Universidad de las Ciencias Informáticas

Facultad 2

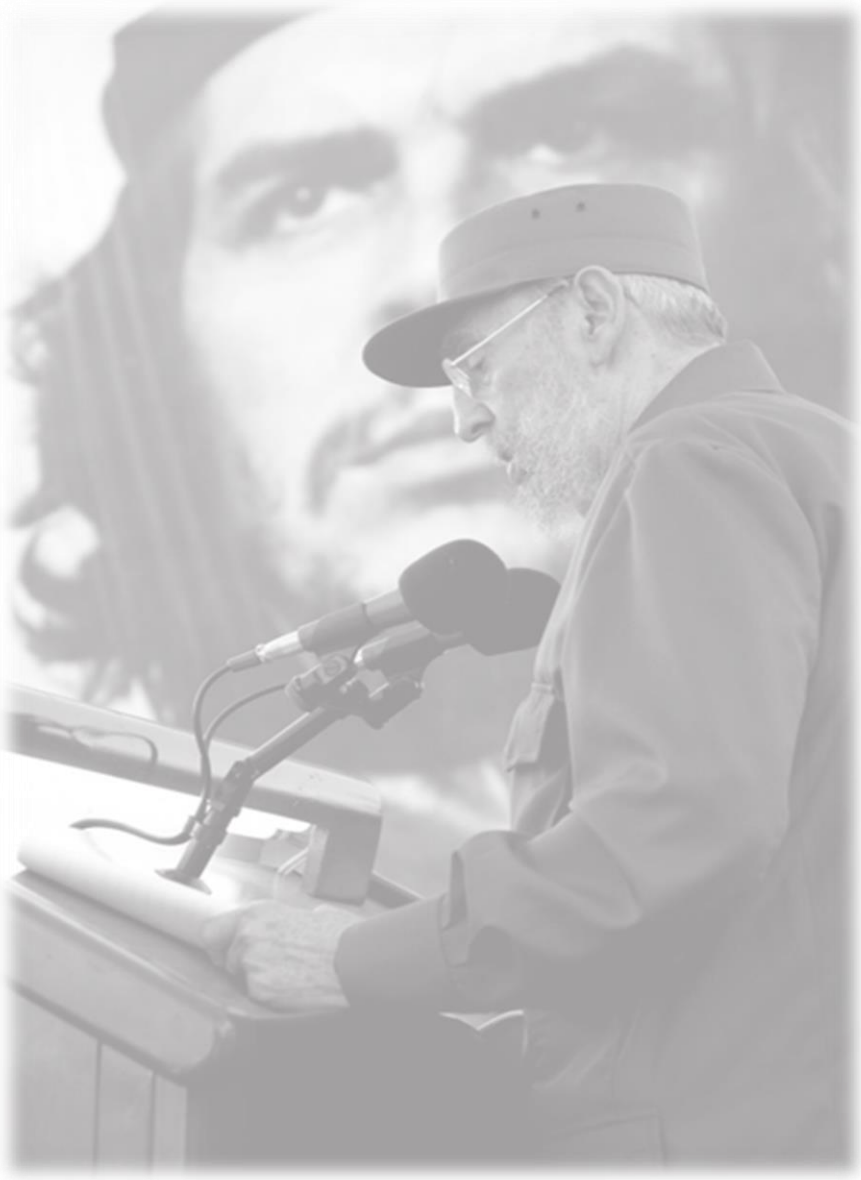
Módulo para la representación gráfica de la información del Juez en Línea Caribbean Mind Forge (CMF)

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

Autores: Yanet Hernández Herrera
Jessica Pedraza Azúa

Tutores: Ing. Javier de León Barral
Ing. Leandro Roura Sixto

La Habana, junio de 2016
"Año 58 de la Revolución"



“El futuro de nuestra patria tiene que ser necesariamente un futuro de hombres de ciencia, tiene que ser un futuro de hombres de pensamiento, porque precisamente es lo que más estamos sembrando; o que más estamos sembrando son oportunidades a la inteligencia (…)”

Fidel Castro Ruz

Declaración de autoría

Declaro ser autor de la presente tesis y concedo a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año 2016.

Yanet Hernández Herrera	Jessica Pedraza Azúa
Ing. Javier de León Barral	Ing. Leandro Roura Sixto

Agradecimientos

***A ti mami**, por ser la luz que me ha guiado hasta éste, el mayor de mis logros, a ti, por tu persistencia y dedicación inagotable, por esa confianza tan infinita que depositaste en mí, de la cual nunca habría imaginado que podría existir tanta en el corazón de una sola persona.*

***A ti papi**, por tu apoyo incondicional, por tu amor en todo momento, por siempre estar cuando te necesito y por malcriarme tanto.*

***A mi esposo,** por su amor incondicional, por su comprensión, por su ayuda más que inmensa, por ser mi mejor amigo, por tantas horas de amor y dedicación. Por lo que representa dentro de mi vida. Sin ti, no sé si lo hubiese logrado, Te Amo Javi ❤.*

***A mi hermana**, por su preocupación, por su extraño amor y por darme en los últimos 5 años la alegría de tener el más hermoso sobrino, "Kevito".*

***A mis abuelos** Clara y Miguel, por su enorme confianza hacia mí, por su amor, por su preocupación infinita. Gracias por ser los mejores abuelos del mundo.*

***A mi suegra**, por haberme acogido en su familia con mucho amor y confianza, por malcriarme demasiado, por quererme como una hija más. Gracias por todo.*

***A mi cuñado** de toda la vida, por su preocupación, cariño y amistad.*

***A mis amigos y amistades:** Gracias por la ayuda durante todos estos años. Yeni & Angelito, por tantas horas de risa y compañía, por la confianza y el cariño. Kenier y Raúl gracias por la ayuda y el apoyo que me han dado en todo este tiempo, las carcajadas infinitas y los momentos de estrés en los que siempre me acompañaron. A Yeni que por cariño le digo "Lola", por la amistad tan buena y por la ayuda incondicional que me has dado en todo este tiempo que nos conocemos.*

***A mis profesores, tutores y compañera de tesis**, gracias por la ayuda en este trayecto de mi preparación profesional.*

***A la vida**, por haberme dado la oportunidad de estudiar y conocer personas hermosas en esta universidad.*

A todas las personas que hoy han hecho este sueño posible.

Yanet Hernández Herrera

A lo largo de mi vida que no es muy larga tampoco, han pasado muchas personas que han aportado su granito de arena, y algunas que otras una piedra o un ladrillo, para que yo hoy me convirtiera en lo que soy, una profesional. Con tanto aporte ya me he armado mi propia casa que es ser una mujer preparada para enfrentarme a la vida. Así que quiero agradecerles a todas las personas que formaron parte de esta construcción y empezaré por los arquitectos, quienes más sino ***mis padres****, ellos han dibujado todo el plano de mi vida, han estado a pie de obra desde el momento en que solo había terreno, han llevado sus sueños y los míos a la realidad sin importar el costo, gracias por tanto sacrificio. A los constructores, que no son más que* ***mi familia*** *y los buenos* ***profesores*** *que he tenido durante toda la construcción de mi vida, gracias, sin su trabajo nada de lo que soy ahora hubiese sido posible. A los pintores, que lograron una armonía de colores tanto alegres como tristes, ellos son* ***mis amigos****, los quiero a todos. A los de decoración de interiores, fue un gusto compartir con ustedes los últimos toques de la obra,* ***mis tutores*** *y* ***mi compañera de tesis****. Y finalmente a la persona que hizo de mi casa suya, y lleva viviendo en ella casi 5 años de su vida, sin importar en las condiciones que se encontraba, mi más sincero amor para ti* ***Adrián****.*

Jessica Pedraza Azúa

Dedicatoria

A mis padres, para darles otro motivo que los haga sentirse orgullosos de su hija más pequeña.

A mi esposo Javi.

Yanet Hernández Herrera

A mis padres, por todo su apoyo y esfuerzo para que yo me convirtiera hoy en una profesional.

A mi familia, por estar pendiente siempre de mí en todos los ámbitos de mi vida.

A mi novio Adrian, por ser un compañero incondicional en estos casi 5 años.

A todos los buenos profesores que he tenido durante toda mi vida.

Jessica Pedraza Azúa

Resumen

La Universidad de las Ciencias Informáticas cuenta hoy con un juez en línea para el desarrollo, ejercitación y realización de concursos de habilidades mentales: Caribbean Mind Forge. Actualmente este sistema no hace uso de la gran cantidad de datos que genera, por lo cual a los administradores se les dificulta tomar decisiones que favorezcan a los usuarios y al propio sistema. El objetivo del presente trabajo es desarrollar un módulo para el juez en línea que represente y procese la información almacenada en la base de datos del sistema Caribbean Mind Forge y apoye la toma de decisiones. Para ello se realizó un estudio de sistemas similares a nivel internacional y nacional basándose en el contexto de la problemática tratada. Además, se documentaron los elementos que se tienen en cuenta para el desarrollo en cuanto a herramientas, lenguaje de programación y metodología a emplear, destacándose como elemento principal el uso la librería gráfica FusionCharts para la representación de la información. Se establecen los requisitos funcionales y no funcionales, descripción de historias de usuarios que fundamentan el proceso de desarrollo del software y se detallan los resultados de las pruebas. En consecuencia, se obtuvo un módulo muy completo que abarca todas las áreas que el sistema maneja y que a su vez satisface al cliente.

Palabras claves: administradores, juez en línea, módulo, tomar decisiones.

Índice

INTRODUCCIÓN

El desarrollo de software educativo en los últimos años ha pasado en Cuba de ser concebido como un presentador de información a ser un elemento didáctico interactivo que se elabora a partir de la representación de conocimiento y que facilita en el usuario su construcción gracias a la utilización de elementos que permiten solucionar problemas e impactar su estructura cognitiva (1).

La Universidad de las Ciencias Informáticas (UCI) pretende informatizar el país y desarrollar la industria de software para contribuir al desarrollo económico del mismo. Su misión es producir software y servicios informáticos a partir de la vinculación estudio – trabajo como modelo de formación y forjar profesionales comprometidos con su Patria, altamente calificados en la rama de la informática (2).

Dentro de la universidad se desarrolla una línea de investigación relacionada con las herramientas de apoyo al proceso de Enseñanza y Aprendizaje (3), particularmente con los Jueces en Línea en las asignaturas de matemática y programación. El 5 de junio de 2013 en la UCI surge el nuevo juez de agilidad mental, Caribbean Mind Forge (CMF) con el propósito de aportar al proceso educativo[1] y de promover el desarrollo del conocimiento de manera recreativa y competitiva de las siguientes materias (4):

- Resolución de problemas de agilidad mental, lógica, matemática y cultura general.
- Memoria fotográfica.
- Cálculo de fechas.
- Operaciones matemáticas.
- Tecleo rápido.
- Sudoku.
- Geografía visual.

CMF ha propiciado a sus usuarios un espacio destinado a la auto-preparación, el fortalecimiento y aprendizaje de algunas materias y el intercambio de conocimientos, convirtiéndose en una plataforma de ejercitación y entrenamiento en las distintas modalidades que maneja. Desde su publicación en la red hasta el momento se ha percibido un creciente interés de los estudiantes, profesores, trabajadores y personal

[1] Proceso de formación profesional que gira en torno al aprendizaje de los seres humanos, desde una óptica de la construcción del conocimiento y el cultivo de la inteligencia en todas sus formas.

externo de la universidad por la actividad diaria de los usuarios. Los volúmenes de información se han ido incrementando considerablemente. Durante el transcurso diario se genera abundantes datos asociados a los distintos recursos que en el sistema se manejan, como usuarios, problemas, sentencias, concursos y envíos.

Hasta el momento, los datos generados no son utilizados en función a las necesidades de los administradores puesto que no permite determinar ágilmente informaciones importantes como pueden ser: conocer los usuarios inactivos en el sistema, la distribución por fechas de los envíos de un concurso o las universidades que más usuarios tienen participando en competencias. El sistema no cuenta con una forma de representar los datos para la toma de decisiones por parte de los administradores, es por esta razón que en ocasiones esto es efectuado al azar o simplemente no se realiza.

Teniendo en cuenta lo planteado se ha identificado el siguiente **problema a resolver:** ¿Cómo contribuir al apoyo de la toma de decisiones en el juez en línea Caribbean Mind Forge?

Se define como **objeto de estudio:** el procesamiento y la representación de los datos. Se plantea como **objetivo general:** desarrollar un módulo para el juez en línea Caribbean Mind Forge que represente y procese datos almacenados en la base de datos para el apoyo a la toma de decisiones, especificándose como **campo de acción:** el procesamiento y la representación de los datos generados en los jueces en línea.

A partir del análisis del objetivo general se derivan los siguientes **objetivos específicos:**

- Realizar el marco teórico de la investigación para obtener una visión global de cómo representar información de forma gráfica que permita tomar decisiones.
- Diseñar e implementar la solución modelada que permita representar gráficamente los distintos recursos con los que trabaja el sistema.
- Validar la solución obtenida mediante las pruebas de validación para la comprobación del correcto funcionamiento del módulo.

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

- Caracterización de los jueces en línea existentes en cuanto a la representación gráfica de los datos para obtener información acerca de los distintos tipos de información que se pueden mostrar mediante gráficos.
- Fundamentación de la selección de la metodología, las herramientas y las tecnologías a utilizar para realizar la implementación del módulo.

- Especificación de los requisitos funcionales y no funcionales del módulo para establecer lo que el software debe hacer y bajo qué circunstancias debe hacerlo.
- Elaboración del modelo de diseño del módulo para obtener una mejor visión de su desarrollo.
- Implementación de las funcionalidades del módulo haciendo uso de las tecnologías seleccionadas para dar cumplimiento a la descripción de la solución planteada.
- Definición de los tipos de pruebas para determinar el comportamiento del software bajo determinadas circunstancias.
- Realización de las pruebas de software para la comprobación del correcto funcionamiento del módulo.

Durante la investigación se utilizan los siguientes **métodos científicos**:

Métodos teóricos:

- Analítico-sintético: para el análisis de la teoría, documentación y demás antecedentes relacionados con el objeto de estudio, permitiendo concluir las características principales que debe tener el módulo propuesto, así como las herramientas y tecnologías óptimas para su desarrollo.
- Modelación: para representar los procesos, datos y características inherentes al campo de acción.

Métodos empíricos:

- Observación: el cual permitió la obtención de conocimiento e información acerca de la representación de la información en los jueces en línea.

El documento está estructurado en tres capítulos:

Capítulo I: Fundamentación teórica

En el capítulo quedan sustentadas las bases teóricas de la investigación. Se exponen los conceptos fundamentales asociados al objeto de estudio del trabajo. Se realiza un estudio de los jueces en línea existentes a nivel mundial y nacional en busca de soluciones para la representación y procesamiento de la información. Además, se hace referencia a la metodología, herramientas, lenguajes y tecnologías utilizadas durante el desarrollo de la investigación.

Capítulo II: Análisis y diseño

Se hace alusión a todo lo referente a las características del módulo, donde se plasman los requisitos funcionales definidos y se encapsulan en historias de usuario. Además, se explica cómo está estructurada la arquitectura de la aplicación desarrollada y los patrones de diseño que se utilizan.

Capítulo III: Implementación y Pruebas

En este tercer y último capítulo se incluyen ejemplos de código de las diferentes clases que explican el funcionamiento del módulo. Se generan los diagramas de componentes y de despliegue y se concluye con la ejecución de las pruebas para comprobar si la aplicación cumple sus objetivos, mostrando los resultados obtenidos.

CAPÍTULO 1

Fundamentación teórica del módulo para la representación gráfica de la información del juez en línea Caribbean Mind Forge

1.1 Introducción

En el presente capítulo se lleva a cabo la fundamentación teórica del trabajo, teniendo como principal objetivo estudiar y analizar los elementos principales relacionados con la investigación. Además, se hace un estudio de sistemas existentes tanto en el ámbito internacional como nacional, considerando la representación de la información para la toma de decisiones. Se culmina el capítulo con la elaboración del perfil tecnológico realizando la selección de las herramientas, tecnologías y metodología que se utilizan en el desarrollo del proyecto.

1.2 Conceptos asociados al dominio del problema

Aplicación web

En la ingeniería de software se denomina aplicación web a aquellas aplicaciones que los usuarios pueden utilizar accediendo a un servidor web a través de Internet o de una intranet mediante un navegador. Una de las ventajas de las aplicaciones web cargadas desde internet (u otra red) es la facilidad de mantener y actualizar dichas aplicaciones sin la necesidad de distribuir e instalar un software en, potencialmente, miles de clientes. También la posibilidad de ser ejecutadas en múltiples plataformas. Es una aplicación que se codifica en un lenguaje soportado por los navegadores web. Permite al usuario interactuar directamente con la aplicación y los datos. Además, son aquellas en que los usuarios acceden a ellas en un servidor Web a través de Internet o de una intranet (5) (6).

Juez en Línea

Un juez en línea es un sistema, por lo general web, que permite juzgar programas de computación que intenten solucionar tareas propuestas. Estos sistemas pueden compilar y ejecutar códigos fuente, y ponerlos a prueba con los juegos de datos definidos para la tarea seleccionada. Las posibles soluciones se ejecutan con restricciones tales como: límite de tiempo de ejecución, límite de memoria, tamaño de código fuente, restricciones de seguridad, entre otras (7). La salida de cada programa enviado por el usuario es capturada por el sistema y comparada contra la salida que se tiene de la tarea en cuestión o será evaluada por un evaluador externo.

Acerca de los evaluadores automáticos de soluciones se ha hecho y documentado bastante. Sin embargo, no existe una teoría unificada para desarrollar evaluadores, sino más bien las pautas para hacerlo dependen de las necesidades de la organización o el usuario que lo desarrolle. Si el evaluador cumple una necesidad, o de la visión particular del equipo de desarrollo que emprenda tal tarea, si el evaluador se pretende utilizar para brindar un servicio general y no para cumplir una necesidad puntual. A esta última categoría corresponden los jueces en línea publicados en Internet, en lo que se refiere a funcionalidades y tipos de concursos (8) (9).

Los jueces en línea son aplicaciones bastante conocidas en el mundo, pero no son muy comunes ni frecuentadas dado que en su mayoría están orientadas a un público muy específico. Los jueces en línea se asocian generalmente con entornos académicos, en especial universidades, debido a que en este entorno la búsqueda del conocimiento teórico se estimula de forma más activa ya que los objetivos y el funcionamiento de un juez en línea no es fácil de convertir en lucrativo a no ser las actividades colaterales que estén relacionadas con el mismo como publicaciones, capacitaciones y concursos (10).

