Universidad de las Ciencias Informáticas

Facultad 6

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Título: Plataforma para la integración de componentes en el Sistema de Laboratorios Virtuales y a Distancia.

Autores: Yisell Gato Álvarez

Eric Gutiérrez Cabrera

Tutor: MSc. Omar Mar Cornelio

Ing. Osvel Chávez Hernández

La Habana

"Año 57 de la Revolución".

"Para ir delante de los demás,

se necesita ver más que ellos."

Declaración de autoría

Declaramos ser autores de la presente tesis y reconocemos 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 _____.

Yisell Gato Álvarez

Firma del autor

_____

Eric Gutiérrez Cabrera

Firma del autor

_____

MSc. Omar Mar Cornelio

Firma del tutor

_____

Ing. Osvel Chávez Hernández

Firma del tutor

_____

De Yisell

A mi mamá Gladys Alvarez García por haber sido mi madre y mi padre durante mis últimos 18 años, y porque sin ella este sueño sería imposible.

De Eric

"Dedicado a mis abuelos, por criarme con ese amor infinito y a mis padres, por estar pendiente día a día para lograr esta meta."

De Yisell

Tengo la dicha de poder decir que en estos 5 años siempre hubo a mi lado personas maravillosas que me apoyaron para que este sueño se hiciera realidad y hoy les quiero agradecer:

En primer lugar a mi mamá que sin su apoyo no hubiese logrado llegar a esta meta. Gracias por estar siempre a mi lado y por ser mi amiga leal. Porque con tu paciencia me hiciste crecer como persona ayudándome a salir adelante en momentos difíciles. Porque siempre he sabido que has luchado para darme lo mejor que has podido. Por tu amor, comprensión y soportar mis malcriadeces todo el tiempo. Gracias por hacer de mi lo que hoy soy.

A mi papá que aunque ya no está conmigo sé que sigue y guía mis pasos desde allí arriba.

A mi hermana, porque estoy segura que no hubiese encontrado en la vida mejor amiga que tú. Gracias por escucharme, por guardar mis secretos y por aceptarme a pesar de todas mis faltas. Porque posiblemente pensarás que "no te queda de otra", pero siempre estás conmigo cuando lo necesito.

A mi abuelita Sofía por malcriarme tanto y porque con sus muchos regaños ha sabido guiarme por el buen camino de la vida. A mi abuelita Ofelia y mi tía Estrellita por el amor incondicional que han sabido brindarme y por supuesto a mis pequeñas sobrinas por llenar mi vida de alegría.

A mi compañero de tesis que cuando le pregunte si quería ser mi dúo no lo pensó dos veces para decirme que sí. Gracias por todo el tiempo que hemos compartido juntos, por las fiestas, las

locura, las picardías, pero por sobre todo, por el aliento que me diste cuando pensé que no lo íbamos a lograr. Gracias por el apoyo que me brindaste cuando por problemas familiares me tuve que ausentar de la escuela, eso nunca lo voy a olvidar y por cuidarme ese día de Plenitud que me porté un poquito mal. Porque aunque sé que muchas veces te molestaste conmigo por mis regaños nunca permitiste que eso se interpusiera en nuestra amistad. Gracias por todo, gracias a ti este sueño se hizo realidad.

A los grandes amigos que he conocido en esta universidad y no puedo dejar de mencionar: A mi qqso por haberme regalado 4 años maravillosos, gracias por los mimos, el amor, la paciencia y la ayuda incondicional. A Arle por haberme permitido compartir contigo mis dos primeros años de universidad por las fiestas, los consejos e incluso algunos regaños. A los hermanitos varones que nunca tuve: Andy y Jose. A mis grandes amigas Glennis y Jessie con las que he recorrido media UCI.A Erne por haberme ayudado muchísimo en estos cinco años, por haber sido mi amigo y mi profesor. A Taimi, Yezenia, Nuris 1 y Nuris 2, Sonia, Sara, Lisandra Morgado y Jisel. A Migue, Leo, Alex, Franki, Lachi y a unos amigos que aunque conocí hace poquito han sabido ganarse un lugarcito en mi corazón: a Tata, Handy y Hairon, Gracias a todos por su amistad.

A los profesores que me ayudaron mucho en esta carrera y no puedo dejar mencionar: a Leonardo, Quintero, José Leandro y Jeem. A las personas que de una forma u otra contribuyeron en el logro de esta tesis:a mis tutores, oponente y el tribunal.En fin, gracias a todos los que de una forma u otra han compartido conmigo e hicieron de estos 5 años los mejores de mi vida.

De Eric

Al escribir estas letras las imágenes que solo navegan en mi mente son las de mis 5 años en esta universidad, todos esos recuerdos desde el 1er día que salí de mi casa con destino a la UCI hasta hoy, mi discusión de defensa de tesis. Debo decir que esta meta no es solo mía, pienso que ha sido también de todas las personas que me han apoyado y alentado para cumplir este sueño, ser un universitario. No seré hipócrita como mayormente se acostumbra al leer los agradecimientos de tesis, todas las personas que mencionaré aquí son los que verdaderamente deben estar.

Primeramente quiero mencionar a las dos personas las cuales no hay discusión alguna que deben encabezar esta lista, esos son mis abuelos, Aida y Oscar, la mayor bendición que he tenido en mi vida, mis guías desde que tenía 1 año, un solo sentimiento se destaca en ellos sobre mí, y es el AMOR, no tengo palabras para describir lo que siento por ellos, solo decir: Gracias por todo, ya aquí tienen a su nieto graduado. Aunque dios se llevó a mi viejito, aún tengo a mí otro ángel vivita, mi abuela. Este logro es para ustedes, jamás lo duden.

Siguen mi padres, mejor no los quiero, no creo que existan, fueron mi principal ayuda para vencer la carrera, mi mamá, con su preocupación extrema porque a su hijo no le faltara nada, por saber cómo salió en cada prueba, por ver cómo se siente de salud, por alentarme a seguir cuando a veces pensaba que no lo lograba, ya hace mucho tiempo para ella yo era su ingeniero, pero hoy si lo puedes decir con base: Mi hijo es ingeniero. Mi papá, con su carácter más juvenil que muchos jóvenes, el

amigo el cual nunca me ha abandonado, siempre lo he tenido ahí para todo, para los dos, también gracias por todo, aquí está su fruto formado.

Agradecer también a mis tíos abuelos Pepe y Olimpia, los cuales me brindaron su hogar en mis primeros años de la carrera, con un amor que más que de tíos, solo dos abuelos lo pueden dar, los amo con mi vida, gracias por ayudarme desde que era un niño, gracias por quererme como su nieto, este título también es para ustedes.

A mi tíos que me vieron nacer, mi tío Aliott con su locura, pero muy orgulloso de su sobrino, mi tío Orlando, mi tío Oscarito, mi tío Alexis que Dios lo tenga en la gloria, mi tía Moraima, mi tío Israelito el mejor hermano de mi mamá, para ellos también le dedico este título.

Para mi hermana, la cual me siento muy orgulloso por los pasos que anda, muy linda y buena, tata, tu hermano te dedica este título también.

Mi primo Arian, que nos criamos como hermano y como eso lo quiero yo, "ya maté esto pipo".

Mis padres habaneros, Ira y Pepe, mi familia también, gracias por su ayuda, son lo máximo, como dejar de quererlos.

A mi novia Mariam, mi amiga, mi amor, mi apoyo incondicional, para ti también te dedico esto mon amour.

Es hora de las personas que representan tanto como la familia: mis amigos.

A Jose mi hermano, que mejor que él que ha estado para mi ahí en los momentos buenos y malos, 5 años junto para todos, nunca me falló aunque por error un día nos molestamos, te quiero mi herma.

Frank, amigo como él hay pocos, apoyo en momentos duros para mí en esta universidad, Eduardo el crogueton, Daldy, el negrito lindo de mi mamá: el Yera, Nury la loquita, el Reide con su novia Doinita, Migue el loco, Leo.

Profesores que me ayudaron muchísimos en momentos difíciles de la carrera, no puedo dejar mencionar a Oscar, Yuya mi profe crazy y al Jeembili. Gracias por su amistad.

Abdiel el mortadello, no es mi amigo, es un verdadero hermano para mí, gracias por tu amistad incondicional.

A mis amigos de la infancia, pero de muchos que tengo hay nombres que no puedo dejar de mencionar: el Daro, mi hermano fiel, Rigo, Annerita, el Pepe.

A amigos que aunque hayan entrado tardes a mi vida, se ganaron mi afecto y amistad, empiezo con mi buti y mi gordi, dos amigas incondicionales, las quiero muchísimos, a mi piquete de Cárdenas: Hayron, Handy y el zurdo, gracias por conocerlos, gracias por sus visitas al hospital y ser atentos, se les quiere de verdad.

Agradecer también a las personas que de una forma u otra me dieron su ayuda para vencer esta tesis: tutores, oponente y tribunal.

Cierro con la persona que me demostró ser mi verdadera amiga, mi hermana, juntos para fiestas, para las locuras, para la gozadera, pero también juntos para las enfermedades, los dolores, los malos ratos, los momentos de estrés, de tristeza, la persona que reconozco que sin ella no me hubiera graduado, ayudándome, apoyándome, alentándome, peleándome con sus razones pero atenta en todo, esa es mi compañera de tesis, mi rubia linda, la gatica del piquete, yiselita, tata gracias por estar en mi vida, eres muy especial para mí, me enorgullezco de decirle a todos que eres mi hermana, te quiero muchísimo, aquí está nuestro logro, mil veces gracias.

El surgimiento de las Tecnologías de la Información y las Comunicaciones (TICs) y la evolución de la *World Wide Web* (WWW) han sido factores de vital importancia en la transformación de la sociedad, especialmente en el área educacional. El desarrollo de la educación a distancia ha permitido a docentes y especialistas el intercambio de información y de recursos tecnológicos a través de las redes de comunicación sin que sea necesario el contacto físico; recursos que por problemas económicos no se tienen al alcance de todos los centros investigativos y educacionales del país y que a través de la Web se pueden utilizar desde cualquier lugar y momento que se estime conveniente. La presente investigación se enmarca en la concepción e implementación de una plataforma que permita integrar componentes de un Sistema de Laboratorios Virtuales a Distancia (SLD) en el área de la automática con el objetivo de facilitarle a quien lo necesite las herramientas necesarias para el desarrollo de tareas investigativas que se llevan a cabo en muchas instituciones y que las mismas no constan de la instrumentación necesaria para desarrollarlas. El proceso de desarrollo del *software* estuvo guiado por la metodología mínima, completa y extensible: Open Up y se utilizó como lenguaje de programación Java con el fin de obtener buenos resultados.

Palabras claves: Plataforma, Sistema de Laboratorios a Distancia, *portlet*, *Liferay*.

Introducción

Desde el comienzo mismo de la evolución del hombre la información ha jugado un papel fundamental en la vida de este y ha progresado paralela y conjuntamente hasta convertirse en parte indispensable e indiscutible de la práctica diaria a todos los niveles(Duménigo 2012). La gran cantidad de información manejada, genera un haz de conocimientos que aumenta de forma acelerada, lo cual trae consigo una mayor demanda de productos de *software*, soluciones informáticas y servicios que permitan gestionarla eficientemente. Por estas razones fue obligatoria la interrelación de esta materia con otras ciencias y aspectos de la vida social. Así surgen las Tecnologías de la Información y las Comunicaciones (TICs), las cuales han llegado a ser uno de los pilares primordiales de la sociedad actual.

Las TICs son aplicables en casi todos los sectores de la sociedad reportando continuas transformaciones en el modelo económico, social y cultural e incidiendo en la mayoría de los aspectos de la vida cotidiana alcanzando mejoras en la calidad de vida y mayor eficiencia en los procesos de gestión de la información. El conjunto de tecnologías que abarcan las TICs es variado, destacando las Tecnologías Web, las cuales sirven para acceder a los recursos de conocimiento disponibles en la red de redes, más conocida como Internet.

Los Portales Web son destacados dentro de las Tecnologías Web, definiéndose como un punto de entrada común a una colección de recursos electrónicos integrados, donde se ofrecen una serie de servicios complementarios, tales como: búsqueda interna, personalización, herramientas de comunicación, servicios de información y alerta y otros servicios específicos asociados (Martínez 2007). Además pueden ser aplicados en la educación ya que sirven para enviar y recibir información de los alumnos, presentar exámenes, contestar encuestas, almacenar el registro de los alumnos, realizar prácticas basadas en simulaciones e incluso realizar acciones en las que se puede interaccionar de forma real con instrumentos o dispositivos externos.

La sociedad de la información y las nuevas tecnologías inciden de manera significativa en la rama educativa. Las escuelas tienen la necesidad de integrar la alfabetización digital, que es una fuente de información e instrumento de productividad para realizar trabajo y material didáctico (Alcántara 2009).

En los últimos años la mayoría de las universidades tanto del país como del mundo en general se enfrentan a dificultades que incluyen la carencia de recursos en personas, espacios y problemas

presupuestarios para la adquisición de equipos necesarios en el proceso de enseñanza. Una solución a este problema se encuentra en el uso de los avances tecnológicos de la información y las comunicaciones mediante la aplicación de la educación a distancia.

La educación a distancia ha sido beneficiada en su desarrollo, debido al enorme impulso obtenido con el surgimiento de la *World Wide Web*[1] (WWW), facilitando aún más los métodos y estrategias que permitieron el surgimiento del *e-learning*[2], y por ende, el acceso al conocimiento. Los educadores a distancia pueden usar la WWW para ayudar a sus estudiantes a navegar eficientemente y aprovechar todas las ventajas de la red mundial dentro de la que pueden encontrar mucho apoyo para su aprendizaje. Este tipo de educación se torna atractiva en el área de control automático.

Una de las modalidades más utilizadas dentro de la educación a distancia son los laboratorios virtuales y remotos a través de Internet, es por ello que todos los esfuerzos actuales se dirigen hacia la implementación de estos (Santana and Hernández 2011). Los laboratorios remotos permiten acceder a sistemas físicos, permitiendo al alumno desarrollar actividades sin tener presencia física en el laboratorio donde se encuentra el sistema.

Cuba es un país que debido a su situación económica le es difícil poner a disposición de cada centro educativo todos los recursos tecnológicos necesarios para el aprendizaje de los estudiantes. El desarrollo de un Sistema de Laboratorios a Distancia (SLD) sería una buena opción para la solución de este problema. A través de las redes de comunicaciones se realizarían intercambios de recursos entre los estudiantes y los laboratorios a la hora de realizar las prácticas; contribuyendo así a la reducción de importaciones de materiales técnicos para la confección de Laboratorios Convencionales (LC) y se les facilitaría a los alumnos el aprendizaje del contenido de las materias al permitirles estudiar en el momento y lugar que crean conveniente.

En el año 2002 fue creada la Universidad de las Ciencias Informáticas (UCI) con el objetivo de formar profesionales capacitados en el desarrollo de *software*. La UCI es un centro que posee amplios recursos tecnológicos y computacionales para el procesamiento de grandes cálculos y desarrollo de aplicaciones. Además, posibilita la descarga de programas y artículos científicos dado al factible acceso a la red de redes, Internet.

[1] Red de redes.
[2] Aprendizaje electrónico.

En la facultad 6 perteneciente a dicha universidad se encuentran en desarrollo un conjunto de laboratorios virtuales enfocados al área de la automática. Estos permiten a los especialistas y estudiantes de esta área realizar prácticas basadas en simulaciones, observaciones y análisis de muestras. Sin embargo, la utilización de cada uno de ellos se realizaría de manera independiente haciendo engorroso el acceso simultáneo a más de un laboratorio a la hora de realizar las prácticas, provocando así una posible pérdida de tiempo e información. Para solucionar este problema sería más conveniente agruparlos todos en un mismo sistema que brinde estos servicios de forma integrada, segura y confiable.

Teniendo en cuenta lo planteado anteriormente, se tiene como **problema de la investigación**: ¿Cómo integrar los componentes del Sistema de Laboratorios Virtuales y a Distancia para el área de automática?

Para darle solución a la problemática identificada, se plantea el **objetivo general** siguiente: Desarrollar una plataforma para la integración de componentes en el Sistema de Laboratorios Virtuales y a Distancia.

Se define como **objeto de estudio** de la investigación: Los sistemas de información para el aprendizaje electrónico enmarcado en el **campo de acción**: Sistemas de Laboratorios Virtuales y a Distancia para el área de automática.

Para dar cumplimiento al objetivo se identificaron las siguientes **tareas de la investigación**:

- Elaboración del marco teórico de la investigación sobre la plataforma de integración de componentes del Sistema de Laboratorios Virtuales y a Distancia.
- Caracterización de las tecnologías y herramientas a utilizar en el proceso de desarrollo del sistema.
- Diseño de la solución propuesta.
- Implementación de las funcionalidades de la plataforma para la integración de componentes en el Sistema de Laboratorios Virtuales y a Distancia.
- Validación de la solución mediante pruebas al sistema.

Resultados esperados:

Un sistema que permita la integración de contenido de diferentes fuentes que a su vez constituyen los laboratorios virtuales de la plataforma de aprendizaje a distancia enfocada al área de la automática.

