Universidad de las Ciencias Informáticas

Facultad 2

Despliegue distribuido de servidores GRHS

Trabajo de Diploma para optar por el Título de

Ingeniero en Ciencias Informáticas

Autores: Adyana Pileta Gammalame

Leobanis Acosta Alvarez

Tutores: Ing. Yaniel Alfredo Velázquez Bruceta

Msc. Ramón Alexander Anglada Martínez

La Habana, día 19 del Mes de junio del 2015

“Año 57 de la Revolución.”

Los resultados que consigues estarán en proporción directa al esfuerzo que aplicas.

Denis Waitley.

DECLARACIÓN DE AUTORÍA

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

Para que así conste firmamos la presente a los 19 días del mes de junio del año 2015.

Autores:

_____	_____
Adyana Pileta Gammalame	Leobanis Acosta Alvarez

Tutores:

_____	_____
Yaniel Alfredo Velázquez Bruceta	Ramón Alexander Anglada Martínez

DEDICATORIA

Leobanis:

Mi madre, Maritza, que me dio la vida, me ayudo a llegar a donde estoy. Por ser comprensiva, darme consejos y apoyarme cada vez que lo necesitaba.

A mi padre, Lázaro, que es mi guía y mi ejemplo a seguir.

A mi hermano, Yovanis y mi hermanita, Ailín por siempre estar preocupándose por el hermano del medio.

Adyana:

A mi Dios y Salvador por rescatarme justo a tiempo, por darme fuerzas cuando pensaba que no podía, por guiarme por el buen camino.

A mis padres , mi mamá Idelsi por ser incondicional, por darme la vida y por ser mi madre, mi amiga, mi compañera, por decirme que siempre estará orgullosa de mi y sobre todo por haberme sabido educar.

A mi papá Pedro, porque siempre me cuidó, se ocupó de mi, supo darme buenos consejos y siempre decir que está orgullosos de hasta donde he llegado.

A mi familia, mi tía Yadira, mis tíos Monchi, Ivan Y Nelson a mis tíos por parte de Padre.

A mi abuela por malcriarme y consentirme.

AGRADECIMIENTOS

Leobanis

Primeramente a mis tutores Yaniel y Anglada que estuvieron disponibles cada vez que nos hacia falta, a mi compañera de tesis Adyana y a Andres Escobar que cada vez que estábamos estancados nos sacaba del apuro y a su novia Greizy que nos permitió molestarlos. A mi familia y amigos que de una forma u otra me ayudaron a seguir adelante.

Adyana:

Agradezco al que es el primero, el principio, el alfa y la omega, el salvador, el restaurador, el sanador, el que viene justo en el momento correcto, al que es tres en uno ***Padre, Hijo y Espíritu Santo****. A mi* ***mamá Idelsi****, por apoyarme en todas las decisiones que tomo, por darme aún más de lo que necesito, aun cuando no puedes te sacrificas y me los das, por no fallarme, por ser mi mamá gracias, por darme la vida. A mi* ***papá Pedro*** *por darme aún más de lo que me puedes ofrecer por apoyarme por estar orgulloso de mí por darme buena educación porque siempre puedo contar contigo. A mi tía* ***Yadira****, a mi* ***abuela Reina*** *a mis tíos todos en general y a mi extensa familia.*

A mis hermanos en cristo por apoyarme y darme palabras de aliento, porque son una familia maravillosa por compartir conmigo buenos y malos momentos. A Yaneisi, Juan, Kiri, Viset, Diana, José Orlando, Giselle, Handy, porque ocupan un lugar lindo en mi corazón y espero seguir viéndolos los quiero mucho. A los muchachos de mi aula desde primero hasta quinto los que se gradúan conmigo y los que no, a mis compañeras de apto en estos años a Yunexy a Yanelis que aunque la conocí este año me soporto a Yuliet a cristina y al Luisma por que hemos hecho un bonito vínculo que espero no se pierda. A mi compañero de tesis Leo sin él hubiese sido imposible a mis tutores Yaniel por siempre estar disponible para nuestras dudas por aconsejarnos, por ayudarnos cuando estábamos trabados mejor imposible, a Anglada gracias por los consejos y críticas que hicieron que mejoráramos. A Andrés por siempre ayudarnos cuando estábamos estancados y darnos ideas de cómo avanzar gracias y a su novia.

Al que es el principio y el fin a Jesucristo gracias mil veces gracias.

RESUMEN

El Gestor de Recursos de Hardware y Software (GRHS) es un producto que permite realizar el inventario de hardware y software sobre una red de computadoras, capaz de mantener un control sobre los recursos de hardware presentes en las computadoras de la red que han sido inventariadas, así como conocer el software que se utiliza en las estaciones de trabajo. Este producto posee tres aplicaciones: Gserver también llamado controlador de inventario, Gadmin interfaz gráfica de administración y Gclient colector de inventarios. Actualmente la comunicación entre servidores y clientes está basada en una estructura jerárquica en forma de árbol que posee desventajas como no garantizar el envío de la información por varios caminos hacia el servidor central, no poseer un módulo que permita desde Gadmin gestionar las configuraciones las cuales son efectuadas directamente en el archivo de configuración. GRHS no brinda la posibilidad de mostrar gráficamente la estructura de servidores configurados, así como los caminos definidos entre los mismos. El presente trabajo tiene como objetivo resolver las deficiencias descritas anteriormente. Para la implementación de la solución se escogieron varias herramientas entre las que se encuentra el lenguaje de programación python y el framework Django. Del lenguaje de programación se instalaron dos dependencias que ayudan a la visualización del grafo que se muestra estas son: pygraphviz y networkx, además de todas las ventajas que ofrece Django como framework. Finalmente se obtiene una solución que permite enviar la información por diferentes caminos al servidor central utilizando una estructura en forma de grafo dirigido sin ciclos, se logra gestionar las configuraciones desde la interfaz de administración y mostrar los caminos definidos entre ellos.

Palabras claves: grafo dirigido no cíclico, réplica de información, servidores, GRHS

TABLA DE CONTENIDOS

INTRODUCCIÓN

Debido a la creciente evolución del hombre y los paulatinos avances de las tecnologías, se han puesto la informática y las telecomunicaciones en la cúspide de las soluciones más viables, donde el concepto Nuevas Tecnologías de la Información y las Comunicaciones (NTIC) ha sido consecuencia de esto. Diversas instituciones han abierto la posibilidad de manejo de la información de manera rápida y eficiente, mediante el uso de sistemas informáticos que proporcionan como resultado el poder solucionar problemas sociales de la actualidad de forma eficaz.

La evolución de la ciencia y la tecnología han planteado diferentes dificultades que han llevado al enfoque de una solución informática, esto indica que la misma va camino a ser la guía para todo tipo de actividades. Cuba, no está exenta de aplicar estas nuevas tecnologías, a pesar de pertenecer a los países del tercer mundo, está abierta a la mejora de la sociedad mediante la utilización de ellas.

Hoy día el gran cúmulo de información que generan las empresas hacen que sea un proceso engorroso el trasladar información de un lugar a otro e incluso almacenarla. Esta continua acumulación de datos conllevó a utilizar herramientas informáticas que permitieran proveer diferentes servicios a las redes de datos, tanto inalámbricas como las basadas en cable. Todo esto posibilita un aumento en la velocidad del traspaso de información, permitiendo accesos a cuentas de correo electrónico, administración de dominios empresariales, hospedaje y dominios Web entre otras funciones; a este conjunto de servicios se les llama servidores (1). A medida que avanza la ciencia los servidores se hacen necesarios para todo tipo de actividades, ya sea almacenar documentos y distribuirlos a los clientes de la red, guardar, recibir o enviar correos electrónicos incluso almacenar los documentos que son accesibles a través de Internet, haciendo esto que su uso cotidiano sea más frecuente e inevitable (2).

A causa de que las tecnologías en el mundo han avanzado de manera rápida, en Cuba se creó la Universidad de las Ciencias Informáticas (UCI) con el fin de impulsar el desarrollo de estas tecnologías y de preparar informáticos comprometidos con la Revolución y con un amplio saber de la informática y sus derivados.

En la UCI, el volumen de información es muy alto debido a los disímiles proyectos que se realizan en la misma por lo que el uso de servidores es sumamente importante para la realización de cualquier actividad correspondiente al desarrollo productivo de software. Dentro de la misma, existen diferentes centros de desarrollo de software que utilizan o programan aplicaciones del lado del servidor. Uno de estos centros es el

Centro de Telemática (TLM) de la Facultad 2. Actualmente el centro dispone de un producto denominado Gestor de Recursos de Hardware y Software (GRHS), el cual, en un principio estaba constituido por un servidor que centralizaba la información, el cual se descentralizó mejorando así su estructura haciendo que funcionase jerárquicamente en forma de árbol. Este producto permite realizar el inventario de Hardware y Software en una red de computadoras. El mismo está capacitado para mantener el control sobre los recursos de hardware presentes en las computadoras de la red que han sido inventariadas, así como conocer el software que se utiliza en las estaciones de trabajo con el fin de contabilizar los cambios realizados en cada una de las computadoras y tomar acciones automáticas en caso de cambios no autorizados (incidencias) (3).

GRHS tiene tres aplicaciones: Gserver también llamado controlador de inventarios (4), Gadmin que es la interfaz de administración (4) y Gclient el cual es el colector de inventarios (4). La aplicación Gclient es la encargada de obtener las propiedades de las piezas y programas instalados en una computadora, así como, detectar los cambios, determinar las incidencias y tomar algunas acciones automáticas cuando ocurren incidencias. La aplicación Gserver se encarga de recibir la información enviada por los colectores, enviar órdenes a las computadoras inventariadas y notificar las alertas a los interesados cuando ocurren incidencias, mientras que Gadmin es la aplicación con la que interactúa el administrador de GRHS (3).