La manera en que un juez en línea comprueba la corrección de una solución es sencilla: para cada problema a resolver hay que aplicar un conjunto de reglas y criterios para probar la corrección de las soluciones enviadas por los usuarios, siendo casi siempre estas, ficheros de entrada y de salida. La solución enviada se considera correcta si la verificación de estas reglas fue exitosa y en caso contrario se considera incorrecta (10).

En la siguiente figura se muestra el funcionamiento de los jueces en línea de manera general, donde se evidencia que los usuarios pueden resolver problemas y enviar las soluciones; el juez se encarga de manera automática analizas las soluciones y emitir un veredicto.

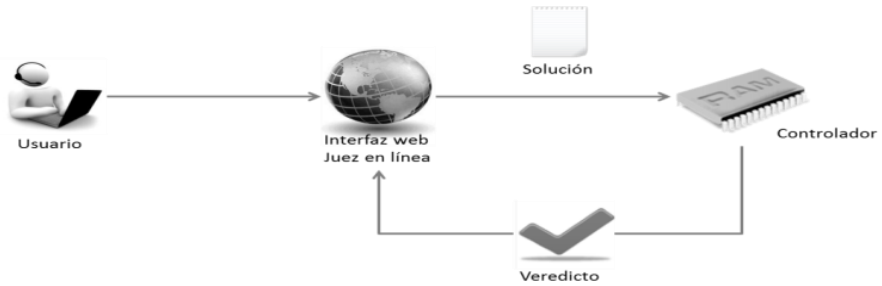


Figura 1 Esquema de funcionamiento general de los jueces en línea **(9)**

Procesamiento de datos

Es la técnica que consiste en la recolección de los datos primarios de entrada, que son evaluados y ordenados para obtener información útil, que luego serán analizados por el usuario final para que pueda tomar las decisiones o realizar las acciones que estime conveniente (11). Es el conjunto de métodos y procedimientos que permiten obtener información. Estos pueden ser distinguidos entre manuales, automáticos, electromecánicos y electrónicos.

El procesamiento de datos es el proceso de transformación de datos en información y puede hacerse utilizando o no una computadora. Pero si se considera que las computadoras procesan datos con mucha mayor velocidad que lo que puede hacer el hombre, y los resultados que arrojan son totalmente exactos; entonces el procesamiento de manera automática permite la toma de decisiones en forma oportuna (12).

Es en general, "la acumulación y manipulación de elementos de datos para producir información significativa." El procesamiento puede involucrar diversas funciones, entre ellas (13):

- Validación: Asegurar que los datos suministrados son limpios, correctos y útiles.
- Clasificación: Ordena elementos de cierta secuencia o en diferentes conjuntos.
- Recapitulación: Reducir los detalles de los datos a sus principales puntos.
- Agregación: Combinación de múltiples piezas de datos.
- Análisis: La colección, organización, análisis, interpretación y presentación de datos.
- Información: Lista detallada o resumen de los datos de información computarizada.

Representación de información

Debe distinguirse entre dato e información. Dato es cualquier número, imagen, sonido, símbolo o nombre que no permite tomar alguna decisión porque no tienen un significado concreto, el mismo debe procesarse para dotarlo de significado y convertirlo en información. Eso es lo que ocurre cuando se trabaja con un sistema informático, los resultados que muestra el ordenador es información que ayuda en alguna tarea. Entonces, la representación de la información es la aplicación de una serie de datos organizados con los que se puede describir una situación que sustituye a la realidad, y pueden ser vista en innumerables aspectos cotidianos (14) (15).

La manera en la que se transmite un mensaje tiene un alto impacto en la que el receptor asume la información recibida (16). Esto quiere decir, ajustado al contexto, que en dependencia de la forma, intención o importancia que tenga la representación de la

información será más aprovechada por el receptor. Para ello existen las siguientes técnicas de representación (17):

- **Tabular:** Cuando los datos estadísticos se presentan a través de un conjunto de filas y de columnas que responden a un ordenamiento lógico; es de gran importancia para el uso de los usuarios ya que constituye la forma más exacta de presentar las informaciones.
- **Textual:** Cuando una serie de datos incluye pocos valores, por lo cual resulta más apropiada la palabra escrita como forma de escribir el comportamiento de los datos; mediante la forma escrita, se resalta la importancia de las informaciones principales.
- **Gráfica:** Proporciona al lector o usuario mayor rapidez en la comprensión de los datos, una gráfica es una expresión artística usada para representar un conjunto de datos. Además, son imágenes que, combinando la utilización de sombreado, colores, puntos, líneas, símbolos, números, texto y un sistema de referencia (coordenadas), permiten presentar información cuantitativa.

1.3 Tendencias actuales

1.3.1 Principales Jueces en línea

Durante la etapa de investigación se identificaron alrededor de 75 jueces en línea (18), de los cuales su mayoría está concentrada en Asia. A continuación, se describen algunos de los más significativos identificando sus principales características dentro del contexto de la problemática tratada.

Juez en línea de la Universidad de Valladolid (UVa)

El UVa es uno de los jueces en línea más antiguos y prestigiosos del mundo. Posee uno de los mejores archivos de problemas de competencias oficiales del mundo relacionadas con el Concurso Internacional Universitario ACM[2] de Programación (ACM-ICPC por sus siglas en inglés) (19).

Como rasgos distintivos se pueden resaltar (20):

- Integración a la plataforma EduJudge, convirtiéndolo en uno de los pocos jueces en línea que poseen una integración con el proceso de enseñanza-aprendizaje.
- Implementado sobre plataforma UNIX.
- Tabla de posiciones, dedicada al desempeño de los usuarios en competencias.

[2] Es una de las primeras sociedades científicas y educativas acerca de la computación.

- ❖ Módulo de estadísticas, el cual representa información respecto a las estadísticas de uso de los problemas, usuarios y concursos.

Sphere Online Judge (SPOJ)

SPOJ es un juez en línea desarrollado por el Sphere Interest Project, entidad afiliada a la Universidad de Tecnología de Gdansk en Polonia, dedicado al desarrollo de tecnologías en línea para proveer herramientas de evaluación automática de habilidades. SPOJ se ha convertido en uno de los jueces en línea más populares en la comunidad internacional, contando hasta la fecha con más de 399274 usuarios registrados, más de 16318091 envíos realizados y más de 5859 problemas publicados, convirtiéndolo en uno de los archivos más grandes de problemas y de más rápido crecimiento en el mundo (21).

Cuenta con una pequeña herramienta administrativa que solo representa la información referente a los usuarios y permite compararlos (22).

Juez en Línea de Pekín (POJ)

El POJ es uno de los jueces en línea de mayor demanda entre la comunidad universitaria china, puesto que es uno de los más antiguos de esa región. Posee características similares a varios jueces en línea de otras universidades chinas y otros lugares del mundo, debido a que por algún tiempo era posible descargar una versión disminuida y restrictiva de la aplicación. Ofrece un servicio de mensajería interna para los usuarios registrados, funcionalidad que se considera muy útil en este tipo de sistemas.

Su característica más distintiva es el módulo de estadísticas donde representa información de manera gráfica, sin embargo, se considera que la información ofrecida es pobre, limitándose solo al comportamiento de los envíos, mensajes y *mails* (23).

Codeforces

En la actualidad, Codeforces es nombrado como uno de los jueces en línea más populares. Surge en el año 2010 como iniciativa de Mike Mirzanayov, cambiando la dinámica de los jueces en línea existentes hasta el momento. Constituye un sistema predominante para la realización de competencias de programación en varios estilos existentes. También aplica una funcionalidad que se considera vital para incrementar la preparación de los concursantes y es la referente a mostrar los códigos fuentes de las soluciones una vez finalizado cada concurso. Por otra parte, como resultado de aplicar

la Web 2.0, ofrece la posibilidad de autenticarse en el sistema con el mismo usuario de Google (24).

Hasta el momento no cuenta con algún modo de representar la información de manera resumida que ayude a los administradores a la toma de decisiones.

TopCoder

TopCoder es una compañía que organiza concursos de programación de diferentes estilos: Algoritmos, Diseño, Marathon Matches y Competencias de detección de errores. El principal objetivo de la compañía es proveer una plataforma donde las diferentes empresas puedan reclutar personal talentoso, basados en el desempeño de los mismos en competencias. TopCoder funciona además como una fuente de creatividad para las empresas que contratan los servicios de la plataforma. El rasgo más distintivo de la plataforma es la integración como plataforma educativa y empresarial al mismo tiempo, además de incluir novedosos estilos de competencias, así como nuevos sistemas de puntuación (25) pero cumple con la misma condición de Codeforces en cuanto al apoyo a la toma de decisiones.

Caribbean Online Judge (COJ)

El desarrollo del Juez en línea caribeño de programación (COJ por sus siglas en inglés) tiene sus antecedentes en el sistema base (*Xtreme Online Judge*) en el año 2006, compuesta principalmente por estudiantes y profesores de la Facultad 8 de la UCI. Luego de que la UCI se unió al movimiento ACM-ICPC y lideró la creación de la Comunidad Caribeña del ACM-ICPC, el *Xtreme Online Judge* fue seleccionado para publicarse en internet como COJ v1.0. El COJ está disponible en internet desde el 5 de junio de 2010 (26).

Hasta la fecha el sistema cuenta con más de 20000 usuarios registrados, 14 lenguajes, más de medio millón de soluciones evaluadas a los miles de problemas publicados en su archivo. Hay usuarios de al menos 144 países distintos y dentro de ellos de 1010 instituciones. Ha sido anfitrión de cientos de competencias desde el nivel local hasta el nivel regional del ACM-ICPC del Caribe y de otras competencias nacionales de programación (27).

El COJ posee un módulo estadístico para el apoyo a la toma de decisiones con funcionalidades importantes en el mismo sistema y en su parte administrativa, algunas son:

- Mostrar los nuevos registros de usuarios en el COJ en un periodo de tiempo.

- Comparar usuarios en cuanto a veredictos.
- Mostrar línea de tiempo por desempeño de usuario en el concurso.
- Mostrar distribución de estado de los problemas en concurso.
- Comparar usuarios en cuanto a veredictos.

Análisis de la investigación de los jueces en línea

A partir de la investigación realizada a cada uno de los jueces en línea se determinó que ninguno satisface el objetivo del presente trabajo ya que no existe una tendencia a representar la información de todos los recursos que manejan de manera gráfica, para así facilitar la toma de decisiones a los administradores. De los seis analizados, cuatro de ellos poseen un módulo estadístico, aunque no siempre de manera gráfica. El más completo de ellos es el COJ, pero como deficiencia se le encontró que su módulo no se encuentra totalmente disponible. El resto cuenta con funcionalidades para la representación de la información de manera gráfica, pero no tratan todas las áreas requeridas para dar solución al problema que fundamenta la necesidad de la presente investigación.

1.4 Metodología de desarrollo de software

Una metodología de desarrollo de software se define como: “un conjunto de procedimientos, técnicas, herramientas y un soporte documental que ayuda a los desarrolladores a realizar un software. La metodología indica cómo hay que obtener los distintos productos parciales y finales (28)”.

1.4.1 Matriz de Boehm y Turner

Para fundamentar la selección de la metodología se aplica el modelo propuesto por Barry Boehm y Richard Turner en su libro *"Balancing Agility and Discipline: A Guide for the Perplexed"* (29) que muestra una evaluación de las características del proyecto para determinar la idoneidad de un enfoque ágil o tradicional.

Las características medidas son (29):

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

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

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

Tamaño: Se refiere a la cantidad de miembros que integren el equipo de desarrollo. Los métodos ágiles son más fáciles de establecer, ejecutar y gestionar en equipos pequeños. Esto no quiere decir que grandes equipos no pueden trabajar con métodos ágiles, sólo que es mucho más difícil de lograr.

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

1.4.1.1 Resultados

Personal

Personal y Tamaño		
Niveles	Cantidad	%
Nivel -1	0	0.00
Nivel 1B	0	0.00
Nivel 1A	0	0.00
Nivel 2	2	100
Nivel 3	0	0.00
Total	2	100

Tabla 1 Personal

El valor que toma la escala Personal tiende a 0% en el nivel 1 y a 35% correspondiente al Nivel 2 y 3.

Dinamismo: El sistema está sujeto a varios cambios, dependiendo de la satisfacción y necesidad del cliente. Se estuvieron integrando algunas funcionalidades en el transcurso del desarrollo por lo que fueron encontradas varias no conformidades en el proceso. También, cuenta con la aprobación de un tribunal. Entonces tiende a acercarse al 50% el valor que toma el dinamismo.

Cultura: El equipo de desarrollo se adapta bien a los cambios que pueden surgir, aunque dada la complejidad y cantidad de funcionalidades a implementar, entonces el tiempo actúa como factor en contra de dichos cambios. Por lo cual el valor se encuentra aproximadamente en el 40%.

Tamaño: El equipo dispone de dos personas que se desempeñan en todos los roles. El valor que toma la escala correspondiente al Tamaño es 2.

Criticidad: Para la implementación del sistema se utiliza una copia reciente de la base de datos del juez en línea CMF. Previamente es probada antes de integrar cualquier funcionalidad del mismo con la supervisión de los administradores del sistema. Por lo cual los fallos que pudieran ocurrir no darán ninguna pérdida.

Teniendo presente cada uno de los valores planteados se realiza la matriz que se presenta en la figura siguiente.

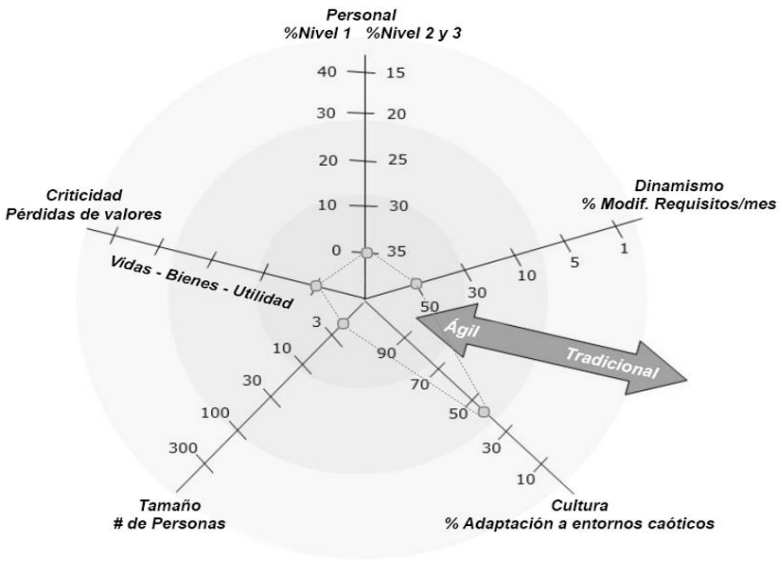


Figura 2 Aplicación de la matriz de Boehm y Turner

La organización de estas variables en un grafo estrella donde la convergencia es hacia metodologías de desarrollo ágiles y la divergencia en al menos una de las variables conlleva a la sugerencia de empleo de metodologías más pesadas. Por lo tanto, debido al análisis de la matriz, se refleja mediante los resultados obtenidos que la utilización de una metodología ágil es la más factible.

Selección de la metodología de desarrollo

Para guiar el proceso de desarrollo, se selecciona la metodología AUP-UCI. No es considerada la metodología utilizada para la creación de CMF (*eXtreme Programming*[3]) porque AUP-UCI está orientada a que se adapte al ciclo de vida definido para la actividad productiva de la universidad (30).

[3] *eXtreme Programming e*s una metodología de desarrollo de la ingeniería de software formulada por Kent Beck

Con esta metodología se logra estandarizar el proceso de desarrollo de software, dando cumplimiento a las buenas prácticas que define CMMI-DEV[4] v1.3 y se logra hablar un lenguaje común en cuanto a fases, disciplinas, roles y productos de trabajo (30).

La misma no es más que una variación de la metodología AUP (*Agile Unified Process* / Proceso Unificado Ágil) de Scott Ambler (30).

1.5 Herramientas y tecnologías

Los creadores de CMF desarrollaron un análisis de las herramientas y tecnologías que eran necesarias para su implementación, demostrando su eficiencia en el contexto para el cual fueron utilizadas.

Al formar parte modular de CMF, la solución desarrollada debe enfocar la selección en evitar que exista inconsistencia en el código y en la integración con el sistema. Por lo cual el módulo está sujeto a las herramientas y tecnologías que usa el juez en línea CMF.

1.5.1 Lenguaje de programación

Groovy 1.7 es un lenguaje dinámico que se ejecuta en la máquina virtual de Java que puede ser integrado a aplicaciones Java ya existentes, lo que demuestra que Groovy y Java están muy estrechamente ligados. Todas las librerías o bibliotecas implementadas para la plataforma Java son accesibles en Groovy, lo que aumenta la cantidad de funcionalidades y el potencial que Groovy puede dar (31) (32).

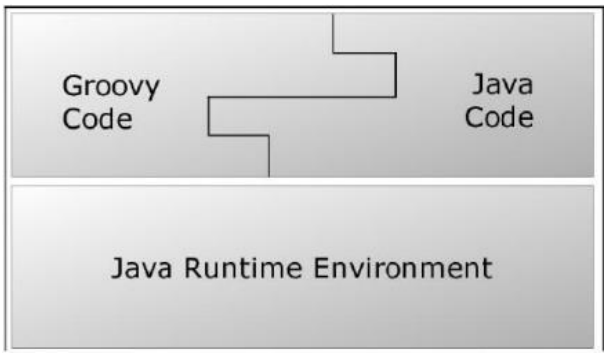


Figura 3 Integración entre Groovy y Java

Desde un punto de vista del entorno de ejecución, se puede decir que Groovy es una nueva forma de crear clases Java, estos lenguajes son totalmente compatibles lo que permite que Groovy haga uso de todas las funcionalidades provista por la API

[4]CMMI-DEV acrónimo de Capability Maturity Model Integration, en español desarrollo y mantenimiento de productos y servicios

(*Application Programming Interface* / Interfaz de Programación de Aplicaciones) de Java, y que Groovy pueda ampliarla y extenderla, y a su vez, Java puede hacer uso de las clases creadas con Groovy (31) (32).