Durante el desarrollo de la investigación se utilizan métodos científicos y técnicas de recopilación de información que guíen la misma. A continuación se especifican los métodos y técnicas seleccionadas y el motivo de su elección.

Métodos teóricos:

- Analítico - Sintético: para analizar y estudiar la documentación y teoría relacionada con plataformas de laboratorios virtuales y a distancia, permitiendo así, extraer los elementos más coherentes e importantes y sintetizarlos en la confección de la solución propuesta.
- Modelación: Creación de abstracciones de la plataforma que constituyan una guía a la hora de implementarla, posibilitando además tener una idea de la interacción de los diferentes componentes de la misma.

Las **técnicas de recopilación de información** utilizadas fueron:

- Tormenta de ideas: Los autores plantearon un conjunto de ideas acerca de los requisitos funcionales que consideraron importantes a la hora de implementar el sistema. Posteriormente el tutor organizó dichas ideas priorizando las de mayor importancia.

El presente documento consta de tres capítulos los cuales se enuncian y describen brevemente a continuación:

- **Capítulo 1: Fundamentos teóricos metodológicos de la plataforma para la integración de componentes en el Sistema de Laboratorios Virtuales y a Distancia:** En este capítulo se exhiben los conceptos relacionados al tema de la investigación para a partir de un estudio de los mismos lograr una solución adecuada al problema identificado. Se seleccionan además las tecnologías y herramientas a utilizar para el desarrollo de la plataforma.
- **Capítulo 2: Análisis y diseño de la plataforma para la integración de componentes en el Sistema de Laboratorios Virtuales y a Distancia:** En este capítulo se elabora el modelo conceptual de la plataforma y se definen los requisitos funcionales y no funcionales que el *software* deberá cumplir. Se realizan los diagramas de casos de uso y clases del diseño así como también quedan definidos los patrones arquitectónicos y del diseño que se ponen de manifiesto en el sistema. Queda definida la arquitectura del *software*.

- **Capítulo 3: Implementación y prueba de la plataforma para la integración de componentes en el Sistema de Laboratorios Virtuales y a Distancia:** Finalmente en este capítulo una vez terminado el sistema se seleccionan las pruebas que se le realizarán al mismo para comprobar que fue culminado exitosamente.

Capítulo I Fundamentos teóricos metodológicos de la plataforma para la integración de componentes en el Sistema de Laboratorios Virtuales y a Distancia.

Introducción.

En el presente capítulo se exhiben los conceptos fundamentales asociados al dominio para alcanzar una mayor comprensión de la investigación. Se realiza un análisis de las soluciones existentes para a partir de ellas desarrollar la nueva solución. Se explican las tecnologías y herramientas a utilizar en el desarrollo de la plataforma que permite la integración de componentes en el SLD y se define además la metodología que guiará el proceso de desarrollo del sistema.

1.1 Sistemas de información.

El uso y acceso a la información ha constituido siempre un cambio trascendental para el desarrollo del hombre en todas las esferas de la vida. A medida que crecía gradualmente su importancia, también variaba la manera en que esta se gestionaba y tramitaba, llegando incluso a crear disciplinas estrictamente destinadas al estudio de la misma.

En el seno del progreso nacieron los Sistemas de Información (SI) como un conjunto formal de procesos que, operando sobre una colección de datos e información estructurados según las necesidades de la organización, recopilan, elaboran y distribuyen la información necesaria para las operaciones, las actividades de dirección y la toma de decisiones. Actualmente los SI apoyados en herramientas informáticas han alcanzado un gran auge, a tal punto de casi desplazar a los tradicionales formatos sobre físicos (Duménigo 2012).

Se vive en una época de constantes innovaciones tecnológicas que están orientadas a impulsar el desarrollo de los mecanismos para incrementar el almacenamiento, tratamiento y transmisión de la información; desarrollo que hemos denominado las Nuevas Tecnologías de la Información y la Comunicación, TIC (Domínguez 2008).

Una de las principales funciones de las TIC en los centros educativos es la fuente abierta de información (WWW e Internet en general). Desde el punto de vista del proceso de aprendizaje, la enseñanza a través de Internet puede incidir favorablemente en las técnicas de comunicación y los conocimientos informáticos de los estudiantes, la competencia en los distintos modos de comunicación y la motivación personal. Puede servir también para satisfacer necesidades específicas de los estudiantes e incluso permitirles trabajar en realidades virtuales. El enorme impulso obtenido con el surgimiento de Internet y las TIC han permitido el desarrollo de entornos que simulen aulas, laboratorios, sesiones de clase y hasta planes universitarios completos.

1.1.1 Aprendizaje electrónico.

Los sistemas de aprendizaje electrónico varían por la forma de presentar los datos, el potencial de interacción, la flexibilidad de los medios de enseñanza y los métodos didácticos utilizados. Todos los sistemas tienen en común la utilización de recursos multimedia que combinan texto, sonido, gráficos o video en una plataforma informática (Amutabi and Oketch 2003).

La expresión genérica Aprendizaje Electrónico (AE) describe toda forma de aprendizaje por medios electrónicos, incluso la utilización de programas de enseñanza en un ordenador personal al empleo de una Intranet o de Internet para la interacción en un sistema de redes.

El uso de Internet ayuda a estudiantes a navegar eficientemente y aprovechar todas las ventajas de la red mundial dentro de la que pueden encontrar mucho apoyo para su aprendizaje. La creación de espacios virtuales sobre la WWW para la generación, experimentación y el descubrimiento de conocimientos conocidos como laboratorios virtuales abre nuevas perceptivas que se potencian con el empleo de la inteligencia artificial, la realidad virtual, la simulación y los sistemas cooperativos.

La enseñanza presencial se puede complementar con la enseñanza a través de Internet, lo que puede hacer posible, por ejemplo, impartir un curso sincronizado en dos lugares distantes entre sí. El material didáctico se puede presentar en línea y la interacción entre los estudiantes de ambos lugares se facilita mediante el correo electrónico, los chats o los foros de debate. Los cursos a distancia pueden resultar adecuados para aquellos estudiantes que viven en zonas remotas y no pueden trasladarse a un centro educativo.

La educación a distancia y el aprendizaje electrónico son conceptos similares, la educación a distancia permite a personas que no pueden asistir a un centro de educación, auto-instruirse, capacitarse e incluso realizar cursos de interés mediante aulas virtuales contando con profesores conectados a tiempo real, eso también constituye aprendizaje electrónico.

1.2 Sistema de Laboratorios Virtuales y a Distancia.

Un laboratorio es un lugar dotado de los medios necesarios para realizar investigaciones, experimentos y trabajos de carácter científico o técnico(DRAE 2012). Sin embargo, a pesar de su papel relevante en los centro investigativos y educacionales para el estudio de las ciencias, presentan algunos inconvenientes como: costo inicial elevado, requiere de un constante mantenimiento, incrementa el consumo energético y además requieren la presencia física del alumno o supervisor. Una alternativa para solucionar este problema es la implementación de Laboratorios Virtuales (LV) aprovechando al máximo las bondades que brindan las TICs.

Los LV son definidos como simulaciones de prácticas manipulativas que pueden ser realizadas por un estudiante lejos de la universidad y el docente (Monje et al. 1999); no se requiere asistir a un espacio físico para realizar las prácticas, ni existe para su ejecución un horario establecido. La creación de los mismos apoya el aprendizaje y la investigación de técnicos y profesionales en el campo de las ciencias, como parte del proceso docente-educativo que se desarrolla en algunas instituciones.

El aprovechamiento de las TICs combinando las tecnologías Web con la implementación de modelos de simulación; permite al alumno evitar la asistencia a los laboratorios convencionales (LC), planteando como alternativa el uso de laboratorios virtuales a distancia.

Un laboratorio a distancia está basado en un entorno cliente – servidor, donde los clientes (estudiantes o investigadores desde un sitio a distancia), solicitan servicios o contactan a un servidor (equipo de cómputo, programas informáticos de acceso, dispositivos a manipular), a través de distintos medios de interconexión (Intranet, Internet) (Zamora 2012).

El trabajo con LVD conlleva a un nuevo paradigma y una nueva cultura de la ciencia. No se considera que vayan a suplantar a los laboratorios tradicionales; sino que deben ser vistos como una extensión de los

mismos, como generadores de nuevas perspectivas que no pueden ser exploradas en un laboratorio real a un costo accesible y con un riesgo aceptable.

1.3 Plataformas para Laboratorios Virtuales y a Distancia.

La educación a distancia como se ha planteado se desarrolla mediante operaciones electrónicas completamente virtualizadas que contienen herramientas de apoyo al aprendizaje, a menudo a este tipo de sistema complejo se le denomina plataforma.

Una plataforma es un sistema que sirve como base para hacer funcionar determinados módulos de hardware o programas informáticos. Queda definida por un estándar alrededor del cual se establecen los tipos de módulos, sistemas operativos, lenguajes de programación y/o interfaz de usuario compatibles (Morales 2013).

Una plataforma para los LVD debe facilitar la manipulación de modelos de simulación desarrollados en diferentes herramientas. Estos modelos deben configurarse en un entorno local y ejecutarse en un entorno a distancia que debe dar soporte a los simuladores. Con el objetivo de que el usuario extraiga conclusiones de la experimentación realizada, los resultados de la simulación deben retornar a la máquina local, y la plataforma debe ofrecerle al operador un conjunto de funcionalidades básicas que facilite su interpretación.

1.3.1 Análisis de plataformas existentes de aprendizaje electrónico.

En la actualidad existen implementadas varias plataformas para LVD como por ejemplo: la Plataforma para Laboratorios Virtuales y Remotos Orientados a la Simulación de Sistemas Dinámicos. A continuación se analizan brevemente las tecnologías y herramientas utilizadas en la implementación por la misma.

Plataforma para Laboratorios Virtuales y Remotos Orientados a la Simulación de Sistemas Dinámicos Controlados (Blas et al. 2013)**.**

Uno de los principales objetivos de este proyecto fue la implementación de una herramienta que permitiera, tanto a docentes como alumnos, la manipulación de diferentes LVD relacionados al área de Control Automático.

Java fue el lenguaje de programación seleccionado. Permite interactuar con múltiples entornos de simulación por medio de librerías especiales, lo que presenta una ventaja puesto que la plataforma requiere de un fuerte intercambio de datos e información con este tipo de componentes. Brinda además un alto nivel de portabilidad, lo que fomenta la usabilidad y contribuye a la realización de un desarrollo independiente del entorno de ejecución.

La plataforma se implementó utilizando Eclipse como entorno de desarrollo integrado y debido a que el conjunto de datos a resguardar en la plataforma no es significativo ni complejo, puesto que sólo referencia a las características generales de los modelos de simulación que representan a los laboratorios, se decidió utilizar MySQL.

Se implementaron en Java los módulos correspondientes a las aplicaciones cliente y servidor, los cuales se explican brevemente a continuación:

- Módulo 1: “Configuración de Modelos/LVR”

Este módulo se realizó en término de un conjunto de clases genéricas que permiten al aplicativo cliente instanciar una configuración específica y posteriormente determinar la validez del conjunto de parámetros configurados por el usuario.

- Módulo 2: “Ejecución de LVR”

Con el objetivo de atender las solicitudes de simulación, este módulo crea un hilo de ejecución para cada LVR a correr. Dicho hilo es el encargado de invocar al módulo responsable de la simulación, habiéndose definido su accionar en función de la instancia de la clase Configuracion.java configurada en el aplicativo cliente.

- Módulo 3: “Visualización de Resultados”

Finalmente este módulo define el formato de presentación de los resultados obtenidos a lo largo del proceso de simulación. Permite visualizar dicha información bajo dos esquemas: tablas y gráficas.

Esta plataforma de aprendizaje electrónico; logró satisfacer las necesidades de la organización para las que fue destinada, sin embargo, luego de un análisis a las tecnologías empleadas, se concluye que la plataforma que se desea desarrollar en la presente investigación debe implementar varios sistemas independientes evolucionando hacia una arquitectura orientada a componentes que permita la fácil integración de aplicaciones de terceros y sean altamente portables.

Tecnologías y herramientas para el desarrollo de la plataforma.

En el proceso desarrollo de la aplicación es fundamental la correcta elección de las tecnologías y herramientas que mejor se adapten, teniéndose en cuenta las tendencias actuales y las novedades de cada una de estas.

1.3.2 Plataformas de desarrollo Web. Java 2 Enterprise Edition.

En los inicios de la Web era difícil la construcción de aplicaciones sofisticadas. La primera generación de aplicaciones Web era primitiva, en general basada en formularios con información y aplicaciones de búsqueda. A través del tiempo, el conocimiento necesario para construir aplicaciones ha sido reducido. Hoy día, es relativamente sencillo construir aplicaciones sofisticadas utilizando las modernas plataformas y lenguajes, como por ejemplo PHP, Python, C# y Java.

Java es uno de los lenguajes de programación más utilizado para la creación de aplicaciones. Ha desarrollado tres ediciones de plataformas diferentes, cada una de ellas destinada a cubrir un conjunto diferente de necesidades de programación. La plataforma Java 2 Edición Estándar (J2SE, por sus siglas en inglés) enfocada para crear una amplia variedad de aplicaciones y applets, la plataforma Java 2 Edición Micro (J2ME, por sus siglas en inglés) permite la creación de aplicaciones para micro-dispositivos y la plataforma Java 2 Edición Empresarial (J2EE, por sus siglas en inglés) destinada a crear aplicaciones de servidor(Allamaraju et al. 2004).

Las aplicaciones desarrolladas bajo la plataforma Java pueden ser desplegadas en cualquier sistema operativo. J2EE es una unidad de *software* funcional que está ensamblada dentro de las aplicaciones Java con sus clases relacionadas y archivos que lo comunican con otros componentes. Dispone de varias herramientas de código abierto como *IDEs*, servidores, marcos de trabajo y *Applications Programming Interface* (*APIs* por sus siglas en inglés). Posibilita una serie de librerías que facilitan el trabajo de los desarrolladores, tales como *Java Database Connectivity* (JDBC), *Enterprise Java Beans* (EJB), *Java Server Pages* (JSP), *Java Tag Library* (JSTL) y *Java Message Service* (JMS). Cuenta con varios marcos de trabajo que facilitan el desarrollo de las aplicaciones, tales como Spring, Struts y Java Server Faces. Además posee varios CMS con muchas funcionalidades agregadas, entre ellos se pueden mencionar *Liferay Portal, Apache Jeetspeed* y *JBoss Portal*. Tanto para la plataforma J2EE y para el lenguaje Java

en general existen múltiples bibliografías en varios idiomas y se han creado comunidades que promueven la utilización de esta tecnología.

1.3.3 Componentes de la plataforma Java 2 Enterprise Edition.

El éxito y difusión alcanzado por Java no es sólo gracias al lenguaje, sino también al resto de la plataforma, que integra múltiples tecnologías en su seno, tecnologías para el desarrollo de aplicaciones Web tales como *Servlets,* JSP (siglas en inglés *de JavaServer Pages*) y *Portlet.*

Servlets

La primera generación de tecnologías para desarrollar aplicaciones Web en Java fueron los *servlets*. Un *Java Servlet* es un objeto Java que responde a las solicitudes HTTP y se ejecuta dentro de un contenedor de *servlets* (ej: *Tomcat*). El uso más común de los *servlets* es generar páginas Web de forma dinámica a partir de los parámetros de la petición que envíe el navegador Web (Cabrera 2014).

Por otra parte el principal problema que presenta el uso de *servlets* es que la página HTML debe ser escrita en el código de la aplicación utilizando excesivas sentencias de salida con el método *"println"*. De esta forma se generaba mucho código innecesario y se hacía muy complicado crear estéticas interfaces de usuario. Además trae consigo dependencia entre el código HTML y el código de la aplicación en Java, lo que constituye esfuerzo adicional por parte de los programadores. Este conjunto de deficiencias marcaron el surgimiento de nuevas tecnologías como las páginas JSP.

JSP

Una página JSP es un documento basado en texto que contiene dos tipos de texto: datos estáticos (que se pueden expresar en cualquier formato basado en texto como HTML, WML y XML) y elementos JSP, que determinan cómo la página construye contenido dinámico (Oracle 2010).

JSP está diseñado específicamente para simplificar la tarea de crear textos produciendo objetos *HttpServlet* y lo hace mediante la eliminación de todas las partes redundantes de codificación de un *servlet.* Todo JSP está diseñado para usarse con HTTP y generar contenido dinámico para el Internet.

JSP es un gran paso de avance, sin embargo muchas veces se necesita incluir código Java dentro de la página para realizar acciones específicas, como iterar sobre una lista de valores, establecer una condición

o conectarse a una base de datos. Teniendo en cuenta que este tipo de operaciones puede repetirse en varias páginas JSP, es necesario repetir el código en cada una de ellas. Para solventar esta desventaja se desarrolló una tecnología denominada JSTL[3].

Portlets

Dentro de J2EE existen componentes que permiten combinar varias potencialidades de la plataforma, incluyendo el desarrollo de interfaces de usuario, ejemplo de ello lo constituyen los *portlets*.

