Facultad 4

Módulo de concursos virtuales para el Juez en Línea Caribeño

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor(es): Alejandro Cancio Machado

Osvaldo Posada López

Tutor(es): Ing.Yonny Mondelo Hernández

Co-tutor: Ing.Orlando Grabiel Toledano López

Julio 2016


La única manera
de hacer
un trabajo genial
es amar lo que haces

DECLARACIÓN DE AUTORÍA

Declaramos que somos los únicos autores de este trabajo y autorizamos a la Facultad 4 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmamos la presente a los ____ días del mes de _____ del año _____.

Alejandro Cancio Machado

_____

Firma del autor

Osvaldo Posada López

_____

Firma del autor

Ing. Yonny Mondelo Hernández

_____

Firma del tutor

Ing. Orlando Gabriel Toledano López

_____

Firma del tutor

AGRADECIMIENTOS

De Alejandro Cancio Machado

agradezco primero a mi familia por apoyarme en todo desde principio a fin sin ellos nunca hubiese llegado tan lejos , a Mi abuelo José R. Cancio quien ha sido como un padre durante todos estos años sin mencionar que siempre que llamaba "Abuelo tírame un cabo para acá" al otro día siempre puntual aunque nunca me zafaba de una construcción cuando iba a villa clara , a mis abuela Elda León que siempre tan quisquillosa me decía lo que tenía que hacer , que no se me quedara nada por traer que siempre me esperaba con una buena comidita en la casa , a mi otra abuela Irma que siempre me complació con los gustos más caros aunque no fuese la más cariñosa siempre me demostró amor a su manera . A mis madrinas por tantos viajes y tanto apoyo durante tantos años , A mi mama (mi gordis) sobre todo por ser mi confesora mi mejor amiga que sin ella nunca hubiese llegado tan lejos gracias por ese apoyo incondicional todos estos años , A mi compañero de tesis que sin el seguro que no me graduó por tantos años haciendo trabajos juntos , a mis amigos presentes y no presentes que siempre me apoyaron en todo momento , a mis tutores Orlando , Tomas aunque no se encuentra presente y Yonny por su gran apoyo durante este periodo final de esta carrera. Gracias

De Osvaldo Posada López

A mí familia por todo el apoyo brindado en todo momento. Quiero agradecer de forma especial a mí padre, a mí novia y principalmente a mí madre por su dedicación, paciencia y amor hacia mí y por estar siempre presentes en los momentos cruciales de mí vida. Agradezco a mi compañero de tesis. Agradezco a mis compañeros de aula y amigos. Agradezco a todos los profesores que influyeron en mí formación.

RESUMEN

Los jueces en línea son aplicaciones web que evalúan automáticamente las propuestas de solución de problemas computacionales. Varios de los jueces del mundo brindan la posibilidad de que el usuario pueda revivir competencias pasadas como forma de entrenamiento para el Concurso Internacional Universitario de Programación (ACM-ICPC). En la Universidad de las Ciencias Informáticas (UCI) se encuentra desplegado el Juez en Línea Caribeño del cual se requiere un módulo que posibilite a los usuarios gestionar concursos virtuales. El objetivo del presente trabajo es desarrollar un módulo de gestión de concursos virtuales para potenciar el entrenamiento de los usuarios del Juez en Línea Caribeño. Para dar cumplimiento al objetivo del trabajo se realizó un estudio del arte sobre sistemas similares y las tecnologías, el sistema fue desarrollado mediante la metodología AUP versión UCI, haciendo uso del *framework* Spring y el lenguaje de programación Java. Se realizaron los flujos de Modelado, Implementación y Prueba, indicados por la metodología elegida, obteniéndose un sistema que cumple con los requerimientos funcionales y no funcionales.

Palabras Clave: Juez en línea, concursos virtuales, Juez en Línea Caribeño.

ÍNDICE

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

INTRODUCCIÓN

El uso de las tecnologías ha tenido un gran impacto para el desarrollo de la sociedad, permitiendo grandes avances en diferentes campos de esta. Uno de estos campos es la formación de profesionales de la computación el cual está integrado por varias disciplinas que constan con altos niveles de dificultad, formando parte de estas disciplinas se encuentra la programación que es considerada la que presenta mayor grado de complejidad para el aprendizaje, por lo que se han buscado variantes y métodos que resulten atractivos para enseñar y mantener motivados a los interesados en aprender programación. Una de las variantes utilizadas para el aprendizaje de programación son los concursos de programación, cuya realización y entrenamiento previo está sustentado actualmente en los jueces en línea de programación que no son más que aplicaciones web que evalúan automáticamente las propuestas de solución a un conjunto de problemas computacionales.

Los jueces en línea de programación han tenido un impacto significativo desde hace varios años en la motivación y el apoyo por el aprendizaje de la programación de computadoras, siendo sus dos contextos de aplicación más importantes el entrenamiento a participantes del Concurso Internacional Universitario de Programación (ACM-ICPC) así como el apoyo a los docentes en cursos de programación de distintas universidades. Entre las formas de entrenamiento sobresalen los concursos virtuales, los cuales permiten a competidores y entrenadores revivir en tiempo real concursos pasados, convirtiendo el proceso de análisis de un concurso anterior en una experiencia real, emocionante y dinámica; pues el concursante tiene la oportunidad de volver a enfrentarse al concurso o de simular que compiten en uno en el cual no estuvo presente, en un ambiente virtual donde todo se desarrolla con la misma dinámica de la competencia pasada. Por otra parte existen los concursos de práctica, los cuales permiten crear competencias sin quedar en los datos históricos del sistema, elementos que pueden ser usados también en las actividades docentes de la universidad.

El Juez en Línea Caribeño (COJ) es el de mayor cantidad de usuarios en el área del Caribe, con más de 24500 en representación de 175 países y más de 1000 instituciones. Actualmente incluye más de 2600 problemas y ha soportado 500 competencias. Se destaca por un significativo conjunto de características que lo distinguen incluso de varios de los mejores jueces en línea del mundo. A pesar de ello muchos de sus usuarios reclaman características que resultan muy necesarias para su mejor aprovechamiento. Entre las más solicitadas por los usuarios se encuentran los concursos virtuales y los concursos de práctica.

Hasta finales del 2014 el COJ contaba con esta funcionalidad lo que sumó varios adeptos a dicho sistema. Pero tras varios cambios y refactorización a la plataforma, y en consecuencia de lo reducido e inestable que se ha comportado el equipo de desarrollo, la funcionalidad de los concursos virtuales y las bases de código necesarias se perdieron. Como consecuencia surgen dificultades para el entrenamiento de los participantes en concursos de programación como (ACM-ICPC) y por otra parte se limita a los usuarios a poner en práctica su aprendizaje y nivel alcanzado en la programación. Desde la pérdida de la funcionalidad son muchos los concursantes, estudiantes, entrenadores y profesores que reclaman el restablecimiento de dichas funcionalidades.

A partir de la situación anterior se define como **problema a resolver:** El Juez en Línea Caribeño carece de un módulo que permita la gestión de concursos virtuales de programación.

Siendo el **objeto de estudio** concursos virtuales de programación de computadoras; y su **campo de acción** proceso de gestión de concursos virtuales de programación de computadoras en jueces en línea.

Se define como **objetivo general:** Desarrollar un módulo de gestión de concursos virtuales para potenciar el entrenamiento de los usuarios del Juez en Línea Caribeño; derivándose los siguientes **objetivos específicos:**

- Analizar los principales elementos teóricos relacionados con el uso de concursos virtuales en la programación competitiva.
- Definir las tecnologías y herramientas necesarias para el desarrollo de la propuesta de solución.
- Diseñar el módulo de gestión de concursos virtuales.
- Desarrollar el módulo de gestión de concursos virtuales.

Para dar cumplimiento a los objetivos se definen las siguientes **tareas de investigación:**

- Fundamentación teórica:
    - Revisión bibliográfica para fundamentar la selección de la metodología de desarrollo, las tecnologías y herramientas que serán utilizadas en la investigación.
    - Analizar sistemas similares.
    - Investigar sobre la arquitectura del COJ y demás aspectos necesarios para incorporar nuevas funcionalidades.

- Seleccionar la metodología de desarrollo, las tecnologías y herramientas que serán utilizadas en el desarrollo de la propuesta de solución.
- Desarrollar los artefactos necesarios de acuerdo a la metodología elegida.
- Implementar las funcionalidades de la propuesta de solución.
- Validar la propuesta de solución mediante pruebas funcionales.

Para darle cumplimiento a los objetivos antes planteado se proponen las siguientes **preguntas científicas**:

- ¿Cuáles son los elementos teóricos y metodológicos que fundamentan el desarrollo del módulo de concursos virtuales para el COJ?
- ¿Cuál es el estado actual de los jueces en línea que cuentan con un módulo de concursos virtuales?
- ¿Cuáles son las herramientas y tecnologías necesarias para el desarrollo del módulo de concursos virtuales para el COJ?
- ¿Cuáles son los pasos a seguir para definir el diseño de la propuesta a desarrollar?
- ¿Qué tipo de pruebas de software se aplican para validar el correcto funcionamiento del módulo de concursos virtuales para el COJ?

Para desarrollar las tareas anteriormente expuestas se emplea el método general dialectico materialista en el que se analizan las interacciones entre todos los componentes y se aplican las leyes de la naturaleza, sociedad y pensamiento.

Los métodos teóricos a emplear en la presente investigación son el Histórico Lógico, el Analítico-Sintético y la Inducción - Deducción. El primero posibilita estudiar la trayectoria histórica y la lógica del desarrollo de concursos virtuales, permitiendo conocer de forma general el funcionamiento de los mismos. El segundo, nos permite dividir y estudiar los componentes existentes y así poder identificar qué elementos pueden ser útiles para dar solución al problema. El tercero posibilita arribar a conclusiones y resultados a partir de todo el estudio teórico.

Los métodos empíricos utilizados en la presente investigación son: la observación, la entrevista, la consulta a expertos y estadísticos; el primero permite recopilar información de los jueces en línea que tienen implementado un módulo de concursos virtuales, permitiendo la obtención de conocimiento del tema a partir del análisis de las funcionalidades de los mismos; el segundo permite la obtención de información para

perfeccionar la investigación; el tercero posibilita tener certeza de lo que se ha realizado y estadísticos que posibilitan avalar científicamente los resultados obtenidos.

El documento está organizado en tres capítulos, de la siguiente forma:

Capítulo 1: Fundamentación teórica.

En este capítulo se exponen los datos y observaciones realizadas a sistemas similares y se describen las soluciones informáticas que serán utilizadas (metodologías, modelo de desarrollo de software, tecnologías y herramientas).

Capítulo 2: Análisis y diseño del sistema.

En este capítulo son descritos los principales conceptos relacionados con la investigación a través de un modelo de dominio. Se identifican los requisitos funcionales y no funcionales los cuales son agrupados en casos de uso. Se obtienen los diagramas de clases del análisis, colaboración, así como los diagramas de clases del diseño, además es construido el modelo de datos y son realizadas las descripciones correspondientes.

Capítulo 3: Implementación y prueba.

Se describen los principales aspectos del desarrollo y se procede con la implementación de la propuesta de solución, donde se obtiene un módulo de concursos virtuales para el COJ que brinde los servicios requeridos por el cliente. Se describen las pruebas realizadas al sistema una vez concluida la implementación.

A partir de todo lo explicado se esperan los siguientes **resultados:** Al concluir esta investigación se obtendrá un módulo del COJ donde se pueda gestionar todo lo referente a concursos virtuales y concursos de práctica.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

En este capítulo se caracterizan los antecedentes de los concursos virtuales y los concursos de entrenamiento. Se expone un estudio acerca de los principales sistemas de este tipo que existen en la actualidad en los diferentes ámbitos, para obtener conocimiento de cuáles son la tendencias que rigen el desarrollo de los mismos. Además, se seleccionan las herramientas a utilizar durante el proceso de desarrollo de la solución propuesta.

1.1 Conceptos relacionados con el dominio del problema

Para la comprensión del presente trabajo se debe tener en cuenta una serie de conceptos y definiciones asociadas al dominio del problema.

Jueces en línea

Como parte del aprendizaje electrónico surgen los “Jueces en Línea”, que son sistemas, por lo general Web, que permiten evaluar de forma automática programas de computación que intenten solucionar tareas propuestas. La salida de cada código enviado por un usuario es capturada por el sistema y comparada contra la solución que se tiene de la tarea en cuestión o será evaluada por un evaluador externo en el caso que así sea requerido (1).

Antecedentes de los concursos virtuales

De los jueces en línea el primero en implementar un módulo de concursos virtuales fue el *Saratov State University Online Contester* implementándolo de una forma novedosa a la hora de presentar las competencias de entrenamiento dándoles paso a demás jueces en línea una idea de cómo entrenar mejor a sus concursantes.

Concurso Virtual

En los jueces en línea se realizan una serie de competencias que tributan a la ACM-ICMP o como parte de entrenamientos para los estudiantes. Las competencias son guardadas y representadas en los jueces dando a conocer la puntuación de los equipos y los lugares, así como los envíos, las respuestas incorrectas, respuestas acertadas, límite de tiempo excedido entre otros. Para el entrenamiento de los concursantes

existen los concursos virtuales que consisten en revivir estas competencias ya realizadas integrando a los concursantes dentro de la competencia en tiempo real compitiendo contra los participantes.

1.2 Ejemplos de soluciones similares en el mundo

Principales jueces en línea con concursos virtuales implementados

Durante la investigación realizada se han identificado alrededor de 20 jueces en línea, de los cuales la mayor cantidad está concentrada en Asia; usualmente dichos sistemas se encuentran en instituciones de Educación Superior, debido a que el mayor número de usuarios de los mismos son estudiantes universitarios. Por otra parte, los jueces en línea también han sido utilizados como herramientas de apoyo al proceso de enseñanza-aprendizaje.

Juez en línea A2 (A2 *online judge*)

El A2 es un juez en línea que cuenta con 18644 concursos, en este se pueden crear concursos de cualquier tipo usando problemas existentes o agregados por el usuario donde no necesariamente tienes que ser un administrador del sistema, se pueden agregar problemas de forma aleatoria o de concursos realizados anteriormente como las finales regionales, se pueden invitar participantes o hacer el concurso de forma pública. La siguiente imagen muestra las distintas posibilidades a crear un concurso virtual en el Juez en línea A2.

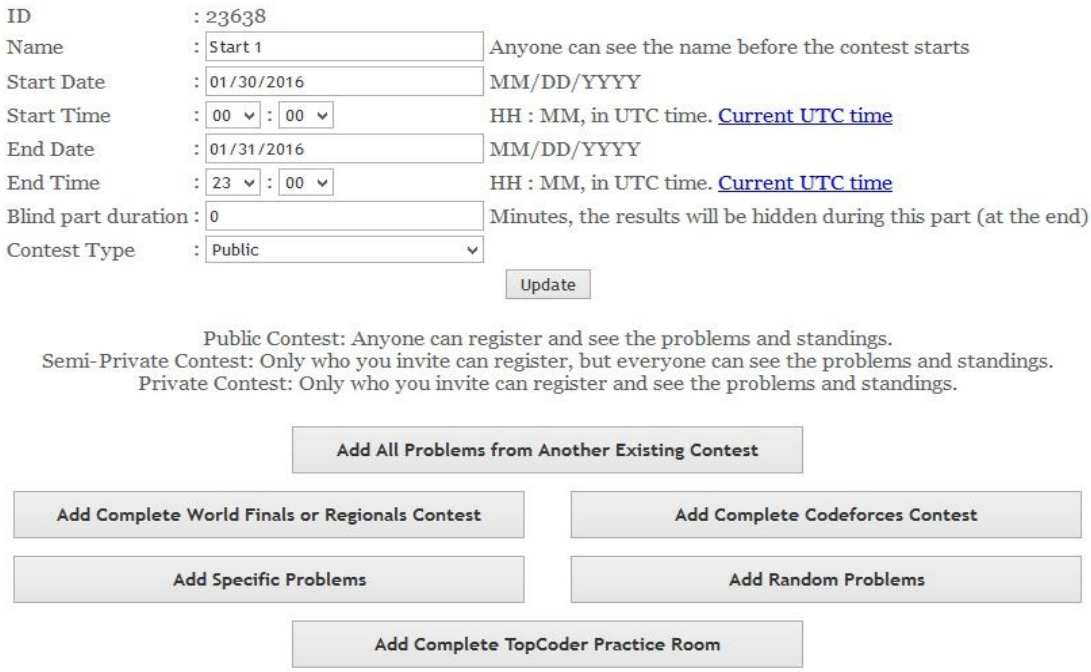


Ilustración 1Página principal de creación de concursos virtuales A2

Saratov State University Online Contester

Este juez en línea fue uno de los primeros sistemas de este tipo desarrollados por y para universidades rusas. Una de las características que presenta el SGU es el módulo de Concursos Virtuales, el cual contiene reglas importantes para el su uso. Para crear un concurso virtual se entra en el sistema con el usuario y defines el concurso. Se pueden borrar sus concursos. Permite recrear concursos anteriormente realizados. No se pueden definir concursos que se crucen con concursos anteriormente creados. Los resultados se añadirán a las estadísticas del concurso, pero solo por una semana. No se puede definir una competencia dos veces.

UVa Virtual Contest Generator (Herramienta para generar concursos virtuales del Juez en Línea de la Universidad de Valladolid)

Esta es una herramienta que conserva las estadísticas del juez en línea de la Universidad de Valladolid. Permite hasta 100 usuarios del *UVa Online Judge*, los problemas pueden ser hasta 100 de los publicados por el juez en línea de Valladolid. Los concursos virtuales pueden ser publicados de dos formas: La primera

y más fácil es que se generan los problemas de forma aleatoria escogidos del Uva y no son mostrados hasta que el concurso comience, la segunda es publicar el concurso en tu propio sitio web, esta herramienta genera una página HTML para que lo publiques en un servidor propio, de esta forma solo los usuarios que puedan ver este sitio accedan a la competición. La competición se genera introduciendo los números de los problemas y los identificadores de los usuarios que van a participar generando una vista previa (2)**.** En la siguiente imagen se muestran los datos necesarios para crear un concurso en la herramienta.

Ilustración 2Página principal creación de concurso virtual Uva

Juez en Línea de la Universidad de Tianjin(TJU)

Este juez permite crear concursos virtuales introduciendo los identificadores de los problemas, siendo obligatorio al menos tres problemas para la competición, de igual forma le pide al usuario que al menos tenga resuelto cinco problemas en el juez, también deja introducir una breve descripción del concurso.