1.5.2 Framework

Grails 2.0.3 es un *framework* para el desarrollo de aplicaciones web construido sobre cinco fuertes pilares. Es de código abierto compatible 100% con Java. Combina patrones tales como "convención sobre configuración" (*Convention Over Configuration*) y "no te repitas" (*Don't Repeat Yourself*) junto a una serie de *frameworks* de código abierto como Hibernate (para mapeo objeto-relacional), Spring (para inyección de dependencias) y SiteMesh (para plantillas). Todo esto, unido al lenguaje dinámico Groovy construido sobre Java. Grails pretende ser un *framework* altamente productivo, proporcionando un entorno de desarrollo estandarizado y ocultando gran parte de los detalles de configuración al programador (33).

Puede ser ampliado a través de *plugins*[5]. Por su dinamismo es capaz de acortar el ciclo de desarrollo ahorrando tiempo de trabajo y agilizándolo. Esta combinación puede reducir el tiempo inicial del proyecto y el tiempo de instalación del desarrollador a minutos en lugar de horas (33).

Como la mayoría de los *frameworks* de desarrollo web, Grails está basado en una especialización del patrón Modelo-Vista-Controlador en su versión 2 (MVC2). En Grails las clases de dominio son entidades persistentes. Los controladores por su parte, permiten gestionar las peticiones a la aplicación. Por último, Grails implementa las vistas mediante *Groovy Server Pages (gsp)*. Sin embargo, las vistas son los únicos componentes de Grails que no son Groovy ni Java y que residen en la carpeta *grails-app/views* del proyecto (33).

Grails como *framework* principal para el desarrollo presenta las siguientes ventajas:

- Utiliza varios *frameworks* como son Hibernate, SiteMesh, Ant y el propio Spring.
- Proporciona bibliotecas de etiquetas dinámicas para crear fácilmente componentes web.
- Buen soporte de *Ajax*[6] que es fácil de extender y personalizar.

[5] es una aplicación o parte de ella que se relaciona con otra para aportarle una función nueva
[6] Asinchronic Javascript XML

- Proporciona un entorno completo de desarrollo, incluyendo un servidor web, una base de datos, sistemas de construcción, pruebas y recarga automática de recursos.

1.5.3 Librería de Javascript

Jquery 2.0 es una biblioteca de Javascript (JS), es libre y de código abierto. Esta biblioteca hace más simple la manera de interactuar con los documentos HTML, manipular el árbol DOM[7], agregar interacción con AJAX a páginas web, así como manejar eventos y desarrollar animaciones. Jquery ofrece una serie de funcionalidades basadas en JS que de otra manera requerirían mucho más código. Con las funcionalidades de esta biblioteca se logran mejores resultados en menos tiempo y espacio. Jquery es un solo fichero JS que contiene las funcionalidades del DOM, los eventos, efectos y AJAX (34) (35).

1.5.4 Librerías Gráficas de Javascript

La solicitud de la representación de la información sobre los datos de trabajo, suele venir acompañada de una petición para visualizar mediante un gráfico los datos procesados en la aplicación. Esto es debido a que toda la recopilación y tratamiento de la información no tiene ninguna utilidad si finalmente no se puede realizar una representación gráfica que permita la comprensión de todo el trabajo anterior.

En la investigación se identificaron alrededor de 50 librerías y a continuación se mencionan algunas de las más significativas (36):

- ✓ Charts.js
- ✓ FusionCharts
- ✓ Highcharts
- ✓ Epoch

Análisis de la selección de la librería Gráfica de Javascript

Después de realizar un análisis comparativo de las librerías anteriores, se decide utilizar FusionCharts en su versión libre como librería principal para la representación de la información, ya que presenta las siguientes ventajas (37):

- Fácil integración con otras librerías, *frameworks* y lenguajes.
- Cuenta con las gráficas básicas (línea, columna, pastel, 2D y 3D) hasta las más complejas (cascada, Gantt, candelabro o *zoomline*).

[7] Modelo en Objetos para la Representación de Documentos

- Permite exportar a imagen, PDF o SVG.
- Todos los gráficos presentan la misma apariencia en todos los navegadores web.
- Permite personalizar la apariencia de los gráficos como el color de fondo, los colores de la trama y fuentes.

1.5.5 Servidor de base de datos

PostgreSQL 8.4 Es un avanzado sistema de bases de datos relacionales basado en Open Source. Esto quiere decir que el código fuente del programa está disponible a cualquier persona libre de cargos directos, permitiendo a cualquiera colaborar con el desarrollo del proyecto o modificar el sistema para ajustarlo a sus necesidades. PostgreSQL está bajo licencia BSD[8]. Un sistema de base de datos relacionales es un sistema que permite la manipulación de acuerdo con las reglas de álgebra relacional. Soporta una capacidad de almacenamiento en el orden de los *Tera bytes* (TB). Los datos se almacenan en tablas de columnas y renglones. Con el uso de llaves, esas tablas se pueden relacionar unas con otras (38) (39).

1.5.6 Servidor web

Apache Tomcat 7.0.54 es desarrollado en un entorno abierto, participativo y publicado bajo la licencia Apache versión 2, por miembros de la *Apache Software Foundation* y voluntarios independientes. Además, puede funcionar como servidor web por sí mismo. Al principio de su desarrollo existió la percepción de que la utilización de Tomcat de forma autónoma era sólo recomendable para entornos de desarrollo y entornos con mínimos requisitos de velocidad y gestión de transacciones. Actualmente ya no existe esa percepción y Apache Tomcat es usado como servidor web independiente en entornos con alto nivel de tráfico y alta disponibilidad (40).

1.5.7 Herramienta CASE para el modelado

Visual Paradigm 8.0 es una herramienta CASE: Ingeniería de Software Asistida por Computación. Soporta el ciclo de vida completo del desarrollo de software: análisis y diseño Orientados a Objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una construcción más rápida de aplicaciones de mayor calidad y a un menor coste. Permite diseñar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y documentación (41).

[8] Es la licencia de software otorgada principalmente para los sistemas BSD (Berkeley Software Distribution)

Se caracteriza por la disponibilidad en múltiples plataformas (Windows, Linux). Ostenta características como diseño centrado en casos de uso y enfocado al negocio que genera un software de mayor calidad. El modelo y el código permanecen sincronizados en todo el ciclo de desarrollo. Cuenta con las ventajas que permite la generación de bases de datos y transformación de diagramas de Entidad- Relación en tablas de base de datos. Posee compatibilidad entre sus ediciones (41).

1.5.8 Herramienta para Pruebas de Carga y Estrés

JMeter 2.3.1 es una herramienta Java desarrollada dentro del proyecto Jakarta, que permite realizar Pruebas de Rendimiento y Pruebas Funcionales sobre aplicaciones web. Efectúa pruebas web clásicas, pero también permite realizar *test* de FTP, JDBC, LDAP, SOAP/XML-RPC y WebServices. Permite la ejecución de pruebas distribuidas entre distintos ordenadores. Además, brinda la posibilidad de activar o desactivar una parte del *test*, lo que es muy útil cuando se está desarrollando una prueba larga, y se permite deshabilitar ciertas partes iniciales que sean muy pesadas o largas. Tiene la forma de generar un caso de prueba a través de una navegación de usuario (42).

JMeter manipula datos de entrada para la ejecución de las pruebas y permite la comparación de los resultados para la evaluación. Almacena los resultados de las pruebas y se generan gráficos que representan los aspectos que se han probado. Además, se clasifica por su especialización en una herramienta de grabación y reproducción (42).

Conclusiones del Capítulo

- En el capítulo se profundizó en los conceptos relacionados con la representación y procesamiento de la información gráfica generada en los jueces en línea.
- Se analizó los sistemas existentes documentando sus características y no se logró identificar una tendencia en los jueces en línea en cuanto a la representación gráfica.
- Mediante la investigación se detectaron cuatro jueces que poseen módulos que representan la información. Los mismos no satisfacen en su totalidad la necesidad de la investigación, pues solo muestran gráficos generales sobre el comportamiento de los usuarios y no llegan a realizar estudios más profundos sobre los problemas, envíos o concursos.
- Se hizo necesario el desarrollo de un módulo para la representación de la información de CMF porque no es posible contar en su totalidad con la experiencia de sistemas similares para afrontar la presente problemática.

- Para la implementación del módulo se tuvo en cuenta la metodología, lenguajes, tecnologías y herramientas utilizadas en el desarrollo de CMF, por lo tanto, se eligió Groovy como lenguaje de programación, Grails como framework de desarrollo, FusionCharts como librería gráfica para la representación de la información y AUP-UCI como metodología para guiar el proceso de desarrollo.

CAPÍTULO 2. Descripción de la solución

2.1 Introducción

El presente capítulo comienza con la representación del modelo de dominio, seguidamente se describe el sistema a grandes rasgos dando paso al planteamiento de los requisitos funcionales y no funcionales que debe cumplir este para la satisfacción del cliente. Luego se encapsulan los requisitos funcionales en historias de usuarios las cuales permiten una descripción de cada uno de estos. Y finalmente se plantean los distintos patrones de diseño que se emplean y la conformación de la arquitectura del sistema.

2.2 Modelo de dominio

Para el desarrollo del módulo no se aprecian claramente los límites del proceso, haciéndose difícil determinar el conjunto de las actividades que se desarrollan en el negocio. Por tal razón no se desarrolla Modelo de Negocio y se ha determinado realizar un Modelo de Dominio.

El Modelo de Dominio es una representación visual de los objetos, conceptos y procesos del mundo real que resultan de interés para el problema, proveniente de la base de conocimientos de expertos o de conocimiento asociado a sistemas similares.

A continuación, se describen los principales conceptos relacionados en el modelo de dominio:

CMF: Juez en línea de agilidad mental, lógica, matemática y cultura general que tiene asociado usuarios, competencias y un conjunto de problemas.

Concurso: Competencia que se realiza haciendo uso de diversos problemas.

Problema: Situación a resolver.

Envío: Solución que emite un usuario a los problemas publicados.

Sentencia: Palabra con la que se expresa la calificación del envío.

Usuario: Persona registrada en CMF que tiene permisos determinados y puede enviar soluciones a diversos problemas.

Administrador: Usuario que tiene permisos administrativos en el sistema y maneja los reportes gráficos.

Público: Usuarios registrados en el sistema sin permisos administrativos.

País: Territorio que constituye una unidad geográfica o política, limitada natural o artificialmente

Institución: Organismo que desempeña una función de interés público, especialmente benéfico o docente.

Reporte Gráfico: Informe que organiza y exhibe la información contenida en la base de datos. Su función es aplicar un formato determinado a los datos para mostrarlos por medio de un diseño atractivo y que sea fácil de interpretar por los administradores.

En la siguiente figura se muestran las principales clases del dominio del módulo y las relaciones existentes entre ellas.

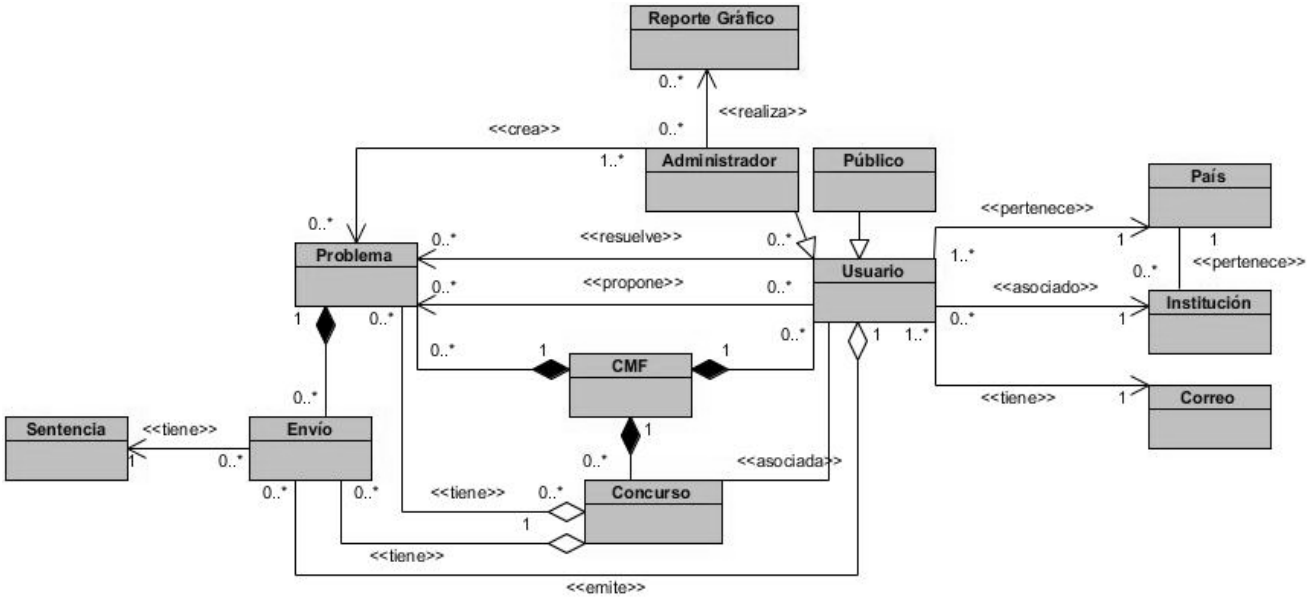


Figura 4 Modelo de Dominio

2.3 Descripción del sistema

Se identificaron luego de analizar la información almacenada en la aplicación, áreas de interés para los administradores como usuarios, envíos, problemas y competencias. Para su procesamiento se propone implementar un módulo que ayude a los administradores a tomar decisiones en el sistema, este se integra al juez en línea CMF. La solución descrita presenta información objetiva de las áreas definidas permitiendo llevar de manera organizada los datos y así poder ayudar en la toma de decisiones. Para ello se hace uso de una de las técnicas de representación de la información: representación gráfica (17). Esta técnica permite la visualización y resumen de los datos originados a partir del estudio de la información almacenada en CMF y la fácil comprensión del fenómeno estudiado. Los datos son resumidos de forma gráfica y numérica mediante barras, puntos, tablas y áreas.

Los usuarios con rol administrador pueden seleccionar en el módulo la funcionalidad deseada para analizar información determinada del sistema y así tomar decisiones.

Seguidamente se realiza una petición al servidor donde se hace una consulta a la base de datos con los parámetros pertinentes a la funcionalidad seleccionada. Esto devuelve como resultado una lista de datos, los cuales son procesados por el servidor, enviados al cliente y representados por la librería gráfica FusionCharts.

Figura 5 Descripción de la solución

2.4 Especificación de Requisitos

2.4.1 Requisitos funcionales

Los requisitos funcionales (RF) de un software definen lo que el sistema es capaz de realizar, funcionalidades requeridas y restricciones que son aprobadas en mutuo acuerdo con el usuario final, describen las transformaciones que el sistema realiza sobre las entradas para producir salidas. Estos requisitos al tiempo que avanza el proyecto de software se convierten en los algoritmos, la lógica y gran parte del código del sistema (43).

A continuación, se muestran las funcionalidades que desarrolla el presente módulo:

- **RF_1 - Mostrar distribución de registros de usuarios.**
  El sistema debe mostrar los nuevos usuarios registrados por fechas en una línea de tiempo.
- **RF_2 - Mostrar cuentas de usuarios activos e inactivos en el sistema.**
  El sistema debe mostrar una gráfica con la cantidad de cuentas de usuarios activos e inactivos del sistema, un usuario se considera inactivo cuando no haya accedido al sistema en al menos un mes atrás.
- **RF_3 - Mostrar distribución de envíos.**
  El sistema debe mostrar los envíos totales por fechas.
- **RF_4 - Mostrar distribución de veredictos.**

El sistema debe mostrar una gráfica por fecha de los dos estados posibles de los envíos [Aceptado, Denegado].

- **RF_5 - Mostrar distribución de problemas por período de tiempo.**
  El sistema debe mostrar la distribución de publicación de problemas, incluyendo los que están en cola, próximos a publicar automáticamente por el sistema.
- **RF_6 - Mostrar total de participantes por modalidad en eventos colaterales.**
  El sistema debe mostrar una gráfica con el total de participantes en los eventos colaterales en cada modalidad.
- **RF_7 - Mostrar distribución de envíos en concurso.**
  El sistema debe mostrar los envíos de un concurso según el estado [Aceptado, Denegado].
- **RF_8 - Comparar usuarios en cuanto a veredictos.**
  El sistema debe mostrar una gráfica de [Aceptados vs Denegados] de los dos usuarios.
- **RF_9 – Mostrar total de participantes en los eventos colaterales.**
  El sistema debe mostrar una gráfica con el total de participantes en los eventos colaterales por facultad.
- **RF_10 - Mostrar los momentos de la semana más enviados.**
  El sistema debe mostrar una gráfica donde se evidencie los momentos más enviados.
- **RF_11 - Mostrar los momentos de la semana más visitados.**
  El sistema debe mostrar una gráfica donde se evidencie los momentos más visitados.
- **RF_12 - Mostrar la distribución por género de los usuarios.**
  El sistema debe mostrar una gráfica con la cantidad de usuarios de cada género [Masculino, Femenino].
- **RF_13 - Mostrar los envíos por año.**
  El sistema debe permitir mostrar una gráfica de los envíos por año.
- **RF_14 - Mostrar los envíos de los últimos siete días.**
  El sistema debe mostrar una gráfica con los envíos realizados en los últimos siete días.
- **RF_15 - Mostrar los envíos de los últimos 30 días.**
  El sistema debe mostrar una gráfica con los envíos realizados en los últimos 30 días.
- **RF_16 - Exportar las gráficas a PNG, JPG, PDF y SVG.**
  El sistema debe exportar las gráficas a los formatos PNG, JPG, PDF y SVG.
- **RF_17 - Enviar por correo las estadísticas.**

El sistema debe enviar un correo a los administradores de forma automática con las estadísticas de las áreas relevantes.

- **RF_18 - Mostrar los diez países con más miembros activos.**
  El sistema debe permitir mostrar una gráfica con los diez países con más usuarios activos.
- **RF_19 - Mostrar los diez términos más buscados de problemas.**
  El sistema debe mostrar una gráfica con los diez términos que han sido más buscado en el listado de los problemas.
- **RF_20 - Mostrar las diez tablas con más peso.**
  El sistema debe mostrar una gráfica con las diez tablas de la base de datos que más peso tienen.

2.4.2 Requisitos no funcionales

Los requisitos no funcionales (RNF) son cualidades que el producto debe cumplir, características que hacen al producto usable, rápido, confiable y son de vital importancia para una puesta en marcha exitosa del software, además de lograr que este responda a las expectativas del usuario (43).