Los *portlets* son componentes Web gestionados por un contenedor que tras recibir la petición de un usuario de portal generan y presentan contenidos dinámicos. Permite la personalización, presentación y gestión de la seguridad.(Berbel 2009)

Según la JSR 168 los *portlets* son componentes Web basados en Java, administrados por un contenedor de *portlets*, que procesa solicitudes y genera contenido dinámico. Son mini-aplicaciones Web que pueden agruparse e interaccionar para formar un portal. Estos *portlets* constituyen aplicaciones Web independientes.

Una ventaja excepcional es la capacidad de reagrupar en una sola página funcionalidades diferentes fuentes que operan datos distintos. Es decir, permite agregar en una interfaz uniforme datos y aplicaciones heterogéneas sin obligar al usuario a viajar entre diferentes aplicaciones para realizar las funciones que necesita.

[3] Siglas en inglés de *JavaServer Page Standard Tag Library*

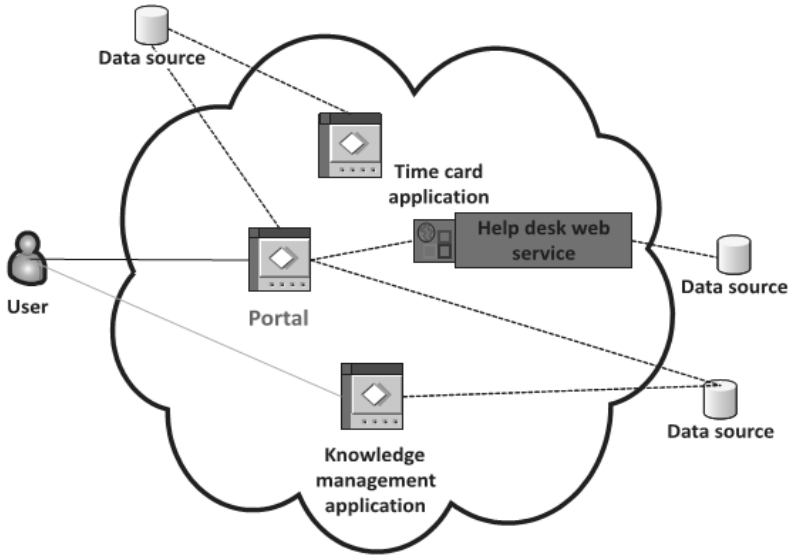

Fig.1: Ejemplo de como el portal de la intranet de una organización recoge el contenido de diferentes fuentes de datos.

Los *portlets* presentan tres características fundamentales:

- Desarrollo independiente. La independencia de los *portlets* permite desarrollarlos de forma separada, y en paralelo con otros desarrollos. Esta característica acelera el tiempo de implementación.
- Personalización. El aspecto final del portal dependerá de los *portlets* que contenga y éstos pueden ordenarse y cambiar esta ordenación de una manera sencilla.
- Seguridad frente a fallos. Un fallo en un *portlet* no se transmite al resto de la aplicación por lo que el único componente que deja de funcionar es el *portlet* que dio el fallo.
- Estados de los *Portlets*: Por último, se puede configurar el estado de la ventana del *portlet*, es decir, indica la cantidad de espacio que el portal asignará al contenido generado por el *portlet*.

Para que un *portlet* pueda mostrarse, debe existir un contenedor de *portlets*, que cumpla con las Java Specification Request (JSR por siglas en inglés) 168 y 286. Un contenedor de *portlets* es el entorno de ejecución de un *portlet*, este maneja su ciclo de vida, sus características, como deben lucir y manipula las peticiones desde el portal (Clay Richardson et al. 2004).

Otro aspecto fundamental es que hoy en día las especificaciones son más maduras y te permiten gestionar con los *portlets* todo lo que se requiere de una aplicación Web. Además existen muchos productos que son verdaderos *marco de trabajo* para el desarrollo de *portlets* que proporcionan muchas funcionalidades de una aplicación Web, ya listas para el uso. Muchas de las funcionalidades que tendría que desarrollar en una aplicación Web están disponibles en estos productos, por ejemplo: gestión de usuarios, roles y grupos, la gestión del menú de aplicaciones y la gestión del diseño de las páginas.

1.3.4 Contenedor de portlets: Liferay.

Liferay es una plataforma para el desarrollo, la integración y la colaboración. Ofrece todas las características necesarias para la creación de portales y aplicaciones Web, todo ello desarrollado sobre una plataforma de código abierto completamente equipada.(Scamercio 2010)

Liferay Portal basa su contenido en *Portlets*. Incluye muchas aplicaciones *portlets* tales como mensajería instantánea, foros y biblioteca de documentos. Soporta múltiples base de datos como: *PostgreSQL*, *MySQL*, *Oracle*, *SQL Server*, *Sybase* e *InterBase*, se puede desplegar con muchos servidores como *Jetty*, *JBoss*, *Sun GlassFish*, *Oracle AS* y *Apache Tomcat*, es multiplataforma ya que puede ejecutarse en cualquier sistema operativo como Windows, Linux y MacOS.(Jonas 2008).

Liferay Portal fue nombrado el mejor producto en portal horizontal en el año 2011, 2012, 2103 y 2014 en la publicación anual a las mejores soluciones *Gartner's Magic Quadrant* de la empresa estadounidense de consultoría e investigación de las TICs: Gartner, además fue nombrado el CMS más popular escrito en Java por Water & Stone en el 2010 y seleccionado por InfoWorld, el mejor portal de código abierto en el mercado(*Liferay* 2014).

Actualmente cuenta con una amplia gama de herramientas y complementos que se pueden adherir a entornos de desarrollos integrados de la plataforma Java como Eclipse y NetBeans.

Liferay Portal está basado en Java y funciona en cualquier plataforma informática capaz de ejecutar el Java Runtime Environment y un servidor de aplicaciones. Por lo antes expuesto, se utilizará para la gestión de los *portlets* del sistema Lifaray Portal en su versión más reciente 6.2.0.

1.4 Marcos de trabajo para desarrollar portlet.

Un marco de trabajo o *framework* por sus siglas en inglés es una pieza de *software* estructural. Se dice estructural porque estructura es el objetivo de un *marco de trabajo* que especifica todo requerimiento funcional. Un marco de trabajo trata de hacer generalizaciones acerca de las tareas comunes, intenta proveer una plataforma donde las aplicaciones pueden ser rápidamente construidas.(Brown and Chad Michael 2007)

Para el desarrollo de *portlets* existen varias tecnologías entre las que se pueden mencionar, Spring *Portlet* MVC, JSF (siglas en inglés de Java Server Faces) y *Liferay* MVC.

Spring *Portlet* MVC.

Spring *Portlet* MVC se basa en Spring MVC para proporcionar un conjunto de controladores que soportan a las aplicaciones *Portlets* de Java (Walls and Breidenbach 2008).

Spring Portlet MVC se diferencia de otros marcos porque es diseñado exclusivamente para el desarrollo de *portlets*; evita las limitaciones que vienen con el desarrollo de *portlets* en un marco web existente. Cuando se usa Spring *Portlet* MVC se integra el marco de trabajo Spring a la aplicación *portlet* de tal forma que se pueden aprovechar todas las características de este potente marco de trabajo, entre las que se pueden mencionar: *IoC, AOP y Bean Factory.*

Java Server Faces

Java Server Faces (JSF por sus siglas en inglés) es un marco de trabajo estándar en Java para construir interfaces de usuarios para aplicaciones Web. Simplifica el desarrollo de las interfaces que es a menudo una de las más difíciles y tediosas tareas del desarrollo en la Web.(Burns et al. 2010)

La tecnología JSF constituye un marco de trabajo de interfaces de usuario del lado de servidor para aplicaciones Web basadas en tecnología Java y en el patrón MVC (Modelo Vista Controlador).

Los principales componentes de la tecnología JSF son:

- Una API y una implementación de referencia para: representar componentes de interfaz de usuario y manejar su estado, manejar eventos, validar en el lado del servidor y convertir datos, definir la

navegación entre páginas, soportar internacionalización y accesibilidad, proporcionar extensibilidad para todas estas características.

- Una librería de etiquetas *Java Server Pages* (JSP) personalizadas para dibujar componentes UI[4] dentro de una página JSP que permite conectar eventos generados en el cliente a código de la aplicación en el lado servidor, mapear componentes UI a una página de datos en el lado servidor, construir una interfaz de usuario con componentes reutilizables y extensibles.

***Liferay* MVC**

Liferay MVC es un marco de desarrollo de *portlets* donde se puede desarrollar en las normas JSR 168 y 286 y desplegar estos en el *Liferay* Portal. Este marco de trabajo tiene muchas características y utiliza las API de *Liferay* para el desarrollo rápido y el fácil uso de los *portlet*. Es ligero y es incorporado a través del *Liferay* Plugins SDK. El *Liferay* IDE tiene soporte para crear y desplegar los *portlets* de *Liferay* MVC en servidores(Prince 2014).

Generalmente cuando se desarrolla *portlets*, los desarrolladores necesitan utilizar más código, pero cuando se usa *Liferay* MVC se tiende a reducir el esfuerzo de desarrollo además que posee muy buen mecanismo para manejar el ciclo de vida de *portlets*.

En la siguiente imagen se muestra la jerarquía del *Liferay* MVC.

[4] Siglas en inglés de *User Interface* (Interfaz de Usuario)

Fig.2: Jerarquía del *Liferay* MVC. Como se puede apreciar la raíz es la Interface *Portlet* que brinda el *portlet* API a través de la JSR 168, la que todo *portlet* debe implementar.

Al utilizar la clase Generic*Portlet* es necesario sobrescribir los métodos *doView* (-), *processAction* (-) para cada uso de las fases tanto acción como render. Al usar *Liferay* MVC no se tiene que sobrescribir cualquiera de los métodos anteriores porque están ya implantados y tienen buen mecanismo diseñado para hacer visitas y realizar la acción *portlet.*

Para el desarrollo del *portlet* se utilizó *Liferay* MVC por las características ventajosas que esta presenta en la implementación del mismo, teniendo en cuenta además que el sistema a desplegar no es muy complejo y que las curvas de aprendizaje de JSF y Spring *Portlet* MVC son muy elevadas en comparación con *Liferay* MVC.

1.5 Portlet en el entorno de Liferay Portal. Alloy UI.

Alloy UI (aleación de interfaz de usuario): Es una biblioteca JavaScript, un marco CSS, un conjunto de recetas HTML y una biblioteca *TagLib,* todo combinado para ayudar a los desarrolladores a través de equipos polivalentes entregar aplicaciones ricas y dinámicas.(*Liferay* 2010)

Alloy es un metaframework de interfaz de usuario que proporciona un sólido y sencillo entorno de desarrollo para la creación de aplicaciones Web con acceso a los tres niveles del navegador; estructura,

estilo y comportamiento. Hace uso de tres lenguajes de diseño Web: HTML, CSS y JavaScript. Ideado por el Director de UI Engineering de *Liferay*, AlloyUI es hoy un proyecto colaborativo que ha recibido importantes contribuciones de la Comunidad *Liferay* y del equipo de UI de Yahoo.

Está basado en la Interfaz de Usuario Yahoo (YUI) y tiene soporte para una variedad de funciones JavaScript avanzadas, diseñadas específicamente para los *portlets*. Está disponible bajo la licencia BSD y está diseñado para aprovechar e incorporar patrones de las mejores bibliotecas para hacer la construcción de aplicaciones Web sólidas y flexibles.

1.6 Monitorización de Liferay.

Un aspecto fundamental en el mantenimiento y prevención de desastres en un sistema es la monitorización de los diferentes servicios que proporciona. La plataforma para el SLD debe componerse de varias estaciones de trabajo para el desarrollo de las tareas e investigaciones a realizar por especialistas y estudiantes en el área de automática por lo que sería de vital importancia impedir posibles fallas en dichas estaciones evitando así interrupciones en los trabajos. Desarrollar un *portlet* que permita la monitorización del sistema en vísperas de tener un control del servidor de la misma sería una buena solución.

Liferay proporciona acceso a estadísticas claves de rendimiento (tiempo máximo por peticiones por ejemplo) para todos los *portlets* y páginas del portal mediante *Java Management eXtension* (JMX por sus siglas en inglés). Estos datos son una importante ayuda para los administradores a la hora de monitorizar el rendimiento del portal y a optimizar mejor los recursos. Por su parte, mediante la funcionalidad de auditoría se permite a los administradores rastrear y gestionar la actividad de los usuarios dentro del portal. En la plataforma J2EE es muy común utilizar JMX.

1.6.1 JMX.

JMX es una tecnología que proporciona herramientas para la gestión y monitorización de aplicaciones basadas en Java como:

- Aplicaciones y dispositivos
- Servidores y servicios

- *Java Virtual Machine* (JVM por sus siglas en inglés).

Permite entre otras cosas consultar o cambiar una determinada configuración, conocer estadísticas y comportamiento de una aplicación, conocer cambios de estado, crear manejadores de recursos y publicarlos en una API[5] e interoperar con otras tecnologías.

El uso de una solución basada en Java tales como JMX ofrece varios beneficios:

- **Facilidad de uso**: Tiene una ventaja significativa sobre tecnologías como SNMP[6] porque un programador con experiencia en Java puede recoger rápidamente los conceptos de JMX y ser productivos. El nivel de conocimientos necesarios para dominar SNMP es significativo: el desarrollador debe conocer el lenguaje de programación utilizado y dominar el conceptos de SNMP, que no son fáciles de entender.
- **Se aprovecha las tecnologías existentes**: Cuando se está construyendo un entorno de administración JMX, no hay necesidad de deshacerse de su estructura de gestión existente: las herramientas de gestión existentes pueden integrar en la tecnología JMX. Proporciona la capacidad para la construcción de la comunicación con cualquier protocolo (como SNMP o HTTP[7]) y la conectividad con cualquier otro medio de transporte (como Java RMI[8]).
- **Orientado a componentes**: Permite construir la solución de administración por componentes. Se puede exponer dispositivos completos o aplicaciones, o simplemente un subconjunto de las características configurables.
- **Alertas, eventos y estadísticas**: Se puede instrumentar la aplicación para insertar información sobre su estado actual de salud, así como estadísticas útiles. Se recopila información de otros recursos administrados como el servidor Web o bases de datos.
- **Soluciones de monitorización rápidos**: Con JMX, cada equipo de desarrollo sólo es responsable de desarrollo de beans gestionados (MBeans) para su aplicación(Sullins and Whipple 2003).

[5] *Application Programming Interface* (Interfaz de programación de aplicaciones)
[6] Siglas en inglés de *Simple Network Management Protocol.* (Protocolo Simple de Administración de Red)
[7] Siglas en inglés de *Hypertext Transfer Protocol* (Protocolo de transferencia de hipertexto)

[7] Siglas en inglés de *Remote Method Invocation.*

1.6.2 JMX Bridge.

JMX Bridge o puente de JMX, es el componente principal de administración de servicios web que actúa como un puente entre la colección de recursos gestionados por JMX y los servicios Web. Dicho puente posee una clase encargada de establecer la conexión con el *MBean Server* y así obtener la información de cada *MBean* solicitado. Luego la clase controladora del *JMX Bridge* gestiona la información y la exporta en formato JSON.

En la plataforma de integración de componentes en el Sistema de Laboratorios Virtuales y a Distancia es preciso usar el *JMX Bridge* debido a que se necesita este puente para acceder a la información de los ***MBeansServerLiferay*** y obtener así los datos de monitoreo que se desean mostrar. Este mecanismo fue necesario debido a que para poder acceder a los MBeans de *Liferay* es preciso utilizar la librería portal-impl.jar la cual no puede ser utilizada por ningún *portlet*. Además los datos se exponen como servicios web que puede ser accedido por cualquier aplicación o dispositivos garantizando la portabilidad y disponibilidad de los datos.

1.7 Servicios Web RESTful.

La Transferencia de Estado Representacional (*Representational State Transfer*, REST por sus siglas en inglés) es un estilo arquitectónico que especifica las limitaciones de la web, como la interfaz uniforme, que si se aplica a un servicio web inducen propiedades deseables, tales como el rendimiento y la escalabilidad que permiten a los servicios funcionar óptimos en entornos Web. En el estilo arquitectónico REST, los datos y la funcionalidad se consideran recursos y se accede mediante Identificadores Uniformes de Recursos (*Uniform Resource Identifiers*, URIs por sus siglas en inglés), típicamente enlaces en la Web. Los recursos son accionados mediante el uso de un conjunto de operaciones simples y bien definidas. El estilo arquitectónico REST restringe cualquier arquitectura a una arquitectura cliente-servidor y está diseñado para utilizar un protocolo de comunicación sin estado, normalmente HTTP(Jendrock et al. 2013).

Los siguientes principios alientan a las aplicaciones RESTful para ser simple, ligero y rápido:

- **Identificación de recursos a través de URI**: Un servicio web RESTful expone un conjunto de recursos que son identificados por URIs.

- **Interfaz uniforme**: Los recursos son manipulados utilizando un conjunto fijo de cuatro operaciones: crear, leer, actualizar y eliminar: *PUT, GET, POST* y *DELETE*.