Análisis comparativo

Para realizar un análisis comparativo entre los jueces a nivel internacional que cuentan con un módulo de concursos virtuales se expone en la siguiente tabla:

Tabla 1Comparación de concursos virtuales

	A2 Online Judge	**Saratov**	***Uva Virtual Contest Generator***	**TJU**
Problemas	Aleatorios o de anteriores concursos	De antiguas competiciones	Aleatorios o de anteriores concursos	Aleatorios o de anteriores concursos
Tipo de concurso	Concursos de entrenamiento	Recreación de antiguas competiciones	Concursos de entrenamiento	Concursos de entrenamiento
Tipo de acceso	competiciones públicas o con usuarios específicos	competiciones públicas o con usuarios específicos	competiciones públicas o con los usuarios que puedan acceder al sitio	competiciones públicas o con usuarios específicos
Restricciones	Usuarios conectados	Usuarios conectados, no se puede cruzar con otro concurso	Usuarios conectados	Usuarios conectados, obligatorio tener más de 5 problemas resueltos, permite hasta 100 problemas y 100 usuarios en la misma competencia
Modificación	Permite modificación	Permite modificación	Permite modificación	Permite modificación
Eliminación	Permite eliminación	Permite eliminación	Permite eliminación	Permite eliminación

Analizando la tabla se concluye que el módulo debe permitir crear concursos virtuales con problemas de competencias anteriores y de entrenamiento con problemas aleatorios escogidos por el sistema o por el propio usuario, deben ser competencias públicas o por usuarios específicos, los concursos puedan ser eliminados por el creador o un administrador según lo requiera. Otra característica importante es la libertad que los usuarios puedan entrar en el concurso siendo dos vías importantes: la de permitir que todos los usuarios entren o la de invitar a usuarios específicos a la competición.

Tendencias de los concursos virtuales

Los jueces en línea tienen una característica muy común todos, consumen gran cantidad de recursos en hardware causado en gran medida por los pedidos de evaluar los problemas, por consecuencia los concursos virtuales deben aumentar los requerimientos para poder gestionar la cantidad de peticiones realizadas por los usuarios y la vista concurrente de las competencias. Los concursos virtuales tienden a ser objeto de ataques por los usuarios, entrando a la aplicación como usuarios principiantes permitiéndoles realizar varias veces la misma competición por lo que los concursos virtuales deben contener una serie de reglas para los usuarios que deseen acceder o crear un concurso. Otra característica importante es que los concursos en línea están compuestos por dos partes importantes: la capa de presentación y el módulo de evaluación que permite juzgar los problemas.

1.3 Tecnologías de desarrollo de software

El Juez en Línea Caribeño está programado en Java, haciendo un uso del *framework* de negocio *Spring* y la estructura de las tablas de la base de datos. Cualquier nueva funcionalidad que desee aprovechar las características de COJ e interactuar con su núcleo debe ser escrita utilizando estas tecnologías. Además del lado del cliente COJ utiliza CSS y JavaScript para dotar a sus páginas de una presentación de calidad y una mejor interactividad. Estas características son las que deciden que para el desarrollo de la solución del problema identificado en la presente investigación sean seleccionadas dichas tecnologías.

1.3.1 Lenguaje de programación

Un lenguaje de programación es aquel elemento dentro de la Informática que permite crear programas mediante un conjunto de instrucciones, operadores y reglas de sintaxis, poniéndose a disposición del programador para que éste pueda comunicarse con los dispositivos de hardware y software existentes (3).

Java 8.0

Java es un lenguaje de programación de alto nivel desarrollado por la compañía Sun Microsystem a principios de la década del 90, con la idea original de utilizarlo en la creación de páginas web. Su sintaxis es muy parecida a la de C y C++ ya que fueron inspirados en los mismos. Los criterios de diseño de Java fueron: independiente de la máquina, seguro para trabajar en red y potente para sustituir código nativo. La principal característica de Java es la de ser un lenguaje compilado e interpretado. Todo programa en Java

debe compilarse y el código que se genera *bytecodes* es interpretado por una máquina virtual lográndose de esta forma la independencia de la máquina, el código compilado se ejecuta en máquinas virtuales que son dependientes de la plataforma (4).

Entre las principales características del lenguaje se encuentran:

- Orientado a objetos: Soporta las características esenciales del paradigma de la programación orientada a objetos: encapsulamiento, herencia y polimorfismo.
- En el diseño de Java se prestó especial atención a la seguridad. Existen varios niveles de seguridad en Java, desde el ámbito del programador, hasta el ámbito de la ejecución en la máquina virtual.
- Una fuente común de errores en programación proviene del uso de punteros. En Java se han eliminado los punteros, el acceso a las instancias de clase se hace a través de referencias.
- Otra característica de Java es que está preparado para la programación concurrente sin necesidad de utilizar ningún tipo de biblioteca.
- El mismo código Java que funciona en un sistema operativo, funciona en cualquier otro que tenga instalada la máquina virtual de Java por lo que se considera multiplataforma (4).

Para el desarrollo de la solución se escoge **Java** por su eficiencia en la tecnología, seguridad que aporta y ser multiplataforma lo han convertido en una tecnología ideal para el desarrollo de aplicaciones, además el equipo de desarrollo cuenta con conocimientos sobre el mismo y el COJ está desarrollado en este lenguaje.

1.3.2 Framework

Un *framework* es una estructura software compuesta de componentes personalizables e intercambiables para el desarrollo de una aplicación. En otras palabras, un *framework* se puede considerar como una aplicación genérica incompleta y configurable a la que podemos añadirle las últimas piezas para construir una aplicación concreta.

Los objetivos principales que persigue un *framework* son: acelerar el proceso de desarrollo, reutilizar código ya existente y promover buenas prácticas de desarrollo como el uso de patrones.

Un *framework* Web, por tanto, podemos definirlo como un conjunto de componentes, ejemplo clases en Java y descriptores y archivos de configuración en Lenguajes de Marcas Extensibles por sus siglas en inglés (XML), que componen un diseño reutilizable que facilita y agiliza el desarrollo de sistemas Web (5). Para

Java existen varios marcos de trabajo, entre los que se destacan Spring MVC, JSF, Vaadin, GWT (*Google Web Toolkit*), Grails, Play 2, entre otros.

Spring 3.2.3

Spring es un *framework* de código abierto muy popular y ampliamente desarrollado, es el *framework* Java utilizado por excelencia para el desarrollo de aplicaciones empresariales. Es uno de los *frameworks* más utilizados hoy en día. El objetivo de Spring es simplificar el desarrollo de aplicaciones empresariales Java. A grandes rasgos, un *framework* es un conjunto de clases que nos permiten resolver un problema en específico. En el caso particular de Spring, nos permite resolver muchos de los problemas que se presentan al desarrollar aplicaciones con tecnología JEE (*Java Enterprise Edition*). Una de las mayores ventajas de Spring, es la forma modular en el que fue creado, permitiendo habilitar/deshabilitar las características a utilizar según se requiera. Spring es utilizado en proyectos muy diversos, como puede ser en instituciones bancarias, aseguradoras, instituciones educativas y de gobierno, entre muchos otros tipos de proyectos y empresas (6).

Entre sus características principales se encuentran:

- Spring permite desarrollar aplicaciones flexibles, altamente cohesivas y con un bajo acoplamiento.
- Spring permitió simplificar el desarrollo JEE al utilizar clases Java Simples (*POJO – Plain Old Java Object*) para la configuración de servicios.

Debido a que muchos proyectos muestran las mismas tareas a realizar una y otra vez, tales como Localización de Servicios, Manejo de Transacciones, Manejo de Excepciones, Parametrización de la aplicación, entre muchos más. Spring permite resolver muchos de estos problemas de manera muy simple.

Para lograr lo anterior el *framework* se basa en dos conceptos fundamental:

- DI (*Dependency Injection*): Este patrón de diseño permite suministrar objetos a una clase (POJO) que tiene dependencias.
- AOP (*Aspect Oriented Programming*): AOP es un paradigma de programación que permite modularizar las aplicaciones y mejorar la separación de responsabilidades entre módulos y/o clases.

Las características anteriores son la base para la creación de contenedores ligeros (*lightweight containers*). Spring es uno de los contenedores ligeros más completos y populares al día de hoy.

Spring cuenta con varios Módulos sus principales son:

- Spring Core: Este módulo provee la funcionalidad básica de la fábrica de Spring. El componente principal es *BeanFactory*, el cual aplica el concepto de *Inversion of Control* (IoC) o también conocido como *Dependency Injection* (DI).
- Spring Context: Aquí es donde se realiza la configuración del *framework*. Incluye la configuración de servicios empresariales tales como JNDI, EJB, Internacionalización, validación, entre varios más.
- Spring AOP: Permite aplicar los conceptos de Programación Orientada a Aspectos (AOP), además incluye clases de soporte para el manejo transaccional, la seguridad, entre varias clases más, permitiendo desacoplar estas características de nuestra aplicación.
- Spring DAO: Permite aplicar conceptos de la capa de datos *Data Access Object* (DAO) a través de POJOs, abstrayendo la complejidad, permitiendo crear un código JDBC más limpio y simple.
- Spring ORM: Permite integrarse con tecnologías tales como JPA, Hibernate, entre otras.
- Spring Web: Permite el desarrollo y la integración con tecnologías como Struts, JSF, Tapestry, entre otros.
- Spring MVC: Este módulo implementa el patrón MVC para ser utilizado en la capa de presentación (6).

Pare el desarrollo del trabajo será necesario el uso del *framework* Spring, debido a que permite desarrollar aplicaciones flexibles, es de código abierto, pero fundamentalmente porque el COJ fue desarrollado en Spring y para poder interactuar con su núcleo es necesario el uso de este *framework* en específico.

1.3.3 Servidor de base de datos

Servidores de Bases de Datos. También conocidos como RDBMS (acrónimo en inglés de *Relational DataBase Management Systems*), son programas que permiten organizar datos en una o más tablas relacionadas. Los servidores de Bases de Datos se utilizan en todo el mundo en una amplia variedad de aplicaciones. Los servidores de base de datos son programas que proveen servicios de base de datos a otros programas u otras computadoras, como es definido por el modelo cliente-servidor. También puede hacer referencia a aquellas computadoras (servidores) dedicadas a ejecutar esos programas, prestando el servicio (7).

PostgreSQL 9.3

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD y con su código fuente disponible libremente. Es el sistema de gestión de bases de datos de código abierto más potente del mercado y en sus últimas versiones están a la altura de otros sistemas gestores propietarios. PostgreSQL utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando (8).

La última serie de producción es la 9.3. Sus características técnicas la hacen una de las bases de datos más potentes y robustas del mercado. Su desarrollo comenzó hace más de 16 años, y durante este tiempo, estabilidad, potencia, robustez, facilidad *de administración e implementación de estándares* han sido las características que más se han tenido en cuenta durante su desarrollo. PostgreSQL funciona muy bien con grandes cantidades de datos y una alta concurrencia de usuarios accediendo a la vez al sistema (8).

Entre sus principales características se encuentran:

- Es una base de datos 100% ACID (características de los parámetros que permiten clasificar las transacciones de los sistemas de gestión de bases de datos: atomicidad, consistencia, aislamiento, durabilidad)
- Integridad referencial
- Replicación asincrónica/sincrónica / *Streaming replication - Hot Standby*
- Copias de seguridad en caliente (*Online/hot backups*)
- Juegos de caracteres internacionales
- Regionalización por columna
- Múltiples métodos de autentificación
- Acceso encriptado vía capa de puertos seguros por sus siglas en inglés (SSL)
- Completa documentación
- Licencia BSD
- Disponible para Linux y UNIX en todas sus variantes (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64) y Windows 32/64bit (9).

Para el desarrollo de la solución será necesario el uso de PostgreSQL**,** debido a que la información se encuentra almacenada en una base de datos gestionada por este sistema, además de que es un software libre del cual se dispone de una vasta documentación en la universidad y al estar la herramienta incluida en el programa de la disciplina de Bases de Datos el equipo está adiestrado en su uso.

1.3.4 Entorno Integrado de Desarrollo

Un IDE (*Integrated Development Environment*) es un entorno de programación que ha sido empaquetado como un programa de aplicación, o sea, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica. Los IDEs pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes. Estos pueden ser usados para uno o varios lenguajes de programación (10).

Un IDE debe tener las siguientes características:

- Multiplataforma
- Soporte para diversos lenguajes de programación
- Integración con Sistemas de Control de Versiones
- Reconocimiento de Sintaxis
- Extensiones y Componentes para el IDE
- Integración con Framework populares
- Depurador
- Importar y Exportar proyectos
- Múltiples idiomas
- Manual de Usuarios y Ayuda.

NetBeans 7.4

Permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos. Un módulo es un archivo Java que contiene clases de Java escritas para interactuar con las APIs de NetBeans y un archivo especial (*manifest file*) que lo identifica como módulo. Las aplicaciones construidas a partir de módulos pueden ser extendidas agregándole nuevos módulos. Debido a que los módulos pueden ser desarrollados independientemente, las aplicaciones basadas en la plataforma NetBeans pueden ser extendidas fácilmente por otros desarrolladores de software.

NetBeans es un proyecto de código abierto de gran éxito con una gran base de usuarios, una comunidad en constante crecimiento, y con cerca de 100 socios en todo el mundo. Sun MicroSystems fundó el proyecto de código abierto NetBeans en junio de 2000 y continúa siendo el patrocinador principal de los proyectos.

La Plataforma NetBeans es una base modular y extensible usada como una estructura de integración para crear aplicaciones de escritorio grandes. Empresas independientes asociadas, especializadas en desarrollo de software, proporcionan extensiones adicionales que se integran fácilmente en la plataforma y que pueden también utilizarse para desarrollar sus propias herramientas y soluciones (10).

Entre las características de la plataforma están:

- Administración de las interfaces de usuario (menús y barras de herramientas)
- Administración de las configuraciones del usuario
- Administración del almacenamiento (guardando y cargando cualquier tipo de dato)
- Administración de ventanas
- *Framework* basado en asistentes (diálogos paso a paso)

1.3.5 Servidor web

Es un programa que gestiona cualquier aplicación en el lado del servidor realizando conexiones bidireccionales y/o unidireccionales y síncronas o asíncronas con el cliente generando una respuesta en cualquier lenguaje o aplicación en el lado del cliente. El código recibido por el cliente suele ser compilado y ejecutado por un navegador web. Para la transmisión de todos estos datos se utiliza algún protocolo. Generalmente se utiliza el protocolo HTTP para estas comunicaciones, perteneciente a la capa de aplicación del Modelo OSI (11). El término también se emplea para referirse al ordenador que ejecuta el programa. Para Java existen varios servidores como: GlassFish, JBoss y Apache Tomcat.

Apache Tomcat 7.0.3

Tomcat (también conocido como Jakarta Tomcat o Apache Tomcat) es una implementación de software de código abierto de Java Servlet y tecnologías *JavaServer Pages* (JSP). Apache Tomcat es desarrollado, en un entorno abierto y participativo y publicado bajo la licencia Apache versión 2, por miembros de la *Apache*

Software Foundation y voluntarios independientes. El proyecto tiene la intención de ser una colaboración de los mejores desarrolladores de su clase de todo el mundo.

Tomcat es un servidor Web con soporte para *servlet* y *JSP*. No es un servidor de aplicaciones, como *JBoss* o *JOnAS*. Trae incluido el compilador *Jasper*, que compila *JSP* convirtiéndolas en *servlets*. El motor de *servlets* de Tomcat a menudo se presenta en combinación con el servidor Web Apache. A partir de la versión 4.0, Tomcat utiliza el contenedor de *servlets* Catalina. *Tomcat* puede funcionar como servidor web por sí mismo. El hecho de que *Tomcat* fue escrito en Java, hace posible que funcione en cualquier sistema operativo que disponga de la máquina virtual Java (12).

Para el desarrollo del trabajo de diploma se escoge Apache Tomcat versión 7.0.3 como servidor web ya que el COJ está montado sobre este servidor web.

1.3.6 Herramientas CASE para el modelado.

Computer Aided Software Engineering (CASE) Ingeniería de Software Asistida por Computación. Esencialmente un CASE es una herramienta que sirve de ayuda al ingeniero a desarrollar y mantener software. Se pueden utilizar en todo el ciclo de vida de desarrollo del software en tareas como el diseño de proyectos, planificación, diseño de diagramas, cálculos de costo, implementación de código automático basado en un diseño dado, compilación de código y documentación entre otros (13).

Visual Paradigm 8.0

Es una herramienta CASE: Ingeniería de Software Asistida por Computación. La misma propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación. *Visual Paradigm* ha sido concebida para soportar el ciclo de vida completo del proceso de desarrollo del software a través de la representación de todo tipo de diagramas. El *Visual Paradigm* constituye una herramienta de software libre de probada utilidad para el analista de un proyecto. Se caracteriza por la disponibilidad en múltiples plataformas (Windows, Linux). Posee características como, diseño centrado en casos de uso y enfocado al negocio que genera un software de mayor calidad. Además de que el modelo y el código permanecen sincronizados en todo el ciclo de desarrollo. Posee la ventaja que permite la generación de BD,

transformación de diagramas de Entidad Relación en tablas de BD. Posee compatibilidad entre ediciones (14).

Rational Rose.

IBM *Rational Rose Enterprise* proporciona un conjunto de prestaciones controladas por modelo para desarrollar muchas aplicaciones de software, incluidas aplicaciones Ada, ANSI C++, C++, CORBA, Java, Java EE, Visual C++ y Visual Basic. El software permite acelerar el desarrollo de estas aplicaciones con código generado a partir de modelos visuales mediante el lenguaje UML (*Unified Modeling Language*). *Rational Rose Enterprise* ofrece una herramienta y un lenguaje de modelado común para simplificar el entorno de trabajo y permitir una creación más rápida de software de calidad.

- Modelado de las aplicaciones más habituales: proporciona prestaciones de modelado visual para desarrollar muchos tipos de aplicaciones de software.
- Desarrollo de aplicaciones para la web: contiene herramientas web y XML para el modelado de aplicaciones web.
- Integración del diseño de aplicaciones con el desarrollo: unifica el equipo del proyecto proporcionando una ejecución y una notación de modelos UML comunes (15).

Fundamentación de la elección

La herramienta CASE que se utilizará para el sistema en desarrollo será el *Visual Paradigm*, eficaz para el proceso de modelado de todo tipo de diagramas necesarios así como interfaces de usuarios. Pertenece a la familia de software libre y la tendencia que se lleva en la UCI es promover su uso.

