. 2014. A Guide to Database Management Systems - DBMS. [En línea] 2014. [Citado el: 22 de Marzo de 2015.] http://dbms.ca/concepts/introduction.html.

Ambler, S.W. 2003. UML 2 Class Diagrams: An Introduction. [En línea] 2003. [Citado el: 15 de 02 de 2015.] http://www.agilemodeling.com/artifacts/classDiagram.htm.

Bascón, E. 2011. El patrón de diseño Modelo Vista Controlador (MVC) y su implementación en Java Swing. [En línea] 2011. [Citado el: 18 de Febrero de 2014.] http://ucbconocimiento.ucbcba.edu.bo/index.php/ran/article/download/84/81.

Bernot, Y. M. 2011. Sistema Gestor de Base de Datos. [En línea] 2011. [Citado el: 3 de Diciembre de 2014.] http://www.ecured.cu/index.php/Sistema_Gestor_de_Base_de_Datos.

Boehm, Barry y Turner, Richard. 2003. *Balancing Agility and Discipline: A Guide for the Perplexed.* 2003.

Cunningham. 2014. Portland Pattern Repository. [En línea] 2014. [Citado el: 23 de Marzo de 2015.] http://c2.com/cgi/wiki?EngineeringTask.

Eguiluz, J. 2014. CSS avanzado. [En línea] 2014. [Citado el: 4 de Diciembre de 2014.] http://librosweb.es/css_avanzado/capitulo_5.html.

Herlocker, J, et al. 2004. Evaluating Collaborative Filtering Recommender Systems. [En línea] 2004. [Citado el: 3 de Diciembre de 2014.] http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.97.5270&rep=rep1&type=pdf.

LibrosWeb. 2015. LibrosWeb. [En línea] 2015. [Citado el: 22 de Marzo de 2015.] http://librosweb.es/libro/css/capitulo_1.html.

Lobaina, J. C y Roque, J. L. 2012. Desarrollo de la versión 2 del Juez en línea Caribeño. [En línea] 2012. [Citado el: 3 de Diciembre de 2014.] http://bibliodoc.uci.cu/RDigitales/2012/octubre/31/TD_05129_12.pdf.

López Gaona, A. 2011. Facultad de Ciencias, UNAM. [En línea] 2011. [Citado el: 19 de Febrero de 2015.] http://hp.fciencias.unam.mx/~alg/normas/estilo.html.

Matos, R. M. 1999. *Diseño de Base de Datos.* 1999.

Microsoft. 2015. MSDN - The microsoft development network. [En línea] 2015. [Citado el: 6 de Abril de 2015.] https://msdn.microsoft.com/en-us/library/aa292128%28v=vs.71%29.aspx.

Mozilla. 2015. Mozilla Developer Network. [En línea] 2015. [Citado el: 22 de Marzo de 2015.] https://developer.mozilla.org/en-US/docs/Web/HTML.

Muñoz Caro, C, Niño, A y Vizcaino Barceló, A. 2002. *Introdución a la programación con orientación a objetos.* s.l. : Prentice-Hall, 2002.

Oracle. 2014. Oracle. [En línea] 2014. [Citado el: 11 de Noviembre de 2014.] https://docs.oracle.com/javase/specs/jls/se8/html/jls-1.html.

Osherove, Roy. 2014. *The Art of Unit Testing.* New York : Manning Publications Co., 2014. ISBN: 9781617290893.

2012. Pattern Definitions. [En línea] 2012. [Citado el: 18 de Febrero de 2015.] http://st-www.cs.illinois.edu/patterns/definition.html.

Peñalver, G. M. 2008. MA-GMPR-UR2 Metodología ágil para proyectos de software libre. [En línea] 2008. [Citado el: 3 de Diciembre de 2014.] http://repositorio_institucional.uci.cu/jspui/bitstream/ident/TD_1309_08/1/TD_1309_08.pdf.

Pressman, R. S. 2002. Ingeniería del Software Un enfoque práctico: Quinta edición. [En línea] 2002. [Citado el: 3 de Diciembre de 2014.] http://unpocodejava.wordpress.com/2013/05/09/tecnicas-para-la-captura-de-requisitos.

Pullés, Y. D. 2003. Herramienta CASE. [En línea] 2003. [Citado el: 3 de Diciembre de 2014.] http://www.sisman.utm.edu.ec/libros/FACULTAD%20DE%20CIENCIAS%20INFORM%C3%81TICAS/CARRERA%20DE%20INGENIER%C3%8DA%20DE%20SISTEMAS%20INFORMATICOS/09/TECNICAS%20DE%20IV%20GENERACION/Libro.pdf.

Radigan, Dan. 2014. Atlassian. [En línea] 2014. [Citado el: 15 de April de 2014.] https://www.atlassian.com/agile/backlogs.

Silveira-Romero, D y Hernández-Legrá, L. 2012. Convocatoria de participación. [En línea] 2012. [Citado el: 3 de Diciembre de 2014.] http://www.instec.cu/acm-icpc/.

Tutorial - UML Package Diagram. 2011. Visual Paradigm. [En línea] 2011. [Citado el: 2 de Abril de 2015.] http://www.visual-paradigm.com/product/vpuml/tutorials/packagediagram.jsp.

Ubuntu. 2014. Official Ubuntu Documentation. [En línea] 2014. [Citado el: 22 de Marzo de 2015.] https://help.ubuntu.com/lts/serverguide/web-servers.html.

Vaillant, M y Labaut, R. 2013. Portal web de la subsede cubana de la final caribeña del ACM-ICPC. [En línea] 2013. [Citado el: 3 de Diciembre de 2014.] http://bibliodoc.uci.cu/RDigitales/2013/septiembre/24/TD_06531_13.pdf.

VisualParadigm. 2014. Visual Paradigm. [En línea] 2014. [Citado el: 11 de Noviembre de 2014.] http://www.visual-paradigm.com.

Walls, C. 2011. *Spring in Action Third Edition.* New York : Manning Publications Co, 2011. ISBN 9781935182351.

Wells, Don. 2013. Extreme Programming. [En línea] 2013. [Citado el: 6 de Abril de 2015.] http://www.extremeprogramming.org/rules/functionaltests.html.

—. 2013. Extreme Programming:. [En línea] 2013. [Citado el: 23 de Marzo de 2015.] http://www.extremeprogramming.org/rules/userstories.htm