- **Mensajes Autónomos Descriptivos**: Los recursos son desacoplados de su representación para que su contenido se puede acceder en una variedad de formatos, como HTML, XML, texto plano, PDF, JPEG, JSON, y otros.
- **Interacciones con estado a través de hipervínculos**: Cada interacción con un recurso no tiene estado; es decir, los mensajes de petición son autónomos.

1.7.1 Spring RESTful.

Los servicios web REST se han convertido en el número uno de los medios para la integración de aplicaciones en la web. Spring a partir de su versión 3.0 ofrece soporte para servicios Web RESTful. Antes de que Spring soportara REST, los programadores usaban otras implementaciones, como *Restlet, RestEasy*, y *Jersey,* para ayudar a construir servicios web RESTful en el mundo Java(Ming Huang and Fei Wu 2010).

El apoyo REST se integra a la perfección en la capa de Spring MVC y puede ser adoptada fácilmente por las aplicaciones que se construyen en este marco de trabajo. Las características principales de Spring REST son:

- Anotaciones, como @RequestMapping y @PathVariable, para apoyar la identificación de recursos y asignaciones URIs.
- Posee la clase ContentNegotiatingViewResolver para apoyar diferentes representaciones con diferentes tipos de contenido MIME[9].
- Perfecta integración en la capa original MVC con un modelo de programación similar.

[9] Siglas en inglés de *Multipurpose Internet Mail Extensions* (Extensiones Multipropósito de Correo de Internet).

1.8 Entorno de Desarrollo Integrado.

Un Entorno de Desarrollo Integrado (IDE) es una herramienta de *software* dedicada exclusivamente al desarrollo de programas informáticos, brindando una serie de complementos que facilitan el desarrollo ágil de *software*. Para el desarrollo de aplicaciones en el lenguaje de programación Java se utilizan fundamentalmente dos IDEs: Eclipse y NetBeans, sin descartar a IntelliJ, pero este por ser comercial y de elevado costo no posee un buen respaldo en la comunidad.

Eclipse Foundation (http://www.eclipse.org/) es una comunidad de código abierto con proyectos enfocados en proveer una plataforma de desarrollo de marcos de trabajo y herramientas para desarrollar y gestionar los ciclos de vida en el desarrollo de *software*. La plataforma de desarrollo es denominada Proyecto Eclipse y la última versión del IDE es la 4.4 llamada Eclipse Luna que es además la utilizada para el desarrollo de la plataforma de integración de componentes del SLD. Se optó por usar esta versión, ya que en la práctica resultó ser más económico en la utilización de los recursos de hardware.

1.9 Marco de trabajo de desarrollo.

En la plataforma J2EE existen marcos de trabajos para diversos propósitos, dígase manipular bases de datos, desarrollar aplicaciones web, visualizar datos y gestionar peticiones AJAX (acrónimo del inglés Asynchronous JavaScript And XML). Dicha plataforma cuenta además con marcos de trabajo muy populares para desarrollar aplicaciones web, uno de los más populares en la comunidad de Java es Spring Framework.

Spring es un marco de trabajo de código abierto, creado por Rod Johnson y descrito en su libro "*Expert One-on-One: J2EEDesign and Development"*. Incluye varias librerías que ayudan a solventar problemas comunes en el desarrollo de aplicaciones. Spring soporta la Programación Orientada a Aspecto, implementa el patrón MVC, separando la interfaz de usuario, los controladores y la capa de acceso a datos y utiliza el Objeto de Acceso a Datos (DAO, siglas en inglés de Data Access Object), que es un patrón de diseño paragestionar las bases de datos (Walls and Breidenbach 2008).

Entre las características principales de este marco de trabajo:

- Es ligero: en términos de tamaño y los gastos generales Spring es muy ligero. El grueso del marco de trabajo se puede distribuir en un único archivo .JAR que pesa poco más de 2,5 MB.

- Inyección de dependencia: promueve acoplamiento flexible a través de una técnica conocida como inversión de control (IoC).
- Orientado a aspectos: viene con un soporte rico para la programación orientada a aspectos (AOP en sus siglas en inglés) que permita al desarrollo coherente separando la lógica empresarial de la aplicación de los servicios del sistema (como la auditoría y gestión de transacciones).
- Contenedor: es un contenedor en el sentido de que contiene y administra el ciclo de vida y la configuración de los objetos de aplicación. Puede declarar como cada uno de sus objetos de aplicación se debe crear, cómo debe configurarse, y cómo deben estar asociados entre sí.
- Framework: hace posible configurar y componer aplicaciones complejas a partir de componentes simples. Los objetos de aplicación se componen de forma declarativa, por lo general en un fichero XML. Spring se compone de varios módulos bien definidos los cuales se representan en la Figura 3.

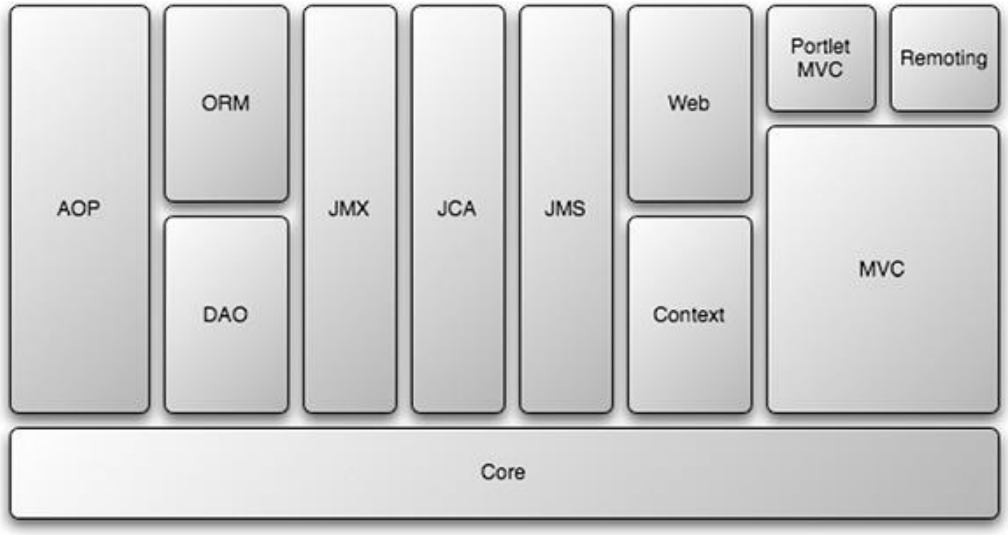

Fig.3: Módulos de Spring. Ejemplifica como Spring se puede utilizar por módulos y no representa un marco de trabajo “Todo o nada”.

Cuando se toma en su conjunto, estos módulos proporcionan todo lo que se necesita para desarrollar aplicaciones empresariales. Spring ofrece puntos de integración con otros marcos de trabajos y bibliotecas por lo que no se tiene que escribir por cuenta propia.

Se seleccionó Spring Framework en su versión 4.0 para el desarrollo del *JMX Bridge* o puente JMX para el monitoreo de la plataforma, pues además de ser un marco de trabajo de código abierto, puede ser

usado en cualquier aplicación desarrollada en Java y existen variadas extensiones para la construcción de aplicaciones web sobre la plataforma Java EE. Ofrece facilidad de integración con los estándares J2EE y herramientas comerciales existentes.

1.10 Sistema Gestor de Bases de Datos(SGBD).

Un SGDB permite la utilización y/o la actualización de los datos almacenados en una o varias bases de datos por uno o varios usuarios desde diferentes puntos de vista y a la vez. Los programas de aplicación operan sobre los datos almacenados en la base de datos utilizando las facilidades que brindan los SGBD, los que en la mayoría de los casos, poseen lenguajes especiales de manipulación de la información que facilitan el trabajo de los usuarios.(Gibert 2007)

Algunos ejemplos de gestores más conocidos y utilizados son: MySQL, Microsoft SQL Server, PostgreSQL.

MySQL es un sistema gestor de bases de datos relacionales, rápido, sólido y flexible. Es idóneo para la creación de bases de datos con acceso desde páginas Web dinámicas, así como para la creación de cualquier otra solución que implique el almacenamiento de datos, posibilitando realizar múltiples y rápidas consultas. Está desarrollado en C y C++, facilitando su integración en otras aplicaciones desarrolladas también en esos lenguajes. Es un sistema cliente/servidor y es uno de los sistemas gestores de bases de datos más utilizado en la actualidad, utilizado por grandes corporaciones como Yahoo! Finance, Google, Motorola, entre otras.

Microsoft SQL Server es un sistema gestor de base de datos relacionales producido por Microsoft. Es un sistema cliente/servidor que funciona como una extensión natural del sistema operativo Windows. Proporciona integridad de datos, optimización de consultas, control de concurrencia y backup y recuperación. Es relativamente fácil de administrar a través de la utilización de un entorno gráfico para casi todas las tareas de sistema y administración de bases de datos. Utiliza servicios del sistema operativo Windows para ofrecer nuevas capacidades o ampliar la base de datos, tales como enviar y recibir mensajes y gestionar la seguridad de la conexión.

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD y con su código fuente disponible libremente. Es el sistema de gestión de bases de datos de código abierto

más potente del mercado y en sus últimas versiones no tiene nada que envidiarle a otras bases de datos comerciales.(Martinez 2010)

PostgreSQL utiliza un modelo cliente/servidor y usa multiprocesos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando. Algunas de las características más importantes y soportadas por PostgreSQL son la integridad referencial, la replicación asincrónica/sincrónica y las copias de seguridad en caliente. Es además Unicode, posee una completa documentación, licencia BSD y disponible para Linux y UNIX en todas sus variantes (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64) y Windows 32/64bit.

A continuación se muerta un tabla comparativa entre los sistemas gestores de base de datos anteriormente mencionados.

Tabla 1: Comparación entre Sistemas Gestores de Base de Datos.

	Libre	Código abierto	Multiplataforma	Versión Actual	Costo aproximado
MySQL	Si	La versión Comunity server	Si	V5.6	Enterprise Edition-$2000usd, Standard Edition-$5000usd Anual
PostgreSQL	Si	Si	Si	v9.2.2	Libre
Microsoft SQL Server	No	No	No	SQL Server 2012	$2500 usd

Por las características antes mencionadas y basándose en la comparación realizada se selecciona a PostgreSQL en su versión 9.1 como el SGBD más adecuado para el desarrollo del sistema que se desea obtener pues muestra un excelente rendimiento a la hora de trabajar con grandes volúmenes de datos y posee además una comunidad muy amplia donde se puede encontrar información variada y actualizada.

1.11 Metodologías de desarrollo de software.

Una metodología es una colección de procedimientos, técnicas, herramientas y documentos auxiliares que guiarán a los desarrolladores de sistemas a elegir las técnicas más apropiadas en cada momento del

proyecto y también a planificarlo, gestionarlo, controlarlo y evaluarlo.(Avison and Fitzgerald 1995). Existen dos clasificaciones de metodologías: tradicionales y ágiles.

Toda metodología debe ser adaptada al contexto del proyecto, teniendo en cuenta los recursos técnicos y humanos, tiempo de desarrollo y tipo de sistema. Para guiar el presente proceso de desarrollo de *software* debido a las ventajas que presenta se selecciona una metodología ágil pues aportan mayor flexibilidad, retroalimentación de la calidad, eficacia y rapidez, reduciendo el margen de error y los riesgos del proyecto.

1.11.1 Metodología OpenUp.

OpenUp es un proceso unificado ágil y liviano, que aplica un enfoque iterativo e incremental dentro de un ciclo de vida estructurado y contiene un conjunto mínimo de prácticas que ayuda al equipo a ser más efectivo desarrollando *software*. Abraza una filosofía pragmática y ágil de desarrollo, que se enfoca en la naturaleza colaborativa del desarrollo de *software*. Se trata de un proceso con herramientas neutrales y de baja ceremonia que puede extenderse para alcanzar una amplia variedad de tipos de proyectos(Gimson 2012).

Es un proceso de desarrollo iterativo del programa informático que es mínimo, completo, y extensible, significando cada uno:

- Mínimo: Solo incluye el contenido del proceso fundamental.
- Completo: Puede ser manifestado como proceso entero para construir un sistema.
- Extensible: Puede ser utilizado como base para agregar o para adaptar más procesos.

Se utilizó OpenUp para desarrollar la solución pues incluye las características fundamentales del Proceso Unificado de Desarrollo (RUP por sus siglas en inglés). Es un proceso de desarrollo de *software* de código abierto diseñado para pequeños equipos organizados. Se caracteriza además por estar centrado en la arquitectura y guiado por los casos de uso y utiliza el Lenguaje Unificado de Modelado (UML por sus siglas en inglés) para el modelado de sistemas.

1.12 Herramienta CASE. Visual Paradigm.

Las herramientas CASE (del inglés *Computer Aided Software Engineering*, que se traduce al español como: Ingeniería de *Software* Asistida por Computadora), son herramientas con el objetivo de incrementar la productividad y la calidad de los productos de *software*, mejorar la planificación del proyecto así como reducir el tiempo y coste de su desarrollo.

Una herramienta CASE muy utilizada en la universidad es el Visual Paradigm, este soporta el ciclo de vida completo de desarrollo de *software*, desde la fase de análisis hasta el despliegue del mismo. Permite realizar ingeniería directa o inversa sobre el programa informático y es capaz de, a partir de un modelo relacional en diferentes sistemas gestores de base de datos, desplegar todas las clases asociadas a las tablas.(Incencio 2014)

Visual Paradigm ofrece:

- Navegación intuitiva entre la escritura del código y su visualización.
- Potente generador de informes en formato PDF/HTML.
- Documentación automática Ad-hoc.
- Ambiente visualmente superior de modelado.
- Sofisticado diagramador automáticamente de layout.

Se selecciona a Visual Paradigm for UML v8.0 *Enterprise Edition* v5.0 como herramienta CASE, para ser usada en el desarrollo de la aplicación, teniendo en cuenta las características y beneficios que brinda para la construcción de *software*, especialmente referente al modelado. Además de ello, se tuvo en cuenta que constituye la herramienta que utiliza la universidad ya que se encuentra con licencia libre.

Conclusiones del capítulo.

Después de analizada la bibliografía científica sobre el desarrollo de plataformas para la integración de componentes en un SLD, se pudo identificar que las soluciones existentes no satisfacen las necesidades del sistema que actualmente se desea implementar por lo que se propone la implementación de una nueva plataforma haciendo uso de las tecnologías analizadas, entre las que se destacan los *portlets* por

su portabilidad, reusabilidad y escalabilidad y de *Liferay* Portal por ser uno de los mejores contenedores de *portlets* existentes.

Una vez analizadas las tendencias sobre las herramientas metodologías para el desarrollo de *software,* se seleccionó Eclipse como entorno de desarrollo integrado y como marco de trabajo se optó por Spring Framework. Se estableció PostgreSQL como SGBD, el cual se encargará de almacenar los datos persistentes. El proceso de desarrollo estará guiado por la metodología de desarrollo de *software* OpenUp y se eligió como herramienta CASE al Visual Paradigm. Finalmente se concluye que con los datos tributados por la investigación teórica y los principales elementos de las herramientas a emplear, se cuenta con suficientes elementos para solucionar la problemática planteada en este documento.

Capítulo II Análisis y diseño de la plataforma para la integración de componentes en el Sistema de Laboratorios Virtuales y a Distancia.

Introducción.

En el presente capítulo se describe la solución propuesta para obtener una noción en general de la Plataforma para la integración de componentes en el Sistema de Laboratorios Virtuales y a Distancia. Se identifican los requisitos funcionales y no funcionales del sistema, los casos de uso, patrones y se modelan los diagramas pertinentes para proceso de diseño logrando así guiar el proceso de desarrollo de *software*.

2.1 Modelo conceptual

Un paso esencial en el análisis de construcción de *software* es descomponer el problema en conceptos u objetos individuales en un modelo conceptual. El modelo conceptual de la plataforma para la integración de componentes en el SLD se muestra en la figura 4, el mismo explica cuáles son y cómo se relacionan los conceptos relevantes en la descripción del problema.

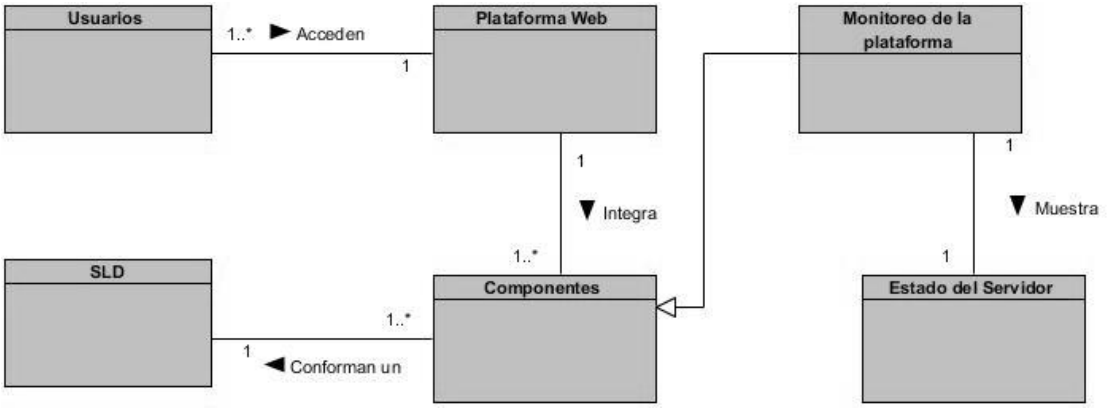

Fig.4: Modelo conceptual de la plataforma para la integración de componentes en el Sistema de Laboratorios Virtuales y a Distancia.