- **RNF_1 - Requerimientos de software**
  Cliente: Al presentar una interfaz construida a base de estándares como HTML, pueden ser ejecutadas desde cualquier plataforma, con el único requisito de contar con un navegador que soporte dichos estándares. Esto actualmente no es una limitante en ningún sistema operativo con interfaz gráfica.
  Servidor: En el caso del servidor de aplicaciones web se debe tener instalado cualquier versión igual o superior al Apache Tomcat 7.0.54 y en el servidor de base de datos se debe tener instalado PostgreSQL 8.4.
- **RNF_2 - Requerimientos de hardware**
  El servidor debe tener al menos 4GB de RAM, un procesador con velocidad igual o mayor a 3.0 GHz y una tarjeta de red, además de tener 40 GB mínimo de capacidad de disco duro. Además, en el cliente se debe tener un navegador web y conexión a internet.
- **RNF_3 - Requerimientos de seguridad**
  La seguridad se garantiza mediante la gestión de roles con el fin de mantener la integridad de los datos, por lo cual la información es accedida por los roles que tengan permisos para hacerlo.
- **RNF_4 - Requerimientos de usabilidad**

Para un mejor entendimiento del funcionamiento de la aplicación, en cada página aparece una descripción en la parte superior indicando de forma explícita el contenido de la misma.

- **RNF_5 - Requerimientos de apariencia o interfaz externa**
- El módulo proporciona claridad y una correcta organización de la información para los usuarios, que permite la interpretación de la misma. La interfaz gráfica tiene un diseño sencillo que permite realizar cualquier tipo de interacción con el sistema de manera fácil. Esto se garantiza con el uso de una plantilla común para todas las interfaces en el panel de administración. Cada representación gráfica del módulo consta de una leyenda permitiendo su debida comprensión y entendimiento.
- **RNF_6 - Requerimientos de rendimiento**

  El rendimiento del producto está dado en gran medida por el aprovechamiento de los recursos en una arquitectura distribuida. El tiempo en que el módulo tarda en responder a las peticiones no debe ser prolongado. El tiempo de respuesta a las peticiones no debe exceder los 30 segundos.
- **RNF_7 - Requerimientos de portabilidad**

  La solución al ser desarrollada sobre la plataforma Java la convierte en multiplataforma. Además, puede ser usada bajo cualquier distribución de Linux, Windows o cualquier otro sistema operativo en el que se pueda instalar la versión 1.7.60 de la máquina virtual de Java *o JVM (Java Virtual Machine).*

2.5 Historias de Usuario (HU)

Las Historias de Usuario representan la técnica utilizada para especificar los requisitos del software, realizándose una por cada característica principal del sistema; tienen el mismo propósito que los casos de uso en las metodologías de desarrollo de *software* pesadas, aunque no son lo mismo ya que son escritas por los propios clientes desde su perspectiva del sistema, por lo que son descripciones cortas y escritas en el lenguaje del usuario.

A continuación, se muestran las historias de usuario definidas para la elaboración del módulo. En el anexo 1 se encuentra el resto.

Número: 1	**Nombre del requisito:** Mostrar distribución de registros de usuarios.
Programador: Yanet	**Iteración Asignada:** 1

Prioridad: Alta	**Tiempo Estimado:** 3
Riesgo en Desarrollo: Medio	**Tiempo Real:** 3
Descripción: Inicialmente al seleccionar esta funcionalidad se muestra la distribución de registros de usuarios desde la creación de CMF hasta el momento por años mediante un gráfico de área. El administrador tiene la posibilidad de: - Filtrar por año, mostrando así la información por meses de ese año seleccionado. - Filtrar por mes del año ya seleccionado anteriormente revelando la información por días de ese mes.	
Observaciones: La distribución de registros en los últimos siete días se muestra al filtrar por el mes actual.	
Prototipo de interfaz: 	

Tabla 2 HU: Mostrar distribución de registros de usuarios

Número: 2	**Nombre del requisito:** Mostrar cuentas de usuarios activos e inactivos en el sistema
Programador: Yanet	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 3
Riesgo en Desarrollo: Media	**Tiempo Real:** 3
Descripción: Muestra mediante un gráfico de barras las cuentas de los usuarios activos e inactivos en el sistema, dividido por: - Cuentas que nunca fueron activadas. - Cuentas inactivas en el último año. - Cuentas inactivas en el último mes. - Cuentas activadas.	

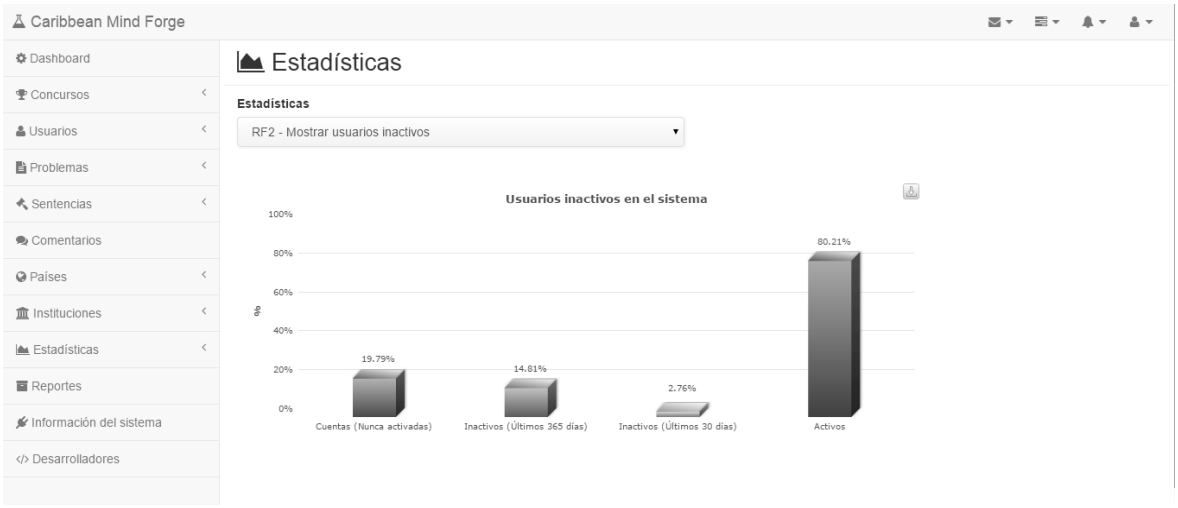
Observaciones: Un usuario se considera inactivo cuando no haya accedido al sistema en al menos un mes atrás.
Prototipo de interfaz: 

Tabla 3 HU: Mostrar usuarios inactivos en el sistema

Número: 3	**Nombre del requisito:** Mostrar los momentos de la semana más enviados
Programador: Yanet	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 5
Riesgo en Desarrollo: Alto	**Tiempo Real:** 5
Descripción: Se muestra una gráfica denominada mapa de calor donde se distribuye por días de la semana y horas del día los momentos más enviados de la semana.	
Observaciones: Cada color representa una intensidad, las mismas son: - Verde: Bajo. - Amarillo: Medio. - Naranja: Medio alto. - Rojo: Alto.	
Prototipo de interfaz:	

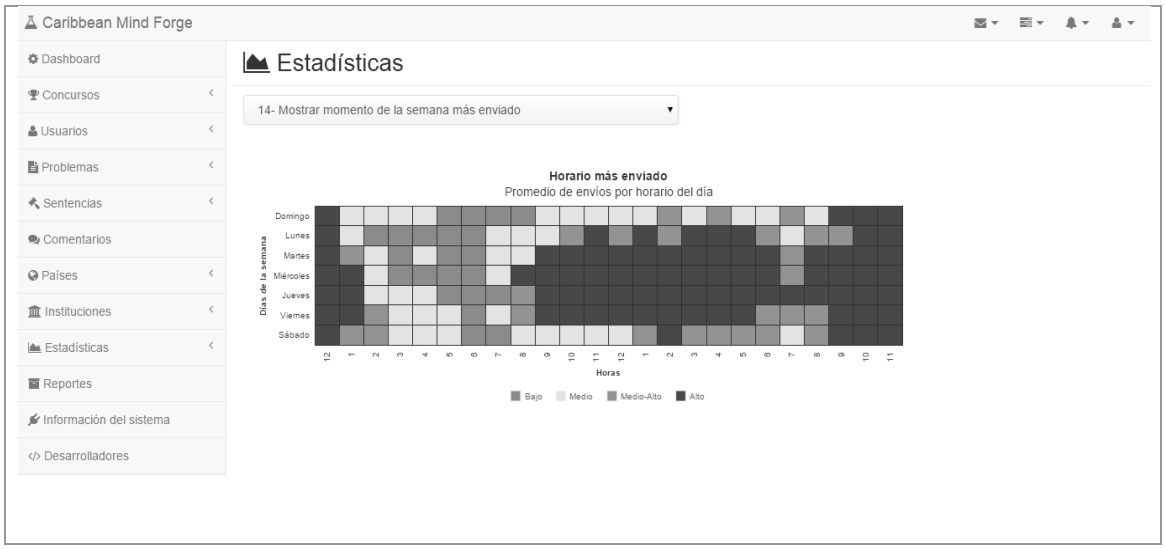


Tabla 4 HU: Mostrar los momentos de la semana más enviados

Número: 4	**Nombre del requisito:** Mostrar distribución de envíos
Programador: Yanet	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:**4
Riesgo en Desarrollo: Medio	**Tiempo Real:** 4
Descripción: Inicialmente muestra el total de envíos a CMF desde su creación hasta el momento por años usando la gráfica de barras. El administrador tiene la posibilidad de: - Filtrar por año, mostrando así la información por meses de ese año seleccionado. - Filtrar por mes del año ya seleccionado anteriormente revelando la información por días de ese mes.	
Observaciones: La distribución de envíos en los últimos siete días se muestra al filtrar por el mes actual.	
Prototipo de interfaz:	

Tabla 5 HU: Mostrar distribución de envíos

Número: 5	**Nombre del requisito:** Mostrar la distribución por género de los usuarios
Programador: Jessica	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 1
Riesgo en Desarrollo: Bajo	**Tiempo Real:** 1
Descripción: Muestra mediante un gráfico de pastel la distribución por género (Femenino y Masculino) de los usuarios.	
Observaciones:	
Prototipo de interfaz:	

Tabla 6 HU: Mostrar la distribución por género de los usuarios

2.6 Arquitectura del sistema

La arquitectura del módulo está en correspondencia con la propuesta por Grails. La misma está fundamentada por la arquitectura N-Capas, basada en el patrón MVC2, el cual contiene como capas lógicas principales: Capa de Presentación o Vistas, Capa de Control o Controladores y Capa de Datos o Modelo, en el caso particular de Grails se añade una Capa de Servicios para la lógica de negocio (44).

Este patrón define que los elementos de una aplicación deben dividirse en tres capas según la responsabilidad de cada uno, estas capas son las siguientes:

- **Dominio o modelo**: el modelo es el responsable de mantener el estado de la aplicación almacenándose normalmente en una base de datos, es el objeto que representa los datos del programa y contiene los componentes que representan y gestionan estos datos, los cuales son manejados por la aplicación (33).
- **Presentación o vista**: las vistas por defecto en Grails son las GSP (*Groovy Server Pages*) y habitualmente nos muestra el contenido en formato HTML, estas son las responsables de generar la interfaz de usuario, normalmente basada en los datos del modelo. Aunque la vista muestre los datos del modelo, nunca manipulará tales datos (33).
- **Control o controlador**: los controladores por su parte, permiten gestionar las peticiones a la aplicación y organizar los servicios proporcionados. Recibe las peticiones o eventos desde el exterior, interactuando después con el modelo y mostrando la vista apropiada al usuario (33).

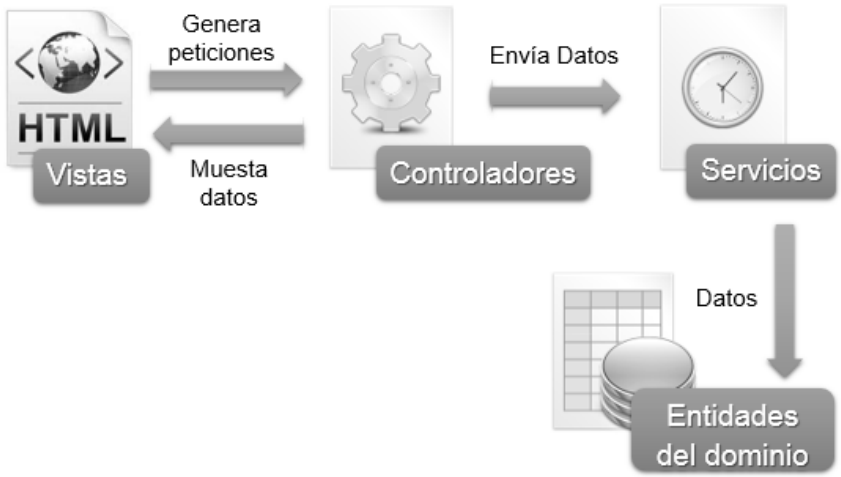


Figura 6 Relación que existe entre las capas en la arquitectura propuesta

2.7 Patrones de diseño

Un patrón es un par problema/solución con nombre que se puede aplicar en nuevos contextos, con consejos acerca de cómo aplicarlo en nuevas situaciones y discusiones

sobre sus compromisos. Un patrón es una descripción de un problema bien conocido que suele incluir: descripción, escenario de uso, solución concreta, las consecuencias de utilizar este patrón, ejemplos de implementación y lista de patrones relacionados (45).

2.7.1 Patrones para Asignar Responsabilidades (GRASP)

Los patrones GRASP (*General Responsibility Assignment Software Patterns* / Patrones Generales de Software para Asignación de Responsabilidades) constituyen un apoyo para ayudar a entender el diseño de objetos, y aplicar el razonamiento para el diseño de una forma sistemática, racional y explicable. Este enfoque para la comprensión y utilización de los principios de diseño se basa en los patrones de asignación de responsabilidades. Estos patrones describen los elementos fundamentales del diseño de objetos y la asignación de responsabilidades, expresados como patrones (45). Los comprendidos en la solución se muestran a continuación:

- **Alta cohesión:** la cohesión es una medida de la fuerza con la que se relacionan las responsabilidades de un elemento, un elemento con responsabilidades altamente relacionadas y que no hace una gran cantidad de trabajo tiene alta cohesión. Define que la información que almacena una clase debe de ser coherente y debe estar (en la medida de lo posible) relacionada con la clase. Asigna una responsabilidad de forma tal que la cohesión siga siendo alta (45).

  Este patrón se pone de manifiesto en las clases de dominio, ya que estas contienen solamente la información que se relaciona con ellas. Un ejemplo es la clase Usuario en la cual los atributos representan en su conjunto la información de un objeto de tipo Usuario de forma clara.

```
class Usuario {

    transient springSecurityService
    String username
    String password
    boolean enabled = true
    boolean accountExpired = false
    boolean accountLocked = false
    boolean passwordExpired = false
    String responsability = 'Usuario'
    Date dateRegister = new Date()
    Date lastLogin = new Date(0000,00,00)
    String correo
    String nombres
    String apellidos
    byte[] foto
    Boolean terminos
    String nick
    boolean genero
    Country country
    Institution inst
    boolean recibirNotificaciones
    String idioma

    static constraints = {...}

    static mapping = {
        password column: '`password`'
    }
```

Figura 7 Clase de dominio Usuario donde se puede apreciar la alta cohesión

- **Bajo acoplamiento:** el acoplamiento es una medida de la fuerza con que un elemento está conectado, tiene conocimiento o confía en otros elementos. Estos elementos pueden ser clases, subsistemas, sistemas, entre otros. Además se deben asignar responsabilidades para que el acoplamiento permanezca bajo (45).

  El Bajo acoplamiento es utilizado en todo el sistema debido a que el mismo aplica la arquitectura N-Capas, basada en el patrón MVC2 permitiendo que la interacción entre las clases sea mínima, de forma tal que las cuestiones de presentación, negocio y acceso a datos están totalmente desligadas, propiciando que un cambio en una capa afecte lo menos posible a las demás capas inferiores.

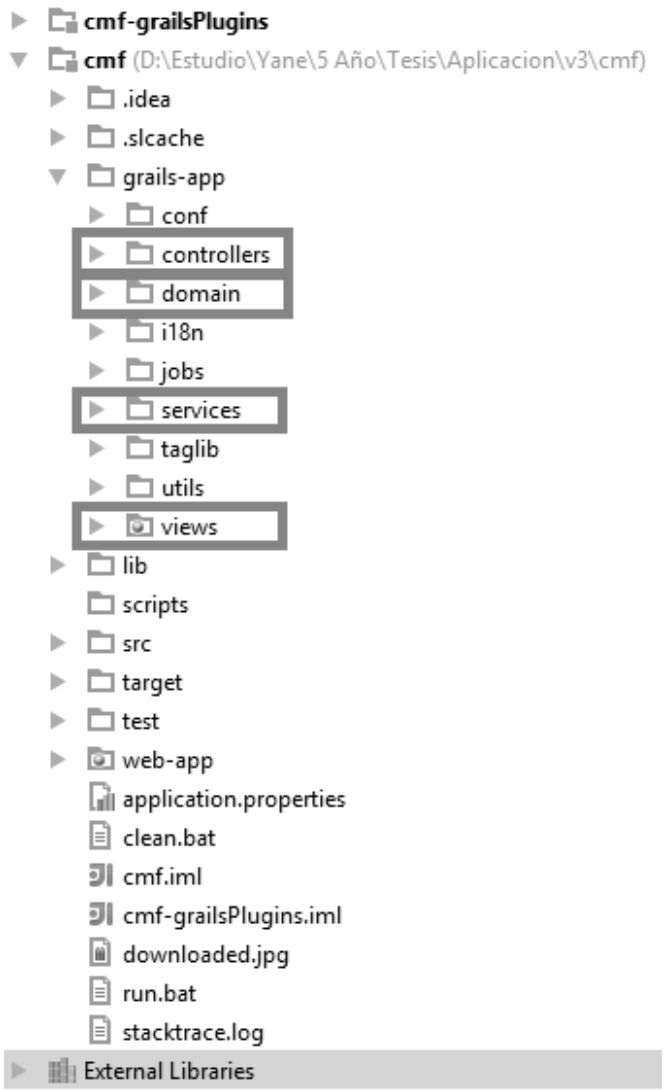


Figura 8 Distribución de las capas Controllers-Views-Domain

- **Experto:** Asigna una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad (45).