GRHS para realizar la tarea de inventario del hardware y el software de la red de computadoras se apoya en (Gclient), cliente del mismo que se instala en cada computadora y de varios servidores que pueden ser instalados en una estructura jerárquica en forma de árbol.

La estructura jerárquica que actualmente se utiliza, permite el envío de información desde los clientes hasta el servidor central haciendo posible que la misma pase de un servidor a otro por los diferentes niveles jerárquicos hasta llegar a su destino, lo cual permite la descentralización de la información y garantiza que si un área queda aislada de la red, se tenga control sobre ella con el servidor de GRHS ubicado en el nivel de la jerarquía correspondiente.

El Gestor de Recursos en la actualidad posee un grupo de desventajas relacionadas con la estructura jerárquica descrita anteriormente, las cuales se describen a continuación:

GRHS no garantiza el envío de la información por varios caminos hacia el servidor central, limitando las vías de comunicación entre los servidores. De ocurrir problemas con la comunicación entre los servidores de un

determinado nivel, dicha información no llega a su destino, permaneciendo en el nivel jerárquico alcanzado hasta que se restablezca nuevamente la conexión. El sistema no posee un módulo que permita desde Gadmin (Interfaz de administración), gestionar las configuraciones relacionadas con la estructura jerárquica mencionada, lo cual obliga al usuario del sistema a establecer las configuraciones directamente en el archivo de configuración. No brinda la posibilidad de mostrar gráficamente la estructura de servidores configurados, así como los caminos definidos entre los mismos, lo cual dificulta la administración de los servidores en cuestión.

Dada la problemática descrita anteriormente y la necesidad de encontrar una solución para las dificultades, se plantea el siguiente **problema a resolver:** ¿Cómo minimizar las dificultades que presenta en la actualidad la estructura jerárquica para el despliegue de servidores GRHS?

Para dar solución a la problemática planteada se traza como **objeto de estudio** el despliegue de servidores en redes de computadoras. Se traza como **Objetivo General** de la investigación desarrollar un módulo para el sistema GRHS que permita desplegar sus servidores (Gserver) de manera distribuida por diferentes caminos.

El **campo de acción** está enmarcado en la red de computadoras del centro TLM donde está desplegado el sistema GRHS.

Para dar cumplimiento al objetivo general planteado se trazan las siguientes **tareas de investigación**:

1. Elaboración del marco teórico de la investigación identificando las principales tendencias, limitaciones y ventajas de software existentes que utilicen el despliegue distribuido de servidores.
2. Realización del estudio del estado del arte del despliegue distribuido de servidores para plantear la solución.
3. Selección de las herramientas, tecnologías, lenguajes y la metodología a utilizar para el desarrollo de la aplicación.
4. Implementación de la propuesta de solución.
5. Validación de los resultados obtenidos con la realización de pruebas sobre la solución para garantizar su correcto funcionamiento.

Durante la presente investigación se hace uso de los siguientes **Métodos Científicos**:

- ✓ **Métodos Teóricos**:

Analítico-Sintético: Se distinguen los elementos de un fenómeno y se procede a revisar ordenadamente cada uno de ellos por separado. Consiste en la extracción de las partes de un todo, con el objeto de estudiarlas y examinarlas por separado, para ver, por ejemplo las relaciones entre las mismas. Es un proceso mediante el cual se relacionan hechos aparentemente aislados y se formula una teoría que unifica los diversos elementos. Consiste en la reunión racional de varios elementos dispersos en una nueva totalidad. El investigador sintetiza las superaciones en la imaginación para establecer una explicación tentativa que someterá a prueba (5).

Se lleva a cabo para analizar y sintetizar los diferentes elementos y definiciones relacionadas con los sistemas existentes que realizan el despliegue distribuido teniendo como objetivo arribar a conclusiones que sustenten la realización de la investigación.

- ✓ **Métodos Empíricos**:

Observación: La observación científica como método consiste en la percepción directa del objeto de investigación. La observación investigativa es el instrumento universal del científico. La observación permite conocer la realidad mediante la percepción directa de los objetos y fenómenos (5).

Se lleva a cabo en la etapa inicial de la investigación teniendo en cuenta el comportamiento del sistema actual para arribar a posteriores conclusiones.

Entrevista: La entrevista es una conversación planificada entre el investigador y el entrevistado para obtener información. Se aplica a especialistas en el tema con el objetivo de obtener criterios de expertos (6).

Se lleva a cabo en la investigación cuando se entabla una conversación con expertos sobre el tema, obteniendo criterios acerca del objetivo al que se desea llegar.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

1.1. Introducción

Durante el desarrollo del presente capítulo se explican los conceptos y definiciones que serán utilizados como términos claves en el transcurso de la investigación para un mejor entendimiento de su contenido. Del mismo modo este capítulo incluye un estudio de las metodologías y tecnologías a utilizar para el desarrollo de la solución, las herramientas y lenguajes existentes que serán significativos en el diseño y desarrollo del sistema y los diferentes sistemas que plantean soluciones similares, constituyendo así un estudio del estado del arte.

1.2. Términos claves

El término **despliegue** es la acción de desdoblar, extender aquello que se encuentra plegado o cerrado. En la informática, se hace referencia a la fase final del desarrollo de software en la que se empaqueta el producto para su posterior distribución y puesta en funcionamiento en su entorno de ejecución final (7).

A instancias de la Informática se conoce como lista de **despliegue** a aquel programa o rutina compuesto por una serie de instrucciones para gráficos y que es ejecutada mediante el procesador; las instrucciones definirán una determinada imagen de salida. Dicha imagen será interpretada a partir de la ejecución de las instrucciones de la lista de despliegue (7).

El término **distribuido** es la acción de dividir algo entre varios elementos, designando lo que a cada uno corresponde, según voluntad, conveniencia, regla o derecho. Dar a algo su oportuna colocación o el destino conveniente (8).

Por lo que **despliegue distribuido** es el acto de desdoblar un conjunto de elementos entidades o programas desde el punto de vista informático que están relacionados entre sí empleando una estructura de grafo, donde el nodo principal es el servidor central y cualquiera de los demás nodos puede ser el origen del grafo y cada nodo puede generar un camino diferente para llevar la información necesaria a su destino. Posibilitando que la información llegue por diferentes caminos sin tener en cuenta los niveles de jerarquía que puedan existir en una determinada entidad.

El término **grafo** se puede definir como un objeto unitario de naturaleza abstracta que abarca a las grafías que componen una letra. La palabra tiene origen griego y significa “imagen” o “dibujo”.

Para las ciencias de la computación y la matemática, un **grafo** es una representación gráfica de diversos puntos que se conocen como nodos o vértices, los cuales se encuentran unidos a través de líneas que reciben el nombre de aristas (7).

Una **réplica** es una copia completa de los datos, la cual posee todas las propiedades del mismo (9).

La **información** está constituida por un grupo de datos ya supervisados y ordenados, que sirven para construir un mensaje (2).

Por lo que **réplica de información** no es más que la copia completa de los datos ya supervisados y ordenados para construir un mensaje.

1.3. Sistemas similares que permiten el envío de información por varios caminos.

1.3.1. HYDRA

El sistema HYDRA (Heterogeneous sYstem Deployment and Remote Administration) se encarga de dar soporte y automatizar las tareas de mantenimiento de estos escenarios, desde la instalación de un sistema operativo, hasta restaurar la configuración de un nodo, pasando por la aplicación de actualizaciones o parches. Entre los principales objetivos que presenta se encuentran:

Instalación automática y masiva: La principal finalidad es poder distribuir aplicaciones y SSOO[1] en un número elevado de computadores de forma automática y completamente desatendida, de forma que ni los administradores ni los usuarios deban tener en cuenta los detalles de gestión y configuración (10).

[1] SSOO: Sistemas Operativos

Instalación nativa: La utilización de una máquina virtual impide el acceso a los dispositivos que esta no emule en su totalidad, lo que restringe, por ejemplo, la programación de drivers, o la utilización de la aceleración por hardware de las tarjetas gráficas. Utilizar el SO de forma nativa evita estos inconvenientes y también se consigue que todo el potencial de la CPU y la memoria queden a disposición del usuario, lo que se traduce en una mejora del rendimiento de la máquina (10).

Recuperación ante fallos: Mediante el uso de la herramienta de administración del sistema, será muy sencillo restaurar un equipo que esté defectuoso o que haya sido atacado. Bastará con indicarle al sistema que vuelva a configurar ese equipo desde cero y esto se hará de forma automática (10).

Configuración a medida: Cada usuario podrá crear una o varias configuraciones de SO y aplicaciones y adaptarlas con los programas y el software necesario acorde a sus necesidades específicas (10).

Planificación: Se deberá permitir algún tipo de planificación del funcionamiento del sistema que permita automatizar la gestión de los despliegues e instalaciones a lo largo del tiempo. De esta forma se reduce la carga de trabajo del personal de administración de sistemas (10).

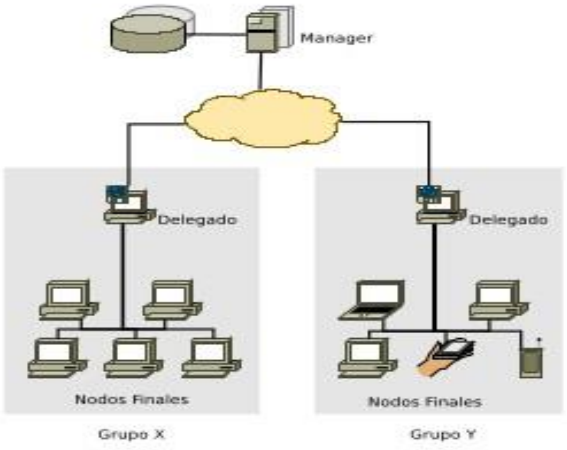


Figura 1: Arquitectura del Sistema HYDRA (10)

Manager se encarga de brindar servicios básicos para las comunicaciones y el despliegue del software, mantener un determinado sistema de archivos que contienen un sistema operativo junto con las aplicaciones, configuraciones y datos personalizados de los usuarios y almacenar las configuraciones de los nodos y los

diferentes despliegues. El Manager se encarga de preparar las imágenes para su distribución, enviarlas a los delegados y coordinar la instalación. Tanto el Manager como la base de datos se encuentran replicadas para garantizar la disponibilidad ante fallos en el servicio. Como se verá más adelante, los usuarios contactarán con este servicio para añadir, borrar y reconfigurar el despliegue de las imágenes HYDRA.