Descripción de los conceptos del modelo de la plataforma:

Usuario: Persona que requiere utilizar herramientas tecnológicas disponibles en la red para realizar prácticas de control automático.

Plataforma Web: Portal Web.

Componentes: Aplicaciones para el consumo propio de la plataforma o un conjunto de laboratorios virtuales.

SLD (Sistema de Laboratorios a Distancia): Colección de laboratorios virtuales que pueden ser utilizados a distancia gracias a la Web.

Monitoreo de la plataforma: Componente que muestra los datos referentes al estado del servidor con el objetivo de monitorizar el rendimiento del portal.

Estado del servidor: Refiere al estado de la memoria RAM, estado del CPU, estado de la memoria de intercambio, estado de las peticiones al portal y a los componentes.

2.2 Especificación de los Requisitos del sistema.

El principal objetivo de determinar las funcionalidades que tendrá el sistema es guiar el desarrollo de la aplicación hacia la obtención de un diseño que cumpla con las expectativas requeridas por el cliente. La especificación de requisitos es la actividad que enumera y describe los requisitos necesarios para el sistema. Se dividen en dos grupos los funcionales (expresan capacidad) y los no funcionales (expresan cualidad).

2.2.1 *Requisitos Funcionales del sistema.*

Los requisitos funcionales (RF) de una aplicación son las capacidades o condiciones que el sistema debe cumplir para satisfacer las necesidades primarias del cliente.

A continuación se definen los requisitos funcionales identificados para el desarrollo del sistema.

RF1: Autenticar usuario.

RF2: Adicionar usuario.

RF3: Editar usuario.

RF4: Eliminar usuario.

RF5: Buscar usuario por nombre, segundo nombre, apellido, nombre de usuario, dirección de correo y estado.

RF6: Asignar permisos a los usuarios.

RF7: Adicionar *portlet*.

RF8: Activar *portlet*.

RF9: Desactivar *portlet*.

RF10: Eliminar *portlet*.

RF11: Mostrar porciento del rendimiento de la memoria RAM mediante una gráfica de pastel.

RF12: Mostrar porciento del rendimiento de la memoria de intercambio mediante una gráfica de pastel.

RF13: Mostrar porciento del rendimiento del CPU mediante una gráfica de hilos.

RF14: Mostrar porciento de las peticiones del portal mediante una gráfica de pastel.

RF15: Mostrar porciento de las peticiones de cada *portlet* mediante una gráfica de pastel.

2.2.2 *Requisitos no Funcionales del sistema.*

Los requisitos no funcionales (RNF) son propiedades o cualidades que el producto debe tener, aunque no formen parte de su función.

A continuación se definen los requisitos no funcionales identificados para el desarrollo del sistema.

RNF 1. Usabilidad:

- Se utilizó tonos de color azul en el diseño de la plataforma, de esta forma no se le dificulta el acceso a los usuarios con problemas de distinción de color.

- La plataforma es legible, el color negro y azul de los textos contrastan con los fondos blancos y azules claros así como los textos de color fuente de color blanco contrastan con los fondos azules fuertes.
- Posee una interfaz visible, los menú Acceder, Inicio, Laboratorios, Biblioteca, Acerca de y Mapa del sitio están a la vista del usuario, no se hace uso de menú desplegables ni indicaciones oculta.
- El proceso de aprendizaje de la plataforma es mínima, el usuario desde el primer momento que se autentica encuentran todas las funcionales de esta de manera cómoda por lo que no es un sitio difícil de navegar.

RNF 2. Requisitos de Software:

Debe estar instalado:

En los ordenadores de los usuarios:

- El navegador Web Mozilla Firefox versión 32.0.

En las estaciones que fungen como servidores:

- Servidor Web, Apache Tomcat 7.0.2 con la aplicación *Liferay* Portal 6.2.0.

RNF 3. Requisitos de Hardware

- El servidor de BD debe tener más de 1GB de RAM y 80GB o más de capacidad de disco duro.
- El servidor de aplicaciones debe tener 2 GB de RAM o superior y 80GB o más de capacidad de disco duro. Un microprocesador igual o superior al Intel Dual Core, con una velocidad de procesamiento de 2.00 GHz en adelante.

RNF 4. Restricciones en el diseño y la implementación:

- Se utilizará el estándar de codificación propuesto por el lenguaje de programación Java.
- Para la programación en el lado del cliente se utilizará JavaScript con la librería Alloy UI para enriquecer la aplicación.

- Para las llamadas asíncronas al servidor se utilizará la tecnología AJAX con asistencia de la librería Alloy UI.
- Para el correcto funcionamiento del JMXBridge es necesario que la librería ATACH.DLL esté disponible en la máquina virtual.

RNF 5. Interfaz:

- La interfaz de la plataforma estará basada en *portlets*.

2.3 Modelo de Casos de Uso del Sistema.

El modelo de casos de uso del sistema permite que los desarrolladores y los clientes lleguen a un acuerdo sobre los requisitos, es decir sobre las condiciones y posibilidades que debe cumplir el sistema. Los CU describen las interacciones entre uno o más usuarios y el sistema, con el fin de proporcionar un resultado observable de valor para el actor.

Los CU describen las interacciones entre uno o más usuarios y el sistema, con el fin de proporcionar un resultado observable de valor para el actor (Framework 2011)

2.3.1 *Actores de sistema.*

A continuación se enuncian y describen los actores que interactúan con el sistema.

Tabla 2: Descripción de los actores del sistema.

Actor del sistema	Descripción
Usuario	Representa una persona. Es la encargada de autenticarse en el sistema.
Administrador	Representa una persona. Es la encargada de gestionar los usuarios que acceden al sistema, administrar los *portlet* que se visualizan en el mismo y además supervisa el monitoreo del servidor de la plataforma.

2.3.2 *Patrones de casos de uso.*

Los patrones de casos de uso son comportamientos que deben existir en el sistema, ayudan a describir qué es lo que el sistema debe hacer, es decir, describen el uso del sistema y cómo este interactúa con los usuarios. Estos patrones son utilizados generalmente como plantillas que describen como debería ser estructurados y organizados los casos de uso. Son patrones que capturan mejores prácticas para modelar casos de uso.

Los patrones de CU utilizados fueron:

- Múltiples Actores: Este patrón se pone de manifiesto en el CU: Autenticar usuario ya que al mismo ingresan más de un actor.
- CRUD Total o Completo: Este patrón se evidencia en el CU: Gestionar usuario ya que permite modelar las diferentes operaciones para gestionar una entidad de información, tales como adicionar, modificar, buscar y eliminar.

2.3.3 *Diagrama de casos de uso del sistema.*

Un diagrama de casos de uso muestra la relación entre los actores y los casos de uso del sistema. Un caso de uso puede agrupar más de un requisito funcional. Los RF de los sistemas definidos anteriormente quedan agrupados en 4 CU presentados en la tabla 3.

Tabla 3: CU del sistema.

Referencia a requisitos	Nombre del caso de uso
RF 1.	CU 1: Autenticar usuario
RF 2, RF 3, RF 4, RF 5, RF 6.	CU 2: Gestionar usuario
RF 7, RF 8, RF 9, RF 10.	CU 3: Administrar *portlet*
RF 11, RF 12, RF 13, RF 14, RF 15.	CU 4: Mostrar estado del servidor

La siguiente figura (Fig. 5) evidencia el modelo de casos de uso de la plataforma para la integración de componentes en el SLD. Se evidencia en la misma los actores del sistema relacionados con los casos de uso que inicializan.

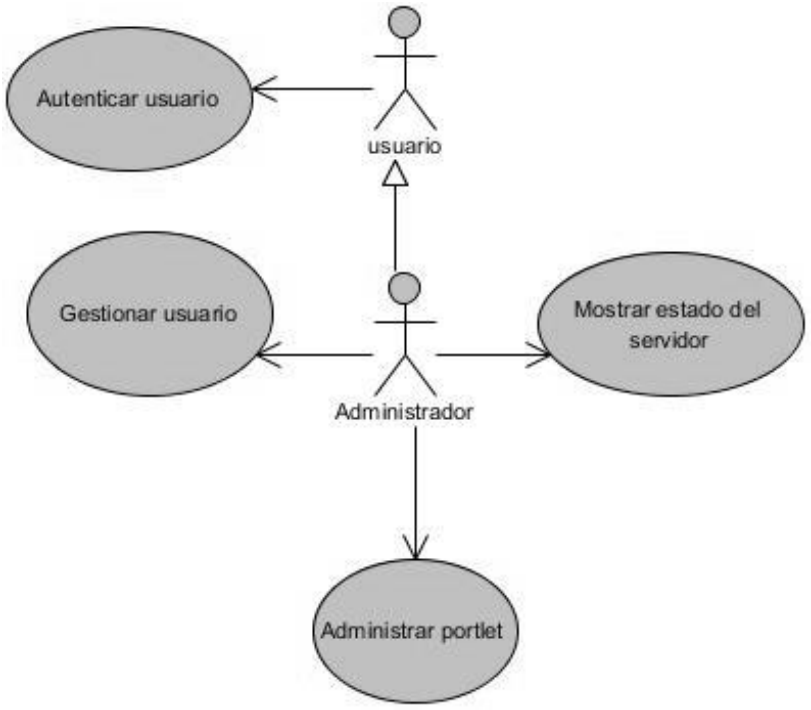

Fig.5: Diagrama de casos de uso del sistema.

2.3.4 *Descripción de casos de uso del sistema.*

La especificación de los casos de uso hace referencia a la descripción de cada una de las partes antes definidas para lograr un mayor entendimiento del funcionamiento de las mismas. Muestran detalladamente paso a paso cómo funciona el sistema ante una acción ejercida por el actor, en este caso, el administrador del sistema. A continuación se describirá el CU Mostrar estado del servidor, por ser el más significativo desde el punto de vista arquitectónico.

Tabla 4: CU: Mostrar estado del servidor.

Objetivo	Mostrar en qué estado se encuentran los componentes del servidor de la plataforma para tener un control del mismo.
Actores	Administrador del sistema (inicia)
Resumen	El caso de uso inicia cuando el actor selecciona la opción ‘’Monitoreo de la plataforma’’. Se muestran cinco gráficos en porciento con la información

	referente a los rendimientos de la memoria RAM, memoria de intercambio, CPU, peticiones al portal y a los *portlets* respectivamente. Finaliza así el CU.		
Complejidad	Alta		
Prioridad	Crítico		
Precondiciones	El usuario debe tener el rol de administrador del sistema.		
Postcondiciones	Se muestran los estados de las memorias, el CPU, peticiones al portal y peticiones a los *portlets* del sistema.		
Flujo de eventos			
Flujo básico Mostrar estado del servidor de la plataforma.			
	Actor	**Sistema**	
1.	El administrador del sistema selecciona la opción ´´Monitoreo del servidor´´		
2.		Muestra la información referente al estado de las memorias, CPU, peticiones al portal y peticiones a los *portlets* finalizándose así el caso de uso.	
3.			
Relaciones	**CU Incluidos**	No precede.	
	CU Extendidos	No procede.	
Requisitos no funcionales	No procede.		
Asuntos pendientes	No procede.		

2.4 Arquitectura del sistema.

La arquitectura de *software* define la estructura de un sistema. Esta estructura se constituye de componentes, módulos o piezas de código que nacen de la noción de abstracción, cumpliendo funciones específicas e interactuando entre sí con un comportamiento definido. Los componentes se organizan de acuerdo a ciertos criterios, que representan decisiones de diseño. En los últimos años la arquitectura de *software* se ha debatido en dos corrientes fundamentales: los estilos arquitectónicos y los patrones arquitectónicos.

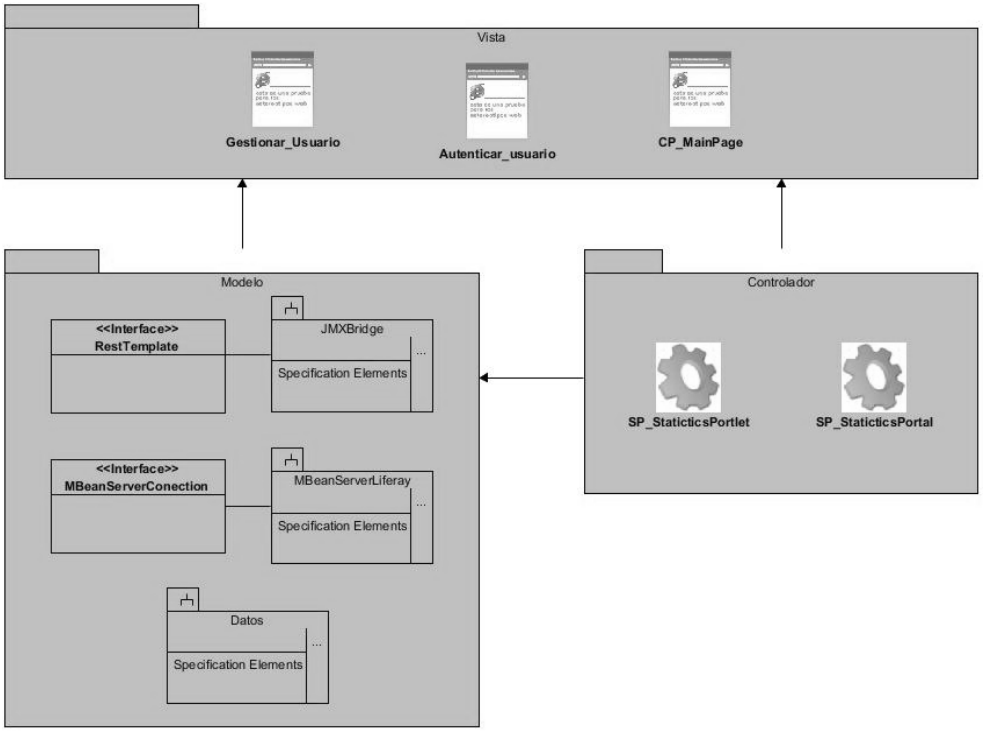

Fig.6: Vista lógica de la arquitectura de la plataforma.

2.4.1 *Patrones arquitectónicos.*

Un patrón arquitectónico es la expresión de un esquema estructural de organización para sistemas de *software*, que proporciona un conjunto de subsistemas predefinidos, especifica sus responsabilidades e incluye normas y directrices para las relaciones entre ellos(Garland and Anthony 2003).

Modelo Vista Controlador

El patrón MVC permite separar los componentes de aplicación en tres niveles, interfaz, lógica y modelo (acceso a datos). Es una especialización de un modelo de capas, con la diferencia que se usa para entornos Web como patrón por excelencia.

Entre las ventajas que este modelo de arquitectura presenta, es importante mencionar las siguientes:

- Existe una clara separación entre los componentes de un programa; lo cual nos permite implementarlos por separado.
- Se puede reemplazar los componentes del sistema (el Modelo, la Vista o el Controlador) sin mucha dificultad.(Catalani 2007).

Las responsabilidades de las capas en MVC son:

Capa modelo: Representa los datos del programa, los maneja y controla todas sus transformaciones.

Capa controladora: Es el eje central de nuestra arquitectura, encargada de gestionar todas las peticiones, validar los inputs recibidos y dirigir cualquier petición de cualquier tipo.

Capa vista: Es la respuesta de cada controlador y lo que se le presenta al usuario final, se puede comunicar con el controlador y el modelo (en algunas ocasiones).

A continuación, en la figura 6 se muestra el funcionamiento del patrón de diseño MVC evidenciándose en él, las tres capas anteriormente mencionadas:

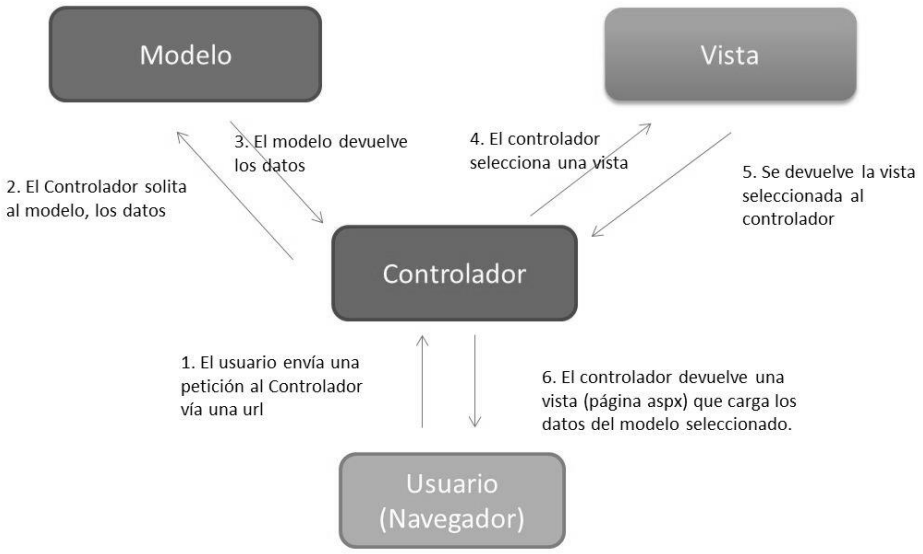

Fig.7: Funcionamiento del patrón Modelo Vista Controlador.