  Este es uno de los patrones que implementa el *framework* Grails al utilizar *GORM* (*Grails Object Relational Mapper* / Mapeador Objeto Relacional de Grails) para el mapeo de objetos de la base de datos. Al generar las clases para la gestión de las entidades según las tablas de la base de datos con las responsabilidades debidamente asignadas, pues cada una de estas clases cuenta con un conjunto de funcionalidades relacionadas directamente con la entidad que representan.
- **Controlador:** este patrón describe alternativas comunes en el manejo del lado del cliente de las peticiones de las operaciones del sistema que se emiten desde la capa de presentación. Un controlador es un objeto responsable de recibir o manejar un evento del sistema, definiendo además el método para la operación

de dicho sistema (45). El uso del patrón Controlador dentro del sistema se evidencia en la implementación de las clases controladoras, estas son las encargadas de gestionar la aplicación de la lógica de negocio luego de recibir las peticiones del usuario.

```
@Secured(['ROLE_ADMIN'])
class EstadisticaController {

    def visitaService

    def index(){}

    def empty(){...}

    /**...*/
    def distribucionNuevosRegistros(){
        def fecha = new Date()
        def annoPivote = 2011
        def data = []
        int totalUsuarios = Usuario.count()
        def meses = [message(code: 'enero'), message(code: 'febrero'), message(code: 'marzo'), message(code: 'abril'),
                     message(code: 'mayo'), message(code: 'junio'), message(code: 'julio'), message(code: 'agosto'),
                     message(code: 'septiembre'), message(code: 'octubre'), message(code: 'noviembre'), message(code: 'diciembre')]
        def diasMeses = [31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]
        float media = 0.0
        String caption = ""
        String xaxisname = ""
        /*
        * Para cuando la distribución por años
        * */
        if(!params.anno){
            int annos = ((fecha.getYear()+1900 - annoPivote)+5)
            annos.times {
                def elemento = [:]
                def enero = new Date(annoPivote-1900, 0, 1, 0, 0, 0)
                def diciembre = new Date(annoPivote-1900, 11, 31, 0, 0, 0)
                elemento.value = new DecimalFormat("##.##").format(Usuario.countByDateRegisterBetween(enero, diciembre))+""
                elemento.label = annoPivote + ""
```

Figura 9 Clase EstadisticaController donde se evidencia el patrón controlador

2.7.2 Patrones GoF (Gang of Four)

Dentro de los patrones de diseño se encuentran Gong of Four (“Pandilla de los Cuatro” en español). Estos definen el comportamiento entre las clases y los objetos. Tratan la relación entre clases, la combinación clases y la formación de estructuras de mayor complejidad. Además permiten crear grupos de objetos para ayudarnos a realizar tareas complejas (45). En el desarrollo del módulo se utilizaron los siguientes patrones GoF.

- **Composición de Vistas (*Composite View Pattern*)**: Consiste en la creación de páginas con estructuras similares en las cuales cada sección de la página varía en situaciones diferentes. Dicho patrón permite la confección de diseños diferentes para cada módulo del sistema reutilizando componentes comunes (45).

  En views/layouts se definen las plantillas a utilizar por las diferentes vistas en el sistema y en el fichero index.gsp en el directorio views/estadística se incluye una

plantilla con el uso de la etiqueta <meta/>. Un ejemplo del código de definición de una plantilla en una vista sería el siguiente:

```
<html>
<head>
    <meta name="layout" content="sb-admin"/>
    <style...>
</head>

<body>
<div id="header" class="row">
    <div class="col-lg-6">
        <h1 class="page-header"><i class="fa fa-area-chart"></i> ${message(code: 'view.acceso.estadisticas')}</h1>
    </div>
```

Figura 10 Definición de una plantilla en una vista

Inversión de Control (IoC): la inversión de control es otro patrón utilizado por el *framework* Grails, donde las dependencias de un componente no deben gestionarse desde el propio componente para que este solo tenga la lógica necesaria para hacer su trabajo (33).

Grails implementa este patrón por defecto para la capa de los servicios, mediante la implementación de un contenedor que se encarga de administrar el ciclo de vida de las instancias. Para utilizarlo solo es necesario declarar las variables de tipo def y con el nombre exactamente igual al del servicio (como se muestra en la Figura 11) y Grails se encarga de hacer el resto.

```
def pdfService
```

Figura 11 Empleo del patrón Inversión de Control

2.8 Base de datos

Grails cuenta con la integración del *GORM* para la generación de la base de datos. Esto les permite a los desarrolladores abstraerse del diseño de la base de datos debido a que los ORM implementan el modelo de objetos a partir de las clases de dominio, las mismas representan los datos persistentes del sistema que deben ser mapeados con una tabla en la base de datos.

Seguidamente, se describe la forma de relacionar las clases manejadas durante el desarrollo del módulo.

- Relación de uno a muchos

Para lograr el mapeo con la base de datos de dos clases de modelo que presenten entre sí este tipo de relación, Grails lo soluciona a través del atributo *hasMany* donde se especifica en la clase de dominio que contiene al otro atributo. A continuación, se

muestra en la clase PreguntaDia a través de este atributo la relación existente con la clase PreguntaRespuesta.

```
package mindpro.domain.gestion.problemas

import ...

class PreguntaDia implements Comparable {

    String titulo
    String title
    String texto
    String respuesta
    Usuario creador
    byte[] imagen
    double puntRelativa
    boolean jurado
    String nota
    ...
    Integer limite
    Integer usuarios
    Double porcientoPromedio

    static hasMany = [respuestas: PreguntaRespuesta]
    static belongsTo = [volumen: Volumen]
```

Figura 12 Relación uno a muchos entre las clases de dominio

Conclusiones del Capítulo

En el capítulo se describió la arquitectura del sistema, esta propuesta se basó en las características no funcionales requeridas. Fueron identificadas las principales funcionalidades a desarrollar, las cuales quedaron plasmadas en las historias de usuario. Además, se explicaron los patrones de diseño utilizados y se concluye que los artefactos generados por la metodología en esta etapa permiten tener una visión más clara de los elementos del sistema, permitiendo una mayor comprensión del mismo guiando de forma efectiva el capítulo de Implementación y Prueba.

CAPÍTULO 3. Implementación y Pruebas

3.1 Introducción

En este capítulo se abordan dos de las fases más importantes en el ciclo de vida de cualquier *software*: Implementación y Pruebas, en esta fase se conforma el *software* mediante su codificación y se realizan las pruebas necesarias para evidenciar si fueron cumplidos los objetivos trazados durante la concepción del módulo.

3.2 Implementación

A la hora de programar se debe tratar que el código no resulte indescifrable para que pueda ser reutilizado, elegir cuidadosamente los nombres de las variables, seleccionar estructuras de datos adecuadas para el problema en cuestión, mantener la lógica de la aplicación lo más sencilla posible, comentar adecuadamente el código y, por último, facilitar la interpretación visual del sistema. Antes de escribir una sola línea de código es fundamental haber comprendido bien el problema que se pretende resolver y haber aplicado principios básicos de diseño que nos permitan construir un sistema de información de calidad (46).

3.2.1 Implementación de las clases controladoras

Las clases controladoras se encuentran ubicadas en la carpeta grails-app/controllers. Estas son las responsables de recibir las solicitudes del usuario, aplicar la lógica de negocio sobre el modelo y decidir la vista que se debe mostrar (33). A continuación, se muestra la clase controladora PreguntaDiaController.

```
package mindpro.controller.gestion

import ...

class PreguntaDiaController {

    static allowedMethods = [save: "POST", update: "POST"]

    def mindProService
    def preguntaDiaService

    def index() {redirect(action: "list", params: params)}

    @Secured(['ROLE_ADMIN'])
    def list() {...}

    @Secured(['ROLE_ADMIN'])
    def create() {...}

    @Secured(['ROLE_ADMIN'])
    def save() {...}

    @Secured(['ROLE_ADMIN'])
    def edit() {...}

    @Secured(['ROLE_ADMIN'])
    def update() {...}

    @Secured(['ROLE_ADMIN'])
    def delete() {...}
```

Figura 13 Implementación de la clase controladora PreguntaDiaController

Esta clase controladora cuenta con métodos para efectuar los procedimientos sobre PreguntaDiaController, las cuales serían: listar, crear, guardar, mostrar, modificar y actualizar.

3.2.2 Implementación de las clases del dominio

Las clases del dominio se encuentran ubicadas en la aplicación en la carpeta grails-app/domain, formando el modelo de dominio de la aplicación. Un modelo del dominio se utiliza para el diseño de los objetos del software y muestra clases conceptuales significativas en un dominio del problema (33). A continuación, se muestra la clase de dominio EstadisticaSentencia:

```
package mindpro.domain.gestion.problemas

class EstadisticaSentencia {

    int anno
    long envios
    long aceptados

    static constraints = {

    }

    static mapping = {
        id generator:'sequence', params:[sequence:'estadistica_sentencia_sequence']
    }
}
```

Figura 14 Implementación de la clase de dominio EstadísticaSentencia

3.2.3 Implementación de la internacionalización

Los ficheros de internacionalización se encuentran ubicados en la carpeta grails-app/i18n. Internacionalizar una aplicación significa diseñarla de forma que la interfaz de usuario y los mensajes que se muestran en la aplicación aparezcan en diferentes idiomas sin que sea necesario modificar el código fuente (33). A continuación, se muestra un fragmento del fichero de internacionalización perteneciente al módulo.

```
# Módulo
estadisticas.sentencias=Sentences statistics
estadisticas.usuarios=Users statistics
usuarios.por.sexo=Users by gender
cargando.grafica=Loading chart, please wait
total.registros=Total records
annos=Years
meses.anno=Months of the year
dias.mes=Days of the month
distribucion.registros.usuarios=Distribution of user records
distribucion.registros.usuarios.anno=Distribution of user records in {0}
distribucion.registros.usuarios.anno.mes=Distribution of user records in {0}/{1}
distribucion.envios.usuarios=Distribution of user submissions
distribucion.envios.usuarios.anno=Distribution of user submissions in {0}
distribucion.envios.usuarios.anno.mes=Distribution of user submissions in {0}/{1}
total.envios=Total submissions
inactivos.ultimo.anno=Idle (Last 365 days)
inactivos.ultimo.mes=Idle (Last 30 days)
```

Figura 15 Implementación del fichero de Internacionalización

3.2.4 Implementación de la clase de servicio

Dentro de la aplicación, las clases servicios se encuentran ubicadas en la carpeta grails-app/servicio, estas clases son utilizadas por las controladoras en la gestión de la lógica de la aplicación. En el caso del módulo se hace uso de la clase de servicio PdfController, a continuación, se muestra un fragmento de código.

```
package mindpro.controller

import ...

class PdfController {

    def pdfService
    def mindProService
    def mensajeService
    def pdfRenderingService

    def index() {redirect(action: "demo")}

    def demo() {...}

    def volumen() {[volumen: Volumen.get(params.id)]}

    def email() {
        if (!params.id) {
            flash.message = "No se ha podido guardar el mensaje."
            redirect controller: "mensaje", action: "bandejaEntrada"
            return
        }
        [mensaje: Mensaje.get(params.id)]
    }

    def pdfLink() {...}
```

Figura 16 Implementación de la clase de servicio PdfController

3.2.5 Paquete Vistas

Grails incluye librerías de etiquetas (*tags* de presentación) que permiten una mejor definición de la lógica de presentación. Además, brinda la posibilidad al desarrollador de crear este tipo de etiquetas para mostrar la sección de la vista que cumpla con la

condición. En la implementación de las vistas se hizo uso de etiquetas iterativas y para formularios, estas las incorpora Grails.

Etiquetas iterativas

Estas etiquetas permiten la inclusión opcional de código en las vistas. La siguiente figura muestra un fragmento de la implementación de colateralesModalidad.gsp donde se evidencia el uso de la etiqueta <g:each></g:each>.

```
<div class="row">
    <div class="col-lg-12">
        <div class="row">
            <div class="col-lg-4">
                <select name="anno" id="anno" value="${anno}" class="form-control" onchange="load('${createLink(action: "colateralesPorModalidad")}?anno=' + this.value)">
                    <option value="0">-Seleccione el año-</option>
                    <g:each in="${2011..2016}" status="i" var="obj">
                        <option value="${obj}" ${obj + "" == anno ? "selected='selected'" : ''}>${obj}</option>
                    </g:each>
                </select>
            </div>
```

Figura 17 Uso de la etiqueta g:each></g:each>

Etiquetas para formularios

Estas etiquetas permiten agrupar los campos y enviar los datos contenidos en éstos a los controladores. La siguiente figura muestra un fragmento de la implementación de la vista distribucionEnvios.gsp donde se crea una lista desplegable haciendo uso de la etiqueta <g:select/>.

```
<div class="row">
    <div class="col-lg-6">
        <div class="row">
            <div class="col-lg-6">
                <g:select name="anno" from="${2011..2016}" value="${anno}" class="form-control" onchange="loadMes(true, this.value)"
                          noSelection="['':'-Seleccione el año-']" id="anno"/>
            </div>
            <div class="col-lg-6">
```

Figura 18 Uso de la etiqueta g:select/>

3.2.6 Paquete Controladores

En los controladores las operaciones se definen como acciones que se ejecutan como parte de la clase. En estas se puede realizar llamadas al método *render* para enviar respuestas al cliente. Durante el desarrollo del módulo, los parámetros de este método que se utilizaron fueron:

- ***Template*:** Para mostrar una plantilla al usuario.
- ***Model*:** Un mapa con el modelo para usar en la vista (33).

La siguiente figura muestra un fragmento de la implementación de la clase EstadisticaController.groovy que evidencia el uso de estos parámetros.

```
if (!params.click) {
    render(template: "compararUsuariosVeredictos", model: [aceptados: aceptados, denegados: denegados])
    return
```

Figura 19 Uso de los parámetros *template* y *model*

3.2.7 Paquete Dominios

Operaciones sobre el modelo de datos

Para la manipulación de las entidades de la aplicación a continuación se explica el método usado y se muestra un ejemplo de su utilización:

- **save():** Sirve para insertar el registro en la base de datos o para actualizarlo si ya existía (33). A continuación, se muestra un fragmento de la implementación donde se hace uso del método *save* para actualizar el valor de la variable pivote en la clase controladora EstadisticaController.groovy.

```
MindPro.withTransaction {
    pivote?.valor = id
    pivote?.save()
}
```

Figura 20 Uso del método *save* en EstadisticaController.groovy

Dynamic Finders ("localizadores dinámicos")

Estos buscadores son muy potentes para realizar consultas básicas, pero no permiten hacer búsquedas avanzadas. A continuación, se explican los métodos usados y se muestran ejemplos de su utilización (33):

- **findAll():** Devuelve la primera instancia que cumpla la primera condición de búsqueda.
- **findAllBy():** Devuelve una lista con todos los resultados que correspondan.
- **countBy():** Devuelve el total de resultados que correspondan.

A continuación, se muestra algunos fragmentos de implementación donde se evidencia el uso de los métodos anteriormente mencionados.

```
elemento = [:]
elemento.label = "Operaciones"
operaciones = Operaciones.findAllByNombreLike("%${params?.anno}%")
elemento.value = operaciones ? Resultado.countByCompetenciaInList(operaciones) : 0
data << elemento
```

Figura 21 Uso de findAllBy() en la acción colateralesPorModalidad

```
def tareaMomentoMasVisitado(){
    def soluciones = []
    def matriz = []
    def pivote = MindPro.findByDescripcion("PIVOTE_ID_VISITA")
    if(!pivote?.valor){...}else{...}

    /*Actualizar la matriz*/
    def id = 0
    soluciones.each {...}
    println matriz
    String lines = ""

    def dataChart = []
        matriz?.each {...}
    MindPro.withTransaction {...}

    new File("matrizvisits.data").write(new String(lines?.substring(0, lines?.length() - 1)?.getBytes("UTF-8"), "UTF-8"))
    saveDataChartVisitados(dataChart)
}

private void saveDataChart(def data){
    long size = PreguntaRespuesta.countByIdLessThan(MindPro.findByDescripcion("PIVOTE_ID_PREGUNTA_RESPUESTA")?.valor)
    println "saveDataChart: " + data
    def daysOfWeek = ["sun", "mon", "tue", "wed", "thu", "fri", "sat"]
    String lines = ""
    int index = 0
    for(int i = 0; i < 7; i++){
        for (int j = 0; j < 24; j++){
            int e = data[index++]
            lines += daysOfWeek[i] + "," + j + "," + e + "," + state(e, size) + "\n"
        }
```

Figura 22 Uso de findBy() y countBy() en los métodos tareaMomentoMasVisitado() y saveDataChart()

3.2.8 Diagrama de componentes

El diagrama de componentes muestra la relación entre componentes de software, sus dependencias, su comunicación su ubicación y otras condiciones. Normalmente los diagramas de componentes contienen interfaces, relaciones de dependencia, instancias de algunas clases y paquetes o subsistemas. Se centra en la modelación de los componentes físicos del sistema. También se pueden utilizar para modelar la gestión de la configuración de los archivos de código fuente, tomando como productos de trabajo precisamente estos ficheros (47).

Seguidamente se muestra el diagrama de componentes perteneciente a la representación gráfica de la información, específicamente de la funcionalidad **Mostrar distribución de nuevos registros.** La clase EstadisticaController.groovy, es donde se maneja la lógica de la funcionalidad analizada y se encuentra en el paquete Controladoras. Además, hace uso de las clases Estadistica.groovy y Usuario.groovy ubicadas en el paquete de Dominio. EstadisticaController.groovy atiende las peticiones que se generan de la vista, por lo que en el paquete correspondiente a esta última se encuentra todo lo relacionado con ella. Este paquete está dividido por GSP, CSS (Cascading Style Sheets) y JS y tienen la responsabilidad de generar la interfaz de usuario.