Delegado actúa como un enlace entre los nodos finales y el Manager. Cada delegado es responsable de un subconjunto de los nodos del sistema. Descarga las imágenes que sus nodos van a necesitar y se ocupa de servirlas. También es el responsable de despertar a sus nodos y ofrecerles una configuración básica para que empiecen a funcionar (10).

El Delegado sirve para varios propósitos en primera instancia, permite descongestionar la red y la carga de trabajo del Manager, al hacer que los nodos obtengan sus imágenes desde el Delegado en lugar de que todos las soliciten al Manager. Por otro lado, sirve de enlace entre los nodos y el Manager, ya que es posible que éste último, al ser una máquina sensible, esté en una red distinta y protegida a la que los nodos no puedan acceder y finalmente, la jerarquía de responsabilidades que existe entre Delegado y Manager permite que la organización adapte la infraestructura de HYDRA a su organización interna.

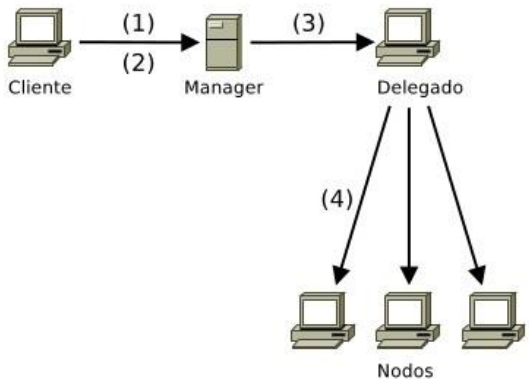


Figura 2: Esquema de funcionamiento de HYDRA (10)

1.3.2. Sistema Distribuido con NI VeriStand

NI VeriStand (National Instrument VeriStand) es un entorno de software fácil de usar para configurar aplicaciones de pruebas en tiempo real. Al usar las funciones de NI VeriStand, usted puede crear este sistema sin la necesidad de diseñar, programar y mantener la arquitectura de software (11).

Uno de los enfoques que posee NI VeriStand es el de recopilar entradas y salidas (E/S) en una ubicación distinta a la de un solo procesador central, generalmente es llamado E/S distribuida, otro es añadir múltiples procesadores a un solo sistema, por consiguiente distribuir el procesamiento. El entorno de software NI VeriStand incluye varias características que hacen del desarrollo de sistemas distribuidos una experiencia agradable. Al usar NI VeriStand, uno o más PCs (servidores) de operador pueden comunicarse con uno o más objetivos de ejecución en tiempo real con mínima configuración. NI VeriStand maneja toda la comunicación entre las PCs de operador (servidores) y objetivos de ejecución en tiempo real. La Figura 3 muestra una sola topología involucrando un servidor y un objetivo (11).

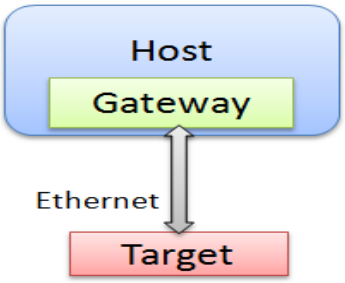


Figura 3: Tipología Simple (11).

El componente de un servidor que se comunica con el objetivo es el NI VeriStand Gateway. Esto es manejado automáticamente, pero es un concepto clave para comprender topologías más grandes. El componente permite que se pueda añadir objetivos fácilmente a una topología dentro del NI VeriStand System Explorer (11).

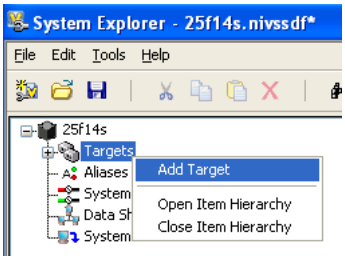


Figura 4: Añadir un objetivo dentro del System Explorer (11)

Un solo archivo de definición del sistema puede contener un número ilimitado de objetivos y hasta combinar diferentes tipos de objetivos (11).

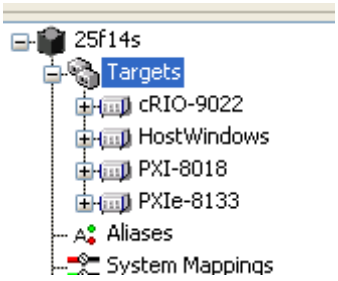


Figura 5: Un solo archivo de definición del sistema puede contener múltiples objetivos (11).

Cada objetivo puede tener su propia configuración de hardware y software y todos los objetivos pueden ser desplegados e interactuar desde un solo Gateway, las PCs principales adicionales pueden comunicarse con la misma topología del objetivo al comunicarse con otro Gateway del servidor (11).

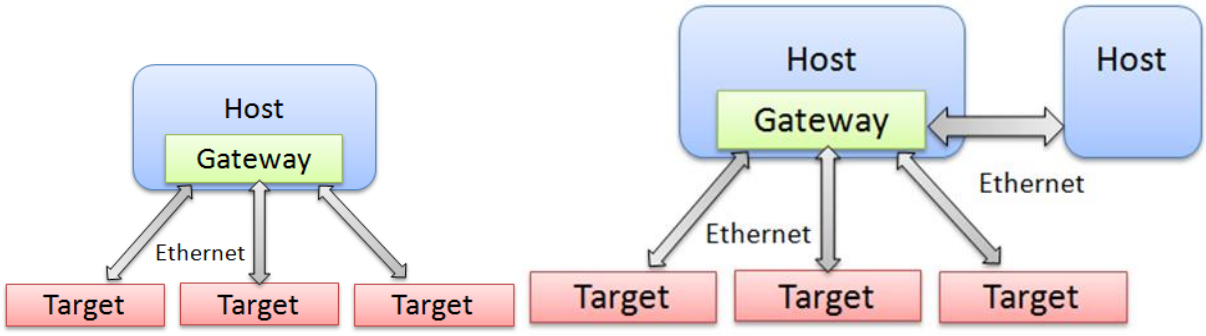


Figura 6: De izquierda a derecha: Múltiples objetivos pueden ser desplegados e interactuar desde un solo servidor y múltiples Servidores y Múltiples Topologías de Objetivos (11)

Para lograr esto, los servidores adicionales simplemente necesitan cambiar la dirección del NI VeriStand Gateway para ser el servidor remoto. El resto de la aplicación se mantiene igual (11).

Elementos que permite NI VeriStand:

Sincronización de Hardware: La sincronización de hardware significa que cada pieza de hardware en el sistema comparte un reloj de referencia de hardware para temporización y un disparo de inicio para comenzar tareas de E/S. Cada pieza de hardware en el sistema obtiene sus propios relojes del mismo reloj de referencia de hardware y cada pieza del hardware inicia al mismo tiempo (11).

Sincronización de Software: La sincronización de software significa que varias piezas del código en el sistema (en este caso, el NI VeriStand Real-Time Engine) comparten un reloj de ejecución y un disparo en estrella para comenzar la ejecución al mismo tiempo (11).

El NI VeriStand Real-Time Engine está diseñado para usar E/S de un solo punto temporizada por hardware (HWTSPIO) cuando los dispositivos de hardware adecuados están disponibles. HWTSPIO es una característica de hardware y software permitiendo bloquear la ejecución del software ante relojes de hardware físicos. El bloqueo de software a hardware está disponible solamente para entrada analógica, así que una configuración del sistema debe tener por lo menos un canal de entrada analógica aunque no sea usado (11).

Por lo tanto, si el hardware es sincronizado como se describe arriba y está presente un canal de entrada analógica en cada configuración, cada software NI VeriStand Real-Time Engine del objetivo es sincronizado automáticamente (11).

1.3.3. PROPHET

PROPHET[2] explora nuevos caminos para aumentar la fiabilidad de las redes en esta sociedad interconectada. Se cuenta con los sistemas en red para proporcionar comunicaciones generalizadas y ubicuas, pero, en los últimos años, se han visto ejemplos de cómo fallos en el software y la configuración puede afectar la fiabilidad de las redes (12).

---

[2] PROPHET: Simplifying Development and Deployment of High-Performance, Reliable Distributed Systems (Simplificación del desarrollo y despliegue de sistemas distribuidos de alto rendimiento y fiables)

PROPHET tiene por objeto simplificar el desarrollo y despliegue de sistemas distribuidos de alto rendimiento y seguros. El proyecto explora la manera de aprovechar el aumento de potencia de cálculo y ancho de banda para hacer los sistemas distribuidos y las redes más fiables y más fáciles de desarrollar y gestionar. Una de las principales áreas de enfoque de este proyecto está en la llamada Red Definida por Software (SDN, Software Defined Networking). En este nuevo e importante ámbito, el potencial para un comportamiento incorrecto de la red debido a errores de software es especialmente grande (13). La investigación del Dr. Kostić [3]aplica técnicas del campo de verificación de software para aumentar la fiabilidad de las redes OpenFlow, una modalidad habitual del paradigma SDN (12).

Tomando como referencia conceptual los sistemas distribuidos, la investigación del Dr. Kostić asimismo se propone construir sistemas distribuidos y redes que imitan la proporcionalidad energética de los sistemas biológicos. Esta investigación paralela para lograr redes y sistemas distribuidos en red energéticamente proporcionales, acomete la creciente necesidad de reducir el consumo de energía en equipos y redes TIC, que ha sido impulsado por el éxito sin precedentes de Internet (13).