Cliente-Servidor

Desde el punto de vista funcional, se puede definir la computación Cliente/Servidor como una arquitectura distribuida que permite a los usuarios finales obtener acceso a la información en forma transparente aún en entornos multiplataforma(Márquez and Zulaica 2004).

En el modelo cliente servidor, el cliente envía un mensaje solicitando un determinado servicio a un servidor, y este envía uno o varios mensajes con la respuesta (ver Fig.8).

Fig. 8: Modelo Cliente/Servidor(Márquez and Zulaica 2004).

La plataforma para la integración de componentes en el SLD presenta los patrones arquitectónicos MVC y Cliente-Servidor. Este último se manifiesta en el paradigma petición-respuesta, dado que es una aplicación Web a la que se accede por el protocolo HTTPS.

2.4.2 *Diagramas de clases del diseño.*

Los diagramas de clases muestran cómo se lleva a cabo la colaboración entre las clases para dar cumplimiento a un requisito determinado. Para la elaboración de estos diagramas se hace uso de estereotipos que ayudan a representar de manera más fácil la función y el carácter de las clases dentro de la realización del requisito.

En la figura 9 se muestra el diagrama de clases del diseño del CU Mostrar estado del servidor, donde se evidencia el patrón arquitectónico MVC ya que se desglosa el mismo en 3 paquetes, Vista, Controladora y Web Services o Modelo. En el paquete Vista se encuentra la interfaz que interactúa con el usuario, compuesta por un formulario con todos los *portlets* a monitorear. La misma envía peticiones a la clase **`PortletStatistics`** del paquete Controlador, estas peticiones se tramitan al JMXBridge el cual obtiene la información de los MBeansServer para luego reenviarla a la clase del paquete Controlador nuevamente para mostrar esa información mediante la interfaz del paquete Vista.

En la figura 10 se visualiza el diagrama de clases de la API JMXBridge del paquete Web Services. Este está compuesto por dos paquetes, Vista y Controlador, donde en el primero se tiene un cliente que hace un link con la clase controladora **`DispatcherServlet`** que a su vez redirecciona a la controladora **`StatisticsController`**. Esta usa las clases **`PortletstatisticsImpl`** y **`PortalStatisticsImpl`**, ambas utilizan **`JMXUtils`** que a su vez usa la librería ATACH.DLL. Finalmente se devuelve la información formato JSOM.

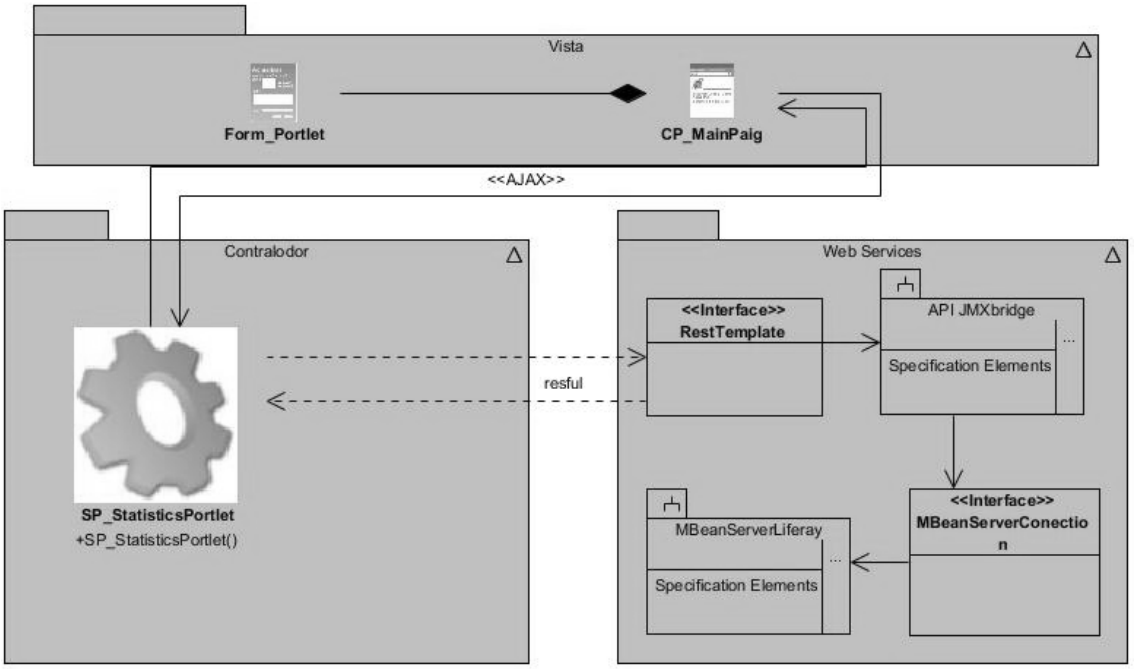

Fig.9: Diagrama de clases del diseño del CU Mostrar estado del servidor.

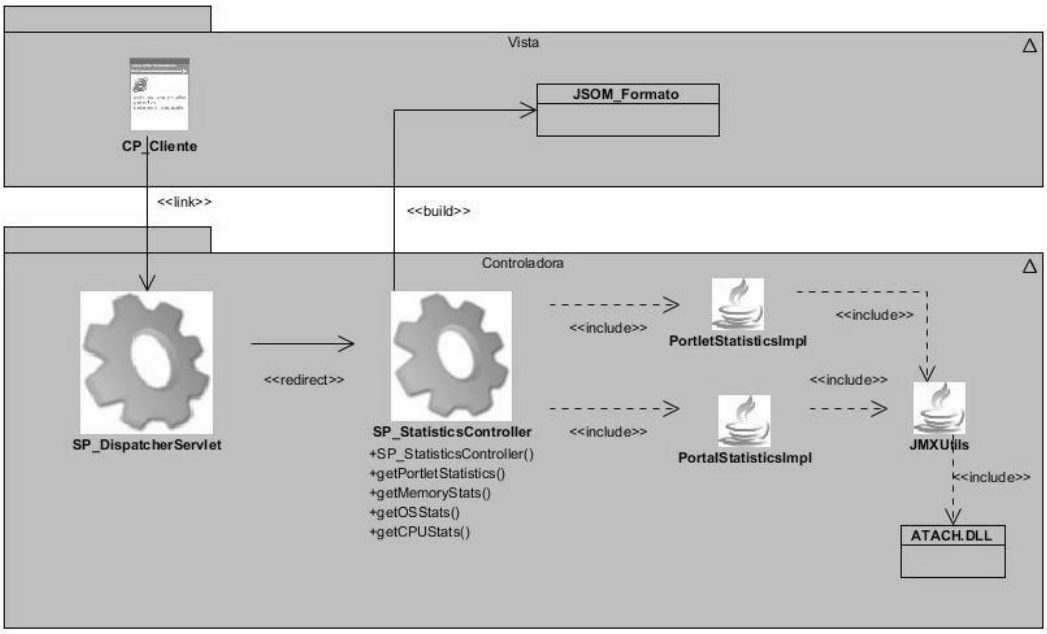

Fig.10: Diagrama de clases del diseño de JMX Bridge.

2.4.3 *Patrones de diseño.*

Los patrones de diseño de *software* son el esqueleto de las soluciones a problemas comunes en el desarrollo de *software*. Brindan una solución a problemas de desarrollo de *software* que están sujetos a contextos similares. Los patrones utilizados en la presente investigación fueron, GRASP, más conocidos como patrones generales de asignación de responsabilidades, los patrones GoF[10] y el Modelo Vista Controlador (MVC).

GRASP

Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones.(Larman 1999). A continuación se mencionan los utilizados en el sistema.

- **Patrón Experto:** Se basa en asignar una responsabilidad a la clase que cuenta con la información necesaria para cumplir dicha responsabilidad. Permite conservar el encapsulamiento, ya que los

[10] Conocidos como patrones de la pandilla de los cuatro (GoF, siglas en inglés de Gang of Four).

objetos se valen de su propia información para realizar lo que se le oriente. Este patrón se evidencia en la clase **`JMXUtils`**, el cual inicializa el **`MBeanServerConnection`** y por ende es el encargado de verificar si el **`MBeanServer`** de *Liferay* Portal está en ejecución para establecer una conexión con el mismo.

- **Patrón Creador:** El patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. El objetivo fundamental de este patrón es encontrar un creador que se conecte con el objeto producido en cualquier evento. Al comportarse como creador, se da soporte al bajo acoplamiento. Teniendo en cuenta que se utilizó el marco de trabajo de Spring, se hace uso del IoC Container, del BeanFactory y Spring Contexto los cuales se encargan de instanciar las clases anotadas dentro del paquete "com.uci.*Liferay*.jmx.bridge", dicha configuración se encuentra en el archivo "servlet-context.xml" del **`JMXBridge`** como se muestra a continuación:

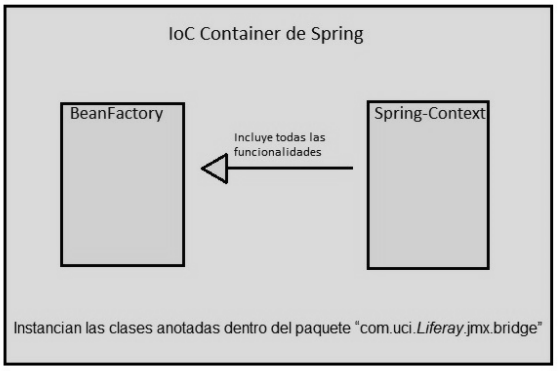

Fig.11: Patrón de diseño GRASP: Creador

- **Patrón Bajo Acoplamiento:** Este patrón asigna una responsabilidad para mantener el bajo acoplamiento. La idea es tratar de que una clase no dependa de muchas otras, así esa clase no tendrá muchas dependencias, lo que facilitará la reutilización de la misma y se reducirá el impacto de los cambios. Se evidencia este patrón manteniendo las relaciones mínimas entre clases.

- **Patrón Alta Cohesión:** Una clase con alta cohesión tiene un número relativamente pequeño de métodos, con funcionalidades altamente relacionadas, y no realiza mucho trabajo, colaborando con otros objetos para compartir el esfuerzo si la tarea es extensa. Las clases con alta cohesión son

relativamente fáciles de mantener, entender y reutilizar. En el sistema se hace evidente la alta cohesión ya que las clases tienen una función bien definida dentro del sistema. Alta cohesión y bajo acoplamiento se retroalimentan juntos, gracias a que Java es un lenguaje Orientado a Objetos equilibramos las funcionalidades de tal forma que una clase no estuviera sobrecargada y que en caso de mantenimiento o errores estos fueran encontrados fácilmente. Por ejemplo, el controlador "**`StatisticsController`**" utiliza dos objetos para acceder a las estadísticas del portal y de los *portlets* respectivamente, estos a su vez dependen únicamente de un objeto **`JMXUtils`** para acceder a las propiedades de los **`MBeans`** publicados por *Liferay*. En la siguiente figura se muestra un pequeño fragmento del diagrama de clases del diseño del API JMXBridge donde se evidencia el uso de los patrones: alta cohesión y bajo acoplamiento.

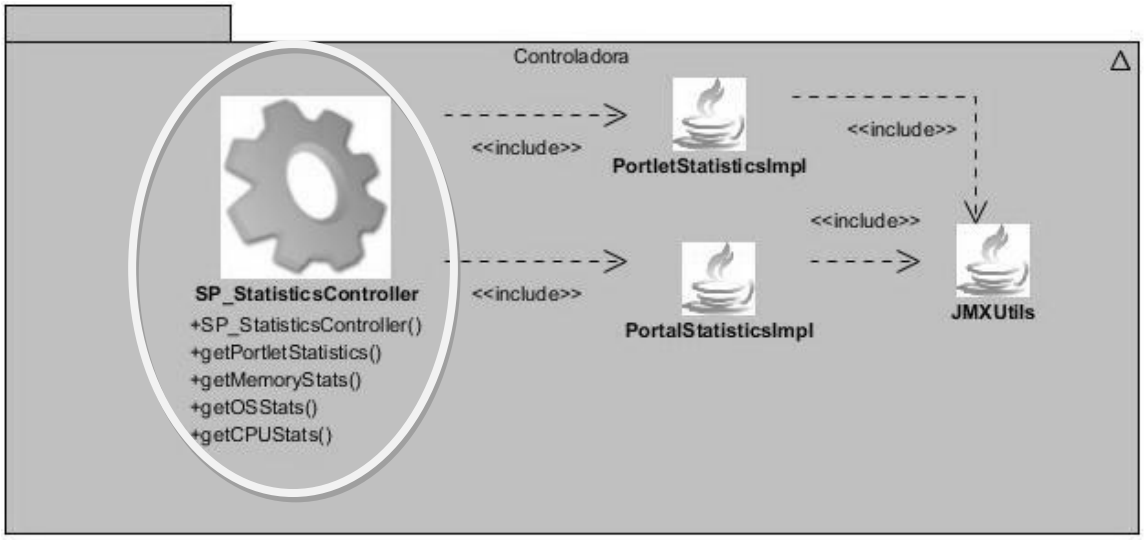

Fig.12: Patrones de diseño GRASP: Alta cohesión y bajo acoplamiento.

- **Patrón Controlador:** Para manejar y controlar los eventos del sistema. Con la utilización de este patrón se logra separar la lógica de negocio de la capa de presentación. De esta manera se logra un mayor control sobre el sistema y se favorece la reutilización de código. El sistema utiliza el **`DispatcherServlet`** de Spring como controlador frontal, una vez llegada una petición asigna el controlador "**`StatisticsController`**" para que la responda, en dependencia de la URL solicitada se retorna la información correspondiente en formato JSON. En el pequeño fragmento

del diagrama de clases del diseño del API JMXBridge que a continuación se muestra, se evidencia lo anteriormente planteado, resaltando en dicho fragmento la clase donde el patrón Controlador se pone de manifiesto.

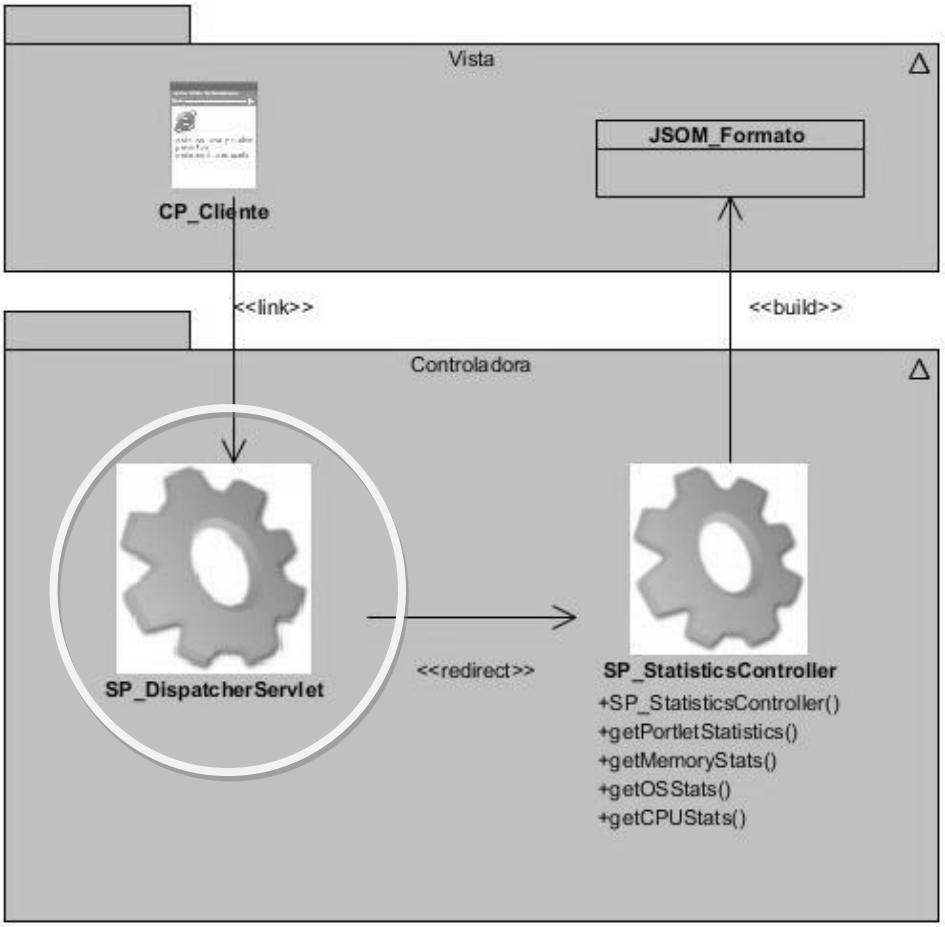

Fig.13: Patrón de diseño GRASP: Controlador.