1.3.7 Metodologías de desarrollo de software

Una metodología es un conjunto integrado de técnicas y métodos que permite abordar de forma homogénea y abierta cada una de las actividades del ciclo de vida de un proyecto de desarrollo. Es un proceso de software detallado y completo (16).

Las metodologías se basan en una combinación de los modelos de proceso genéricos (cascada, incremental). Definen artefactos, roles y actividades, junto con prácticas y técnicas recomendadas (16).

La metodología para el desarrollo de software en un modo sistemático de realizar, gestionar y administrar un proyecto para llevarlo a cabo con altas posibilidades de éxito. Una metodología para el desarrollo de software comprende los procesos a seguir sistemáticamente para idear, implementar y mantener un producto software desde que surge la necesidad del producto hasta que cumplimos el objetivo por el cual fue creado (16).

Podemos destacar que una metodología:

- Optimiza el proceso y el producto software.
- Métodos que guían en la planificación y en el desarrollo del software.
- Define qué hacer, cómo y cuándo durante todo el desarrollo y mantenimiento de un proyecto.

Una metodología de desarrollo de software o metodología de desarrollo de sistemas en ingeniería de software es un marco de trabajo que se usa para estructurar, planificar y controlar el proceso de desarrollo de un sistema de información (16).

Para la selección de la metodología de desarrollo es necesario analizar primero que tipo de enfoque será utilizado. Existen dos tipos de enfoque: el prescriptivo o tradicional y el ágil.

Metodologías tradicionales

Las metodologías tradicionales son denominadas, a veces, como metodologías pesadas. Centran su atención en llevar una documentación exhaustiva de todo el proyecto y en cumplir con un plan de proyecto, definido todo esto, en la fase inicial del desarrollo del proyecto. Otra de las características importantes dentro de este enfoque, son los altos costes al implementar un cambio y la falta de flexibilidad en proyectos donde el entorno es volátil. Las metodologías tradicionales (formales) se focalizan en la documentación, planificación y procesos (16).

Metodologías ágiles

Este enfoque nace como respuesta a los problemas que puedan ocasionar las metodologías tradicionales y se basa en dos aspectos fundamentales, retrasar las decisiones y la planificación adaptativa. Basan su fundamento en la adaptabilidad de los procesos de desarrollo. Estas metodologías ponen de relevancia que la capacidad de respuesta a un cambio es más importante que el seguimiento estricto de un plan (16).

Selección del enfoque

Para la selección del enfoque se utiliza el método de Boehm y Turner que plantea 5 criterios fundamentales mediante los que se estará valorando el proyecto; estos son: tamaño del equipo, criticidad del producto, dinamismo de los cambios, cultura del equipo y personal con que se cuenta. Cada uno de esos criterios tiene elementos que lo discriminan y por tanto se tienen en cuenta a la hora de seleccionar uno u otro enfoque (17).

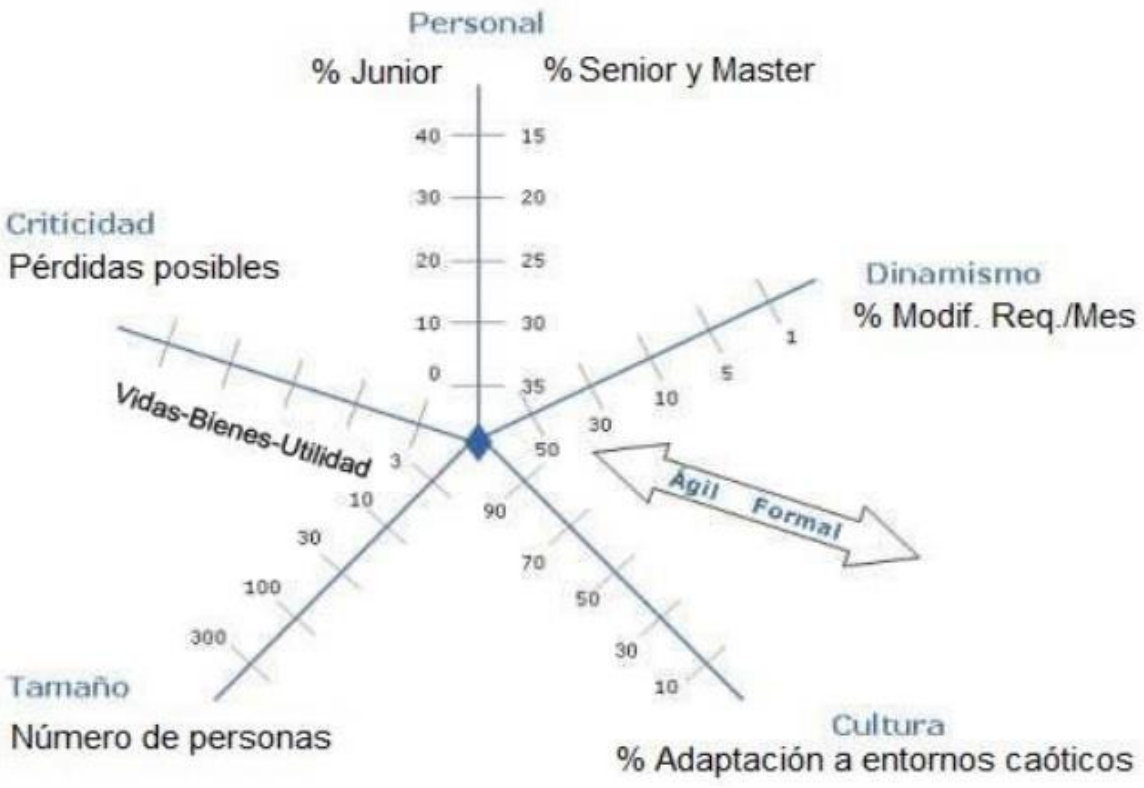


Ilustración 3Estrella de Boehm-Turner (17)

El significado de cada una de las aristas que conforman la estrella se explica a continuación:

Tamaño: Este criterio se utiliza para representar el número de personas involucradas en el proyecto. Pueden tenerse en cuenta el nivel de complejidad que pueda presentarse en la comunicación entre los miembros del proyecto y los costos que pueden provocar cambios esperados.

Criticidad: Se utiliza para evaluar la naturaleza del daño ocasionado por defectos que no hayan sido detectados al producto. Su evaluación puede ser cualitativa.

Dinamismo: Representa la rapidez con la que pueden estar cambiando los requerimientos del proyecto.

Personal: Representa la proporción del personal con experiencia alta, media y baja. Los métodos orientados al plan no se ven afectados negativamente por este factor pues no interesa el nivel de experiencia con la que cuenten los miembros del equipo.

Cultura: Las organizaciones y las personas que relaciona el proyecto pueden depender de la confianza o de la relación contractual. Esto refleja el nivel de ceremonia necesario y aceptado: documentación, control, formalismo en las comunicaciones (17).

Debido a que en la actualidad el ambiente de negocio es muy variable, un proyecto puede sufrir muchos cambios en su transcurso de desarrollo por motivos como la variación de las peticiones del cliente lo cual provoca cambios en los requisitos y funcionalidades del mismo, por lo que es necesarios minimizar la respuesta ante un cambio garantizando la entrega del producto en tiempo. También se cuenta con un equipo conformando por dos personas no expertas, con conocimientos básicos de las tecnologías a emplear. Por lo anterior mencionado se escoge el enfoque ágil para la metodología de desarrollo.

Agile Unified Process variante UCI

El Proceso Unificado Ágil de Scott Ambler o *Agile Unified Process* (AUP) en inglés es una versión simplificada del Proceso Unificado de *Rational* (RUP). Este describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de software de negocio usando técnicas ágiles y conceptos que aún se mantienen válidos en RUP. El AUP aplica técnicas ágiles incluyendo:

- Desarrollo Dirigido por Pruebas (*test driven development* - TDD en inglés)
- Modelado ágil
- Gestión de cambios ágil
- Refactorización de Base de Datos para mejorar la productividad.

Al igual que en RUP, en AUP se establecen cuatro fases que transcurren de manera consecutiva.

Fases AUP variante UCI

- **Inicio:** Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo y decidir si se ejecuta o no el proyecto.

- **Ejecución:** En esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto.

- **Cierre:** En esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto (18).

Disciplinas AUP variante UCI

AUP define 7 disciplinas (4 ingenieriles y 3 de gestión de proyectos), las disciplinas son:

- **Modelado de negocio:** El Modelado del Negocio es la disciplina destinada a comprender los procesos de negocio de una organización. Se comprende cómo funciona el negocio que se desea informatizar para tener garantías de que el software desarrollado va a cumplir su propósito.

- **Requisitos:** El esfuerzo principal en la disciplina Requisitos es desarrollar un modelo del sistema que se va a construir. Esta disciplina comprende la administración y gestión de los requisitos funcionales y no funcionales del producto.

- **Análisis y diseño:** En esta disciplina, si se considera necesario, los requisitos pueden ser refinados y estructurados para conseguir una comprensión más precisa de estos, y una descripción que sea fácil de mantener y ayude a la estructuración del sistema (incluyendo su arquitectura). Además, en esta disciplina se modela el sistema y su forma (incluida su arquitectura) para que soporte todos los requisitos, incluyendo los requisitos no funcionales. Los modelos desarrollados son más formales y específicos que el de análisis.

- **Implementación:** En la implementación, a partir de los resultados del Análisis y Diseño se construye el sistema.

- **Pruebas interna:** En esta disciplina se verifica el resultado de la implementación probando cada construcción, incluyendo tanto las construcciones internas como intermedias, así como las versiones finales a ser liberadas. Se deben desarrollar artefactos de prueba como: diseños de casos de prueba, listas de chequeo y de ser posible componentes de prueba ejecutables para automatizar las pruebas

- **Pruebas de liberación:** Pruebas diseñadas y ejecutadas por una entidad certificadora de la calidad externa, a todos los entregables de los proyectos antes de ser entregados al cliente para su aceptación.

- **Pruebas de Aceptación:** Es la prueba final antes del despliegue del sistema. Su objetivo es verificar que el software está listo y que puede ser usado por usuarios finales para ejecutar aquellas funciones y tareas para las cuales el software fue construido (18).

Roles definidos por AUP variante UCI

- **Administrador de proyecto:** Maneja a los miembros construye relaciones con los *stakeholders*, coordina interacciones con los *stakeholders*, planea, maneja y asigna los recursos.
- **Ingeniero de procesos:** Desarrolla, adapta y apoya sus materiales del proceso del software.
- **Desarrollador:** Escribe, testea y construye software.
- **Administrador de Base de Datos:** Diseña, prueba, desarrolla, y apoya los esquemas de la BD.
- **Modelador ágil:** Crea y desarrolla modelos, bosquejos o los archivos de la herramienta CASE, de una manera evolutiva y de colaboración.
- **Administrador de la configuración:** Un encargado de la configuración es responsable de proporcionar la infraestructura total y el ambiente del CM al equipo de desarrollo.
- ***Stakeholder*.**
- **Administrador de pruebas:** Responsables del éxito de la prueba, incluyendo el planeamiento, la gerencia, y la defensa para la prueba y las actividades de la calidad.
- Probador: Encargado de ejecutar las pruebas (18).

Xtreme Programming

La programación extrema (XP) es un enfoque de la ingeniería del software formulado por Kent Beck. Es el más destacado de los procesos ágiles de desarrollo de software. Al igual que éstos, la programación extrema se diferencia de las metodologías tradicionales principalmente en que pone más énfasis en la adaptabilidad que en la previsibilidad. Los defensores de XP consideran que los cambios de requisitos sobre la marcha son un aspecto natural, inevitable e incluso deseable del desarrollo de proyectos. Creen que ser capaz de adaptarse a los cambios de requisitos en cualquier punto de la vida del proyecto es una aproximación mejor y más realista que intentar definir todos los requisitos al comienzo del proyecto e invertir esfuerzos después en controlar los cambios en los requisitos. Se puede considerar la programación extrema como la adopción de las mejores metodologías de desarrollo de acuerdo a lo que se pretende llevar a cabo con el proyecto y aplicarlo de manera dinámica durante el ciclo de vida del software. XP es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en la realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico. A Kent Beck se le considera el padre de XP (16).

Scrum

Scrum es un proceso ágil que se puede usar para gestionar y controlar desarrollos complejos de software y productos usando prácticas iterativas e incrementales. *Scrum* es un proceso incremental iterativo para desarrollar cualquier producto o gestionar cualquier trabajo. Aunque *Scrum* estaba previsto que fuera para la gestión de proyectos de desarrollo de software, se puede usar también para la ejecución de equipos de mantenimiento de software o como un enfoque de gestión de programas (16).

Análisis de la selección de la metodología de desarrollo

Luego de realizar un estudio de las metodologías anteriores se escoge AUP versión UCI como metodología de desarrollo a utilizar pues está especialmente indicada para pequeños equipos de trabajo donde existe un alto riesgo técnico y se orienta a una entrega rápida de resultados y una alta flexibilidad. AUP tiene bien

definido y organizado el proceso de desarrollo de software lo que permite que estén claros los pasos a seguir para lograr un producto con calidad en el tiempo definido, además que genera los mismo artefactos de la metodología RUP, artefactos con los cuales el equipo de trabajo está familiarizado.

1.4 Conclusiones parciales

Después de haber realizado un estudio de los principales jueces en línea a nivel internacional que cuentan con un módulo de concursos virtuales se decide que el módulo debe permitir crear concursos virtuales con problemas de competencias anteriores y de entrenamiento con problemas aleatorios escogidos por el sistema o por el propio usuario, deben ser competencias públicas o por usuarios específicos teniendo como única restricción que el usuario esté conectado para poder invitarlo a la competencia y por último que los concursos puedan ser modificados y eliminados por el creador según lo requiera. En cuanto al estudio sobre las principales tecnologías a emplear en el desarrollo de la solución debido a que el COJ está programado en Java, haciendo un uso del *framework* de presentación *Spring*, servidor web *Apache Tomcat* y servidor de base de datos PostgreSQL, cualquier nueva funcionalidad que desee aprovechar las características de COJ e interactuar con su núcleo debe ser escrita utilizando estas tecnologías. La metodología escogida permitirá garantiza la calidad y completitud del software guiando procesos como el desarrollo del software, la gestión, la configuración y los cambios en el mismo.

CAPÍTULO 2. ANÁLISIS Y DISEÑO DEL SISTEMA

En el presente capítulo se abordará el proceso de desarrollo del módulo de concursos virtuales para el COJ en las etapas de análisis, diseño haciendo uso de la metodología de desarrollo AUP variante UCI la cual define en cada una de las fases un grupo de artefactos que brindan una comprensión clara del negocio y sistema a desarrollar. En el presente capítulo se definen artefactos como: Modelo de dominio, la especificación de requisitos funcionales y no funcionales, descripción de las historias de usuarios que tendrá la propuesta de solución, diagramas de clases del diseño y diagramas de clases del análisis, así como los diagramas de secuencias. También se describen los patrones arquitectónico y patrones de diseño a utilizar.

2.1 Propuesta de solución

El módulo de concursos virtuales estará enfocado en permitirles a los usuarios gestionar concursos virtuales y de práctica. Los usuarios podrán crear concursos virtuales que es recrear una competencia pasada con todos los acontecimientos de la competencia y podrán crear concursos de práctica en el cual se escoge problemas que se encuentran almacenados en las bases de datos del COJ. Una vez creado un concurso virtual los usuarios tendrán acceso en caso de ser público y en caso de ser por usuarios específicos tendrá acceso de ser invitado al concurso. Los concursos contaran con funcionalidades necesarias para su realización como: listar problemas, descripción del problema seleccionado, tabla de posiciones, envíos realizados, enviar solución, descripción del concurso.

2.2 Modelo de dominio

Un modelo de dominio muestra a los modeladores clases conceptuales significativas en un dominio del problema. Un modelo de dominio es la representación de las clases conceptuales del mundo real, no de componentes de software. No se trata de un conjunto de diagramas que de diagramas que describan clases de software, u objetos software con responsabilidades (19).

2.2.1 Conceptos del dominio

Usuario: persona registrada en el sistema que tiene permisos para enviar intentos de solución y participar en competencias.

Problema: todos los problemas del COJ que los usuarios pueden realizar envíos de soluciones de ellos.

Envío: solución en forma de código que realiza un usuario a un problema del Juez en Línea Caribeño.

Lenguaje: tipos de lenguajes de programación en los que se envían las soluciones.

Concurso Real: Concursos reales que ya tuvieron lugar y quedan registrados en el sistema

Concurso de práctica: concurso creado con problemas escogidos por el creador del mismo y no queda registrado en el sistema.

Concurso virtual: concurso creado a partir de un concurso real pasado, en el cual su transcurso será idéntico a como paso en su momento.