1.3.4. Resumen de los sistemas similares

Después de realizado el análisis de los sistemas similares, teniendo en cuenta las características y posibilidades que proponen, se arriba a la conclusión que los mismos no satisfacen las necesidades, pues no tratan el despliegue distribuido de la manera en que se necesita. Uno de estos sistemas analizados plantea una solución similar a la que se quiere obtener, el cual es el Sistema distribuido con NI VeriStand, pero es un software privado, teniendo en cuenta que nuestro país está en medio de la migración a software libre no es factible su uso, solo se contará con el marco teórico que plantea. Está además el sistema HYDRA que plantea otra de las soluciones, pero es igualmente privativo y su arquitectura está planteada a manera de árbol es decir de forma jerárquica que es la estructura que se quiere mejorar. El proyecto PROPHET plantea la visión a la que se desea arribar pero no se cuenta con un código fuente que ayude

[3] el Dr. Dejan Kostić, merecedor del galardón Starting Investigator Award otorgado por el European Research Council (ERC) y Research Associate Professor en el instituto de investigación madrileño IMDEA Networks, está liderando el proyecto ERC PROPHET*.

a materializar lo que proponen. Por lo antes planteado no se utiliza ninguna de las aplicaciones mencionadas anteriormente.

1.4. Estructuras que permiten el envío de información por varios caminos

1.4.1. P2P

P2P son las siglas en ingles de Peer-to-Peer (Par a Par o Igual a Igual), que significa comunicación entre iguales, no es una red ni un software, más bien está definido como una estructura de red o una forma de organización lógica. Esto significa que P2P no define un protocolo en específico o reglas para su uso, P2P únicamente indica la manera en que se deben realizar las conexiones y la organización de nodos, pero dejando a la implementación definir detalles de coordinación (protocolos), estructura y seguridad (autentificación, sesión, entre otros.). (14)

Usualmente en una red, las computadoras están conectadas a un servidor central y se les llama clientes, de ahí que sea común la referencia a cliente-servidor. Como su nombre sugiere, en una red P2P las computadoras se conectan y comunican entre sí sin usar un servidor central, aprovechando, optimizando y administrando la capacidad de la red (ancho de banda) de modo que usa la mejor ruta entre todos los nodos o computadoras que la conforman (14).

P2P consiste en lograr que todos los nodos se comporten como iguales por lo que sería necesario unificar en una misma aplicación el cliente y servidor de XILEMA GRHS para que pueda comportarse como uno u otro dentro de la estructura de red. Actualmente 3 aplicaciones trabajan en conjunto sobre el sistema GRHS, uno dedicado al cliente, otro al servidor y otro sobre la interfaz web de administración, lo que hace imposible detener el desarrollo de los componentes principales de un sistema que actualmente posee fecha límite durante el tiempo que pueda tomar hacer que las versiones actuales de los mismos funcionen como un todo de forma estable (14).

1.4.2. Estructura de datos grafos

Los grafos, constituyen estructuras de datos en las que se pueden expresar relaciones de conexión entre diversos elementos denominados vértices. Cada conexión se representa por un dato llamado arista (15).

Un grafo es una estructura de datos que almacena datos de dos tipos:

- ✓ **Vértices o nodos**, con un valor almacenado (16).
- ✓ **Aristas o arcos**: cada una conecta a un vértice con otro y puede tener un valor almacenado (16).

- ✓ **Adyacente:** se dice que w es adyacente a v si existe la arista (v, w donde v y w son vértices) (16).
- ✓ **Camino:** secuencia de vértices tales que cada uno es adyacente al anterior (16).
- ✓ **Peso o costo:** las aristas pueden contener datos y uno de ellos puede ser el coste o peso asociado a esa arista se usa para determinar el coste de recorrer el camino (16).
- ✓ **Longitud del camino:** número de aristas que tiene (16).
- ✓ **Coste de un camino:** la suma de los pesos de sus aristas (16).
- ✓ **Camino simple:** es aquel en que todos los vértices son distintos, excepto quizás el primero y el último (16).
- ✓ **Ciclo:** es un camino de longitud al menos 1 que empieza y acaba en el mismo vértice (16).
- ✓ El **recorrido de un grafo** se realiza desde un vértice v dado, visitando el resto de vértices a través de caminos de v (16).

Tipos de recorrido

Recorrido en anchura y recorrido en profundidad.

• **Recorrido en anchura** (16)

- BFS = Breadth First Search[4].
- Visita los vértices de un grafo de forma ordenada:

- comenzando desde un vértice fuente y visitando los vecinos de este vértice.

- continuando para cada uno de los vecinos visitando sus respectivos vecinos y así sucesivamente hasta que todos respectivos vecinos o todos los vértices (alcanzables desde la fuente) se hayan visitado.

• **Recorrido en profundidad** (16)

- DFS = Depth First Search[5].
- Visita los vértices de un grafo de forma ordenada, pero no uniforme:

- expandiendo todos los vértices que se van visitando en un camino concreto hasta que no haya más vértices que visitó en dicho camino.

[4] Búsqueda Primero a lo ancho

[5] Búsqueda primero en profundidad

- regresando de forma recurrente (back tracking), de modo que se repite el paso anterior en cada uno de los vecinos de los vértices que se estén procesando en el camino actual.

1.4.3. Protocolo Spanning Tree Protocol (STP)

El Protocolo de Árbol de Extensión (STP) es un protocolo de capa dos publicado en la especificación IEEE 802.1 (17).

El objetivo del árbol de extensión es mantener una red libre de bucles. Un camino libre de bucles se consigue cuando un dispositivo es capaz de reconocer un bucle en la topología y bloquear uno o más puertos redundantes (17).

STP utiliza el algoritmo de Spanning Tree (STA) para determinar los puertos del switch de la red que deben de configurarse para el bloqueo a fin de evitar que se generen bucles. El STA designa un único switch como raíz y lo utiliza como punto referencia para todos los cálculos de rutas (18).

El protocolo Árbol de extensión explora constantemente la red, de forma que cualquier fallo o adición en un enlace, switch o bridge es detectado al instante. Cuando cambia la topología de red, el algoritmo de árbol de extensión reconfigura los puertos del switch o el bridge para evitar una pérdida total de la conectividad.

Los switches intercambian información (BPDU) cada dos segundos si se detecta alguna anormalidad en algún puerto STP cambiara de estado algún puerto automáticamente utilizando algún camino redundante sin que se pierda conectividad en la red (17).

El algoritmo, basado en costos de un enlace, realiza lo siguiente:

- Escoge un Switch raíz: Los switches poseen un número que consiste de un número de prioridad y la dirección MAC (hay un número por puerto), todo junto forma un ID de Switch (estrictamente es de Bridge). Se elige como raíz al puente con prioridad más baja y si hay coincidencias en prioridad, se escoge el que tenga una dirección MAC más pequeña (comparando desde el bit menos significativo).
- Designa los switches adyacentes a la raíz como switches designados, cuya tarea es administrar las comunicaciones desde la LAN hacia la raíz.
- Cada switch determina su puerto raíz, que es el puerto que conecta la ruta con menor costo hacia el switch raíz.
- Selecciona para cada switch, los puertos que van a formar parte del STP, denominados puertos designados. Los demás son bloqueados (19).

Cada puerto del switch con STP pasa por los siguientes cinco estados:

- **De la inicialización al bloqueo:** Esto es, sólo recibe las BPDU. No se recibe ni envía información de usuario.
- **Del bloqueo a escuchar:** Recibe las BPDU y espera por cualquier nueva información que lo pueda regresar al estado anterior.
- **De escuchar a aprender:** No envía datos de usuario, pero si construye las tablas MAC.
- **De aprender a enviar:** Operación regular.
- **De envío a desactivado:** El puerto pasa a estar administrativamente apagado (19).

1.4.4. Resumen de estructuras que permiten el despliegue distribuido

Por lo antes expuesto se decide que la solución a implementar, esté basada en una estructura en forma de grafo para permitir el despliegue de servidores de manera distribuida, posibilitando que la información llegue por varios caminos aun cuando existan fallos en la conexión. Con esta estructura se maneja el control ante fallos y es más probable que la información llegue a su destino y en menor tiempo. También se contará con elementos que plantea la teoría del protocolo de árbol de expansión como la inexistencia de ciclos.

1.5. Metodología de desarrollo

Una metodología es una colección de procedimientos, técnicas, herramientas y documentos auxiliares que ayudan a los desarrolladores de software en sus esfuerzos por implementar nuevos sistemas de información. Una metodología está formada por fases, cada una de las cuales se puede dividir en sub-fases, 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.

1.5.1. RUP[6]

La metodología de desarrollo seleccionada es RUP y se caracteriza por ser:

[6] Rational Unified Process o Proceso Unificado de Modelado

Dirigido por casos de uso: Los casos de uso describen los requisitos funcionales del sistema desde la perspectiva del usuario y se usan para determinar el alcance de cada iteración y el contenido de trabajo de cada persona del equipo de desarrollo. (20)

Centrado en la arquitectura: La arquitectura permite ganar control sobre el proyecto para manejar su complejidad y controlar su integridad. Hace posible la reutilización a gran escala y provee una base para la gestión del proyecto. (20)

Iterativo e incremental: Se divide en 4 fases: Inicio, Elaboración, Construcción y Transición y cada una de ellas se divide en iteraciones. Cada iteración realizada añade funcionalidades al producto de software o mejora las existentes. Además, en cada iteración se trabaja en un número de disciplinas haciendo énfasis en algunas de ellas. Las disciplinas propuestas por RUP son: Modelado del negocio, Requisitos, Análisis y Diseño, Implementación, Pruebas, entre otras. (20)

Se seleccionó RUP como metodología de desarrollo ya que es una plataforma flexible de procesos de desarrollo de software que ayuda brindando guías consistentes y personalizadas de procesos para todo el equipo de proyecto. RUP describe cómo utilizar de forma efectiva reglas de negocio y procedimientos comerciales probados en el desarrollo de software para equipos de desarrollo de software, conocidos como “mejores prácticas”. Captura varias de las mejores prácticas en el desarrollo moderno de software en una forma que es aplicable para un amplio rango de proyectos y organizaciones. Provee a cada miembro del equipo fácil acceso a una base de conocimiento con guías, plantillas y herramientas para todas las actividades críticas de desarrollo. Crea y mantiene modelos, en lugar de enfocarse en la producción de una gran cantidad de papeles de documentación, además es la utilizada por el equipo de desarrollo de XILEMA GRHS.