GoF

Los patrones GoF complementan a los patrones GRASP y en ocasiones se puede encontrar una contraposición entre este tipo de patrones, e incluso, podría inferirse que algunos patrones GoF son variantes de los patrones GRASP, es por ello que la decisión de utilizar uno u otro debe tomarse con

precaución y aplicarse sólo en el ámbito necesario. Entre los patrones GoF más utilizados se pueden citar el de Agente, Fachada y Agente Dispositivo y el Comando.

- El patrón Agente explica el comportamiento cuando no se desea o no es posible acceder directamente a un componente, para ello este patrón sugiere definir una clase sustituta de *software* que represente al componente y asignarle la responsabilidad de comunicarse con el componente real. Debido que la clase **`StatisticsPortlet`** no puede obtener directamente del **`MBeanServerLiferay`** la información referente al monitoreo del sistema, se define la API JMXBridge como intermediaria entre ambas clases, asumiendo la responsabilidad de acceder al **`MBeanServerLiferay`** y obteniendo de esta la información solicitada por la **`StatisticsPortlet`**. La siguiente figura es un fragmento del diagrama de clases del diseño del CU: Mostrar estado del servidor, la misma se evidencia lo anteriormente planteado.

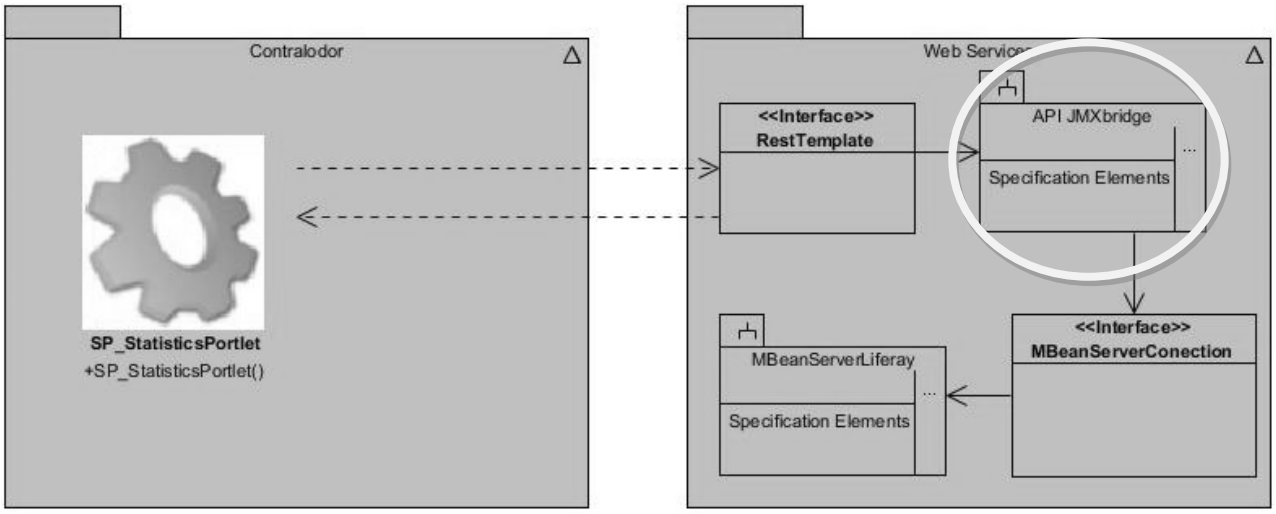

Fig.14: Patrón de diseño GoF: Agente.

- El patrón Fachada es generalmente utilizado para resolver problemas de integración, y se aplica cuando se requiere una interfaz común de comunicación con un conjunto de interfaces o funciones de otro subsistema, en este sentido Fachada define una sola clase que unifique las interfaces y le asigna la responsabilidad de colaborar con el subsistema. El patrón Agente Dispositivo es una

especificidad de Fachada cuando el subsistema que se quiere integrar es un dispositivo externo. Este patrón se pone de manifiesto en el **`JMXUtils`** del API JMXBridge.

- El patrón Comando soluciona el problema de cuando un objeto o sistema recibe varias peticiones o comandos, para ello cada comando define una clase que lo represente y le asigna la responsabilidad de ejecutarse el mismo, de esta forma reduce la responsabilidad del receptor en el manejo de los comandos, aumenta la facilidad con que pueden agregarse otros comandos y ofrece las bases para registrar los comandos, formar colas de espera con ellos y cancelarlos. En la clase **`DispatcherServlet`** se hace uso del mencionado patrón.

Conclusiones del capítulo.

En este capítulo se confeccionó el modelo conceptual de la plataforma para la integración de componentes en el SLD lo que permitió una mejor comprensión del problema y fueron definidos los requisitos funcionales y no funcionales de la misma. Los RF identificados se agruparon en el modelo de casos de uso del sistema, destacando por su impacto en la arquitectura el CU: “Mostrar estado del servidor”, el cual se describe dividido en secciones con el objetivo de lograr una mayor comprensión del mismo.

Se definieron los patrones arquitectónicos y patrones de diseño que posibilitaron definir una mejor arquitectura, entre estos se definió la arquitectura cliente-servidor. Se definió el diagrama de clases del diseño para el CU: “Mostrar estado del servidor”, en la elaboración del mismo se dividió el sistema en paquetes funcionales correspondientes con la arquitectura en capas, obteniendo una capa para la presentación (Vista), una para la lógica de aplicación (Controladora).

Capítulo III: Implementación y prueba de la plataforma para la integración de componentes en el Sistema de Laboratorios Virtuales y a Distancia.

Introducción.

Una vez concluido el modelo del diseño, se dispone de los detalles suficientes para proceder a la construcción del sistema, y una vez concluido este, se procede a la verificación del cumplimiento de los requisitos funcionales mediante las pruebas de *software*. En el presente capítulo se construye el modelo de implementación correspondiente a la plataforma, desglosándolo en los diagramas de componentes y despliegue. Además se determinan los tipos de prueba que serán aplicadas al sistema.

3.1 Modelo de implementación.

El modelo de implementación describe como los elementos del diseño se implementan en términos de componentes tales como ficheros de código fuente, ejecutables, librerías y documentos. Dicho modelo está conformado por los diagramas de componentes y despliegue.

3.2.1 *Diagrama de componente.*

Un diagrama de componentes muestra las dependencias lógicas entre los componentes del sistema, sean estos componentes códigos fuentes, ficheros binarios o ejecutables. Los diagramas ilustrados en la figuras 15 y 16 muestran la vista física a través de componentes y sus relaciones para el CU Mostrar estado del servidor.

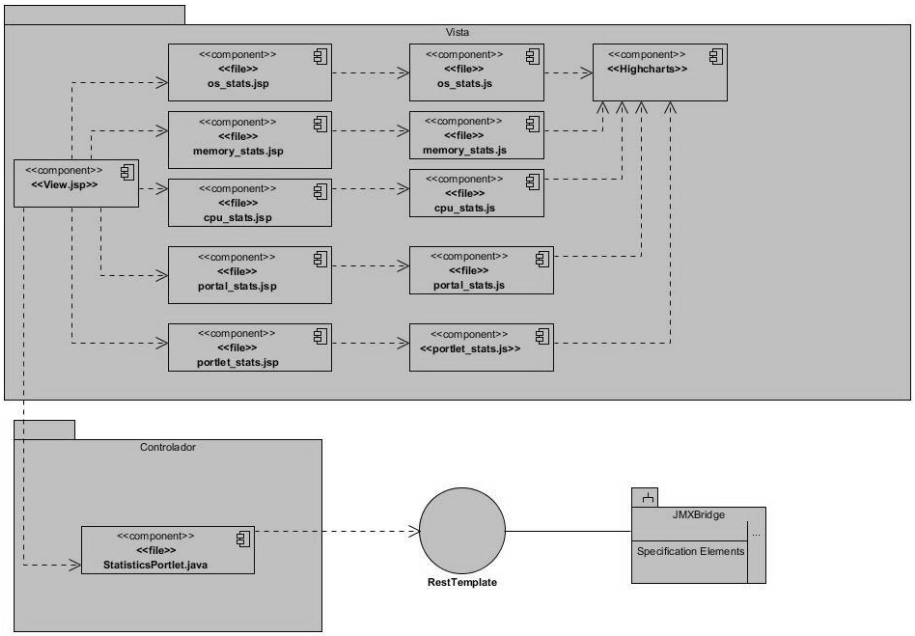

Fig.15: Diagrama de componentes del CU Mostrar estado del servidor.

En el diagrama anterior se muestran todos los componentes del paquete Vista y del paquete Controlador para el CU Mostrar estado del servidor. En el paquete Vista se muestra el componente **`View.jsp`** que incluye todas las páginas jsp que contienen los valores a monitorear. Cada página jsp se relaciona con una clase JavaScript y todas estas a su vez utilizan la librería **`Highcharts`** para graficar los parámetros que se desean monitorizar.

En la figura siguiente (Fig.16) se muestran los componentes para el desarrollo del JMX Bridge. En el paquete controlador se muestra la clase **`DispatcherServlet`**, la cual redirecciona a la clase controladora **`StaticticsContoller`** que contiene las clases **`StatictisPortal`** y **`StatictisPortlet`**. Ambas clases utilizan el **`JMXBridge`** para a través de la interfaz **`MbeansServerConection`** acceder al **`MBeansServerLiferay`** y obtener los datos de monitoreo que se desean mostrar.

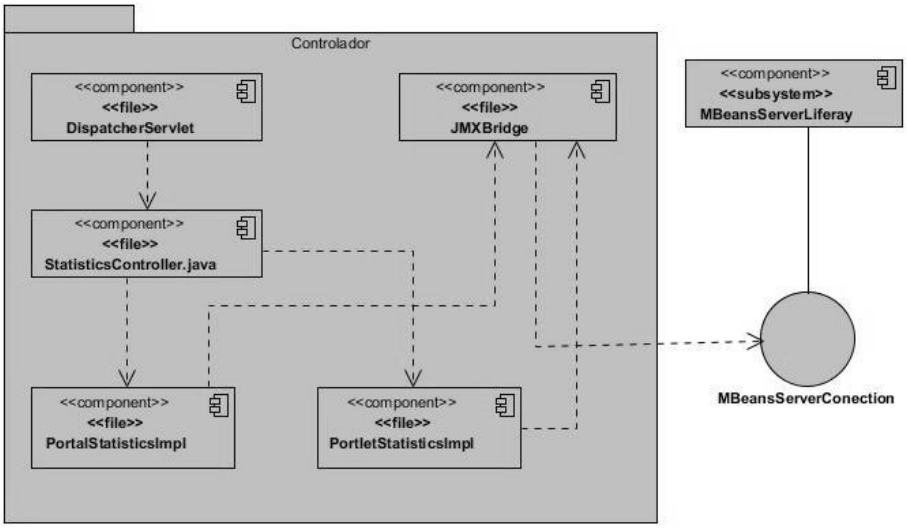

Fig.16: Diagrama de componentes de JMX Bridge.

3.2.2 *Modelo de Despliegue.*

El diagrama de despliegue muestra las relaciones físicas entre los componentes hardware y *software* en el sistema propuesto. Se representa como un conjunto de nodos unidos por conexiones de comunicación**.**

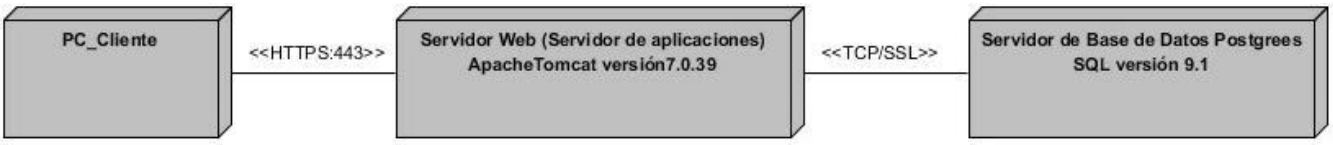

Fig.17: Modelo de despliegue del sistema.

Para la instalación del sistema se requiere de un servidor Web, Apache Tomcat versión 7.0.39, donde se ejecutará la aplicación Web. Se requiere además de un servidor de base de datos donde se ejecutará el SGBD PostgreSQL 9.1. Estos dos servidores se comunicarán mediante el protocolo de transmisión de datos TCP/SSL. El cliente podrá acceder al sistema mediante el protocolo seguro de transferencia de hipertexto HTTPS.

3.2.3 *Estándares de Codificación.*

Los estándares de codificación comprenden todos los aspectos relacionados a la generación de código. Son una forma de normalizar la programación de forma tal que al trabajar en un proyecto, cualquier persona involucrada en el mismo tenga acceso y comprenda el código. Permiten definir la escritura y organización del código fuente de un programa y define la forma en que deben ser declaradas las variables, las clases, los comentarios(Hommel 1999).

Para la implementación de la plataforma para la integración de componentes en el SLD fueron utilizados los estándares de codificación del lenguaje java, los mismos se pueden encontrar en *www.javahispano.com*.

3.2 Pruebas de Software.

Concluido el proceso de desarrollo de la plataforma, es necesario que el sistema sea probado para descubrir y corregir el máximo de errores posibles antes de su entrega al cliente. Las pruebas son un elemento crítico para la garantía de calidad del *software* y representa una visión final de las especificaciones, del diseño y la implementación.

Las pruebas son técnicas de comprobación dinámica. Implican la ejecución del programa, permiten evaluar la calidad de un producto y mejorarlo identificando defectos y problemas. Para ser más eficaces, las pruebas deberían ser realizadas por un equipo independiente, esto garantiza pruebas con alta probabilidad de encontrar errores, que es el objetivo principal de las pruebas. Los niveles de prueba aplicados una vez concluida la plataforma para verificar su correcto funcionamiento fueron:

- Pruebas de integración.
- Pruebas de sistema.

Pruebas de integración

Aun cuando los módulos de un programa funcionen bien por separado es necesario probarlos conjuntamente. Un módulo puede tener un efecto adverso o inadvertido sobre otro módulo; las sub-funciones, cuando se combinan, pueden no producir la función principal deseada, la imprecisión aceptada individuamente puede crecer hasta niveles inaceptables al combinar los módulos y los datos pueden perderse o malinterpretarse entre interfaces. Por lo tanto, es necesario probar el *software* ensamblando

todos los módulos probados previamente. Éste es el objetivo de las pruebas de integración(Juristo et al. 2006).

Las pruebas de integración verifican el correcto ensamblaje entre los distintos componentes con el fin de comprobar que interactúan correctamente a través de sus interfaces, tanto internas como externas; cubren la funcionalidad establecida y se ajustan a los requisitos no funcionales especificados en las verificaciones correspondientes.

Existen dos tipos de pruebas de integración: la incremental y la no incremental. En la integración no incremental se combinan todos los módulos por anticipado y se prueba todo el programa en conjunto. En la integración incremental se construye y se prueba en pequeños segmentos donde los errores son más fáciles de aislar y de corregir. Además esta última es la prueba de integración que se ajusta al desarrollo de *software* basado en componentes ya que facilita que los componentes sean probados, maximizando la facilidad para encontrar errores en la integración de los mismos al sistema.

La integración descendente es una estrategia de las pruebas de integración incremental. En estas se integran los módulos moviéndose hacia abajo por la jerarquía de control, comenzando por el módulo de control principal o programa principal. Los módulos subordinados o secundarios se van incorporando en la estructura, de forma primero-en-profundidad o de forma primero-en-anchura.

Pruebas de sistema

Las pruebas de sistema, están constituida por una serie de pruebas diferentes cuyo propósito primordial es ejercitar profundamente el sistema basado en computadora. Aunque cada prueba tiene un propósito diferente, todas trabajan para verificar que se han integrado adecuadamente todos elementos del sistema y que realizan las funciones apropiadas.

Las pruebas de rendimiento son un tipo de prueba dentro de las pruebas de sistema. Están diseñadas para probar el rendimiento del sistema en tiempo de ejecución dentro del contexto de un sistema integrado. Con esta prueba, el encargado puede descubrir situaciones que lleven a degradaciones y posibles fallos del sistema(Pressman 2005).

Las pruebas de rendimiento se encargan de identificar cómo evoluciona el tiempo de respuesta del sistema en función de varios factores, como la carga de usuarios, o el volumen de datos a tratar. Para

poder estudiar el efecto de estos dos parámetros las pruebas de rendimiento se subdividen en dos, pruebas de carga y pruebas de estrés, las cuales se detallan a continuación:

Pruebas de Carga: Durante las pruebas de carga, se estudiará la evolución del sistema en función del tamaño de los mensajes recibidos, así como del número de operaciones a realizar durante un proceso concreto.

Pruebas de Estrés: Las pruebas de estrés tienen como objetivo, someter al sistema a una carga concurrente de peticiones superior a la esperada habitualmente, para poder así encontrar el punto de corte a partir del cual el sistema se comporta de forma defectuosa.

Finalmente es aconsejable definir unas pruebas mixtas, en las que se empleen los datos de pruebas definidos en los escenarios de integración en situaciones de estrés reales, monitorizando la funcionalidad ante varios hilos concurrentes.