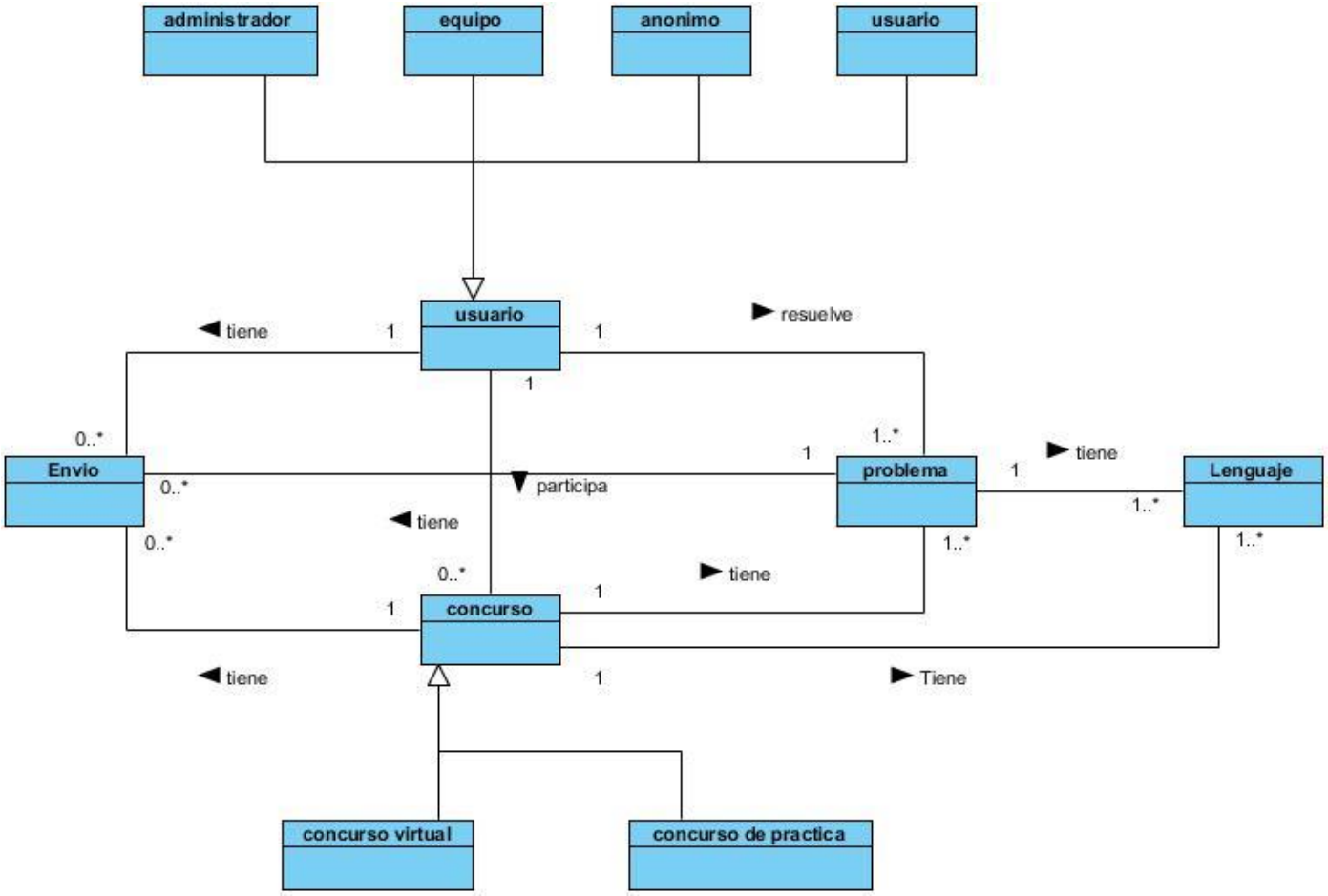


Ilustración 4Modelo de Dominio

2.3 Requerimientos del sistema

Las definiciones de requerimientos del sistema son para especificar las funciones que debe cumplir el mismo y sus propiedades esenciales. Para crear estas definiciones se requiere consultar con los clientes del sistema y usuarios finales. La fase de definición de requerimientos se puede derivar en tres tipos que son: requerimientos funcionales, requerimientos no funcionales o del sistema y características que no debe mostrar el sistema (20).

2.3.1 Requerimientos funcionales

RF1 Incluir concurso virtual
RF2 Eliminar concurso
RF3 Incluir concurso de práctica
RF4 Ver datos de concursos virtuales y de práctica creados por el propio usuario
RF5 Ver datos de todos los concursos virtuales y de práctica
RF6 Realizar concurso virtual o de práctica
RF7 Listar problemas
RF8 Ver problema
RF9 Ver tabla de posiciones
RF10 Ver envíos de soluciones de un concurso
RF11 Enviar solución
RF12 Administrar concursos virtuales
RF13 Listar los concursos virtuales y de práctica creados por el propio usuario
RF14 Listas todos los concursos virtuales y de práctica
RF15 Filtrar búsqueda de concursos
RF16 Ver descripción del concurso actual
RF17 Filtrar búsqueda de envíos
RF18 Filtrar búsqueda de usuario
RF19 Filtrar búsqueda de problemas

Descripción de los requerimientos del software

Las descripciones de las historias de usuarios contienen los detalles del flujo de eventos a ejecutar, las precondiciones, comportamientos válidos y no válidos de los requisitos así como la interfaz de usuario de

cada requisito. A continuación, se presenta la descripción de la HU1 Incluir concurso virtual en la (Tabla 2), HU2 Eliminar concurso en la (Tabla 3) y HU3 en la (Tabla 4).

Tabla 2HU1 Incluir concurso virtual

Número: 1	**Nombre del requisito:** Incluir concurso virtual
Programador: Alejandro Cancio Machado	Iteración asignada:1
Prioridad: Alta	Tiempo estimado: 10 días
N/A	Tiempo real: 12 días

Descripción:

1- Objetivo:

Permitir incluir concursos virtuales en el sistema.

2- Acciones para lograr el objetivo (precondiciones y datos):

Para incluir un concurso virtual hay que:

- Tener en cuenta los siguientes datos: plantilla, fecha, hora de inicio, tipo de acceso, tipo de concurso y usuarios en caso de ser cerrado.
- Estar autenticado en el sistema con el rol usuario o administrador.

3- Comportamientos válidos y no válidos (flujo central y alternos):

Los campos plantilla, fecha, hora de inicio, tipo de acceso, tipo de concurso y usuarios son obligatorios.

Plantilla: campo en el cual el usuario escoge el concurso que desea revivir.

Fecha: está dividido en año, mes y, día en el caso del día admite números del 1 al 31, en el caso del mes admite números del 1 al 12 y en caso del año admite números del año actual y el que le sigue.

Hora de inicio: está dividido en hora, minutos, segundos, en el caso de la hora admite números entre el 0 y el 23, en el caso de los minutos admite números entre el 0 y el 59 al igual que el caso de los segundos.

Tipo de acceso: Campo en el cual el usuario marca si el concurso será público o privado en caso de ser privado

tiene que añadir los usuarios que desee que participen. Tipo de concurso: se escoge entre concurso virtual y concurso de práctica. Usuarios: Se escogen los usuarios que desee que participen en caso de ser privado en acceso al concurso. **4- Flujo de la acción a realizar:** - El sistema debe permitir incluir y/o seleccionar los datos para incluir un nuevo concurso virtual. - Cuando el usuario incluye y/o selecciona correctamente los datos necesarios para incluir un concurso virtual y selecciona la opción Crear, se crea un nuevo elemento y se redirige a la página Mi lista donde se encuentran todos los concursos creados por ese usuario. - Si los datos están incompletos o incorrectos se señalarán los campos en cuestión dando la posibilidad al usuario de realizar nuevamente la acción en cuestión. - Si selecciona la opción Resetear refresca la página con los campos vacíos.
Observaciones: los concursos virtuales tienen dependencia de los concursos reales pasados.
Prototipo de interfaz:

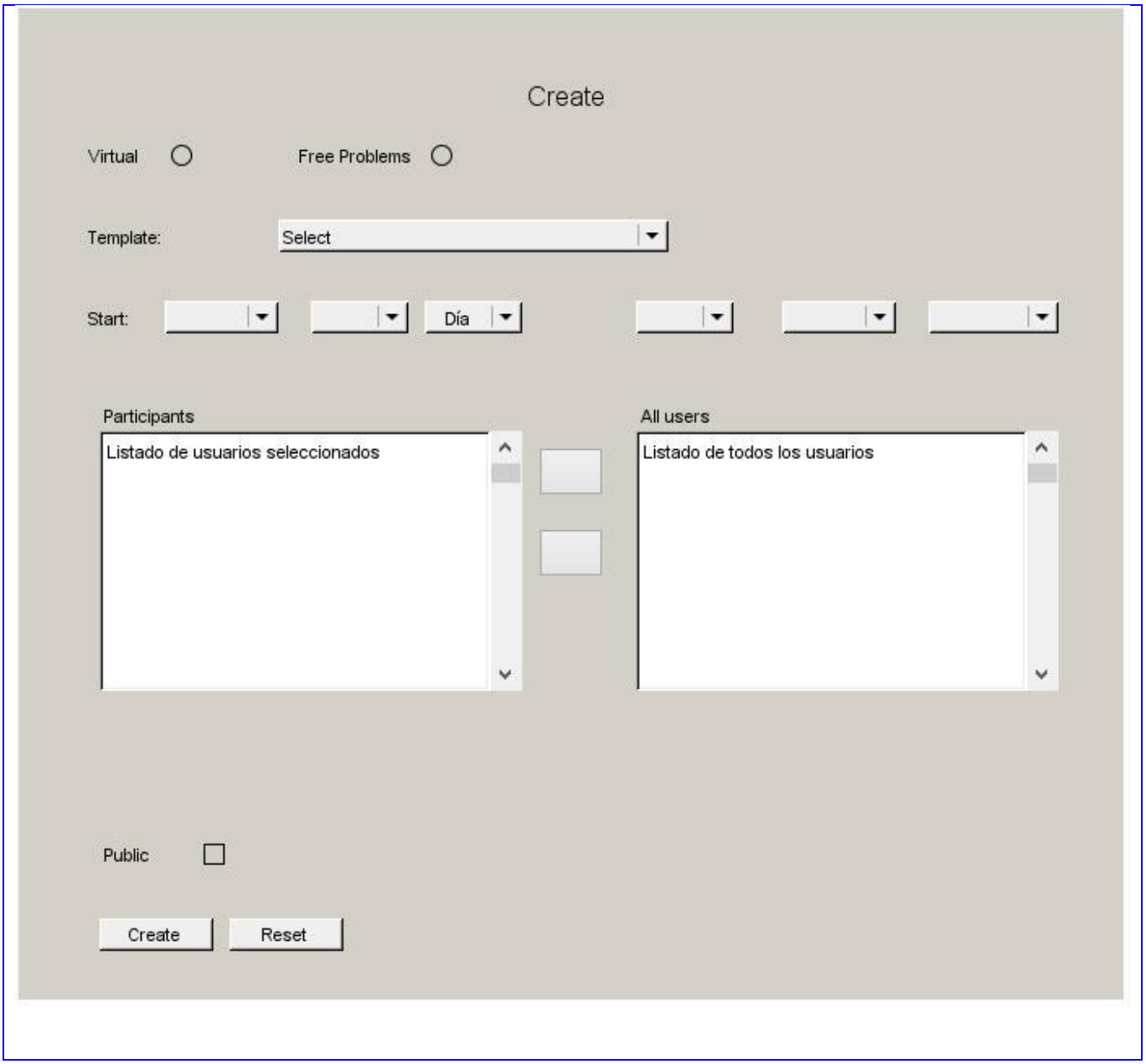

Create
Virtual
Free Problems
Template:
Select
Start:
Día
Participants
Listado de usuarios seleccionados
All users
Listado de todos los usuarios
Public
Create
Reset

Tabla 3HU2 Eliminar concurso

Número: 2	**Nombre del requisito:** Eliminar concurso
Programador: Alejandro Cancio Machado	Iteración asignada:1
Prioridad: Alta	Tiempo estimado: 7 días
N/A	Tiempo real: 7 días
Descripción: **1- Objetivo:** Permitir eliminar un concurso virtual en el sistema. **2- Acciones para lograr el objetivo (precondiciones y datos):** Para eliminar una categoría hay que: - Estar autenticado en el sistema con el rol usuario o administrador. - Debe existir en el sistema el concurso virtual. - El concurso puede ser eliminado por el usuario creador del mismo o un administrador. **3- Flujo de la acción a realizar:** - El sistema permite eliminar un concurso, para ello se puede realizar seleccionando la opción Eliminar de las opciones que muestra el propio elemento o desde el panel administrativo por un administrador. Para ambos casos el sistema muestra un mensaje de confirmación para Aceptar o no la acción que se está realizando. - Si selecciona la opción Aceptar se actualiza el listado de concursos creados y el sistema muestra un mensaje de información.	
Observaciones: los concursos virtuales tienen dependencia de los concursos reales pasados.	
. **Prototipo de interfaz:**	

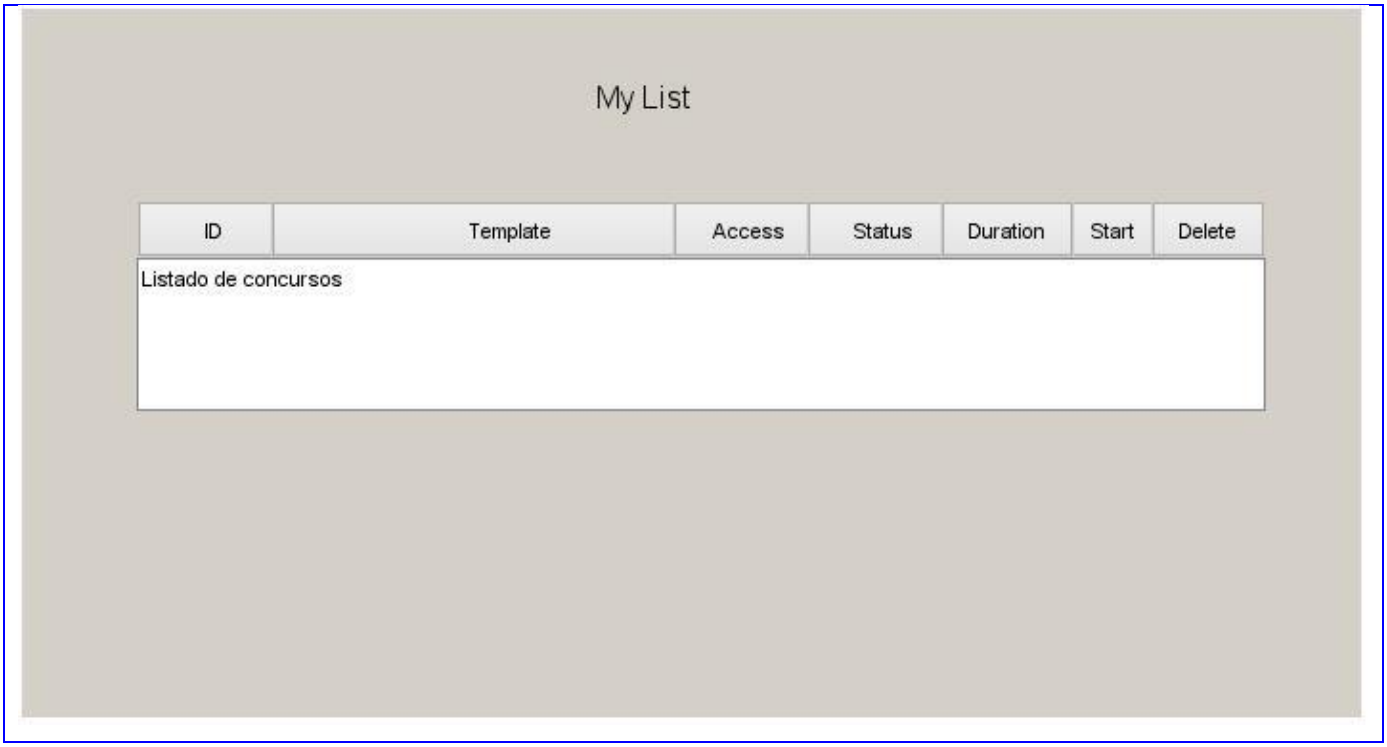


Tabla 4HU3 Incluir concurso de práctica

Número: 3	**Nombre del requisito:** Incluir concurso de práctica
Programador: Osvaldo Posada López	Iteración asignada: 1
Prioridad: Alta	Tiempo estimado: 10 días
N/A	Tiempo real: 5 días

Descripción:

1- Objetivo:

Permitir incluir concursos de práctica para que los usuarios participen y se preparen.

2- Acciones para lograr el objetivo (precondiciones y datos):

Para incluir un concurso de práctica hay que:

- Tener en cuenta los siguientes datos: plantilla, fecha, hora, tipo de acceso, tipo de concurso, problemas son obligatorios.
- Estar autenticado en el sistema con el rol usuario o administrador.

3- Comportamientos válidos y no válidos (flujo central y alternos):

Los campos plantilla, fecha, hora, tipo de acceso, problemas, usuarios, tipo de concurso problemas son obligatorios.

Plantilla: campo en el cual el usuario escoge el concurso que desea revivir.

Fecha: está dividido en año, mes y, día en el caso del día admite números del 1 al 31, en el caso del mes admite números del 1 al 12 y en caso del año admite números del año actual y el que le sigue.

Hora de inicio: está dividido en hora, minutos, segundos, en el caso de la hora admite números entre el 0 y el 23, en el caso de los minutos admite números entre el 0 y el 59 al igual que el caso de los segundos.

Tipo de concurso: se escoge entre concurso virtual y concurso de práctica.

Problemas: Campo en el cual el usuario escoge los problemas que desea añadir al concurso de práctica.

Tipo de acceso: Campo en el cual el usuario marca si el concurso será público o privado en caso de ser privado tiene que añadir los usuarios que desee que participen.

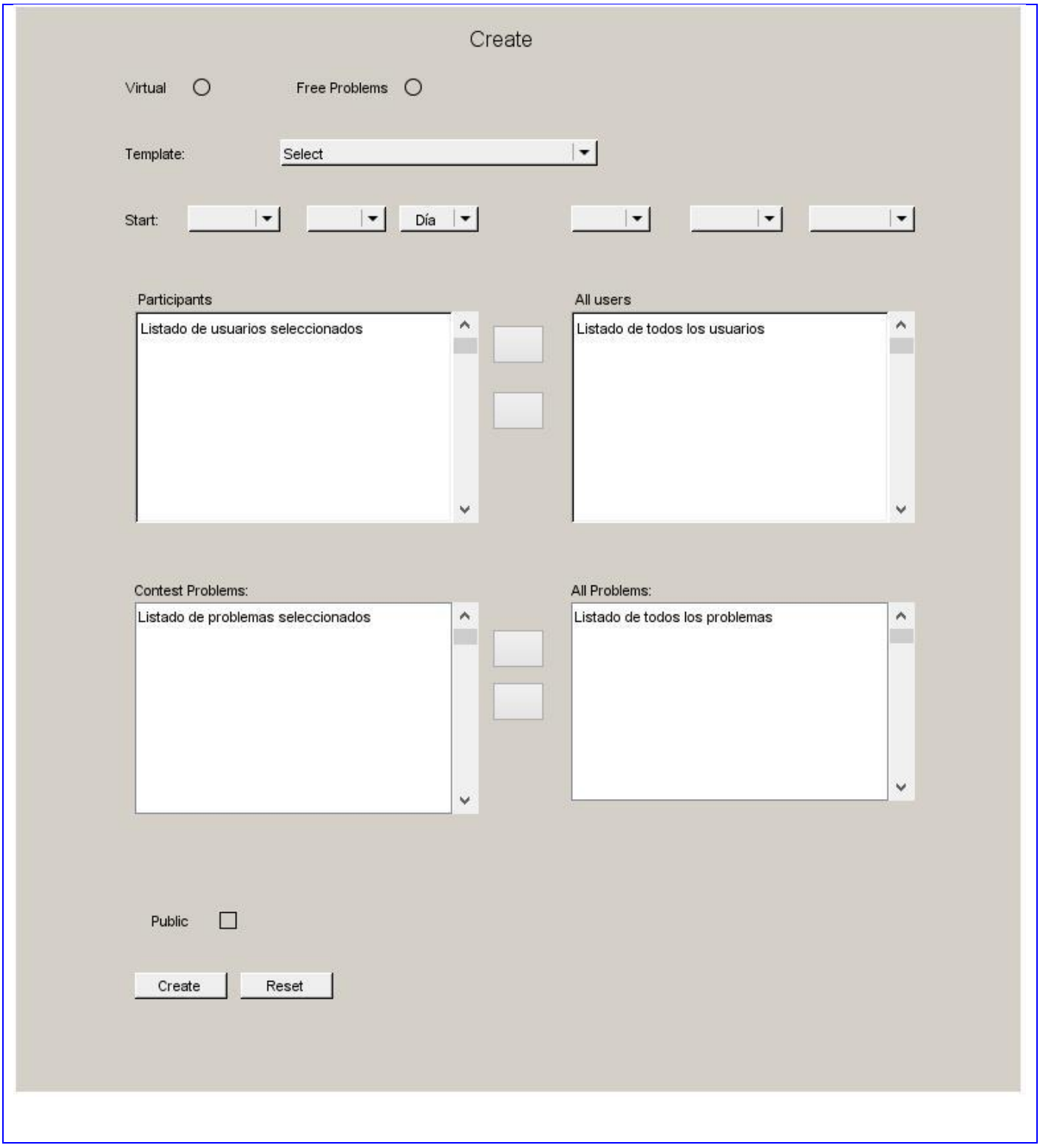
Usuarios: Se escogen los usuarios que desee que participen en caso de ser privado en acceso al concurso.