1.6. Tecnologías y herramientas

1.6.1. Lenguaje de Modelado UML

Lenguaje Unificado de Modelado (UML por sus siglas en Inglés Unified Modeling Language) es la notación estándar para la descripción de métodos software, respaldado por el OMG (Object Management Group). Según su definición, UML es un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema. (21)

El principal objetivo de UML es "proporcionar a los arquitectos de sistemas, ingenieros de software y desarrolladores de software" herramientas para el análisis, diseño e implementación de sistemas basados en software, así como para el modelado de negocios y procesos similares. (21)

Es importante resaltar que UML es un "lenguaje de modelado" para especificar o para describir métodos o procesos. Se utiliza para definir un sistema, detallar los artefactos en el sistema, documentar y construir. En otras palabras, es el lenguaje en el que está descrito el modelo. (21)

Se puede aplicar en el desarrollo de software entregando gran variedad de formas para dar soporte a una metodología de desarrollo de software (tal como el Proceso Unificado Racional o RUP), pero no especifica en sí mismo qué metodología o proceso usar. (21)

Por las características anteriores, además de la integración con la metodología de desarrollo escogida se selecciona UML como lenguaje de modelado.

1.6.2. Herramienta CASE

Las herramientas CASE surgen para auxiliar a los desarrolladores de software, lo que permite el apoyo computarizado en todo o en parte del ciclo de vida del desarrollo de un sistema de software. (22)

CASE comprende una gran variedad de programas que se utilizan para ayudar a las actividades del proceso de software, como el análisis de requerimientos, el modelado de sistemas, la depuración y las pruebas. (23)

Las herramientas CASE han surgido para dar solución a varios problemas inherentes al diseño del software, principalmente nacen para solucionar el problema de la mejora de la calidad del desarrollo de sistemas de mediano y gran tamaño y en segundo término, por el aumento de la productividad. (23)

Para el modelado del sistema se utilizará Visual Paradigm como herramienta CASE.

1.6.3.1. Visual Paradigm 5.0

Visual Paradigm 5.0 es una herramienta de modelado. Su característica más importante es su flexibilidad ya que es multiplataforma, es decir, tiene la capacidad de ejecutarse sobre diferentes sistemas operativos además de constituir una herramienta de software libre. (24)

Fue creada por Visual Paradigm International (VPI), un proveedor de soluciones informáticas que incluye organizaciones para desarrollar aplicaciones de calidad, rápidas y baratas. Sus soluciones se enfocan en

eliminar la complejidad, aumentando así la productividad y disminuyendo el tiempo de desarrollo de las aplicaciones informáticas. (24)

Soporta todo el ciclo de vida del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Como lenguaje de modelado utiliza UML, ofreciendo soluciones de software que permiten a las organizaciones desarrollar las aplicaciones con calidad más rápido, de forma satisfactoria y más baratas. Posee gran facilidad de uso y el entorno gráfico que proporciona es agradable para los usuarios. Presenta una buena interoperabilidad con otras herramientas CASE y con los principales Entornos de Desarrollo Integrado. (24)

Visual Paradigm 5.0 fue la herramienta de modelado de software elegida para representar los artefactos de la aplicación debido a las facilidades que proporciona para la modelación de sistemas ya que agiliza la creación de los diagramas definidos en la metodología de desarrollo RUP. Se puede utilizar en sistemas operativos GNU/Linux incluyendo además que el equipo de desarrollo está familiarizado con el uso de la herramienta.

1.6.3. Lenguaje de Programación

1.6.4.1 Python 2.7.6

Python es un lenguaje de programación de alto nivel creado por Guido Van Rossum a principios de los años 90. Es un lenguaje con una sintaxis simple, sencilla y favorece un código legible. Es interpretado o de script, con tipado dinámico y orientado a objetos. La legibilidad permite acceder al código y tener una mayor comprensión de la implementación. (25)

Es multiplataforma ya que está disponible en plataformas como UNIX, Solaris, Linux, DOS, Windows, OS/2 y Mac OS, lo que posibilita la utilización de librerías específicas de cada plataforma y correr en todos estos sistemas sin grandes cambios. (25)

Ofrece gran soporte e integración con otros lenguajes y herramientas. Ofrece una biblioteca estándar muy amplia que incluye herramientas matemáticas y disímiles funcionalidades de gran ayuda desde el más bajo nivel hasta el más alto, facilitando al programador la implementación de aplicaciones sin la necesidad de recurrir continuamente a bibliotecas externas. Además dispone de una extensa colección de bibliotecas libres disponibles en la mayoría de los repositorios de los sistemas GNU\Linux. (25)

Debido a la portabilidad de su código; al programar las aplicaciones sin utilizar bibliotecas propias a los sistemas operativos, es posible ejecutarlas en cualquier plataforma sobre la que exista el intérprete de Python. (25)

Dichas características unidas a la simplicidad, interactividad, código abierto y excelente documentación, lo convierten en un lenguaje muy apropiado y codiciado para numerosas aplicaciones. (25)

Ofrece un entorno interactivo que facilita la realización de pruebas. El entorno de ejecución de Python detecta muchos de los errores de programación que escapan al control de los compiladores y proporciona información para detectarlos y corregirlos. Posee un amplio juego de estructuras de datos que se pueden manipular de modo sencillo. Además, una de las ventajas fundamentales de Python es la gratuidad de su intérprete (25).

Se selecciona Python como lenguaje de programación para el desarrollo de la aplicación por las características y ventajas presentadas anteriormente, teniendo en cuenta principalmente que el sistema XILEMA GRHS al cual se debe integrar la solución se encuentra desarrollada en el lenguaje Python.

1.6.4. Framework

Un Framework es un conjunto estandarizado de conceptos, prácticas y criterios para hacer frente a un tipo común de problema, que puede ser usado para resolverlo de forma rápida y eficaz. El objetivo de los Framework es proporcionar una estructura común, de modo que los desarrolladores no tienen que hacer el código de cero cada vez y puede volver a utilizar la gran mayoría.

1.6.5.1. Django 1.4.5

Es un marco de trabajo basado en la arquitectura Model-View-Template que solo cambia los nombres del estilo clásico modelo-vista-controlador y le otorga a las vistas toda la responsabilidad del negocio. Usa un motor de plantillas; además, ofrece facilidades en la gestión de usuarios y permisos, así como con el trabajo de las entidades mediante un módulo de administración, garantizando una gestión más sencilla. Tiene un buen ORM llamado django-models que facilita el acceso a los datos tanto para guardar como extraer la información. También permite generar las tablas en varios gestores de base de datos. Se le han hecho varias pruebas de rendimiento siendo mejor que varios marcos de trabajo de Ruby, PHP (Symfony y Zend) y Python (26).

Se decide utilizar el marco de trabajo Django pues la versión actual del sistema se implementó haciendo uso del mismo, además de que es más rentable usar el mismo lenguaje para la parte web, los agentes y servidores para ahorrar tiempo de estudio de otras tecnologías y poder hacer cambios de responsabilidades a cualquiera del equipo de desarrollo sin causar grande retardos en la adaptación. Django es un framework web de código abierto escrito en Python que permite construir aplicaciones web más rápido y con menos código.

1.6.5. Entorno de Desarrollo Integrado

Un Entorno de Desarrollo Integrado IDE (Integrated Development Enviroment por sus siglas en inglés) es un programa compuesto por un conjunto de herramientas útiles para un desarrollador de software. Como elementos básicos, un IDE posee un editor de código, un compilador/intérprete y un depurador.

1.6.5.1 Eclipse Helios

La plataforma Eclipse consiste en un IDE abierto y extensible. Da soporte a lenguajes de programación, como C/C++, Fortran, Java, PHP o Python. A la plataforma base de Eclipse se le pueden añadir extensiones (plugins) para extender las funcionalidades de la herramienta (27).

PyDev es una de las extensiones que existe para Eclipse, permite integrar a la plataforma un IDE para Python, el cual provee entre otras funcionalidades:

- Completamiento de Código.

- Resaltado de Sintaxis.

- Análisis de Código.

- Refactorizar.

- Debuguear.

- Consola Interactiva (28).

A través de la investigación realizada para Eclipse, se pudieron apreciar características significativas que demuestran que Eclipse constituye un IDE apto para el desarrollo de la aplicación que se desarrolla. Además es importante resaltar su alto nivel de integración con el leguaje a utilizar, Python. Por las razones descritas y las ventajas que proporciona se decidió la utilización de dicho entorno de desarrollo integrado.

1.6.6. Gestor de Base de Datos

Un Sistema Gestor de Base de Datos (SGBD, en inglés DBMS: Data Base Management System) es un conjunto de programas que permiten crear y mantener una base de datos, asegurando su integridad, confidencialidad y seguridad. Es un sistema de software que permite la definición de bases de datos; así como la elección de las estructuras de datos necesarias para el almacenamiento y búsqueda de los datos, ya sea de forma interactiva o a través de un lenguaje de programación. Un SGBD relacional es un modelo de datos que facilita a los usuarios describir los datos que serán almacenados en la base de datos junto con un grupo de operaciones para manejarlos. (29)

Los SGBD relacionales son una herramienta efectiva que permite a varios usuarios acceder a los datos al mismo tiempo. Brindan facilidades eficientes y un grupo de funciones con el objetivo de garantizar la confidencialidad, la calidad, la seguridad y la integridad de los datos que contienen, así como un acceso fácil y eficiente a los mismos.