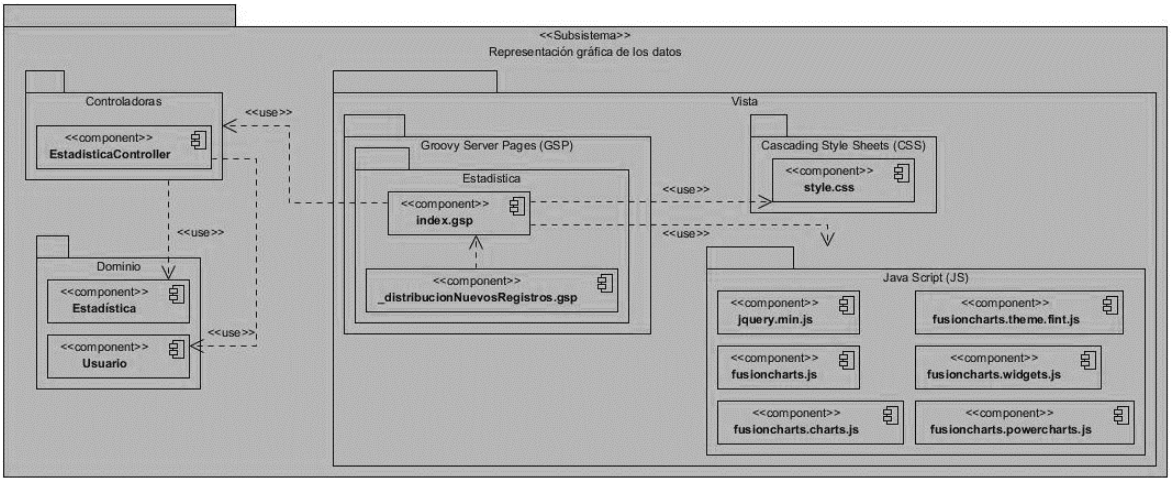


Figura 23 Diagrama de componentes

3.2.9 Diagrama de despliegue

Muestra las relaciones físicas de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos. Los diagramas de despliegue son los complementos de los diagramas de componentes que unidos proveen la vista de implementación del sistema. Describen la topología del sistema y la estructura de los elementos de hardware y el software que ejecuta cada uno de ellos. Los diagramas de despliegue representan a los nodos y sus relaciones. Los nodos son conectados por asociaciones de comunicación tales como enlaces de red, conexiones TCP/IP (48).

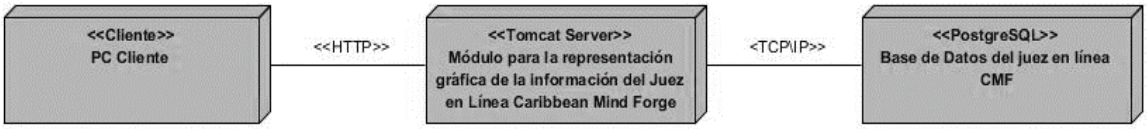


Figura 24 Diagrama de despliegue

3.3 Pruebas

Las pruebas de software son los procesos que permiten verificar y revelar la calidad de un producto de software. Son utilizadas para identificar posibles fallos de implementación, calidad, o usabilidad de un sistema informático. Básicamente es una fase en el desarrollo de software, consistente en probar las aplicaciones construidas. Para determinar el nivel de calidad se deben efectuar unas medidas o pruebas que permitan comprobar el grado de cumplimiento respecto a las especificaciones iniciales del sistema. Existen diferentes tipos y métodos de pruebas que juntos conforman la estrategia de prueba a seguir para lograr la calidad del producto final (49).

3.3.1 Estrategia de Prueba

La estrategia de prueba tiene como propósito fundamental describir el enfoque y los objetivos generales de las actividades de prueba. Con el objetivo de guiar el proceso de

pruebas y garantizar al máximo la eliminación de las no conformidades detectadas por el probador se hace necesario definir la estrategia de prueba. Este proceso consiste en la integración de diferentes factores y pasos que dan como resultado una correcta construcción del software. Incluye los niveles de prueba (desarrollador, unidad, integración, etc.) a ser diseccionados, el tipo de prueba a ser ejecutadas (funcional, stress, etc.) y los métodos de prueba a aplicar (caja negra o caja blanca) (49).

Posteriormente, se especifica y explica la estrategia seleccionada para las pruebas de del módulo (50).

- **Método de prueba:** método de caja negra.
- **Tipo de prueba:** funcionales y no funcionales.
- **Tipo de técnica:** casos de prueba.
- **Nivel de prueba:** pruebas de aceptación, pruebas de sistema.

3.3.2 Método de Prueba

Método de caja negra: Se realizan pruebas que verifican que cada función es operativa y que la entrada y salida se producen de forma correcta. Estas pruebas se llevan a cabo sobre la interfaz del software, obviando el comportamiento interno y la estructura del programa. Las pruebas de caja negra se centran principalmente en los requisitos funcionales del software y pretenden encontrar los siguientes tipos de errores (51):

- Funciones incorrectas o ausentes.
- Errores de interfaz
- Errores en estructuras de datos o en accesos a las Bases de Datos externas.
- Errores de rendimiento.

3.3.3 Tipo de Prueba

Funcionalidad: Tipo de prueba centrada en validar las funciones, ofreciendo los servicios, métodos o casos de usos requeridos. Este tipo de prueba se basa específicamente en la ejecución y revisión de las funcionalidades previamente diseñadas para el sistema, estas pruebas se hacen mediante el diseño de modelos de pruebas que buscan evaluar cada una de las opciones con las que cuenta el software. El factor fundamental que se tiene en cuenta es la capacidad de la aplicación para manejar grandes volúmenes de dato (51).

3.3.4 Tipo de Técnica

Casos de prueba: La generación de cada caso de prueba debe ir acompañada del resultado que ha de producir el software al ejecutar dicho caso para detectar un posible fallo en el programa. Los casos de prueba determinan un conjunto de entradas,

condiciones de ejecución y resultados esperados para un objetivo particular. Cada técnica de pruebas proporciona unos criterios distintos para generar estos casos o datos de prueba.

3.3.5 Nivel de Prueba

Pruebas de aceptación (PA): Son pruebas de caja negra definidas por el cliente para cada historia de usuario, y tienen como objetivo asegurar que las funcionalidades del sistema cumplen con lo que se espera de ellas. Una historia de usuario no se puede considerar terminada hasta tanto pase correctamente todas sus pruebas de aceptación. Por lo que a continuación se muestran una serie de casos de prueba, los cuales servirán como muestra visual del proceso de pruebas realizadas al módulo propuesto (51).

Los casos de pruebas se describen en tablas que contendrán los siguientes campos:

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

La evaluación de la prueba realizada se hará según el resultado de la misma, la tendrá uno de los tres resultados que a continuación se describen:

- **Satisfactorio:** Cuando el resultado de la prueba es exactamente el esperado por el usuario.
- **No Satisfactorio:** Cuando el resultado de la prueba realizada genera un error de codificación en la aplicación o muestra como resultado elementos no deseados o fuera de contexto, trayendo como consecuencia que la funcionalidad requerida por el cliente no tenga resultado, lo que invalida también la HU.

Como parte de la estrategia de prueba se diseñaron los casos de pruebas para cada funcionalidad, con el objetivo de probar que el sistema funciona correctamente. Para consultar los casos de pruebas de las funcionalidades, ver Anexo 2.

Pruebas de Rendimiento: Las pruebas de rendimiento se basan en comprobar que el sistema puede soportar el volumen de carga definido en la especificación, es decir, hay que comprobar su eficiencia. Son utilizadas para evaluar el cumplimiento por parte de un sistema o componente, de los requisitos de rendimiento especificados. Con ellas se puede determinar cómo responde un sistema ante una cierta carga, así como validar otros atributos relacionados con la calidad, como pueden ser la escalabilidad o el uso de recursos entre otros (50). Dentro de este tipo se encuentra las pruebas de carga, pruebas que serán aplicadas al módulo desarrollado mediante la herramienta JMeter.

- **Pruebas de carga**: Las pruebas de carga someten el sistema a cargas de trabajo extremas determinando la capacidad de resistencia límite del programa, se realiza generalmente para observar el comportamiento de una aplicación bajo una cantidad de peticiones esperada. Esta carga puede ser el número esperado de usuarios concurrentes utilizando la aplicación y que realizan un número específico de transacciones durante el tiempo que dura la carga (52).

3.3.6 Resultado de las pruebas

Para evaluar la solución desarrollada se realizaron pruebas de aceptación donde se aplicaron los niveles, método y tipos de prueba expuestos anteriormente. Al ser aplicada este tipo de prueba fueron obtenidas un conjunto de no conformidades donde son expuestas en la siguiente tabla.

No.	No conformidad	Ubicación	Estado
1	No se muestra los números de los días del mes.	Estadística/ Mostrar distribución de nuevos registros.	Resuelta
2	No se muestra los números de los días del mes.	Estadística/Mostrar distribución de envíos de usuarios.	Resuelta
3	Existen campos vacíos.	Estadística/Comparar problemas en cuanto a veredictos.	Resuelta
4	.No se encuentra el formato SVG	Estadística/ Exportar las gráficas a PNG, JPG, PDF, SVG.	Resuelta

5	No se muestra los números de los días del mes.	Estadística/ Mostrar distribución de registros de usuarios.	Resuelta
6	Existen campos vacíos.	Estadística/Mostrar total de participantes por facultades en eventos colaterales.	Resuelta
7	No está internacionalizada la gráfica.	Estadística/ Mostrar los momentos de la semana más visitados.	Resuelta
8	No está internacionalizada la gráfica.	Estadística/ Mostrar los momentos de la semana más enviados.	Resuelta
9	No se muestra correctamente los días de la semana	Estadística/ Mostrar distribución de problemas por período de tiempo.	Resuelta
10	No se muestra la gráfica	Estadística/ Mostrar las diez tablas con más peso.	Resuelta
11	No se encuentra el formato SVG	Estadística/..	Resuelta

Tabla 7 Resultados de las no conformidades en las pruebas

En estas pruebas se detectaron un total de 11 no conformidades. En una primera iteración se revelaron 7 que fueron solucionadas. En una segunda iteración se encontraron nuevas no conformidades, donde fueron resueltas. En la tercera y última iteración no se evidenciaron no conformidades.

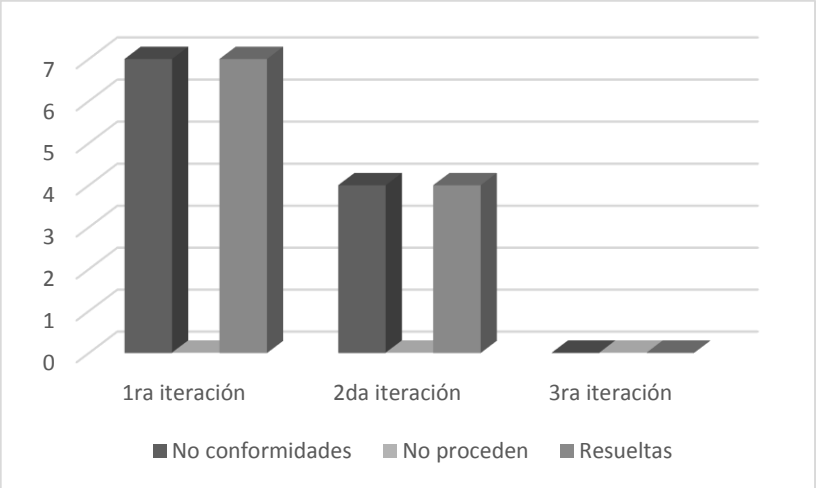


Figura 25 Resultados obtenidos en las pruebas de aceptación

Los resultados de la prueba de rendimiento se muestran en la siguiente figura, donde los tiempos de respuestas obtenidos como resultado para dos usuarios son expresados en milisegundos (ms), donde un segundo equivale a 1000 milisegundos. Como conclusión se observa que son tiempos de espera admisibles y que no llegan a los 30 segundos como se define anteriormente en los requerimientos de rendimiento.

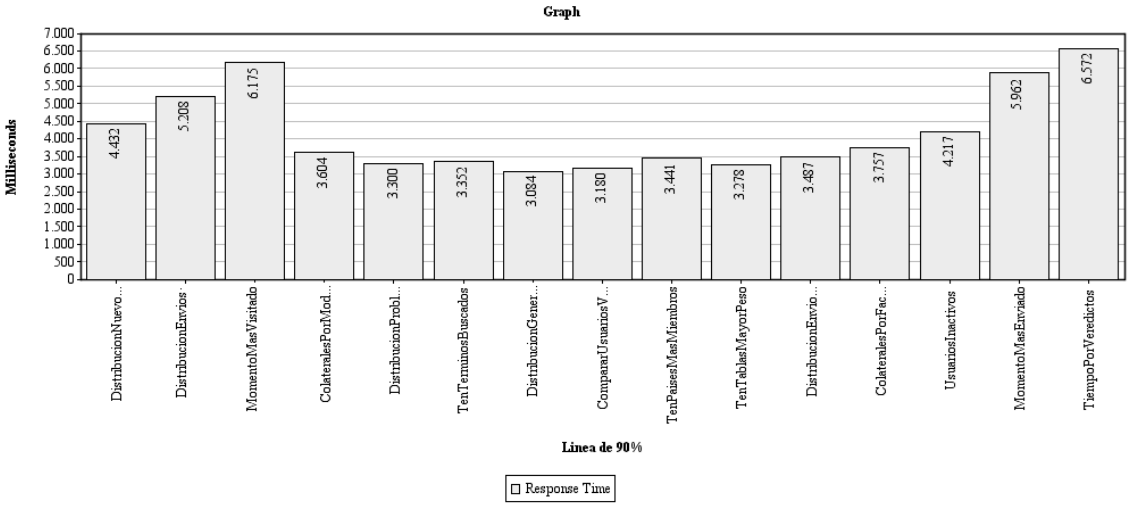


Figura 26 Resultado de las pruebas de rendimiento

Conclusiones del Capítulo

En este capítulo se expusieron los elementos más significativos referentes a las fases de implementación y pruebas. Se realizó el diagrama de despliegue observándose la distribución física del sistema a través de nodos y se mostró las relaciones entre los elementos a través de los diagramas de componentes. Se determinó comprobar la solución mediante casos de prueba utilizando el método de caja negra. Estos casos de pruebas permitieron comprobar el correcto funcionamiento de todas las funcionalidades de la aplicación, demostrando que la solución es totalmente funcional y que responde a los requisitos definidos.

Conclusiones Generales

Una vez concluida la investigación y desarrollado un producto final de acuerdo a los requisitos propuestos por los clientes se arriban a las siguientes conclusiones generales:

- Se elaboró el marco teórico de la investigación donde se realizó un estudio de las posibles variantes de cómo se representa la información para la toma de decisiones en varios jueces en línea, lo cual arrojó, que no se encuentra un sistema que se ajuste completamente al objetivo del trabajo.
- Se realizó el análisis y diseño del módulo de acuerdo a la metodología de desarrollo AUP-UCI para llevar a cabo el proceso de desarrollo y posibilitó mediante la construcción de sus artefactos que se tuviera una mejor visión del sistema a desarrollar.
- El uso de la librería gráfica FusionCharts permitió mostrar la información en las distintas gráficas a las que se ajustaban debido a la gran variedad de gráficos que presenta.
- Se arrojó como resultado de la implementación que el módulo es capaz de representar y organizar grandes cantidades de datos de manera resumida y detallada, ya que mediante las gráficas los administradores son capaces de visualizar comparaciones, tramas y tendencias de los datos.
- Se evidenció con las pruebas realizadas el correcto funcionamiento del sistema y el cumplimiento de todos los requisitos propuestos.

Referencias

1. Reyes Cruz, Sergio. Aplicación del software educativo como herramienta de enseñanza para la obtención de un aprendizaje significativo en educación superior. Facultad de Medicina de Tampico, Universidad Autónoma de Tamaulipas. México : s.n., 2013. pág. 16, Maestría en docencia.

2. UCI. Universidad de las Ciencias Informáticas. [En línea] 2012. [Citado el: 15 de Octubre de 2015.] http://www.uci.cu/?q=mision.

3. Junco Vázquez, Tomás Orlando. El Jurado en Línea, nueva forma de ejercitación y evaluación de las asignaturas de programación. Universidad de las Ciencias Informáticas. La Habana : s.n., 2009. pág. 80, Tesis de grado.

4. de León Barral, Javier y, Roura Sixto Leandro. CMF, Caribbean Mind Forge. [En línea] 5 de junio de 2013. [Citado el: 15 de Octubre de 2015.] http://cmf.uci.cu/faq/read/811525.

5. Nmédia Solutions. Historia del desarrollo de aplicaciones Web. Departamento Informática y Sistemas, Universidad de Murcia. España : s.n., 2010.

6. Alegsa, Leandro. ALEGSA.com.ar. [En línea] 16 de Noviembre de 2010. [Citado el: 18 de Junio de 2016.] http://www.alegsa.com.ar/Dic/aplicacion%20web.php.

7. Lia, Rujia, Manzoor, Shahriar y y Revilla, Miguel A. Competitive Learning in Informatics: The UVa Online Judge Experience. Institute of Mathematics and Informatics. 2010.

8. Moreno Olivos, Tiburcio. La evaluación de competencias en educación. Sinética. [Documento]. 2012. págs. 01-20. ISSN 1665-109X.

9. Barral, Javier de León. Juez en Línea de agilidad mental, lógica y matemátias, Caribbean Mind Forge. Universidad de las Ciencias Informáticas. La Habana : s.n., 2014. pág. 83, Tesis de grado.

10. —. CMF - Juez en línea de agilidad mental, lógica y matemática. [En línea] 2 de Mayo de 2011. [Citado el: 9 de Febrero de 2016.] http://cmf.uci.cu/enlace.

11. Universidad Centroccidental Lisardo Alavarado. [En línea] 2014. http://www.ucla.edu.ve/dac/Departamentos/coordinaciones/informaticai/documentos/PROCESAMIENTO%20DE%20DATOS.html.

12. Carrion, Leonel Martinez. SlideShare. [En línea] 12 de 2011. http://es.slideshare.net/leonel_canceriano/procesamiento-de-datos-6940831.

13. Gómez Molina, Davida. Prezi. [En línea] 26 de Agosto de 2015. https://prezi.com/htyqnoz-6lz9/procesamiento-de-informacion/.

14. Orozco, David. ConceptoDefinicion.de. [En línea] 4 de Octubre de 2011. http://conceptodefinicion.de/informacion/.

15. Sociedad Andaluza de Educación Matemática Thales. [En línea] Creative Commons. http://thales.cica.es/rd/Recursos/rd99/ed99-0307-02/h2_info.html.

16. de la Fuente Valentín, Luis. UNIR - Universidad Internacional de La Rioja. [En línea] 6 de Febrero de 2014. http://blogs.unir.net/2522-visualizacion-de-la-informacion-en-la-toma-de-decisiones. ISSN 2444-1244.