4- Flujo de la acción a realizar:

- El sistema debe permitir incluir y/o seleccionar los datos para incluir un nuevo concurso de práctica.
- Cuando el usuario incluye y/o selecciona correctamente los datos necesarios para incluir un concurso de práctica y selecciona la opción Crear, se crea un nuevo elemento y se redirige a la página mi lista donde se encuentran todos los concursos creados por ese usuario.
- Si los datos están incompletos o incorrectos se señalarán los campos en cuestión dando la posibilidad al usuario de realizar nuevamente la acción en cuestión.
- Si selecciona la opción Resetear refresca la página con los campos vacíos.

Observaciones:

Prototipo de interfaz:


Create
Virtual
Free Problems
Template:
Select
Start:
Día
Participants
Listado de usuarios seleccionados
All users
Listado de todos los usuarios
Contest Problems:
Listado de problemas seleccionados
All Problems:
Listado de todos los problemas
Public
Create
Reset

2.3.2 Requerimientos no funcionales

Los requerimientos no funcionales son aquellos que no se refieren directamente a las funcionalidades específicas que proporciona el sistema, sino a las propiedades emergentes de éste como la fiabilidad, el tiempo de respuesta y la capacidad de almacenamiento (21).

Usabilidad:

- El sistema debe poseer una interfaz gráfica bien formada y fácil de utilizar por cualquier usuario con conocimientos básicos de informática.

Eficiencia:

- El sistema debe ser capaz de procesar hasta 100 concursos virtuales o de práctica.
- Toda funcionalidad del sistema debe responder al usuario en menos de 5 segundos.

Funcionabilidad y Seguridad:

- El sistema debe tener un registro de eventos en los logs.
- El nuevo sistema debe desarrollarse aplicando patrones que incrementen la seguridad.
- Al sistema debe hacerse copias de seguridad cada 24 horas.

Accesibilidad:

- El sistema debe estar disponible desde cualquier estación de trabajo conectada a la red.

Confiabilidad y Recuperabilidad:

- El sistema debe de iniciar automática los servicios una vez reanudado el sistema ante un fallo de energía en el servidor.
- El sistema debe ser capaz de notificarle al usuario la perdida de conexión ante una interrupción de las comunicaciones de red del cliente.

Portabilidad e Instalabilidad:

- El sistema debe poderse instalar en una PC cliente.

2.4 Modelo de análisis

El lenguaje que se utiliza en el análisis se basa en un modelo de objetos conceptuales, que se llama modelo de análisis. El modelo de análisis ayuda a refinar los requisitos, permite razonar sobre aspectos internos del sistema, incluidos sus recursos compartidos internos. Además, el modelo de análisis ofrece un mayor poder expresivo y una mayor formalización. El modelo de análisis también ayuda a estructurar los requisitos y proporciona una estructura centrada en el mantenimiento, en aspecto tales como la flexibilidad ante los cambios y la reutilización (22).

2.4.1 Diagramas de clases del análisis

Una clase de análisis representa una abstracción de una o varias clases y subsistemas del diseño del sistema (22). Los estereotipos empleados en las clases de análisis Interfaz, Control y Entidad se muestran en la (Ilustración 5).

Ilustración 5 Estereotipos del diagrama de clases del análisis

A continuación se presentan los Diagramas de Clases del Análisis de los requisitos Incluir concurso virtual en la (Ilustración 6), Eliminar concurso en la (Ilustración 7) e Incluir concurso de práctica en la (Ilustración 8).

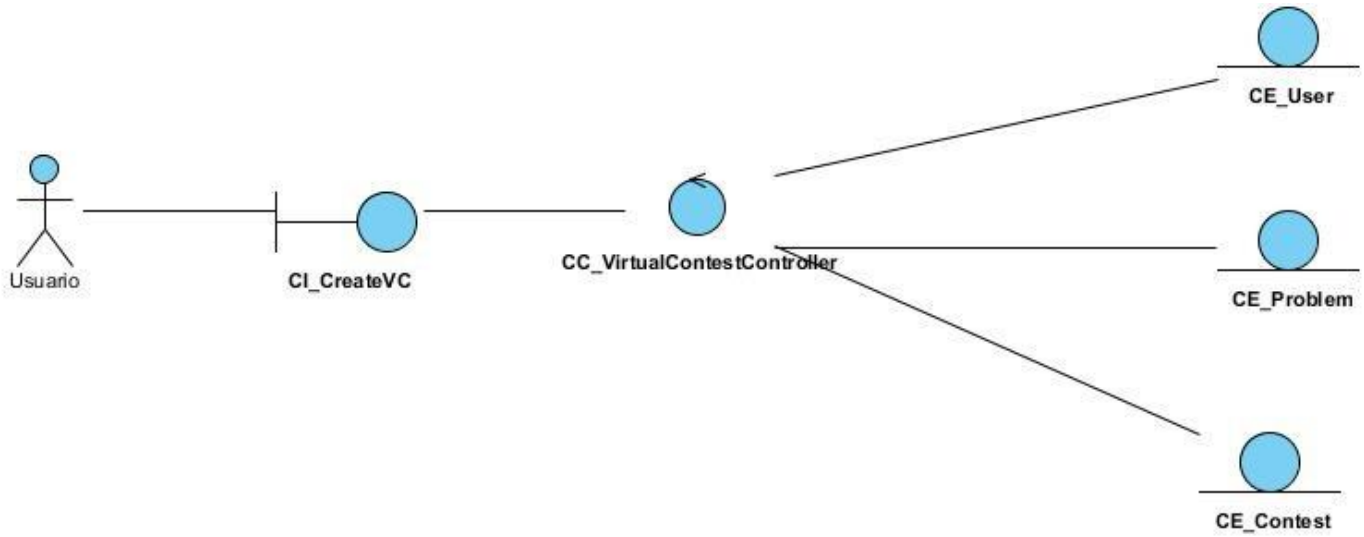


Ilustración 6Diagramade clases del análisis. Incluir concurso virtual

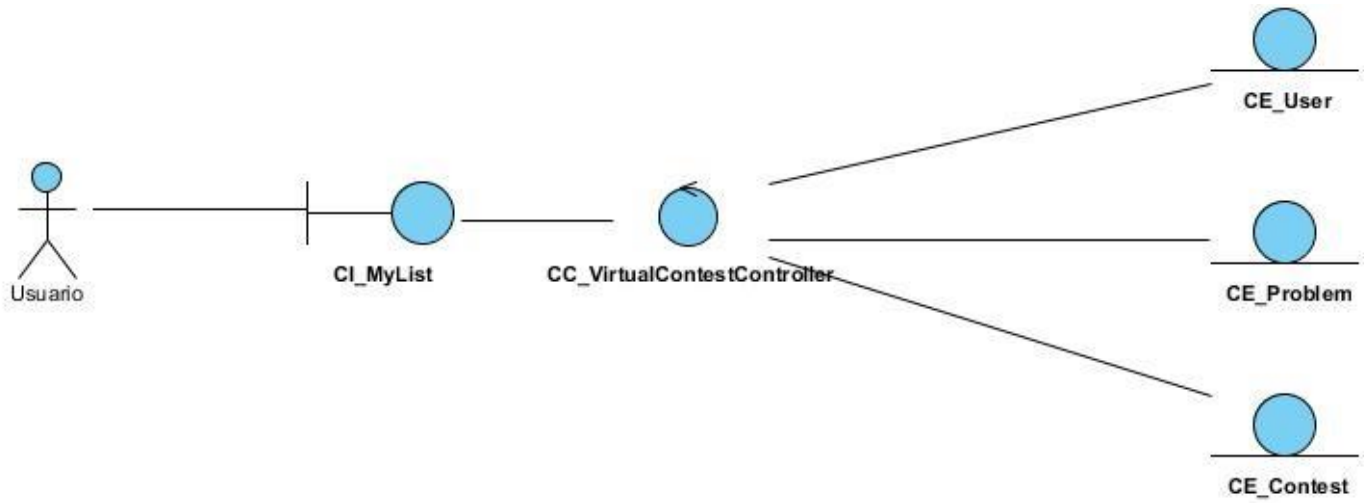


Ilustración 7Diagrama de clases del análisis. Eliminar concurso

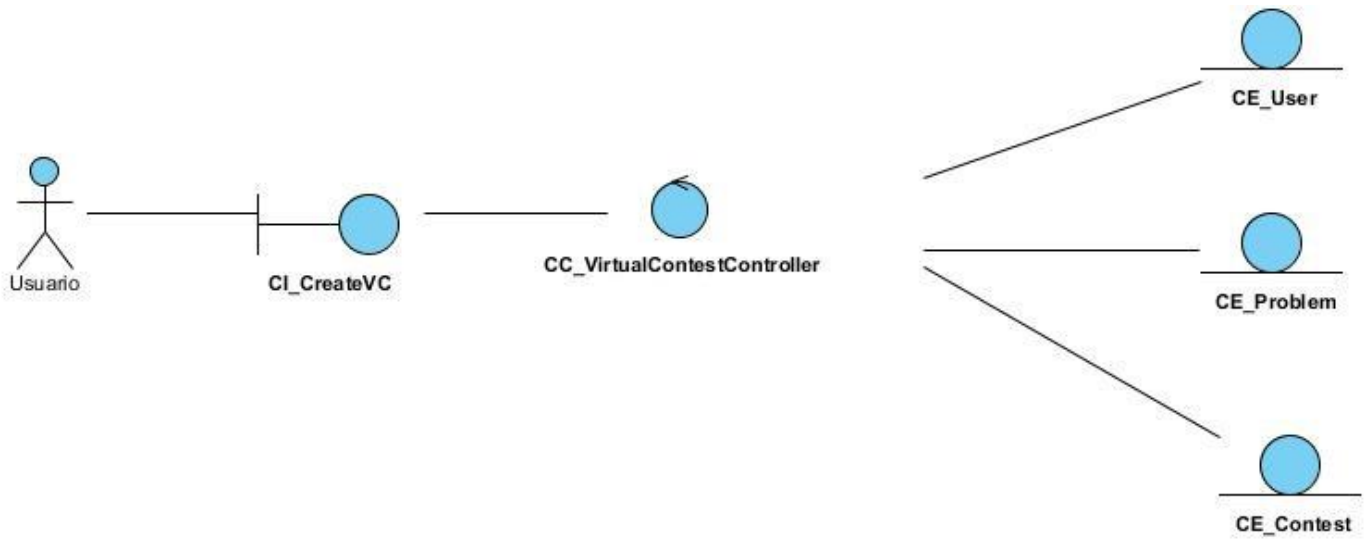


Ilustración 8Diagrama de clases del análisis. Incluir concurso de práctica

2.5 Modelado de Diseño

El diseño de software es un proceso iterativo por medio del cual se traducen los requerimientos en un "plano" para construir el software. Al principio, el plano ilustra una visión holística del software. Es decir, el diseño se representa en un nivel alto de abstracción, en el que se rastrea directamente el objetivo específico del sistema y los requerimientos más detallados de datos, funcionamiento y comportamiento. A medida que tienen lugar las iteraciones del diseño, las mejoras posteriores conducen a niveles menores de abstracción. Esto también puede rastrearse hasta los requerimientos, pero la conexión es más sutil (23)**.**

2.5.1 Diagramas de Clases del Diseño

Un diagrama de clases de diseño (DCD) representa las especificaciones de las especificaciones de las clases e interfaces software en una aplicación. Entre la información general encontramos: clases, asociaciones y atributos, interfaces, con sus operaciones y constantes, métodos, información acerca del tipo de los atributos, navegabilidad, dependencias. A diferencia de las clases conceptuales del Modelo del Dominio, las clases de diseño de los DCD muestran las definiciones de 1as clases software en lugar de los conceptos del mundo real (19). Los estereotipos web utilizados para la creación de los diagramas de clases del diseño son: *Client Page, Server Page, Form*, las relaciones y asociaciones entre las paginas clientes, servidoras y formularios son: *Build, Link, Redirect, Submit, Include* como se muestra en la (Tabla 5).

Tabla 5Leyenda de estereotipos web

Estereotipos	Icono	Descripción
<<Client Page>>		Página cliente es una página web la cual es interpretada por el navegador.
<<Server Page>>		Página servidora es la página que contiene el código la cual se ejecuta del lado del servidor.
<<Form>>		Los formularios forman parte de una página cliente que contenga datos de entrada.
Relaciones		**Descripción**
<<Build>>		Asocia las paginas cliente con las paginas servidora, expresa como una página servidora construye una página cliente.
<<Link>>		Se refiere a un hipervínculo, este siempre se origina de una página cliente y apunta a otra página cliente o una página servidora.
<<Submit>>		Se refiere a la relación entre un formulario y una página servidora, a través de esta asociación el formulario envía los valores de los campos contenidos al servidor para ser procesados.
<<Include>>		Se refiere a la relación de que una página servidora incluya otra página servidora.
<<Redirect>>		Se refiere a la relación de redireccionar el procesamiento de una página a otra.

A continuación se presentan los Diagramas de Clases del Diseño con estereotipos web correspondientes a los requisitos Incluir concurso virtual en la (Ilustración 9), Eliminar concurso en la (Ilustración 10) e Incluir concurso de práctica en la (Ilustración 11).