1.6.6.1. PostgreSQL 9.2

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD y con su código fuente disponible libremente. Utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando. (30)

PostgreSQL puede ser extendido por el usuario añadiendo tipos de datos, operadores, funciones agregadas, funciones ventanas y funciones recursivas, métodos de indexado y lenguajes procedurales. También soporta conexiones que abarcan TCP/IP, sockets Unix y sockets NT, además de soportar completamente ODBC. Es bueno ante grandes informaciones pues soporta hasta 32 TB de capacidad en las tablas con tuplas de hasta 1,6 TB y campos de 1 GB. A pesar poseer las capacidades anteriormente descritas está bajo licencia libre sin ningún costo por su uso. A partir de su versión 9.1 tiene soporte para replicación de tablas siendo necesario en escenarios donde existan muchas estaciones inventariadas. (30)

Se selecciona PostgreSQL como gestor de base de datos por las características antes mencionadas y principalmente porque la versión actual de XILEMA GRHS funciona haciendo uso de dicho gestor.

1.7. Conclusiones parciales

Durante el desarrollo del capítulo se hizo un análisis de los sistemas como HYDRA, NI VeriStand y PROPHET. Los sistemas estudiados no cumplen con las necesidades que plantea el problema. Luego de realizado el estudio se evidencia la necesidad de garantizar la llegada de la información de manera distribuida por sus servidores. Se realizó además un estudio de las estructuras de datos que permiten el despliegue distribuido de servidores como, la estructura de datos grafos, P2P y STP donde la estructura escogida para realizar la solución del problema son los grafos, pues con ellos se garantiza que la información viaje por diferentes caminos.

Por otra parte se realizó un estudio de la metodología de desarrollo y se define RUP como metodología, por ser una plataforma flexible de procesos de desarrollo de software, que ayuda brindando guías consistentes y personalizadas de procesos para todo el equipo de proyecto y por ser la metodología que utiliza el equipo de desarrollo de XILEMA GRHS. Se estudiaron herramientas y tecnologías seleccionando para el lenguaje de modelado UML, vinculado a la herramienta CASE, Visual Paradigm, como lenguaje de programación, Python, como framework de desarrollo, Django, como entorno de desarrollo Eclipse Helios y como Gestor de Base de Datos PostgreSQL.

CAPÍTULO 2. PROPUESTA DE SOLUCIÓN, ANÁLISIS Y DISEÑO

2.1. Introducción

En el presente capítulo se efectuará un análisis y diseño abordando principalmente la propuesta del sistema así como los artefactos que se generan definiendo el modelado del proceso de negocio. Se definirán los requisitos del software especificándose los requerimientos funcionales, para lograr el despliegue distribuido de servidores. De igual forma se definirán los requerimientos no funcionales que puntualizarán las características del sistema, hardware, software y documentación en línea de la aplicación de forma general. También serán definidos los casos de uso del sistema, representándose en el diagrama de casos de uso correspondiente y los diagramas propios de los casos de uso

2.2. Propuesta del sistema

La solución se desarrollará sobre la base de la versión 1.0 del sistema XILEMA GRHS que actualmente permite la realización de inventario de hardware y software y las acciones pertinentes a una PC, así como el envío de información hacia el servidor principal, haciendo posible el despliegue de servidores de XILEMA GRHS mediante un esquema jerárquico en forma de árbol. La aplicación de dicho esquema posibilita tener comunicación entre los servidores y la descentralización de la información que se almacenaba anteriormente en solo un servidor garantizando así que si un área queda aislada de la red, se tenga control sobre ella con el servidor de XILEMA GRHS ubicado en el nivel de la jerarquía correspondiente. Haciendo uso de dicho esquema es posible realizar acciones desde un determinado nivel y no necesariamente desde un servidor central que gestiona a todos los clientes. La comunicación entre servidores se realiza de la misma forma que entre servidores y agentes, haciendo uso del protocolo HTTPS.

La solución propuesta plantea el despliegue distribuido de servidores GRHS, mediante un esquema en forma de grafo como se muestra en la Figura 7.

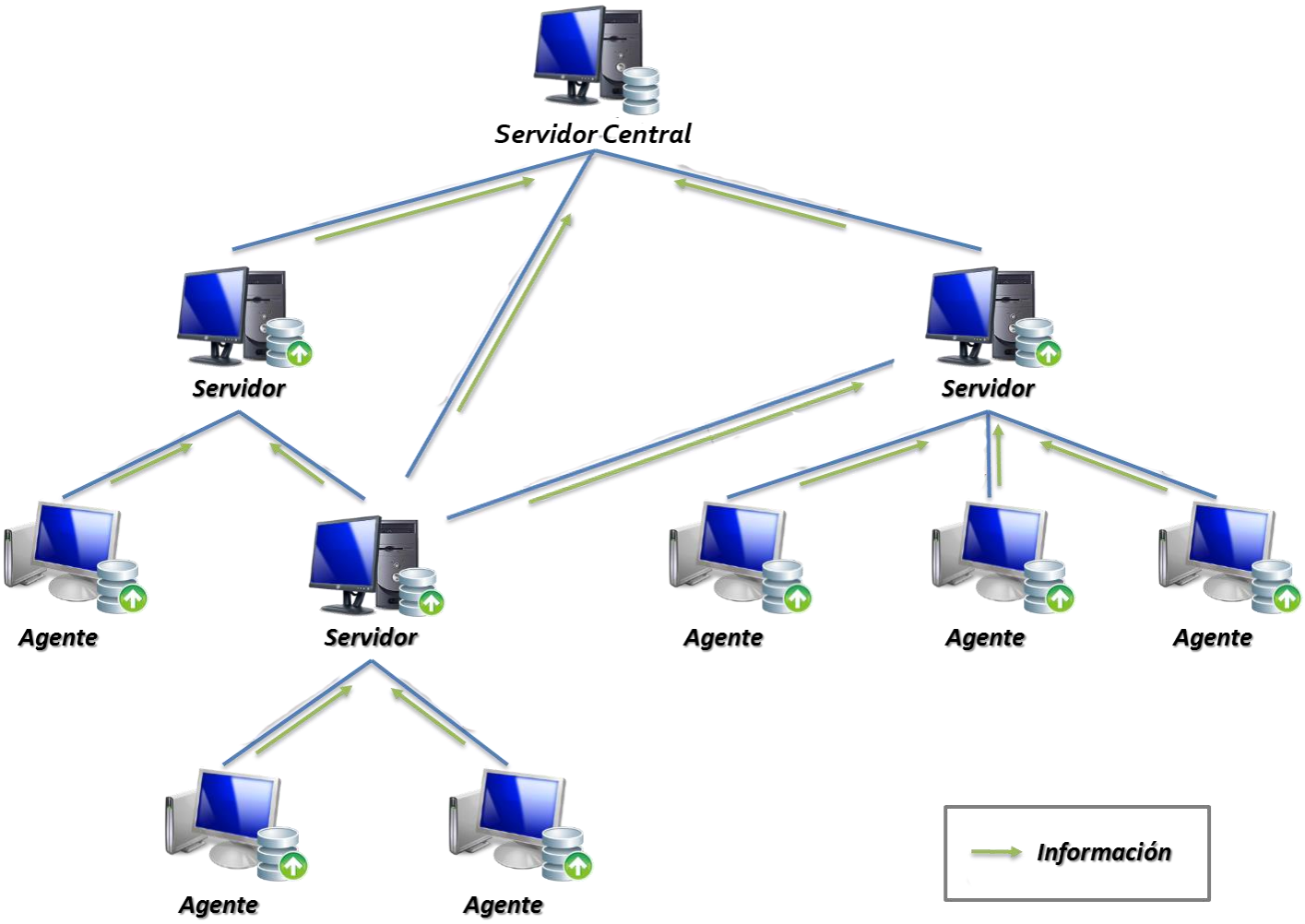


Figura 7: Propuesta de Solución

La aplicación de dicha estructura permitirá realizar el envío de información por varios caminos hacia el servidor principal, posibilitando que si existe fallo en la conexión la misma llegue por otro camino hacia su destino. Para evitar la formación de ciclos se adapta el concepto de zonas que en este contexto no es más que la agrupación de varios servidores los cuales no se pueden conectar entre sí. Permitirá además gestionar la configuración de sus servidores desde una interfaz gráfica en Gadmin facilitando el trabajo del administrador y no hacer engorroso el proceso de gestionar un archivo físico. Garantizará también que se muestre gráficamente la estructura de servidores configurados, así como los caminos definidos entre los mismos

teniendo como objetivo que el usuario pueda observar el diseño de la red por zonas de las conexiones entre servidores.

En la Figura 7 se puede observar el servidor central, que es al cual debe llegar toda la información del resto de los servidores y de sus clientes sin tener en cuenta si están conectados directamente o no. Cada uno de los servidores será capaz de manejar la información de sus clientes o servidores inferiores, permitiendo que si ocurre un problema en la conexión la misma llegue al destino por otro camino.

La solución planteada presenta una estructura de grafo no cíclico y dirigido en el cual es importante el orden del par de nodos que define cada arista. En la Figura 8 se observa un ejemplo del mismo.