17. Jimenez, Alma. SlideShare. [En línea] 20 de Enero de 2010. http://es.slideshare.net/Psicjimenez/tecnicas-de-representacion-de-la-informacion.

18. Juan Carlos Lobaina Guzmán, Jorge Luis Roque Alvarez. COJ. Caribbean Online Judge. [En línea] 1.0 beta, 2011. [Citado el: 20 de Ocubre de 2015.] http://coj.uci.cu/general/links.xhtml.

19. UVa Online Judge. [En línea] [Citado el: 9 de Febrero de 2016.] http://uva.onlinejudge.org/index.php?option=com_onlinejudge&Itemid=8.

20. Módulo de eventos y notificaciones para el Juez en Línea Caribeño. Morciego Lezcano, Angel Miguel y Sánchez Oliü, Yelaine. 2, La Habana : s.n., 2 de Febrero de 2016, Serie Científica de la Universidad de las Ciencias Informáticas, Vol. IX, págs. 113-124. ISSN: 2306-2495.

21. Sphere Online Judge (SPOJ). [En línea] [Citado el: 9 de Febrero de 2016.] http://www.spoj.com.

22. SPOJ. Sphere online judge. [En línea] [Citado el: 30 de Enero de 2016.] http://www.spoj.com/tools/.

23. Xu Pengcheng Ying Fuchen Xie Di. POJ. Xu Pengcheng Ying Fuchen Xie Di. [En línea] [Citado el: 30 de Enero de 2016.] http://poj.org.

24. Mirzayanov, Mike. Codeforces. [En línea] http://codeforces.com/.

25. Topcoder. [En línea] http://www.topcoder.com/.

26. Lobaina, Juan Carlos y Roque Alvarez, Jorge Luis. COJ. Caribbean Online Judge. [En línea] 1.0 beta, 2011. [Citado el: 9 de Febrero de 2016.] http://coj.uci.cu/general/about.xhtml.

27. Lobaina, Juan Carlos y Roque Alvarez, Jorge Luis. COJ. Caribbean Online Judge. [En línea] 1.0 beta, 2011. [Citado el: 9 de Febrero de 2016.] http://coj.uci.cu/contest/past.xhtml.

28. Jacobson, Ivar, Booch, Grady. El proceso Unificado de Desarrollo de Software. [ed.] Andrés Otero. [trad.] Salvador Sánchez, y otros. Madrid : Addison Wesley, 2000. pág. 458. ISBN: 84-7829-036-2.

29. Boehm, Barry y Turner, Richard. Observations on Balancing Discipline and Agility. University of Southern California ; George Washington University. 2003. págs. 32 - 39. ISBN: 0-7695-2013-8.

30. Sánchez, Tamara Rodríguez. Metodología de desarrollo para la Actividad productiva de la UCI. Universidad de las Ciencias Informáticas. La Habana : s.n., 2014. pág. 15, Programa de mejora.

31. Spring Source. Groovy. [En línea] 2016. [Citado el: 23 de mayo de 2016.] http://groovy.codehaus.org.

32. Santana Roldán, Carlos. CODE. [En línea] 30 de Marzo de 2013. [Citado el: 23 de Mayo de 2016.] https://www.codejobs.biz/es/blog/2013/03/30/que-es-groovy.

33. Calahorro Brito, Nacho. Manual de desarrollo web con grails. [En línea] 9 de Junio de 2009. [Citado el: 1 de Abril de 2016.] http://www.manual-de-grails.es. 978-84-613-2651.

34. Alvarez, Miguel Angel. Desarrollo Web. [En línea] [Citado el: 1 de Abril de 2016.] http://www.desarrolloweb.com/.

35. Wellman, Dan. jQuery UI 1.8. The User Interface Library for jQuery. Birmingham : Packt Publishing Ltd, 2011. pág. 424, Manual. ISBN: 978-1-849516-52-5.

36. TEC - COMPUTACIÓN e INFORMÁTICA - LENGUAJE DE PROGRAMACIÓN VI. [En línea] 2014. [Citado el: 15 de Febrero de 2016.] https://sites.google.com/a/innovavirtual.org/tecceilpvi/home/vi-ciclo-2014/emision-de-graficos/alternativa-de-graficos/comparison-of-javascript-charting-frameworks/50-librerias-javascript-para-charts-y-graphs.

37. Fusion Charts. [En línea] http://www.fusioncharts.com/.

38. Denzer, Patricio. Departamento de Electrónica. Universidad Técnica Federico Santa María. [En línea] 23 de Octubre de 2002. [Citado el: 11 de Junio de 2016.] http://profesores.elo.utfsm.cl/~agv/elo330/2s02/projects/denzer/informe.pdf.

39. PostgreSQL: About. [En línea] [Citado el: 10 de Febrero de 2016.] http://www.postgresql.org/about/.

40. F. Lanvin, Daniel. Departamento de Informática. [En línea] 31 de Octubre de 2011. [Citado el: 10 de Febrero de 2016.] http://www.di.uniovi.es/.

41. Extensión de Visual Paradigm for UML para el desarrollo dirigido por modelos de aplicaciones de gestión de información. Cabrera González, Lisbet y Pompa Torres, Enrique Roberto. 10, La Habana : s.n., 15 de Octubre de 2012, Serie Científica de la Universidad de las Ciencias Informáticas, Vol. V, pág. 11. ISSN: | RNPS.

42. Prueba Automática de Carga y Estrés. Sanchez Almenares, Liudmila. 2, La Habana : s.n., 2009, Serie Científica. Universidad de las Ciencias Informáticas, Vol. II, págs. 4-8. ISSN: 2306-2495 .

43. La ingeniería de requerimientos y su importancia en el desarrollo de proyectos de software. Arias Chávez, Michael. 10, Costa Rica : Universidad de Costa Rica, 2006, Redalyc: Red de Revistas Científicas de América Latina y el Caribe, España y Portugal, Vol. VI, págs. 1-13. ISSN: 2215-2458.

44. Abdul-Jawad, Bashar. Groovy and Grails Recipes. [ed.] Steve Anglin y Tom Welsh. Primera. 2009. pág. 424. ISBN: 978-1-4302-1600-1.

45. Larman, Craig. UML Y PATRONES. Una introducción al análisis y diseño orientado a objetos y al proceso unificado. Madrid : s.n., 2003. ISBN- 84-05-3438-2.

46. Castellano, Javier García. Página personal de Javier García Castellano. [En línea] [Citado el: 25 de febrero de 2015.] http://flanagan.ugr.es/docencia.

47. Xavier Ferré Grau, María Isabel Sánchez Segura. Desarrollo Orientado a Objetos con UML. Facultad de Informática-UPM, Universidad Veracruzana. México : s.n., 2011. pág. 53.

48. Martínez Zurita, Alberto. SlideSahre. [En línea] 9 de Marzo de 2013. [Citado el: 18 de Junio de 2016.] http://es.slideshare.net/albertozurita96/diagrama-de-despliegue-17071673.

49. Machado Peña, Yadira, y otros. ESTRATEGIA DE PRUEBAS FUNCIONALES APLICADA EN CALISOFT PARA GARANTIZAR LA CALIDAD DEL SOFTWARE. Departamento de pruebas de software (DPSW), Universidad de las Ciencias Informáticas. La Habana : s.n., 2012. pág. 13. ISBN : 978-959-286-019-3.

50. Abner, Gerardo. SlideShare. [En línea] 29 de Junio de 2013. [Citado el: 16 de Junio de 2016.] http://es.slideshare.net/abnergerardo/pruebas-de-sistemas-y-aceptacion-23663195.

51. Pressman, Roger S. Ingeniería del Software. Un enfoque práctivo. [ed.] Darrel Ince. Quinta. s.l. : McGraw-Hill, 2005. pág. 614.

52. Piattini Velthuis, Mario G. Análisis y Diseño de Aplicaciones Informáticas de Gestión. Incluye una Perspectiva de Ingeniería del Software. s.l. : RA-MA, 2005. 978-84-7897-587-7.

Anexos

Anexo 1: Historias de Usuario

Número: 6	**Nombre del requisito:** Mostrar los momentos de la semana más visitado
Programador: Yanet	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 4
Riesgo en Desarrollo: Medio	**Tiempo Real:** 4
Descripción: Se muestra una gráfica nombrada mapa de calor, donde se distribuye por días de la semana y horas del día los momentos más visitados en esa semana.	
Observaciones: Cada color representa una intensidad, las mismas son: - Verde: Bajo. - Amarillo: Medio. - Naranja: Medio alto. - Rojo: Alto.	
Prototipo de interfaz: 	

Tabla 8 HU: Mostrar los momentos de la semana más visitado

Número: 7	**Nombre del requisito:** Mostrar distribución de veredictos
Programador: Jessica	**Iteración Asignada:** 1

Prioridad: Media	**Tiempo Estimado:** 3
Riesgo en Desarrollo: Medio	**Tiempo Real:** 3
Descripción: Muestra una gráfica con los dos estados posibles de los envíos [Aceptado, Denegado]. El administrador tiene la posibilidad de: - Filtrar por año, mostrando así la información por meses de ese año seleccionado. - Filtrar por mes del año ya seleccionado anteriormente revelando la información por días de ese mes.	
Observaciones: Los datos de los últimos siete días se muestra al filtrar por el mes actual.	
Prototipo de interfaz: 	

Tabla 9 HU: Mostrar distribución de veredictos

Número: 8	**Nombre del requisito:** Mostrar distribución de problemas por período de tiempo
Programador: Yanet	**Iteración Asignada:** 1
Prioridad: Media	**Tiempo Estimado:** 4
Riesgo en Desarrollo: Medio	**Tiempo Real:** 4
Descripción: Se muestra una gráfica de tipo Gantt donde se expone el momento de publicación de los problemas en el mes actual.	
Observaciones: El día actual es señalado con una columna de color rojo.	
Prototipo de interfaz:	

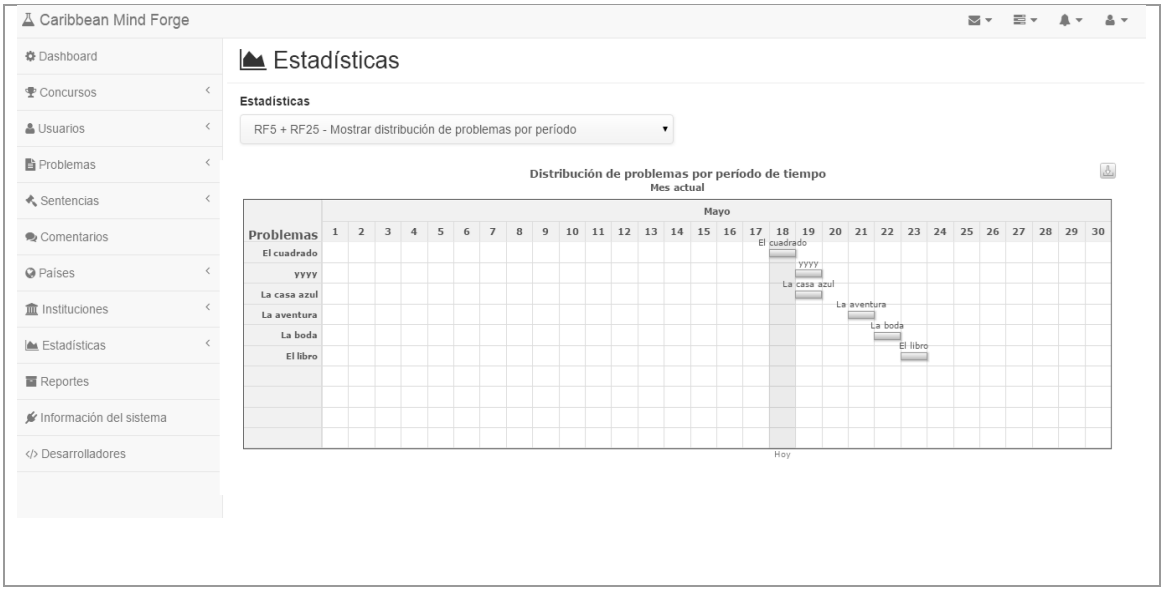


Tabla 10 HU: Mostrar distribución de problemas por período de tiempo

Número: 9	**Nombre del requisito:** Mostrar distribución de envíos en concurso
Programador: Yanet	**Iteración Asignada:** 1
Prioridad: Media	**Tiempo Estimado:** 3
Riesgo en Desarrollo: Medio	**Tiempo Real:** 3
Descripción: Se muestra una gráfica de barras con los envíos de diferentes concursos. El administrador tiene la posibilidad de: - Filtrar por concurso, mostrando así la información del concurso seleccionado.	
Observaciones:	

Prototipo de interfaz:

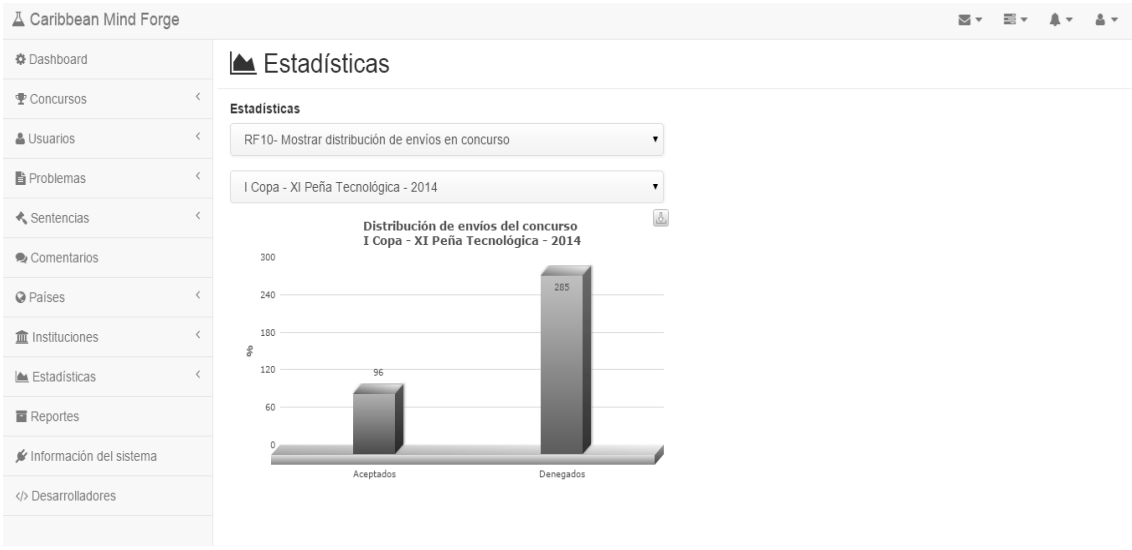


Tabla 11 HU: Mostrar distribución de envíos en concurso

Número: 10	**Nombre del requisito:** Comparar usuarios en cuanto a veredictos
Programador: Jessica	**Iteración Asignada:** 2
Prioridad: Media	**Tiempo Estimado:** 4
Riesgo en Desarrollo: Medio	**Tiempo Real:** 4
Descripción: Dado dos usuarios se muestra en una gráfica de barras la cantidad de envíos [Aceptados vs Denegados] realizados en ambos.	
Observaciones:	

Prototipo de interfaz:

Tabla 12 HU: Comparar usuarios en cuanto a veredictos

Número: 11	**Nombre del requisito:** Mostrar total de participantes por facultad en los eventos colaterales.
Programador: Jessica	**Iteración Asignada:** 2
Prioridad: Media	**Tiempo Estimado:** 3
Riesgo en Desarrollo: Bajo	**Tiempo Real:** 3
Descripción: Se muestra una gráfica de barra dividida por la cantidad total de participantes por facultades, donde el administrador puede: - Escoger el año del evento colateral. - Escoger las facultades.	
Observaciones:	

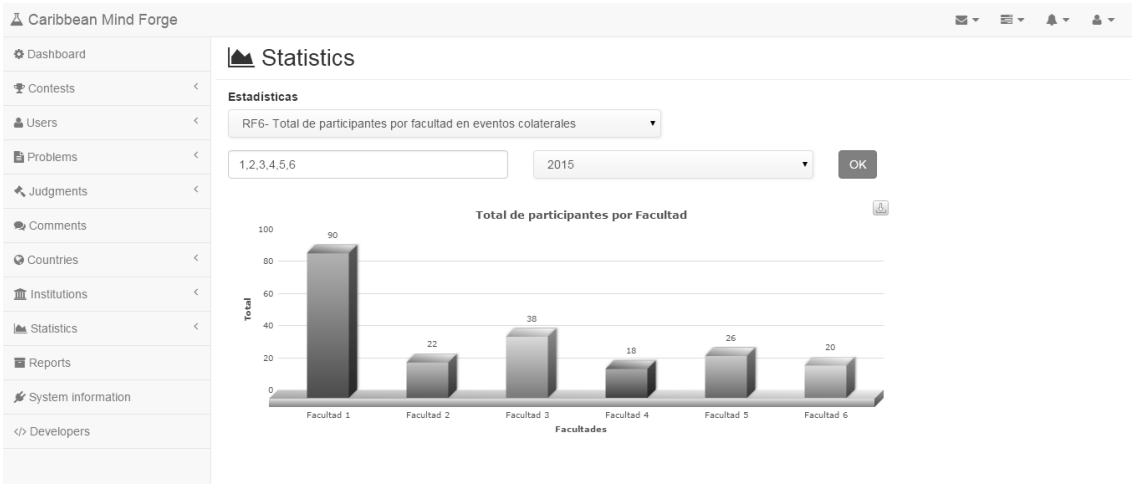
Prototipo de interfaz:


Tabla 13 HU: Mostrar total de participantes por modalidad en eventos colaterales

Número: 12	**Nombre del requisito:** Enviar por correo las estadísticas
Programador: Jessica	**Iteración Asignada:** 2
Prioridad: Media	**Tiempo Estimado:** 4
Riesgo en Desarrollo: Medio	**Tiempo Real:** 4
Descripción: Se les envía a los administradores del sistema un correo con el resumen semanal de lo ocurrido en el sistema en cuanto a: - Usuarios - Problemas - Sentencias - Veredictos	
Observaciones:	
Prototipo de interfaz:	

Tabla 14 HU: Enviar por correo las estadísticas

Número: 13	**Nombre del requisito:** Mostrar total de participantes por modalidad en eventos colaterales
Programador: Yanet	**Iteración Asignada: 2**
Prioridad: Baja	**Tiempo Estimado:** 3
Riesgo en Desarrollo: Bajo	**Tiempo Real:** 3