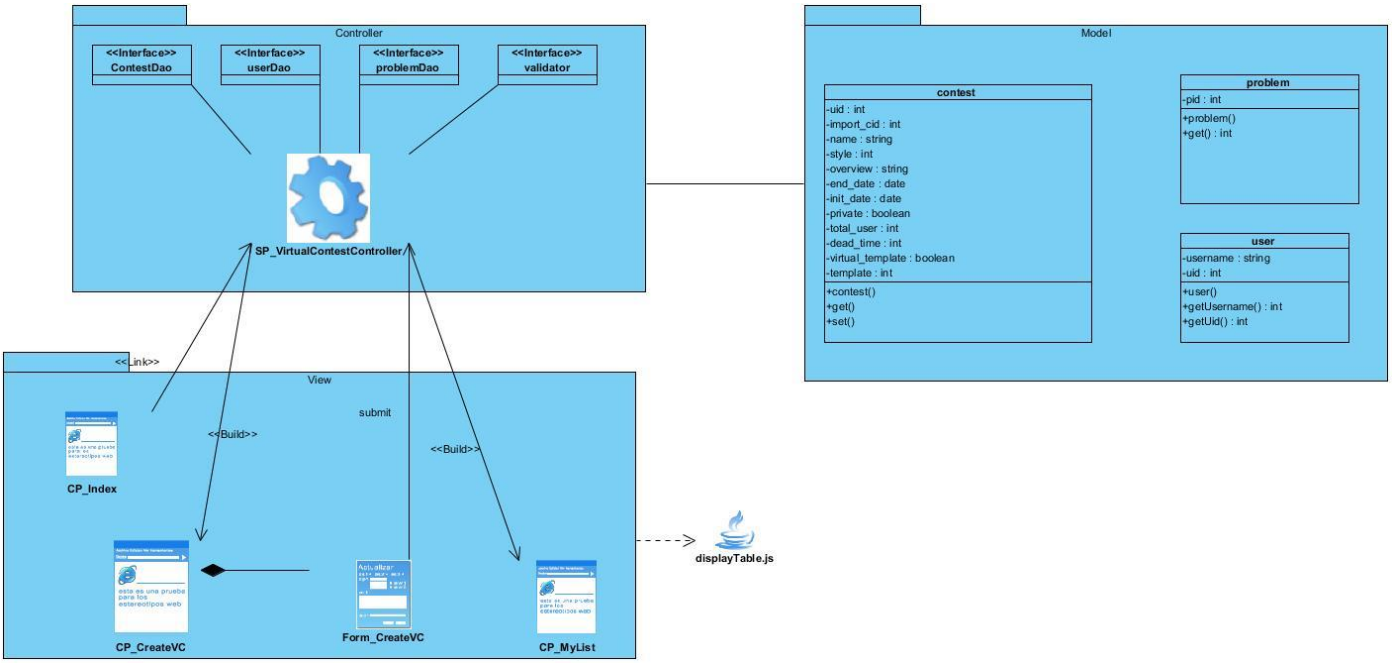


Ilustración 9Diagrama de clases del diseño. Incluir concurso virtual

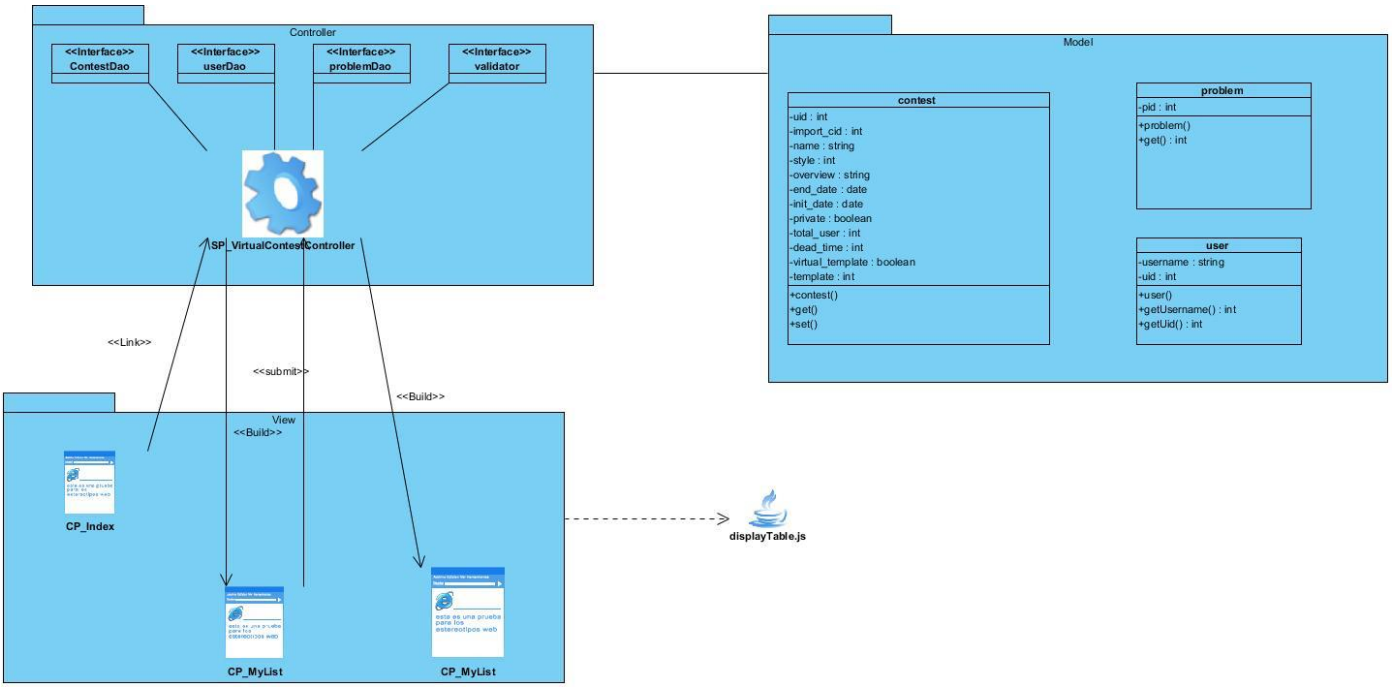


Ilustración 10Diagrama de clases del diseño. Eliminar concurso

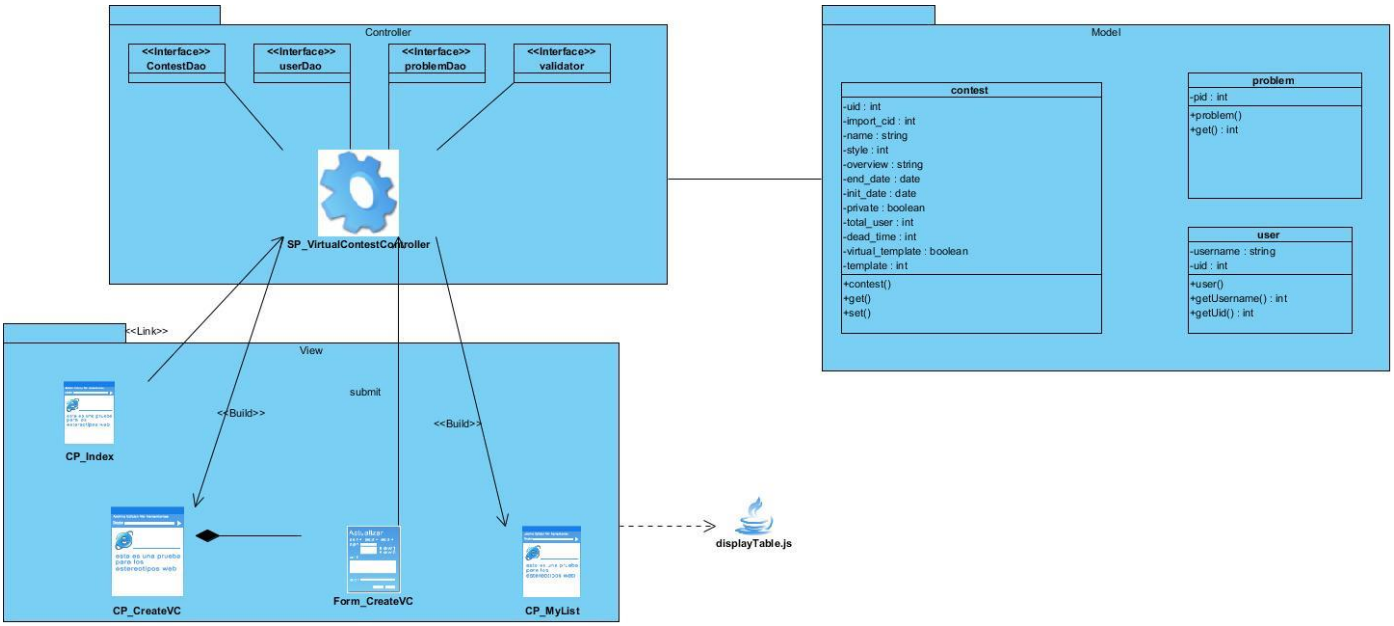


Ilustración 11Diagrama de clases del diseño. Incluir concurso de práctica

2.5.2 Modelo de datos

La mayoría de los sistemas utilizan bases de datos de información de gran tamaño. En algunos casos, esta base de datos es independiente del sistema software. En otros, se crea para el sistema que se está desarrollando. Una parte importante del modelado de sistemas es la definición de la forma lógica de los datos procesados por el sistema. Estos se denominan a menudo modelos semánticos de datos. La técnica de modelado de datos más ampliamente usada es el modelo Entidad-Relación-Atributo, que muestran las entidades de datos, sus atributos asociados y las relaciones entre estas entidades (20). A continuación se presenta el modelo de la base de datos del COJ en la (Ilustración 12) representando las entidades usadas y la relación entre ellas.

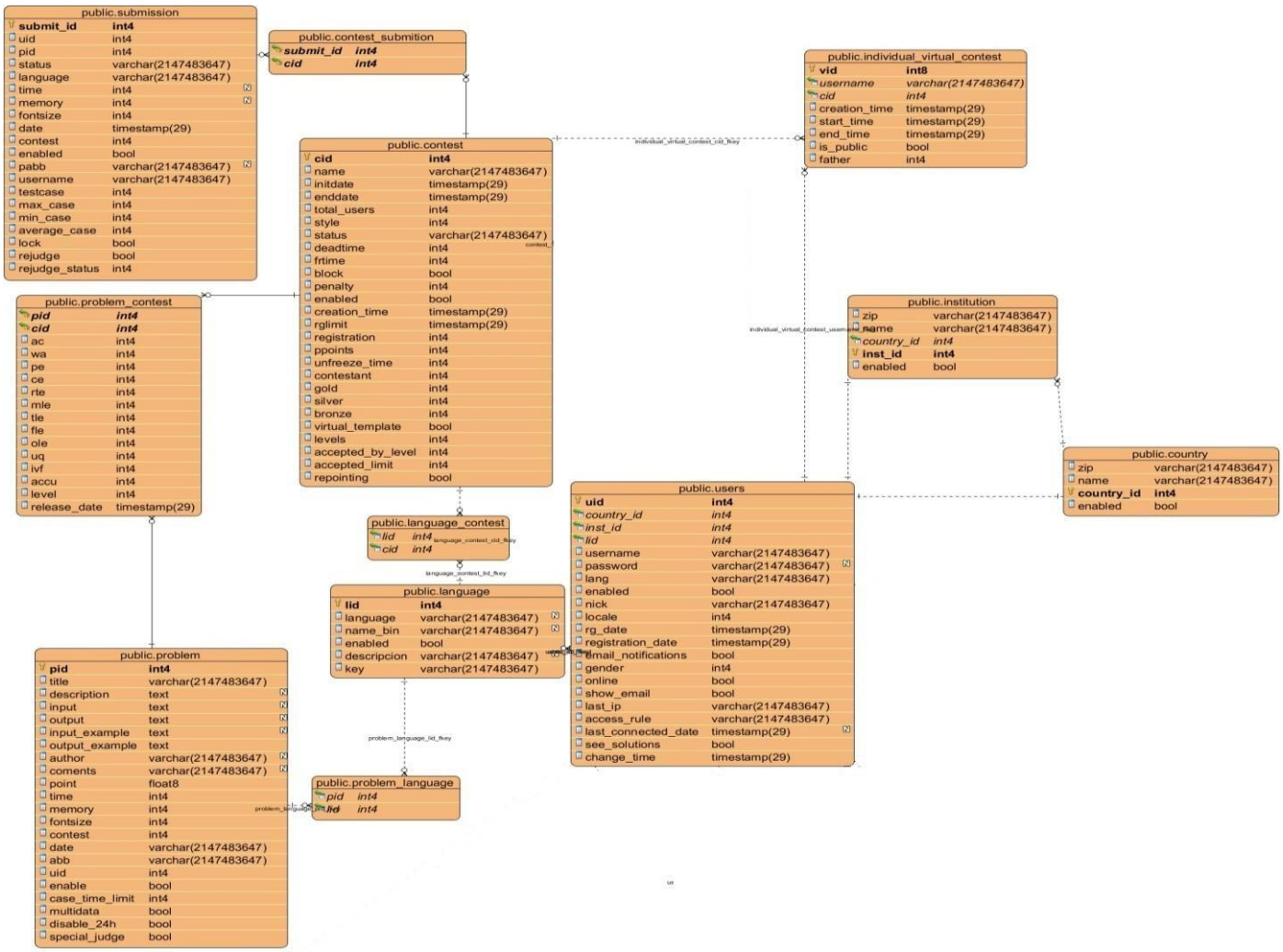


Ilustración 12Modelo de base de datos del COJ

2.5.3 Diagramas de Secuencia

Un diagrama de secuencia muestra las interacciones entre objetos ordenadas en secuencia temporal. Muestra los objetos que se encuentran en el escenario y la secuencia de mensajes intercambiados entre los objetos para llevar a cabo la funcionalidad descrita por el escenario. En aplicaciones grandes además de los objetos se muestran también los componentes y casos de uso (24)

Los estereotipos empleados se muestran en la tabla (Tabla 4). A continuación se presentan los Diagramas de Secuencias de los requisitos Incluir concurso virtual en la (Ilustración 13), Eliminar concurso en la (Ilustración 14) e Incluir concurso de práctica en la (Ilustración 15).

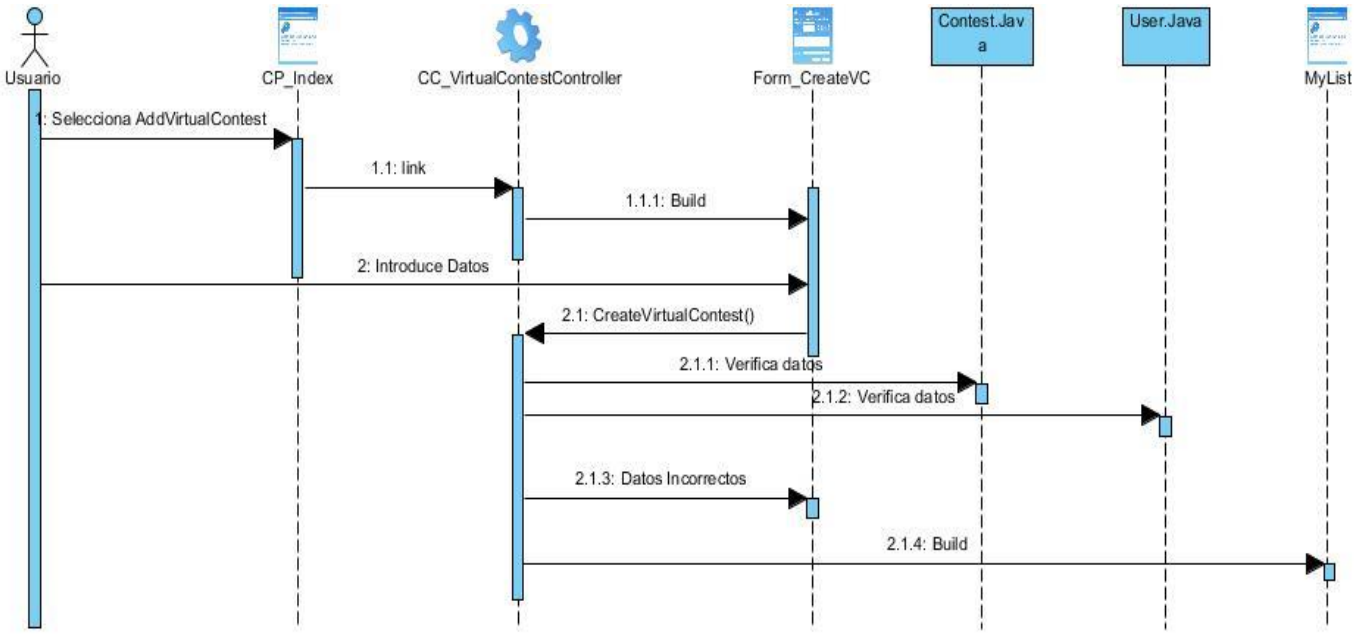


Ilustración 13: Diagrama de secuencia. Incluir concurso virtual

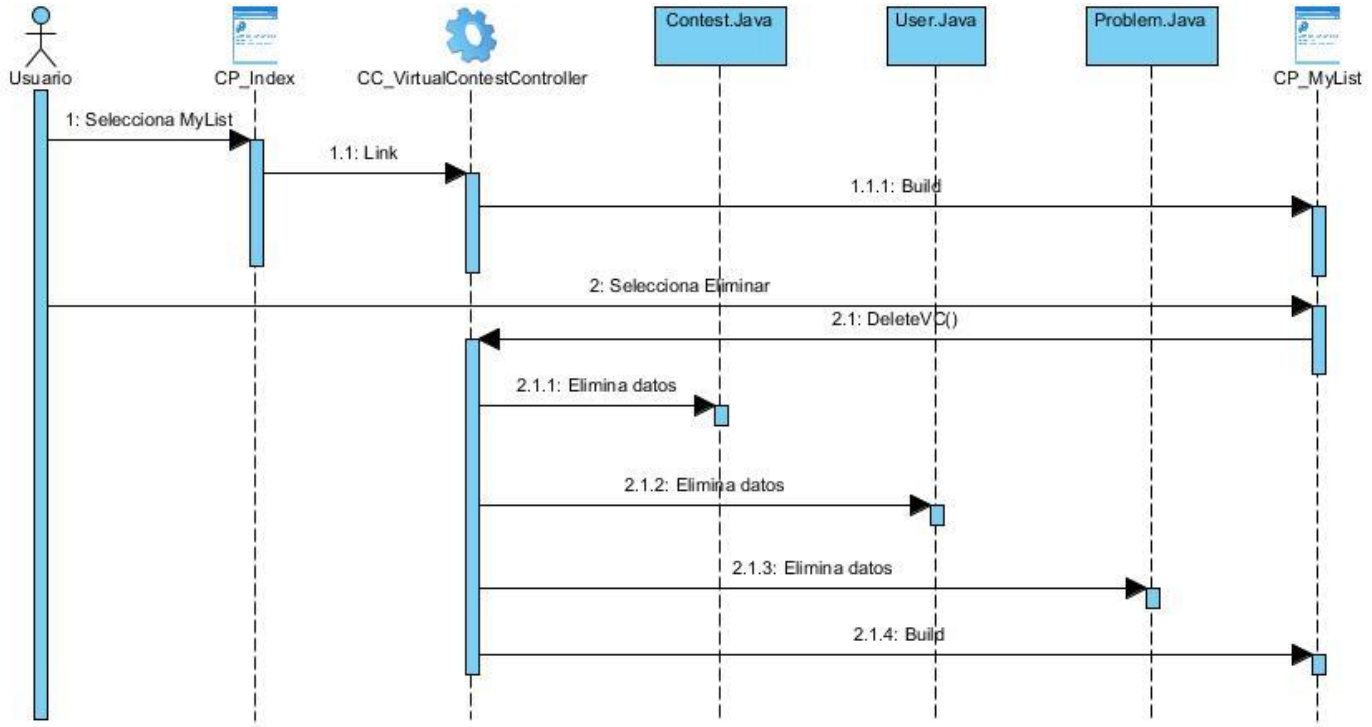


Ilustración 14Diagrama de secuencia. Eliminar concurso

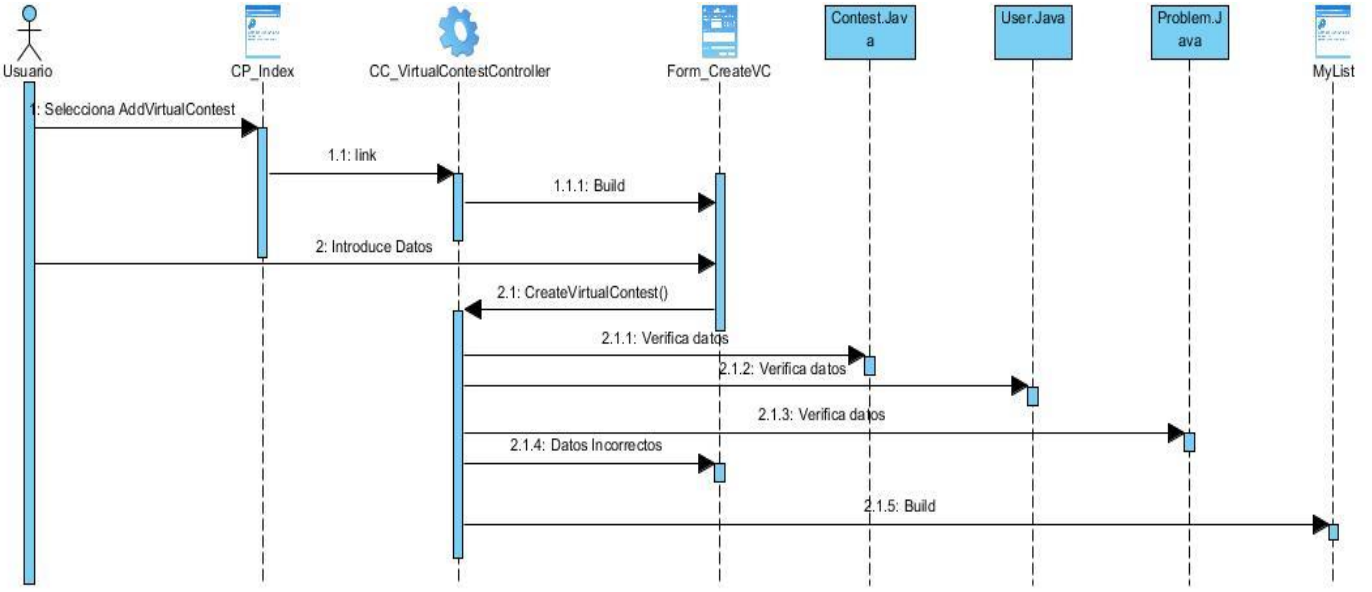


Ilustración 15Diagrama de secuencia. Incluir concurso de práctica

2.6 Arquitectura de la aplicación

La arquitectura de software es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución (25).