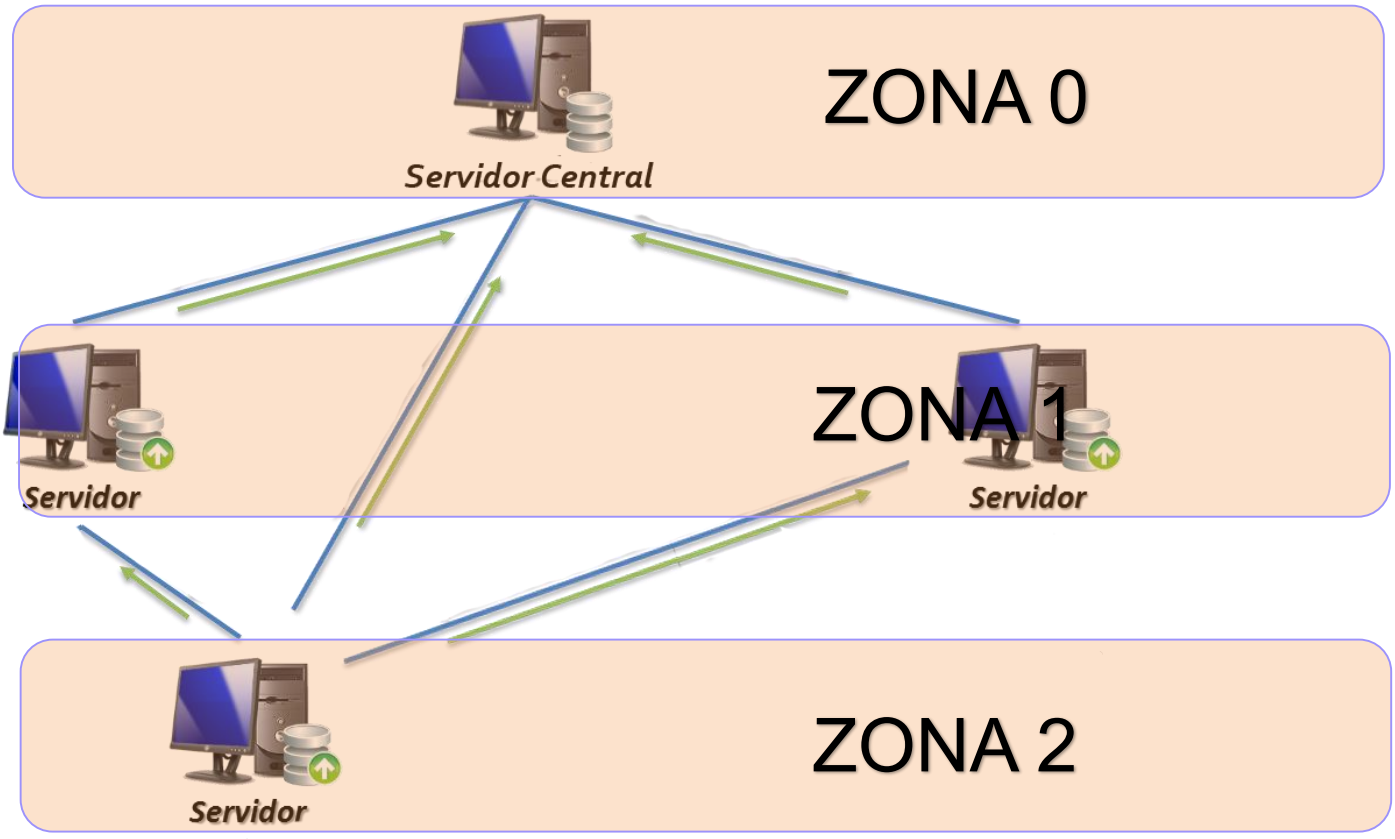


Figura 8: Grafo dirigido acíclico DAG por sus siglas en inglés

En la propuesta de solución se plantea esta estructura, donde está el servidor central de GRHS que se encuentra en la última zona definida (Zona n); el resto de los servidores también delimitados en zonas se

comunican con la zona anterior, ejemplo si el servidor se encuentra en la zona n+3 se puede comunicar con la zona n+2, zona n+1 y zona n, no siendo así la comunicación con la zona n+4, en esta zona la interacción es solo la recepción de la respuesta que manda el superior de si está conectado o no.

2.3. Diagrama de Negocio

A continuación la figura 8 muestra el Diagrama de proceso del negocio que muestra los principales procesos que estarán presentes en la solución.

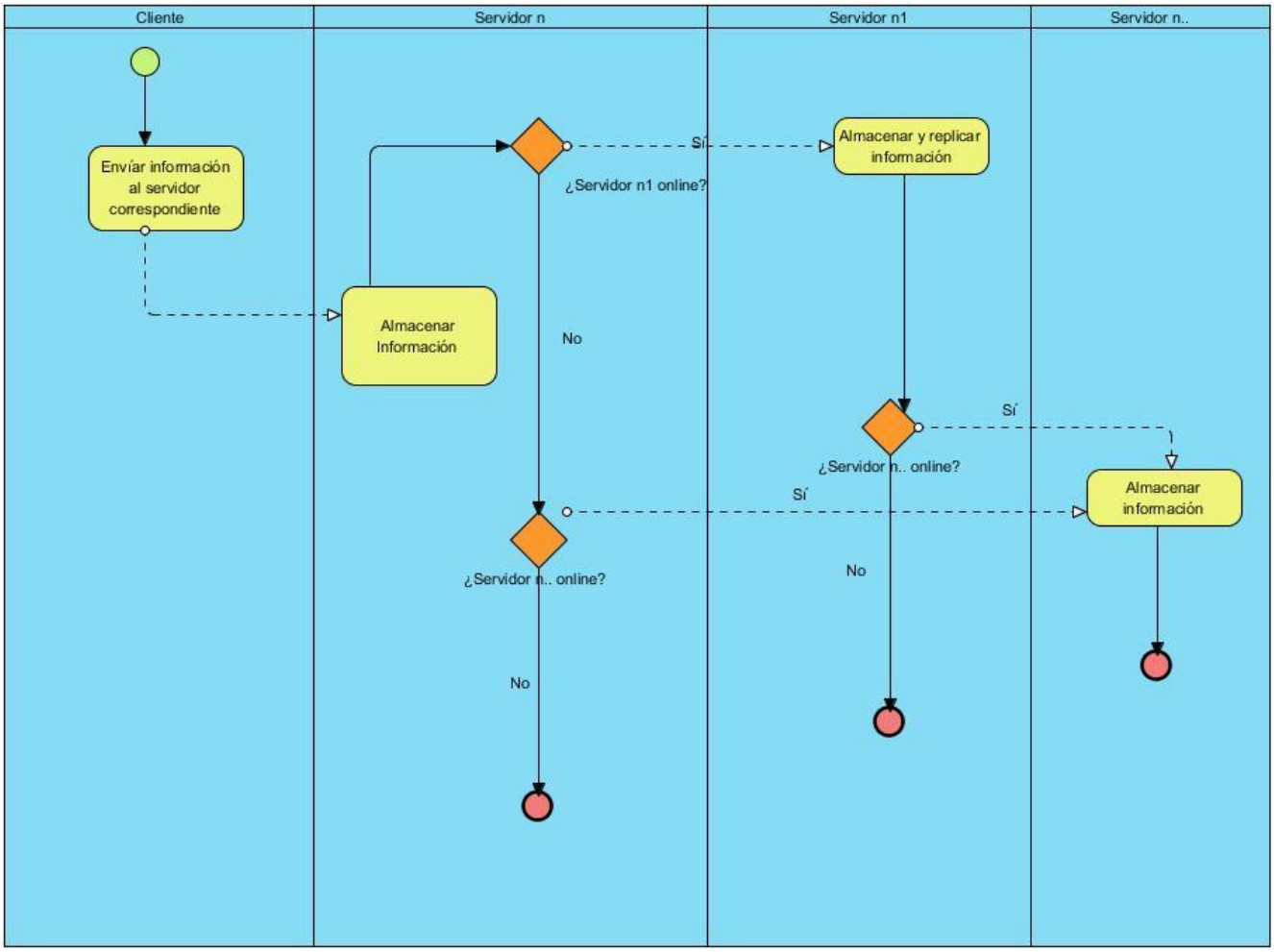


Figura 9: Diagrama de proceso del negocio

El diagrama de negocio representa el flujo de actividades presentes entre el cliente y el grupo de servidores que se encargan de enviar la información de un lugar a otro, replicar la información y almacenarla, de manera

tal que la misma llegue hacia el servidor central donde se almacenan todas las informaciones que han sido enviadas desde otros servidores.

Inicialmente el cliente envía la información al servidor correspondiente. El servidor que atiende esta petición debe identificar a cuál de sus servidores ubicados en la zona superior enviará la información. De no haber comunicación la información pasara a la siguiente zona realizando el mismo procedimiento hasta llegar al servidor central. Así se evidencia la formación de un grafo capaz de enviar la información por diferentes caminos hasta el servidor central.

Después que la información ha sido replicada se mostrara en la interfaz de administración los logs de comunicación, detallando a que servidores se comunicó.

2.4. Especificación de los requisitos del software

Luego de realizado el modelo del negocio se realiza el proceso de captura de requisitos funcionales del sistema, los cuales especifican en la siguiente tabla, siendo estos los encargados de especificar qué es lo que el sistema debe hace y definir las funcionalidades requeridas. (31)

2.4.1. Requisitos funcionales

Nº	Nombre	Descripción
RF 1	Replicar Información.	Posibilita el envío o réplica de información hacia otros servidores por diferentes caminos.
RF 2	Añadir configuración.	Posibilita añadir las configuraciones de los servidores desde la interfaz de administración, dado los campos: ✓ Zone ✓ Server ✓ Port ✓ Https ✓ Retry

RF 3	Modificar configuración.	Posibilita modificar las configuraciones de los servidores desde la interfaz de administración, dado los campos: ✓ Zone ✓ Server ✓ Port ✓ Https ✓ Retry
RF 4	Listar configuración.	Posibilitar listar los servidores que han sido configurados desde la interfaz de administración mostrando los campos: ✓ Zone ✓ Server ✓ Port ✓ Https ✓ Retry
RF 5	Eliminar configuración.	Posibilita eliminar las configuraciones de los servidores desde la interfaz de administración.
RF 6	Mostrar estructura de servidores.	Posibilita visualizar la estructura que posee el grafo de los servidores y los caminos definidos entre ellos en la interfaz de administración.
RF 7	Mostrar logs de comunicación.	Posibilita mostrar los logs de comunicación entre servidores.