Descripción: Se muestra una gráfica de barra dividida por el total de participantes en cada modalidad de los eventos colaterales. El administrador puede: - Escoger el año de los eventos colaterales.
Observaciones:
Prototipo de interfaz:

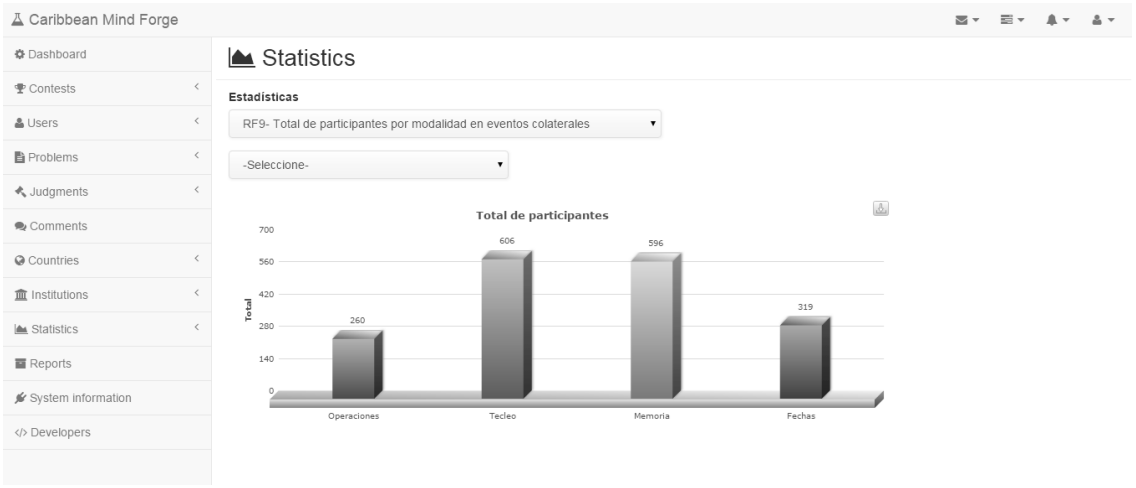


Tabla 15 HU: Mostrar total de participantes por modalidad en eventos colaterales

Número: 14	**Nombre del requisito:** Exportar las gráficas a PNG, JPG, PDF y SVG
Programador: Yanet	**Iteración Asignada:** 2
Prioridad: Baja	**Tiempo Estimado:** 2
Riesgo en Desarrollo: Baja	**Tiempo Real:** 2
Descripción: Permite exportar cada gráfica. El administrador tiene la posibilidad de: - Exportar en formato PNG, JPG, PDF o SVG.	
Observaciones:	

Prototipo de interfaz:

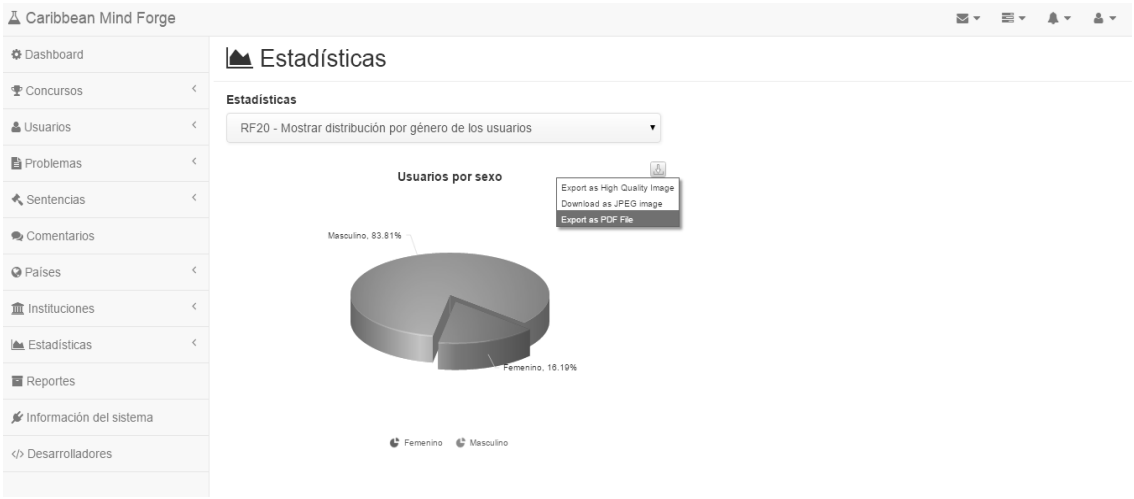


Tabla 16 HU: Exportar las gráficas a PNG, JPG, PDF y SVG

Número: 15	**Nombre del requisito:** Mostrar los diez países con más miembros activos
Programador: Jessica	**Iteración Asignada: 2**
Prioridad: Baja	**Tiempo Estimado:** 3
Riesgo en Desarrollo: Bajo	**Tiempo Real:** 3
Descripción: Se muestra una gráfica de tipo pastel con los diez países que más usuarios activos tiene el sistema.	
Observaciones:	

Prototipo de interfaz:

Tabla 17 HU: Mostrar los diez países con más miembros activos

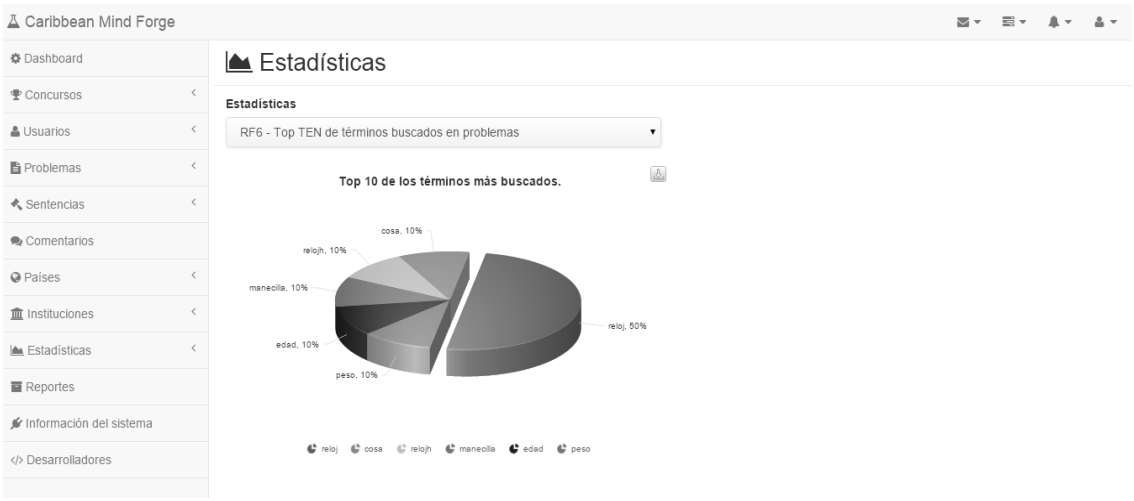
Número: 16	**Nombre del requisito:** Mostrar los diez términos más buscados de problemas
Programador: Yanet	**Iteración Asignada:** 2
Prioridad: Baja	**Tiempo Estimado:** 3
Riesgo en Desarrollo: Bajo	**Tiempo Real:** 3
Descripción: Se muestra una gráfica de tipo pastel dividida en porciento con los diez términos que han sido más buscado en el listado de los problemas.	
Observaciones:	
Prototipo de interfaz: 	

Tabla 18 HU: Mostrar los diez términos más buscados de problemas

Número: 17	**Nombre del requisito:** Mostrar las diez tablas con más peso
Programador: Yanet	**Iteración Asignada:** 3
Prioridad: Baja	**Tiempo Estimado:** 3
Riesgo en Desarrollo: Medio	**Tiempo Real:** 3
Descripción: Se muestra una gráfica de tipo pastel dividida en porciento, donde el administrador puede consultar las diez tablas de la base de datos que más peso tienen.	
Observaciones:	

Prototipo de interfaz:

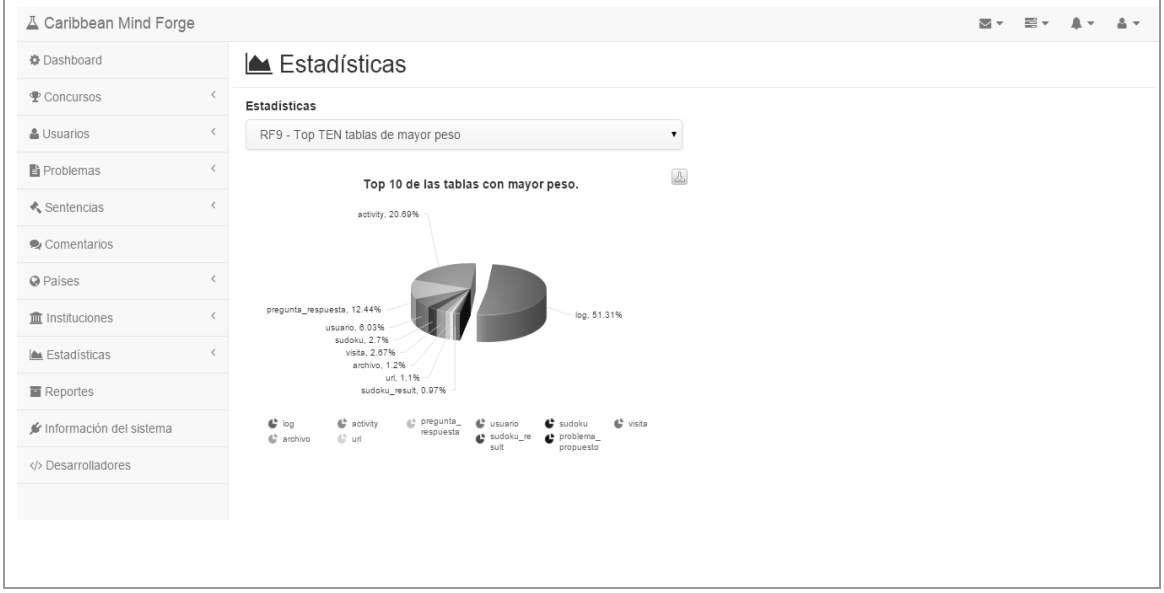


Tabla 19 HU: Mostrar las diez tablas con más peso

Anexo 2: Pruebas de Aceptación

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba	Observaciones
El administrador del sistema accede al módulo en el panel de administración y selecciona “Mostar distribución de registros de usuarios”.		El sistema muestra gráficamente los registros de todos los usuarios de la CMF distribuido en todos los años.	Satisfactorio	
Selecciona el año por el que desea filtrar.		Se muestra gráficamente los registros de todos los usuarios del año seleccionado y permite filtrar por el mes deseado.	Satisfactorio	
Selecciona el mes por el que desea filtrar dependiendo		Se muestra gráficamente los registros de todos los usuarios del	Satisfactorio	No se muestra los números de los días del mes.

del año previamente seleccionado.		año y mes seleccionado.		

Tabla 20 PA: Mostrar distribución de nuevos registros

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba	Observaciones
El administrador del sistema accede al módulo en el panel de administración y selecciona “Mostrar distribución de envíos de usuarios”.		El sistema muestra gráficamente los envíos de todos los usuarios de la CMF distribuido en todos los años.	Satisfactorio	
Selecciona el año por el que desea filtrar.		Se muestra gráficamente los envíos de todos los usuarios del año seleccionado y permite filtrar por el mes deseado.	Satisfactorio	
Selecciona el mes por el que desea filtrar dependiendo del año previamente seleccionado.		Se muestra gráficamente los envíos de todos los usuarios del año y mes seleccionado.	Satisfactorio	No se muestra los números de los días del mes.

Tabla 21 PA: Mostrar distribución de envíos de usuarios

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba	Observaciones
El administrador del sistema accede al módulo en el panel de administración y selecciona “Mostar cuentas activas e inactivas”.		El sistema muestra una gráfica de barra con el total de usuarios activos e inactivos de la CMF dividido en cuentas que nunca que nunca fueron	Satisfactorio	

		activadas, las inactivas en los últimos 365 días y en el último mes. Además se muestran las cuentas activas.		

Tabla 22 PA: Mostrar cuentas de usuarios activos e inactivos en el sistema

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba	Observaciones
El administrador del sistema accede al módulo en el panel de administración y selecciona "Comparar usuarios en cuanto a veredictos". Ingresa los dos usuarios a comparar.		El sistema muestra gráficamente los envíos de los usuarios seleccionados restringidos por los usuarios ingresados.	Satisfactorio	
	El administrador del sistema accede al módulo en el panel de administración y selecciona "Comparar usuarios en cuanto a veredictos". Además de los campos vacíos	El sistema muestra una notificación informando que hay campos vacíos.	Satisfactorio	

Tabla 23 PA: Comparar problemas en cuanto a veredictos

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba	Observaciones

El administrador del sistema accede al módulo en el panel de administración y elige “Mostrar distribución de envíos en concurso” y selecciona el concurso.		El sistema muestra gráficamente la distribución de envíos de los usuarios en esa competencia.	Satisfactorio	

Tabla 24 PA: Mostrar distribución de envíos en concurso

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba	Observaciones
El administrador del sistema accede al módulo en el panel de administración y elige “Mostar tablas de mayor”.		El sistema muestra en una gráfica de pastel dividida en porciento con los nombres de las diez tablas de mayor peso en la base de datos.	Satisfactorio	

Tabla 25 PA: Mostrar las diez tablas con más peso

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba	Observaciones
El administrador del sistema accede al módulo en el panel de administración y elige “Mostrar los diez países con más miembros activos”.		El sistema muestra una gráfica de pastel dividida con el porciento de los diez países con más miembros activos en el sistema.	Satisfactorio	

Tabla 26 PA: Mostrar los diez países con más miembros activos

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba	Observaciones

El administrador del sistema accede al módulo en el panel de administración y elige “Mostrar los diez términos más buscados de problemas”.		El sistema muestra una gráfica de pastel dividida con el porciento de los diez términos con más búsqueda en la sección de problemas.	Satisfactorio	

Tabla 28 PA: Mostrar los diez términos más buscados de problemas

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba	Observaciones
El administrador del sistema accede al módulo en el panel de administración y elige “Distribución de usuarios por género”.		El sistema muestra una gráfica de pastel dividida con el porciento de los usuarios con género femenino y masculino.	Satisfactorio	

Tabla 29 PA: Distribución de usuarios por género

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba	Observaciones
El administrador del sistema accede al módulo en el panel de administración y elige “Distribución de problemas por período”. Selecciona el año y el mes.		El sistema muestra una gráfica con la distribución de problemas a publicar por el sistema para los concursos o sección de “24h” , filtrado por año y mes.	Satisfactorio	

Tabla 27 PA: Distribución de problemas por período

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba	Observaciones

El administrador del sistema accede al módulo en el panel de administración y selecciona “Línea de tiempo por veredictos”.		El sistema muestra gráficamente los veredictos de todos los usuarios de la CMF distribuido en todos los años.	Satisfactorio	
Selecciona el año por el que desea filtrar.		Se muestra gráficamente los veredictos de todos los usuarios del año seleccionado y permite filtrar por el mes deseado.	Satisfactorio	
Selecciona el mes por el que desea filtrar dependiendo del año previamente seleccionado.		Se muestra gráficamente los veredictos de todos los usuarios del año y mes seleccionado.	Satisfactorio	

Tabla 28 PA: Mostrar línea de tiempo por veredictos

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba	Observaciones
El administrador del sistema accede al módulo en el panel de administración y selecciona en el botón que tiene cada gráfica, exportar en formato PNG o JPG o PDF.		El sistema muestra en cada gráfica un botón donde se exporta en cualquiera de los tres formatos.	Satisfactorio	

Tabla 29 PA: Exportar las gráficas a PNG, JPG, PDF, SVG

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba	Observaciones
El administrador del sistema accede al módulo en el panel de administración y selecciona “Mostrar distribución de registros de usuarios”.		El sistema muestra gráficamente los registros de todos los usuarios de la CMF distribuido en todos los años.	Satisfactorio	
Selecciona el año por el que desea filtrar.		Se muestra gráficamente los registros de todos los usuarios del año seleccionado y permite filtrar por el mes deseado.	Satisfactorio	
Selecciona el mes por el que desea filtrar dependiendo del año previamente seleccionado.		Se muestra gráficamente los registros de todos los usuarios del año y mes seleccionado.	Satisfactorio	No se muestra los números de los días del mes.

Tabla 30 PA: Mostrar distribución de registros de usuarios

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba	Observaciones
El administrador del sistema accede al módulo en el panel de administración, selecciona “Total de participantes por facultad en eventos colaterales” e ingresa las facultades y el año		El sistema muestra una gráfica de barras con el total de participantes por facultad en los eventos colaterales en el año seleccionado.	Satisfactorio	

de los eventos colaterales.				
	El administrador del sistema accede al módulo en el panel de administración, selecciona “Total de participantes por facultad en eventos colaterales”. Deja el campo de facultad vacío.	El sistema muestra una notificación informando que hay campos vacíos.	Satisfactorio	

Tabla 31 PA: Mostrar total de participantes por facultad en eventos colaterales

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba	Observaciones
El administrador del sistema accede al módulo en el panel de administración, selecciona “Total de participantes por modalidad en eventos colaterales” e ingresa la modalidad.		El sistema muestra una gráfica de barras con el total de participantes por modalidad en los eventos colaterales en el año seleccionado.	Satisfactorio	

Tabla 32 PA: Mostrar total de participantes por modalidad en eventos colaterales

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba	Observaciones

El administrador del sistema accede al módulo en el panel de administración, selecciona “Momento de la semana más enviado.		El sistema muestra una gráfica de día de la semana vs horario del día donde se evidencia los momentos más enviados.	Satisfactorio	

Tabla 36 PA: Mostrar momento de la semana más enviado

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba	Observaciones
El administrador del sistema accede al módulo en el panel de administración, selecciona “Momento de la semana más visitado.		El sistema muestra una gráfica de día de la semana vs horario del día donde se evidencia los momentos más visitados.	Satisfactorio	

Tabla 37 PA: Mostrar momento de la semana más visitado

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba	Observaciones
El sistema envía un correo con frecuencia semanal con información en cuanto a lo ocurrido con los usuarios, sentencias, veredictos y problemas.		El administrador recibe el resumen de la semana en cuanto a usuarios, sentencias, veredictos y problemas.	Satisfactorio	

Tabla 33 PA: Enviar estadísticas por correo las estadísticas