El patrón de arquitectura de software empleado para el desarrollo del software es el Modelo Vista Controlador (MVC). Este patrón se utiliza principalmente en aplicaciones que manejan gran cantidad de datos. MVC sugiere la separación del software en 3 capas: Modelo, Vista, Controlador. El modelo es la representación de la información que maneja la aplicación. En si son los datos que puestos en contexto del sistema proveen de información al usuario o a la misma aplicación. La vista es la representación del modelo en forma gráfica disponible para la interacción con el usuario. El controlador es la capa encargada de manejar y responder las solicitudes del usuario, procesando la información necesaria y modificando al Modelo en caso de ser necesario (26).

Es seleccionado esta arquitectura debido a las ventajas que brinda a la estructura del módulo, además de ser la utilizada por el COJ aplicación donde será desplegado el módulo de concurso virtual.

2.7 Patrones de Diseño

Un patrón de diseño es una descripción de clases y objetos comunicándose entre sí adaptada para resolver un problema de diseño general en un contexto particular (26).

En el diseño de la aplicación se hace uso de los Patrones Generales de Asignación de Responsabilidad, por sus siglas en ingles (GRASP), los cuales constituyen un apoyo para entender el diseño de objetos, un patrón es una descripción de un problema y la solución a la que se le da un nombre. Los patrones GRASP se dividen en 6 ellos son: experto, creador, controlador, fachada, alta cohesión y bajo acoplamiento (26).

El patrón **Experto** se basa en que la responsabilidad de realizar una labor es de la clase que tiene o puede tener los datos involucrados (26). Este patrón permite identificar las clases que contienen la información necesaria para que puedan ser manipuladas por otras. Esto se evidencia en las clases entidades tales como: ***User, Problem, Contest, Submission***.

El patrón **Creador** el guía la asignación de responsabilidades relacionadas con la creación de objetos, se asigna la responsabilidad de que una clase cree un objeto de otra clase (26). Este patrón se evidencia en las clases encargadas de registrar datos de entidades en el sistema, tales como: ***UserDAO, ContestDAO,***

ProblemDAO, SubmissionDAO. Por otra parte existen varias clases encargadas de crear las vistas tales como: ***VirtualContestController, VirtualProblemCotrnoller, VirtualSubmissionController***.

El patrón **Controlador** asigna la responsabilidad de recibir o manejar un mensaje de evento del sistema a una clase, este define el método para la operación del sistema (26). Este patrón se evidencia en las clases controladoras encargadas de realizar las operaciones de gestión de datos sobre las entidades, posibilitando el acceso a la información almacenada en la base de datos. Estas clases son: ***VirtualContestController, VirtualProblemController, VirtualSubmissioController***.

El patrón **Bajo Acoplamiento** es un principio que asigna la responsabilidad de controlar el flujo de eventos del sistema a clases específicas. Esto facilita la centralización de actividades. El controlador no realiza estas actividades, las delega en otras clases con las que mantiene un modelo de alta cohesión (26).

2.8 Conclusiones Parciales

En el proceso de análisis y diseño del Módulo de concurso virtual para el COJ se llevó a cabo por su factibilidad la elaboración del modelo de dominio a partir de los conceptos fundamentales del problema, contribuyendo a la identificación de los requisitos funcionales y no funcionales que debe cumplir el sistema. Estos elementos dieron paso a la creación de las historias de usuarios permitiendo a los desarrolladores una mejor comprensión de los requisitos funcionales. Posteriormente se procedió con los flujos de análisis y diseño como diagramas de clases del análisis, diagramas de clases del diseño con el uso de estereotipos web, diagramas de secuencia y modelo de datos, los cuales servirán como guía para la implementación del sistema. Por otra parte quedo definida como arquitectura Modelo Vista Controlador para la creación del Módulo de concursos virtuales debido a que será desplegado en COJ aplicación que contiene dicha arquitectura. Se definieron los patrones de diseño ya que su uso permitió aplicar buenas prácticas en el proceso de modelado e implementación del software. Se puede concluir que los artefactos generados en esta etapa basándose en la metodología AUP versión UCI guiarán el proceso de forma satisfactoria.

CAPÍTULO 3. IMPLEMENTACIÓN Y PRUEBA

En el presente capítulo se procede a la implementación y validación del módulo de concursos virtuales para el COJ. En este se describe la implementación en términos de componentes y como estos están organizados en el modelo de despliegue. Una vez terminada la etapa de implementación, se efectuarán las pruebas al sistema desarrollado para validar los resultados obtenidos en él, realizándole pruebas tanto a la interfaz del módulo como al código.

3.1 Modelo de implementación

La implementación se comienza con el resultado del diseño. En esta etapa se implementa el modelo del diseño en términos de componentes por lo que será necesaria la creación del diagrama de componentes y el diagrama de despliegue especificando los nodos físicos con que interactúa el sistema.

3.1.1 Diagrama de despliegue

La vista de despliegue representa las instancias de componentes de ejecución en instancias de nodos. Un nodo es un recurso de ejecución, tal como una computadora, dispositivo, o memoria. Esta vista permite determinar las consecuencias de la distribución y de la asignación de recursos. La vista de despliegue se representa como un diagrama de despliegue (27). A continuación se presentan el Diagrama de Despliegue en la (Ilustración 16).

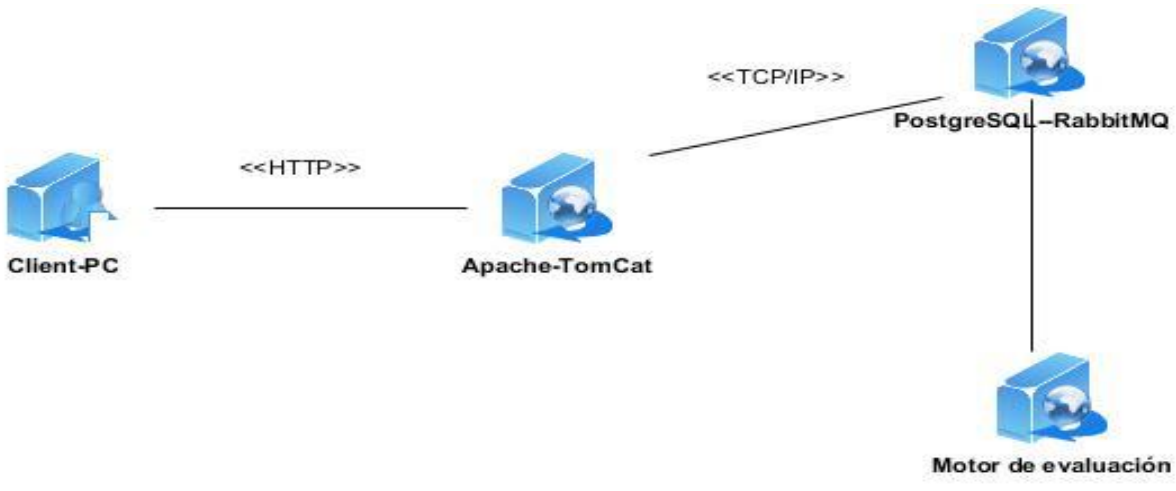


Ilustración 16Diagrama de despliegue

3.1.2 Diagrama de componentes

La vista de implementación modelo los componentes del sistema, a partir de los cuales se construye la aplicación, muestra el empaquetado físico de las partes reutilizables del sistema en unidades sustituibles llamadas componentes. Una vista de implementación muestra la implementación de los elementos del diseño mediante componentes. Los componentes son piezas reutilizables de alto nivel a partir de los cuales se puede construir el sistema. Este modela las dependencias entre los componentes para poder determinar el impacto de un cambio propuesto. También modela la asignación de clases y de otros elementos del modelo a los componentes. La vista de implementación se representa en diagramas de componentes (27). A continuación se presentan el Diagrama de Componentes en la (Ilustración 17).

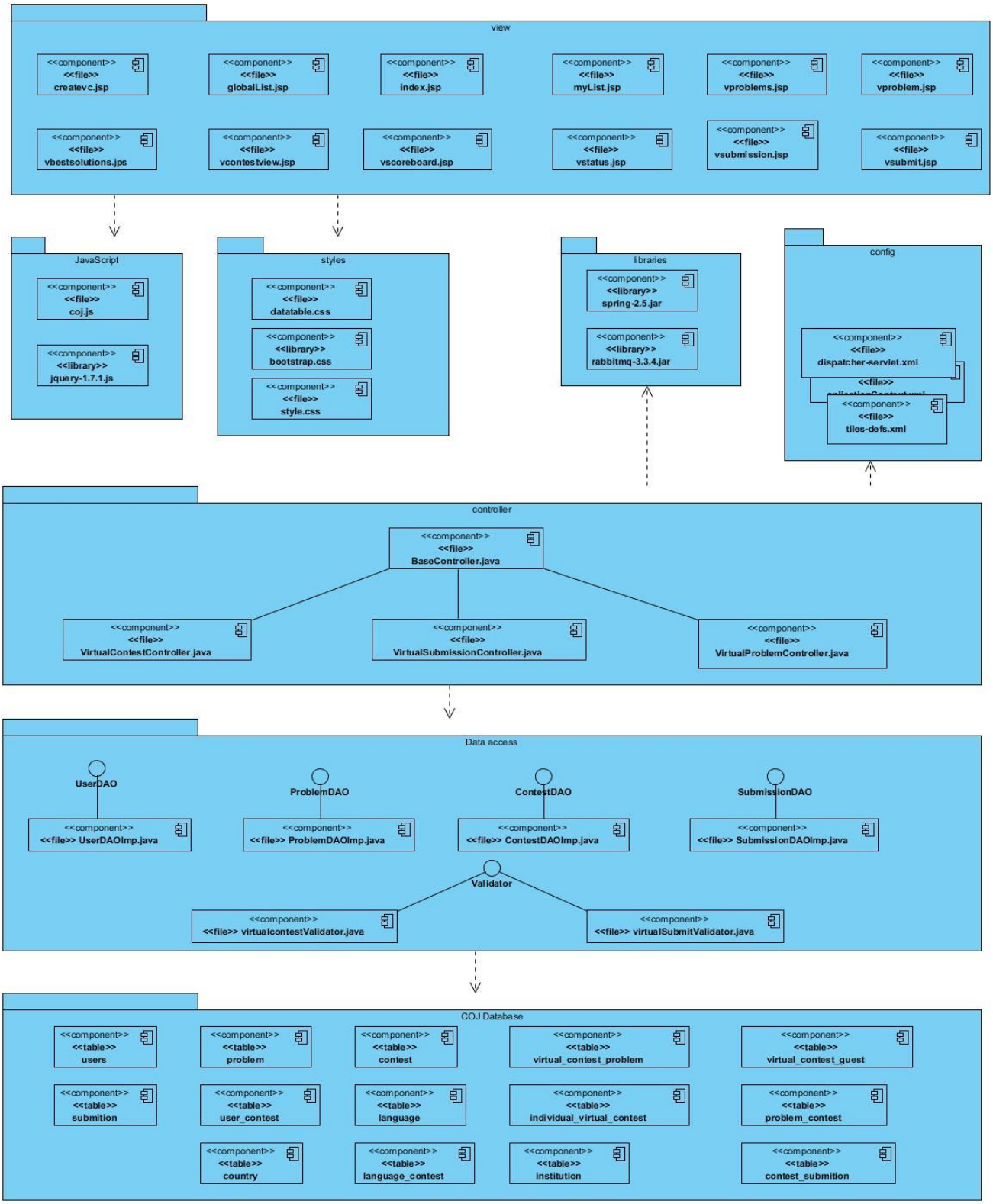

view
<<component>> <<file>> createvc.jsp
<<component>> <<file>> globalList.jsp
<<component>> <<file>> index.jsp
<<component>> <<file>> myList.jsp
<<component>> <<file>> vproblems.jsp
<<component>> <<file>> vproblem.jsp
<<component>> <<file>> vbestsolutions.jps
<<component>> <<file>> vcontestview.jsp
<<component>> <<file>> vscoreboard.jsp
<<component>> <<file>> vstatus.jsp
<<component>> <<file>> vsubmission.jsp
<<component>> <<file>> vsubmit.jsp
JavaScript
<<component>> <<file>> coj.js
<<component>> <<library>> jquery-1.7.1.js
styles
<<component>> <<file>> datatable.css
<<component>> <<library>> bootstrap.css
<<component>> <<file>> style.css
libraries
<<component>> <<library>> spring-2.5.jar
<<component>> <<library>> rabbitmq-3.3.4.jar
config
<<component>> <<file>> dispatcher-servlet.xml
<<file>>
<<component>> <<file>> tiles-defs.xml
controller
<<component>> <<file>> BaseController.java
<<component>> <<file>> VirtualContestController.java
<<component>> <<file>> VirtualSubmissionController.java
<<component>> <<file>> VirtualProblemController.java
Data access
UserDAO
ProblemDAO
ContestDAO
SubmissionDAO
<<component>> <<file>> UserDAOImp.java
<<component>> <<file>> ProblemDAOImp.java
<<component>> <<file>> ContestDAOImp.java
<<component>> <<file>> SubmissionDAOImp.java
Validator
<<component>> <<file>> virtualcontestValidator.java
<<component>> <<file>> virtualSubmitValidator.java
COJ Database
<<component>> <<table>> users
<<component>> <<table>> problem
<<component>> <<table>> contest
<<component>> <<table>> virtual_contest_problem
<<component>> <<table>> virtual_contest_guest
<<component>> <<table>> submition
<<component>> <<table>> user_contest
<<component>> <<table>> language
<<component>> <<table>> individual_virtual_contest
<<component>> <<table>> problem_contest
<<component>> <<table>> country
<<component>> <<table>> language_contest
<<component>> <<table>> institution
<<component>> <<table>> contest_submition

Ilustración 17Diagrama de componentes

3.2 Pruebas de Software

Las pruebas de software (*testing* en inglés) son los procesos que permiten verificar y revelar la calidad de un producto software antes de su puesta en marcha. Básicamente, es una fase en el desarrollo de software que consiste en probar las aplicaciones construidas (28).

Con las pruebas de software se buscan defectos en el sistema permitiendo encontrar comportamientos incorrectos en el mismo y verificar a su vez que este cumple con los requerimientos del cliente.

3.2.1 Tipos de Pruebas

La disciplina de pruebas es una de las más costosas del ciclo de vida software. En sentido estricto, deben realizarse las pruebas de todos los artefactos generados durante la construcción de un producto, lo que incluye especificaciones de requisitos, casos de uso, diagramas de diversos tipos y, por supuesto, el código fuente y el resto de productos que forman parte de la aplicación e infraestructura (28).

Prueba de caja negra: En este tipo de prueba, tan sólo, podemos probar dando distintos valores a las entradas. Los datos de prueba se escogerán atendiendo a las especificaciones del problema, sin importar los detalles internos del programa, a fin de verificar que el programa corra bien. Este tipo de prueba se centra en los requisitos funcionales del software y permite obtener entradas que prueben todos los flujos de una funcionalidad (28).

Prueba de caja blanca: Consiste en realizar pruebas para verificar que líneas específicas de código funcionan tal como está definido. También se le conoce como prueba de caja-transparente. La prueba de la caja blanca es un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para derivar los casos de prueba (28).

Pruebas unitarias: Estas pruebas las ejecuta el desarrollador, cada vez que va probando fragmentos de código o *scripts* para ver si todo funciona como se desea. Estas pruebas son muy técnicas. Por ejemplo, probar una consulta, probar que un fragmento de código envíe a imprimir un documento, probar que una función devuelva un *flag*. El objetivo de las pruebas unitarias es aislar cada parte del programa y mostrar que las partes individuales son correctas (28).

Pruebas de interacción: Consiste en construir el sistema a partir de los distintos componentes y probarlo con todos integrados. Estas pruebas deben realizarse progresivamente. El foco de atención es el diseño y la construcción de la arquitectura de software (28).

Pruebas del sistema: Se le aplican para probar cualquier pieza de software completo, desarrollado o adquirido, ya sea para decidir acerca de su aceptación, para analizar defectos globales o para estudiar aspectos específicos de su comportamiento, tales como seguridad o rendimiento (29).

Al sistema se le realizaran pruebas al código y las interfaces de usuario, para llevar a cabo dichas pruebas se aplicará el método de caja negra a las interfaces haciendo uso de los casos de pruebas presentados a continuación en la (Tabla 6), (Tabla 7) y (Tabla 8).