Para la aplicación de las pruebas de carga y estrés se utilizó como herramienta JMeter por las características que a continuación se mencionan:

JMeter es una aplicación cien porciento Java diseñado para cargar clientes de prueba / *software* de servidor (por ejemplo, una aplicación web). Puede ser utilizado para probar el rendimiento tanto en los recursos estáticos y dinámicos como archivos estáticos, Java Servlets, ASP.NET, PHP, CGI scripts, objetos Java, bases de datos, servidores FTP y más. JMeter se puede utilizar para simular una carga pesada en el servidor, la red o el objeto de probar su fuerza o para analizar el rendimiento general bajo diferentes tipos de carga (Foundation 2015).

JMeter brinda como ventaja activar o desactivar una parte del test, lo que es muy útil cuando se está desarrollando un test largo, y se desea deshabilitar ciertas partes iniciales que sean muy pesadas o largas. En general es una herramienta poderosa, fácil de usar, totalmente gratis y que su estructura en árbol permite que nuestra imaginación no tenga límites a la hora de diseñar el plan de prueba.

3.3.1 *Aplicación de la pruebas.*

Prueba de integración

Para la aplicación de la prueba de integración incremental mediante la estrategia: integración descendente, fueron integrados una vez concluido el programa principal o la plataforma varios módulos subordinados para comprobar el correcto funcionamiento de los mismos de forma conjunta en el sistema. Se seleccionó el módulo Diagnostic-*Portlet* cuyo objetivo es realizar diagnósticos de competencias en la plataforma para efectuar la prueba de integración. Para realizar el proceso fue necesario que el usuario cumpliera con la condición de estar autenticado en el sistema con el rol de administrador. Posteriormente se realizaron los siguientes pasos:

1. En el menú "Administración" que aparece en la página principal de la plataforma se selecciona la opción "Panel de control".
2. Una vez seleccionada esta opción aparece una venta compuesta por un conjunto de menús entre los que se encuentra el menú aplicación seleccionando en este la opción "Gestor de aplicaciones".
3. Posteriormente se da clic a la ventana "Instalar" y se selecciona la opción “Cargar fichero”.
4. Para cargar el fichero se da clic al botón “Examinar” y se busca la ubicación donde se encuentra el *portlet* Diagnostic-*Portlet*.war. Para que un *portlet* se pueda integrar a la plataforma debe cumplir la condición de tener la extensión WAR o LPKG.
5. Para finalizar el proceso, se da clic al botón "Instalar" y el sistema muestra el siguiente mensaje de confirmación: “El plugin ha sido enviado con éxito y ahora está siendo instalado”.

Pruebas de sistema

JMeter a la hora de emitir los resultados brinda varios *listener* (oyentes) para guardar los resultados de las pruebas, se decidió utilizar el *listener* “*Summary Report*” el cual nos permite obtener los datos estadísticos de la prueba realizada, los parámetros de este *listener* son:

- Label: El nombre de la muestra (conjunto de muestras).
- # Muestras: El número de muestras para cada URL.
- Media: El tiempo medio transcurrido para un conjunto de resultados.
- Min: El mínimo tiempo transcurrido para las muestras de la URL dada.
- Max: El máximo tiempo transcurrido para las muestras de la URL dada.
- Std**.**Dev: Desviación estándar de la muestra en el tiempo transcurrido.

- Error %: Porcentaje de las peticiones con errores.
- Rendimiento: Rendimiento medido en base a peticiones por segundo/minuto/hora.
- Kb*/*sec: Rendimiento medido en Kilobytes por segundo.
- Avg. Bytes: Tamaño medio de la respuesta de la muestra medido en bytes (erróneamente, en JMeter 2.3.1 muestra el valor en kB).

La siguiente tabla muestra los datos recogidos de la herramienta JMeter después de realizada cuatro pruebas diferentes con distintas cantidades de usuarios accediendo concurrentemente al portal.

Tabla 5: Resultados que brindó JMeter de las pruebas realizadas.

Prueba	Muestras	Media	Min	Max	Std.Dev	% Error	Rend	Kb/Sec	Avg.Byte
I	1000	6145	684	12943	2726,39	31,05%	80,8/sec	645,08	9470,1
II	900	4017	135	7393	1997,32	12,11%	77,7/sec	886,18	11681,9
III	800	2301	102	5456	1518,05	0,00%	77,0/sec	984,85	13096,4
IV	700	2358	51	8290	2025,43	0,00%	73,5/sec	938,43	13080,0

3.3.2 *Resultados de las pruebas.*

Pruebas de integración

Después de concluido el proceso de integración del *portlet* Diagnostic-*Portlet* a la plataforma, se obtuvo como resultado una satisfactoria instalación del mismo, mostrándose una interfaz con el contenido de dicho *portlet* como se muestra en la siguiente figura (Fig.18).

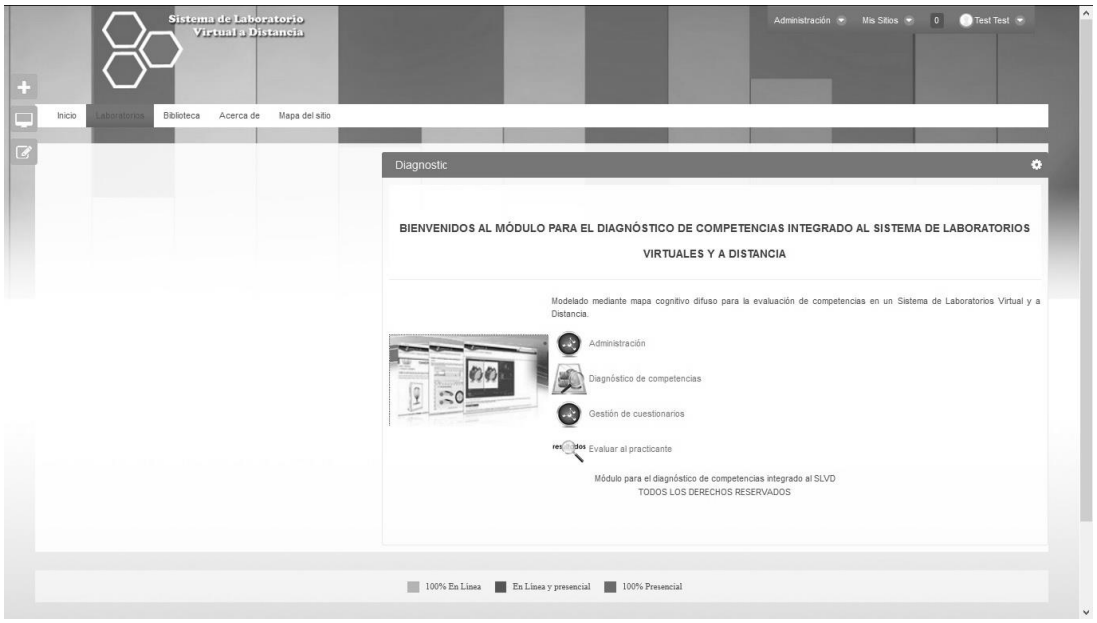

Fig.18: Interfaz del *portlet* Diagnostic-*Portlet* una vez integrado a la plataforma.

Pruebas de rendimiento

Finalizadas las pruebas de carga y estrés se concluye que la plataforma web para la integración de componentes del SLD, funciona eficientemente para una carga de 800 usuarios conectándose concurrentemente y que para una cantidad de usuarios igual o superior a los 900 usuarios la aplicación entra en su estado crítico, aumentando el porcentaje de las peticiones con errores y haciéndose largo el tiempo de respuesta de cada una.

Conclusiones del capítulo.

En este capítulo se construyó el modelo de implementación correspondiente a la plataforma; para ello se realizó el diagrama de componentes, específicamente del CU: “Mostrar estado del servidor” por ser el más relevante del sistema, definiéndose en el mismo los componentes del sistema necesario para el correcto funcionamiento de este CU. Los componentes fueron separados por paquetes logrando una mayor comprensión acerca de su función. Se definió además el modelo del despliegue del sistema que muestra los elementos necesarios para realizar el despliegue e instalación del mismo: al menos un ordenador cliente, un servidor de bases de datos y un servidor de servicios web y se definieron los estándares de

codificación para ser aplicados en la implementación basados en el lenguaje Java sobre el cual se está desarrollando el sistema.

Se le realizaron a la plataforma pruebas de sistema y de integración. Entre los tipos de pruebas de sistema fue seleccionada la prueba de rendimiento mediante la herramienta JMeter la cual arrojó conclusiones importantes. Por otra parte las pruebas de integración se realizaron mediante la estrategia integración descendiente teniendo la misma resultados satisfactorios. Dichas pruebas brindan al cliente conformidad y seguridad ante las funcionalidades del sistema.

Conclusiones Generales

Una vez concluida la presente investigación se arribó a las siguientes conclusiones:

- Se determinó para la confección de la plataforma el uso de componentes de interfaz web denominados *portlets*, que proveen información en formato HTML y pueden ser utilizados por otras aplicaciones que cumplan con las especificaciones JSR 168 y 286.
- El sistema fue desarrollado siguiendo las políticas de la universidad en cuanto al uso de tecnologías libres lo que permite contribuir a la soberanía tecnológica, posee libre acceso al código fuente y reduce el gasto en cuanto al pago de licencias de *software*.
- Para definir las características y condiciones que debe proveer la solución se identificaron requisitos funcionales y no funcionales, agrupándose los primeros en CU que permitieron una correcta planificación en aras de satisfacer las necesidades del cliente.
- Se validaron las funcionalidades implementadas mediante pruebas de sistema haciendo uso de la herramienta JMeter y pruebas de integración incremental mediante la estrategia integración. Las pruebas permitieron encontrar inconformidades en la validación de datos y el tratamiento de errores que contribuyeron a aumentar la calidad del sistema.

Recomendaciones

Debido que se utiliza JMX para monitorizar el sistema se propone implementar mecanismos para tomar acciones correctivas en tiempo real con el objetivo de evitar la sobrecarga del sistema.

Referencias bibliográficas

1. ALCÁNTARA, D. Importancia de las TIC para la educación. In *Revista Digital Innovación y Experiencias Educativas.* 2009.
2. ALLAMARAJU, S., C. BEUST AND J. DAVIES *Programación Java Server con J2EE Edition*. Edtion ed.: ANAYA MULTIMEDIA, 2004. ISBN 978-8441513587.
3. AMUTABI, M. Ñ. AND M. O. OKETCH. Experimenting in distance education: the African Virtual University (AVU) and the paradox of the World Bank in Kenya. In *International Journal of Educational Development.* Kenia, 2003, vol. 23.
4. AVISON, D. AND G. FITZGERALD. METODOLOGÍAS TRADICIONALES VS. METODOLOGÍAS ÁGILES. In *Information Systems Development: Methodologies, Techniques, and Tools.*, 1995.
5. BERBEL, A. Portlet Gestor de Comentarios de Contenidos. UNIVERSIDAD PONTIFICIA COMILLAS, 2009.
6. BLAS, M. J., A. LOYARTE AND J. VEGA. Plataforma para Laboratorios Virtuales y Remotos Orientados a la Simulación de Sistemas Dinámicos Controlados. In *14th Argentine Symposium on Technology, AST 2013.* Argentina, 2013.
7. BROWN, D. AND D. CHAD MICHAEL *Struts2 in action*. Edtion ed.: Manning Publications Co, 2007.
8. BURNS, E., C. SCHALK AND N. GRIFFIN *JavaServer Faces 2.0: The Complete Reference*. Edtion ed.: McGraw-Hill Companies, 2010.
9. CABRERA, E. Introducción a las aplicaciones Web con Java [online]. [Santo Domingo, R. D.]: 2014.
10. CATALANI, E. “Arquitectura Modelo/Vista/Controlador”. In., 2007.
11. CLAY RICHARDSON, W., D. AVONDOLIO, J. VITALE, P. LEN, et al. Professional Portal Development withOpen Source Tools. In. Indianapolis, Indiana: Wiley Technology Publishing, 2004.
12. DOMÍNGUEZ, H. La formación de profesores en Tecnologías de la Información y la Comunicación (TIC) para integrar material académico interactivo en el bachillerato de la Universidad Nacional Autónoma de México (UNAM). In *Revista Iberoamericana de Educación* Universidad Nacional Autónoma de México (UNAM): Organización de Estados Iberoamericanos para la Educación, la Ciencia y la Cultura (OEI), 2008, vol. 48.
13. DRAE. Laboratorio. In *DRAE.* España: 22, 2012.

14. DUMÉNIGO, D. Sistemas de información, aplicación en empresas. In *Contribuciones a la Economía.* 2012.
15. FRAMEWORK, E. P. Concept: Use Case. In., 2011.
16. GARLAND, J. AND R. ANTHONY *Large-Scale Software Architecture*. Edtion ed. Great Britain: John Wiley & Sons Ltd, 2003.
17. GIBERT, M. Y. O. Bases de datos en PostgreSQL. In., 2007.
18. GIMSON, L. Metodologías ágiles y desarrollo basado en conocimiento. Universidad Nacional de La Plata 2012.
19. HOMMEL, S. Convenciones de Código para el lenguaje de programación JAVA™. In.: Sun Microsystems Inc., 1999.
20. INCENCIO , G. SISTEMA INFORMÁTICO PARA LA EVALUACIÓN DE ATRIBUTOS DE CALIDAD EN COMPONENTES BIOMÉTRICOS. In *3C TIC* Cuba, 2014, vol. 3.
21. JENDROCK, E., R. CERVERA-NAVARRO, I. EVANS, D. GOLLAPUDI, et al. The Java EE 6 Tutorial. In. U.S.A: 500 Oracle Parkway Redwood City, CA 94065, 2013.
22. JONAS, Y. *Liferay Portal Enterprise Intranets*. Edtion ed. Birmingham, E.E. U.U: Packt Publishing, 2008. ISBN 978-1-847192-72-1.
23. JURISTO, N., A. M. MORENO AND S. VEGAS. TÉCNICAS DE EVALUACIÓN DE SOFTWARE. Universidad Politécnica de Madrid, Campus de Montegancedo 2006.
24. LARMAN, C. *UML y Patrones. Introducción al análisis y diseño orientado a objetos*. Edtion ed. México: Prentice Hall, 1999.
25. LIFERAY. Alloy UI. In., 2010.
26. LIFERAY. Liferay Inc, Premios y reconocimientos. In., 2014.
27. MÁRQUEZ, B. M. AND J. M. ZULAICA. Implementación de un reconocedor de voz gratuito a el sistema de ayuda a invidentes Dos-Vox en español. Universidad de las Américas Puebla, 2004.
28. MARTÍNEZ, J. Á. Análisis de los usuarios, contenidos y servicios de los servicios públicos electrónicos. In., 2007.
29. MARTINEZ, R. Sobre PostgreSQL. In., 2010.
30. MING HUANG, Y. AND D. FEI WU. Build RESTful web services using Spring 3. In. U.S.A: © Copyright IBM Corporation 2010, 2010.

31. MONJE, J., M. RIVAS ROSSI AND V. HUGO. Internet, multimedia and virtual laboratories in a "Third World" environment: how we solved the 21 basic problems in the Costa Rican Distance Education University. In *X Congreso Internacional sobre Tecnología y Educación a Distancia*

32. San José, Costa Rica, 1999.
33. MORALES, M. Clasificación del Software Libre Orientado a la Automatización Integral de Bibliotecas según el Nivel de Complejidad de la Biblioteca: Bibliotecas Simples, Bibliotecas de Mediana Complejidad y Bibliotecas de Alta Complejidad. In *E-Ciencias de la Información.* 2013, vol. 3.
34. ORACLE. The Java EE5 Tutorial. . 2010.
35. PRESSMAN, R. *Ingeniería de Software. Un enfoque práctico.* Edtion ed.: The McGraw-Hill Companies, 2005.
36. Liferay Inc. [online]. 2014. Available from Internet: <https://www.liferay.com/es/web/meera.success/blog/-/blogs/liferay-mvc-portlet-development-introduction>.
37. SANTANA, I. AND L. HERNÁNDEZ Experiencias del uso de Laboratorios Remotos en la enseñanza de la Automática. In *Relada.* Madrid, España, 2011, vol. 5.
38. SCAMERCIO, F. Introducción a Liferay Portal. In., 2010.
39. SULLINS, B. G. AND M. B. WHIPPLE *JMX in Action*. Edtion ed.: Manning Publications Co., 2003. ISBN 1-930110-56-1.
40. WALLS, C. AND R. BREIDENBACH *Spring in Action, Second Edition*. Edtion ed.: Manning Publications Co., 2008. ISBN 1-933988-13-4.
41. ZAMORA, R. Laboratorios Remotos: Actualidad y Tendencias Futuras. In *Scientia et Technica.* Universidad Tecnológica de Pereira, 2012, vol. 51.

Anexos

Fig.19: Interfaz de autenticación de la plataforma para la integración de componentes en el Sistema de Laboratorios Virtuales y a Distancia.

Fig.20: Interfaz del caso de uso: Mostrar estado del servidor.