Tabla 6Caso de prueba Incluir concurso virtual

CP Incluir concurso virtual									
Descripción general • Permitir incluir nuevos concursos virtuales en el sistema. **Condiciones de ejecución** Para incluir un concurso virtual hay que: • Estar autenticado en el sistema con el rol usuario o administrador.									
Escenario	**Descripción**	**plantilla**	**fecha**	**hora de inicio**	**tipo de acceso**	**tipo de concurso**	**Usuarios**	**Respuesta del sistema**	**Flujo central**
EC 1.1 Opción Incluir	Selecciona la opción							Brinda la posibilidad de introducir o seleccionar de manera obligatoria	Administración/Incluir Concurso Virtual

Concurso Virtual.	de incluir un nuevo Concurso virtual.							los siguientes datos del concurso: • Plantilla • Fecha • Hora de inicio • Tipo de acceso • Tipo de concurso Y de forma opcional: • Usuarios. Permite: • Guardar los datos. • Cancelar la operación en cualquier momento.	
EC 1.2 Opción de Crear.	Introduce y/o selecciona los datos del concurso y selecciona la opción crear.	V	V	V	V	V	N/A	Valida los datos. Crea un concurso virtual. Muestra el listado de los concursos virtuales y de práctica creados.	Administración/Incluir Concurso Virtual/Crear
EC 1.3 Opción de cancelar.	Selecciona la opción de Resetear.							Elimina los datos creados.	Administración/Incluir Concurso Virtual/Resetear
EC 1.4 Datos incompletos	Existen datos incompletos.	I	V	V	V	V	N/A	Muestra un mensaje de información. Muestra un indicador sobre los campos vacíos. Regresa al EC 1.1.	Administración/Incluir Concurso Virtual/Crear
		V	I	V	V	V	N/A		
		V	V	I	V	V	N/A		
		V	V	V	I	V	N/A		
		V	V	V	V	I	N/A		

		V	V	V	V	V	N/A		
EC 1.5 Datos incorrectos	Existen datos incorrectos.	I	V	V	V	V	N/A	Muestra un mensaje de información. Muestra un indicador sobre los campos incorrectos. Regresa al EC 1.1.	Administración/Incluir Concurso Virtual/Crear
		V	I	V	V	V	N/A		
		V	V	I	V	V	N/A		
		V	V	V	I	V	N/A		
		V	V	V	V	I	N/A		
		V	V	V	V	V	N/A		

Tabla 7Caso de prueba Eliminar concurso

CP Eliminar concurso			
Descripción general Permitir eliminar un concurso en el sistema. **Condiciones de ejecución** Para eliminar una institución hay que: • Estar autenticado en el sistema con el rol usuario o administrador. • Debe existir en el sistema al menos un concurso. • EL concurso solo puede ser eliminado por el usuario creador o un administrador.			
Escenario	**Descripción**	**Respuesta del sistema**	**Flujo central**
EC 1.1 Opción de listar mis concursos virtuales o de práctica	Selecciona la opción mi lista.	Muestra la página del listado de concursos creados el usuario donde se puede ver los siguientes datos: • Id • Nombre • Tipo de acceso • Estado • Fecha y Hora • Duración.	Administración/Mi lista
EC 1.2 Opción de eliminar	El usuario selecciona la opción de Eliminar	Muestra la página de mi lista con los restantes concursos.	Administración/Mi lista/Eliminar

EC 1.3 Opción Aceptar	Selecciona la opción Aceptar.	Elimina el concurso. Regresa al listado de concursos actualizado y muestra un mensaje de información.	Administración/Mi lista/Eliminar/Aceptar
EC 1.4 Opción de cancelar.	Selecciona la opción de Cancelar.	Regresa a la vista anterior.	Administración/Mi lista/Eliminar/Cancelar

Tabla 8Caso de prueba Incluir concurso de práctica

CP Incluir concurso de práctica										
Descripción general Permitir incluir nuevos concursos de práctica en el sistema. **Condiciones de ejecución** Para incluir un concurso de práctica hay que: • Estar autenticado en el sistema con el rol usuario o administrador.										
Escenario	**Descripción**	**plantilla**	**fecha**	**hora de inicio**	**tipo de acceso**	**tipo de concurso**	**Problemas**	**Usuarios**	**Respuesta del sistema**	**Flujo central**
EC 1.1 Opción Incluir Concurso Virtual.	Selecciona la opción de incluir un nuevo Concurso virtual.								Brinda la posibilidad de introducir o seleccionar de manera obligatoria los siguientes datos del concurso: • Plantilla • Fecha • Hora de inicio • Tipo de acceso • Tipo de concurso • Problemas Y de forma opcional: • Usuarios.	Administración/Incluir Concurso Virtual

									Permite: • Guardar los datos. • Cancelar la operación en cualquier momento.	
EC 1.2 Opción de Crear.	Introduce y/o selecciona los datos del concurso y selecciona la opción crear.	V	V	V	V	V	V	N/A	Valida los datos. Crea un concurso de práctica. Muestra el listado de los concursos virtuales y de práctica creados.	Administración/Incluir Concurso Virtual/Crear
EC 1.3 Opción de resetear.	Selecciona la opción de Resetear.								Elimina los datos creados.	Administración/Incluir Concurso Virtual/Resetear
EC 1.4 Datos incompletos	Existen datos incompletos.	I	V	V	V	V	V	N/A	Muestra un indicador sobre los campos vacíos. Regresa al EC 1.1.	Administración/Incluir Concurso Virtual/Crear
		V	I	V	V	V	V	N/A		
		V	V	I	V	V	V	N/A		
		V	V	V	I	V	V	N/A		
		V	V	V	V	I	V	N/A		
		V	V	V	V	V	I	N/A		
		V	V	V	V	V	V	N/A		
EC 1.5 Datos incorrectos	Existen datos incorrectos.	I	V	V	V	V	V	N/A	Muestra un indicador sobre los campos incorrectos. Regresa al EC 1.1.	Administración/Incluir Concurso Virtual/Crear
		V	I	V	V	V	V	N/A		
		V	V	I	V	V	V	N/A		
		V	V	V	I	V	V	N/A		
		V	V	V	V	I	V	N/A		
		V	V	V	V	V	I			
		V	V	V	V	V	V	N/A		

3.2.2 Resultados de las pruebas de caja negra

Para la realización de las pruebas se aplicó el método de caja negra sobre las interfaces del sistema permitiendo la ejecución de las mismas verificar el cumplimiento de los requisitos funcionales y no funcionales del software.

A continuación se presentan los resultados de las pruebas realizadas:

Tabla 9Resultados de los casos de pruebas

Iteraciones	Cantidad de casos de pruebas	No conformidades detectadas			
		Alta	Media	Baja	Total
1	18	2	3	10	15
2	18	1	2	7	10
3	18	1	2	4	7

La tabla anterior muestra los resultados de los casos de pruebas los cuales fueron ejecutados en tres iteraciones. En cada iteración se le da solución a las no conformidades encontradas mejorando la calidad del software progresivamente.

3.2.3 Resultados de las pruebas unitarias

Para la realización de las pruebas al código se aplicó el método de pruebas unitarias utilizando la herramienta *JUnit* versión 4 permitiendo la ejecución de las mismas verificar el correcto funcionamiento de las clases controladoras. A continuación se muestran ejemplos de los resultados de las pruebas unitarias:

El siguiente método es sobre la clase controladora virtualcontestcontroller con el objetivo de cargar de forma satisfactoria la vista esperada. El resultado se muestra en la (Ilustración 18) muestra que la vista se cargó de forma satisfactoria.

```
@Test
    public void testContestView() throws Exception {
    r=mockMvc.perform(get("/practice/vcontestview.xhtml")
        .param("cid","6025")
        .principal(principal)).andDo(print())
        .andExpect(status().isOk())
        .andExpect(view().name("/practice/vcontestview"))
```

```
        .andReturn();

        Assert.assertEquals("se mostró la vista correctamente
","/practice/vcontestview.xhtml?cid=6025",r.getResponse().getRedirectedUrl());

}
```

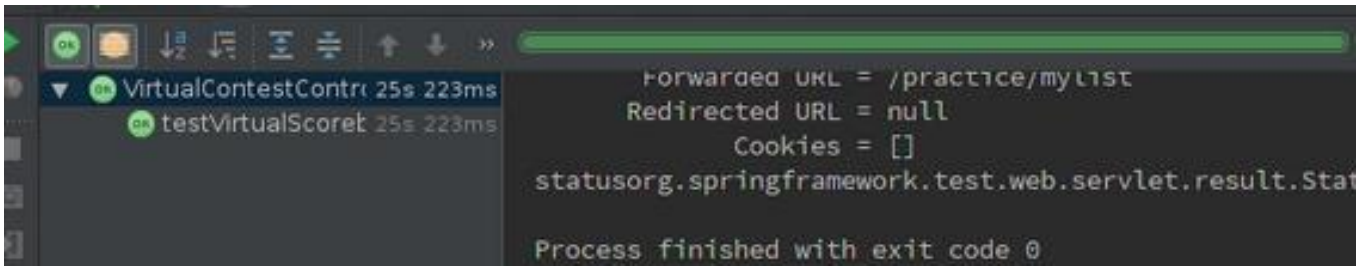


Ilustración 18Resultado del método de prueba a la clase controladora virtualcontestcontroller

El siguiente método se le aplica a la controladora virtualproblemcontroller con el objetivo de cargar de forma satisfactoria la tabla de problemas. El resultado se muestra en la (Ilustración 19) muestra que la vista se cargó de forma satisfactoria.

```
@Test

        public void testVproblemstable() throws Exception {

                when(problemDAOmock.countProblemContest(cid)).thenReturn(6);

                when(problemDAOmock.getContestProblems(6, "en", "admin", contest1, new
                PagingOptions())).thenReturn(new IPaginatedList<Problem>() {

                @Override

                public int[] getNumbers() {...});

                r=mockMvc.perform(get("/practice/vproblemas.xhtml")

                        .param("cid", "6025")

                        .principal(principal)
```

```
        ).andDo(print())
                .andExpect(status().isOk())
                .andExpect(view().name("/practice/vproblemas"))
    .andReturn();
        Assert.assertEquals("se mostró la vista correctamente
    ","/practice/vproblemas",r.getResponse().getForwardedUrl());
}
```

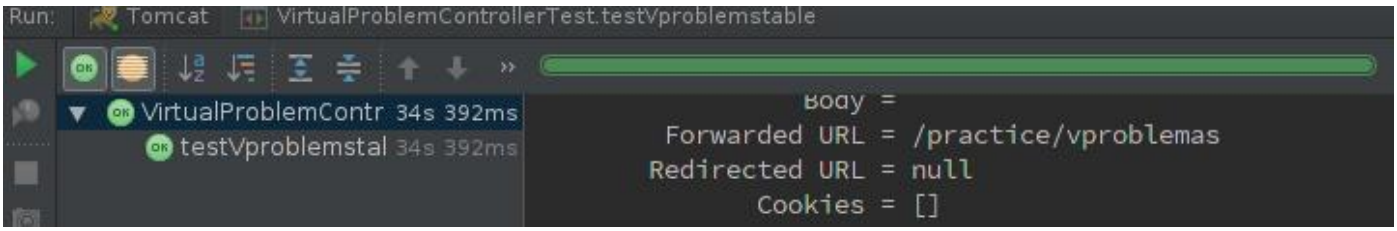


Ilustración 19Resultado del método de prueba a la controladora virtualproblemcontroller

El siguiente método se aplica a la controladora virtualsubmissioncontroller con el objetivo de cargar de forma satisfactoria los envíos a soluciones realizados. El resultado en la (Ilustración 20) muestra que la vista se cargó de forma satisfactoria.

```
@Test
    public void testVstatus() throws Exception {
        when(submissionDAOmock.string("select language from language where key=?",
    "Jaba")).thenReturn("asd");
        ArrayList<Language> ret = new ArrayList<>();
        ret.add(new Language("sasas", 1256321));
        ret.add(new Language("fgdsdf", 5432));
when(contestDAOmock.getContestLanguages(contest1.getCid())).thenReturn(ret);
```

```
        r=mockMvc.perform(get("/practice/vstatus.xhtml")
                .principal(principal)
                .param("cid", "6025")
        ).andDo(print())
                .andExpect(status().isOk())
                .andExpect(view().name("/practice/vstatus"))
        .andReturn();
                Assert.assertEquals("se mostró la vista correctamente
        ","/practice/vstatus",r.getResponse().getForwardedUrl());
}
```

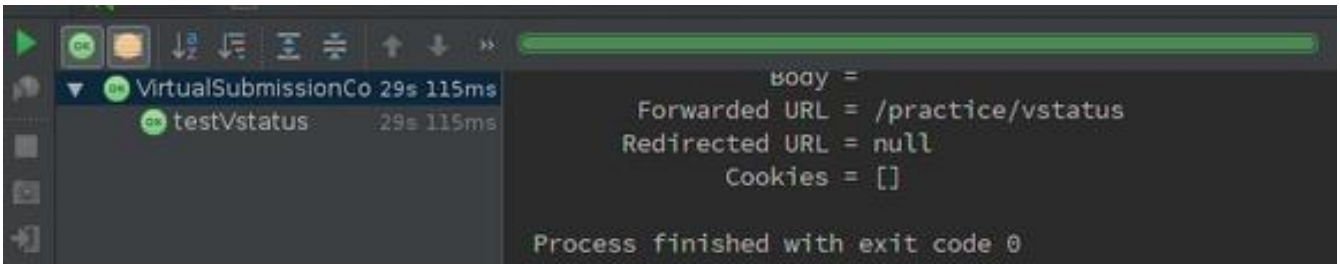


Ilustración 20Resultado al método de prueba a la controladora virtualsubmissioncontroller

3.3 Conclusiones parciales

En este capítulo se realizó el proceso de implementación el cual estuvo guiado por la elaboración del diagrama de componentes permitiendo lograr una estructura de alto nivel del modelo de implementación y organizar los subsistemas de implementación en capas y paquetes representando la vista de cómo se debe implementar el sistema. Para la validación del software se realizaron pruebas mediante la técnica de caja negra basándose en los diseños de casos de pruebas las cuales permitieron detectar una serie de no conformidades en cada iteración. Las pruebas al código fueron realizadas con el uso de la herramienta JUnit versión 4, permitiendo detectar errores al cargar la lista de problemas, listar los problemas, entre otros. Estas pruebas posibilitaron mejorar de forma progresiva la calidad del sistema.

CONCLUSIONES

El presente trabajo se centró en desarrollar un módulo de gestión de concursos virtuales para el Juez en Línea Caribeño. Durante el desarrollo del trabajo se le dio cumplimiento al objetivo propuesto y se arribó a las siguientes conclusiones:

- El estudio de los jueces en línea que contienen la creación concursos virtuales permitió extraer las características comunes entre ellos que fueran de importancia para la creación del módulo.
- La construcción de un módulo de concursos virtuales brinda la posibilidad a los usuarios de recrear competencias pasadas y gestionar tanto concursos virtuales como concursos de práctica.
- Las pruebas realizadas al sistema mediante la técnica de caja negra, permitieron detectar un total de 32 no conformidades en tres iteraciones que fueron corregidas al terminar cada iteración mejorando la calidad del sistema y dando cumplimiento a los requisitos funcionales y no funcionales.
- Al culminar la fase de implementación y pruebas el módulo de concursos virtuales estará disponible entre las herramientas administrativas del Juez en Línea Caribeño en http://coj.uci.cu.

RECOMENDACIONES

Teniendo en cuenta los resultados del presente trabajo, se proponen las siguientes recomendaciones:

- Incorporarle nuevas funcionalidades como un módulo estadístico, que incluyan gráficas y muestren estadísticas relacionadas a los lenguajes de programación más usados en el concurso dividido por usuarios, equipos e instituciones.
- Incorporar un servicio de recomendación de mejores concursos a realizar.

REFERENCIAS BIBLIOGRÁFICAS

1. REVILLA, RUJIA LIU and MIGUEL A. *Competitive Learning in Informatics: The UVa Online Judge Experience.* s.l. : SHAHRIAR MANZOOR, 2008.

2. Halim, Steven. UVa Virtual Contest Generator. [En línea] 2000. [Citado el: 15 de 1 de 2015.] http://uhunt.felix-halim.net/vcontest.

3. Definición.org. Definición.org. [En línea] [Citado el: 9 de 2 de 2016.] http://www.definicion.org/lenguaje.

4. Fernández, Oscar Belmonte. *Introducción al lenguaje de programación Java.* 2004.

5. J.Gutiérrez, Javier.*¿Qué es un framework web?* 2014.

6. Fernández, Norberto.*Introducción a Spring.*

7. Alegsa, Leandro. Alegsa.com.ar. [En línea] 12 de 5 de 2012. [Citado el: 29 de 1 de 2016.] http://www.alegsa.com.ar/Dic/servidor%20de%20base%20de%20datos.php.

8. PostgreSQL-es. PostgreSQL-es. [En línea] [Citado el: 29 de 1 de 2016.] http://www.postgresql.org.es/sobre_postgresql#intro.

9. —. **PostgreSQL-es.** [En línea] [Citado el: 29 de 1 de 2016.] http://www.postgresql.org.es/sobre_postgresql#caracteristicas.

10. Geovanny, Mendoza González. *Herramienta de Desarrollo Netbeans.*

11. Asenjo, Jorge Sanchez. *Servidores de Aplicaciones Web.*

12. Foundation, the Apache Software. apache tomcat. [En línea] [Citado el: 29 de 1 de 2016.] http://tomcat.apache.org/.

13. cyta.com.ar. [En línea] [Citado el: 16 de 2 de 2016.] http://www.cyta.com.ar/biblioteca/bddoc/bdlibros/proyectoinformatico/libro/.

14. Visual paradigm. [En línea] [Citado el: 16 de 2 de 2016.] http://www.visual-paradigm.com/.

15. IBM. IBM. [En línea] [Citado el: 16 de 2 de 2016.] http://www-03.ibm.com/software/products/es/enterprise.

16. INTECO, Laboratorio Nacional de Calidad del Software de. *INGENIERÍA DEL SOFTWARE:METODOLOGÍAS Y CICLOS DE VIDA.* 2009.

17. Mairelys Boeras Velázquez, Laritza Cabrera Barroso, Eileén Llano Castro, Ana María Sanchez Gonzalez, Yaima Oval Riveron, Eylin Hernández Luque. *Aplicando el método de Boehm y Turner.*

18. Sánchez, Tamara Rodréguez. *Metodología de desarrollo para la Actividad productiva de la UCI.*

19. Larman, Craig. *UML y Patrones.*

20. Sommerville, Ian. *Ingeniería de Software 7ma edición.* 2005.

21. Sommerville. *Sommerville_Parte_||_Requerimientos.*

22. Ivar Jacobson, Grady Booch, james Rumbaugh. *El Proceso Unificado de Desarrollo de Software .*

23. S.Pressman, Roger. *Ingeniería del software. Un enfoque práctico.*

24. Universidad de Carlos |||, Madrid. *Diagramas de secuencia.*

25. IEEE. *Arquitectura.*

26. *Capítulo 2.* *Arquitectura de Software.*

27. James Rumbaugh, Ivan Jacobson, Grady Booch. *El Lenguaje Unificado de Modelado. Manual de Referencia.*

28. CIBERTEC. *Pruebas de Software.*

29. Peña, Juan Manuel Fernández. *Capítulo 6 Pruebas de sistema.*

30. Sánchez, Tamara Rodríguez. *Metodología de desarrollo para la Actividad productiva de la UCI.* 2014.

31. Larman, Craig. *UML y Patrones. 2da Edición.*

32. Gamma. *Patrones de diseño.*