2.4.2. Requisitos no funcionales

Describen aspectos del sistema que son visibles por el usuario que no incluyen una relación directa con el comportamiento funcional del sistema. Los requerimientos no funcionales incluyen restricciones como el tiempo de respuesta (desempeño), la precisión, recursos consumidos, seguridad, entre otros. Normalmente

están vinculados a los requisitos funcionales, para determinar la rapidez, la forma de comportamiento del sistema y sus cualidades. La solución presenta los siguientes requisitos no funcionales (32):

2.4.2.1. Software

- Tanto el cliente como el servidor requieren el intérprete de Python en su versión 2.7.6 instalado.
- El servidor de XILEMA GRHS requiere PostgreSQL en su versión 9.1 o superior, instalado y configurado correctamente.
- Deben tener instalada las dependencia de python: pygraphviz y networkx, además de las que utilizan Gadmin y Gserver las cuales están predefinidas

2.4.2.2. Hardware

- XILEMA GRHS:
- Gserver requiere para 500 clientes 2GB de RAM, procesador de 4 núcleos y 10 GB de disco duro
  Se requiere como mínimo 1GB de RAM, procesador Dualcore o superior a una velocidad de 2.2 GHz y 1GB de espacio libre en disco duro.

Los requerimientos de hardware para el servidor pueden variar en dependencia de la cantidad de clientes que se necesite instalar.

2.5. Definición de los casos uso

Los diagramas de casos de uso del sistema describen y documentan el comportamiento de un sistema desde el punto de vista del usuario, bajo la forma de acciones y reacciones.

Los casos de uso representan las funciones que un sistema puede ejecutar. Son descripciones de la funcionalidad del sistema independientes de la implementación, basados en un lenguaje natural, que le proporciona como ventaja principal la facilidad para interpretarlos además de ser muy útiles en la comunicación con el cliente. (33)

2.5.1. Diagrama de casos de uso del sistema

Luego de la captura de requisitos, se desarrolló utilizando el lenguaje de modelado UML el diagrama de casos de uso del sistema, en el cual se reflejan los casos de uso:

Gestionar configuración.

Replicar Información.

Mostrar estructura de servidores.

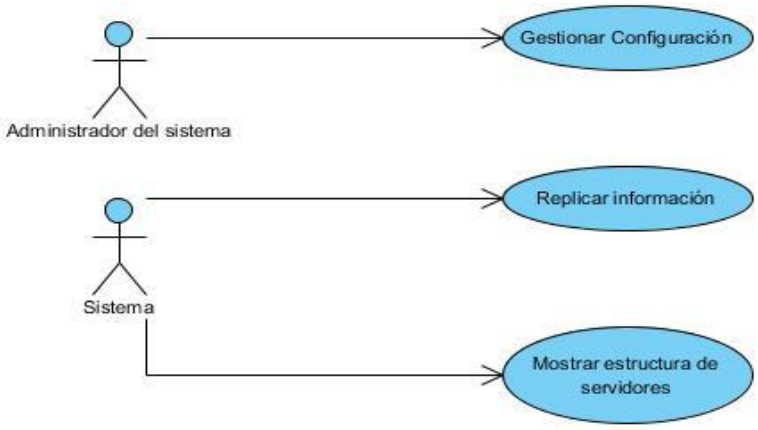


Figura 10: Diagrama de caso de uso

Actor	**Descripción**
Administrador del sistema.	Representa un rol del sistema que está destinado al usuario administrador que posee los privilegios y permisos para acceder a la información del sistema.
Actor	**Descripción**
Sistema.	Representa un actor ficticio, es el evento que desencadena las acciones de replicar información y ejecuta además otras acciones propias del sistema.

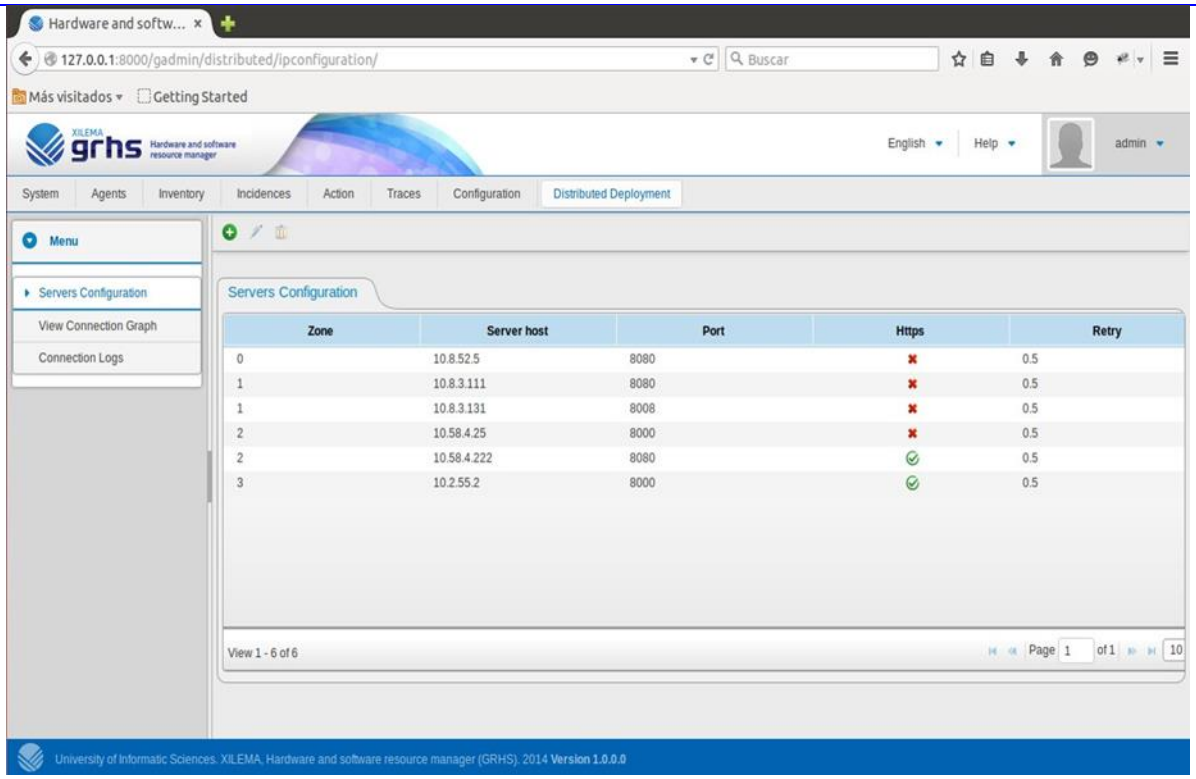
2.6. Descripción de los casos de uso del sistema

La descripción de los casos de uso del sistema tiene el fin de brindar una explicación detallada de las funcionalidades asociadas a cada caso de uso. Su elaboración debe ser clara para lograr un mejor entendimiento sobre lo que debe realizar el sistema en cada caso. Expresa de forma precisa las acciones que

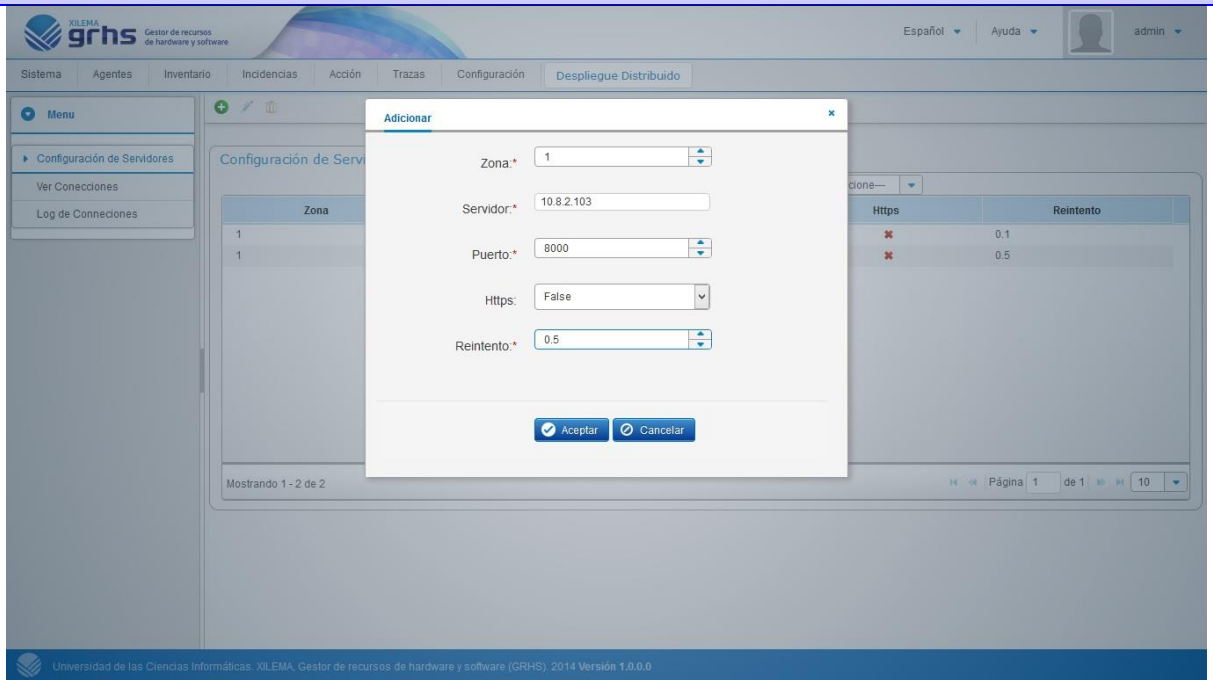
se realizan durante la interacción entre el actor y el sistema, describe el flujo de actividades que desarrolla el actor al hacer uso del sistema y las respuestas que emite el mismo. Las descripciones de los casos de uso del sistema se muestran a continuación. (33)

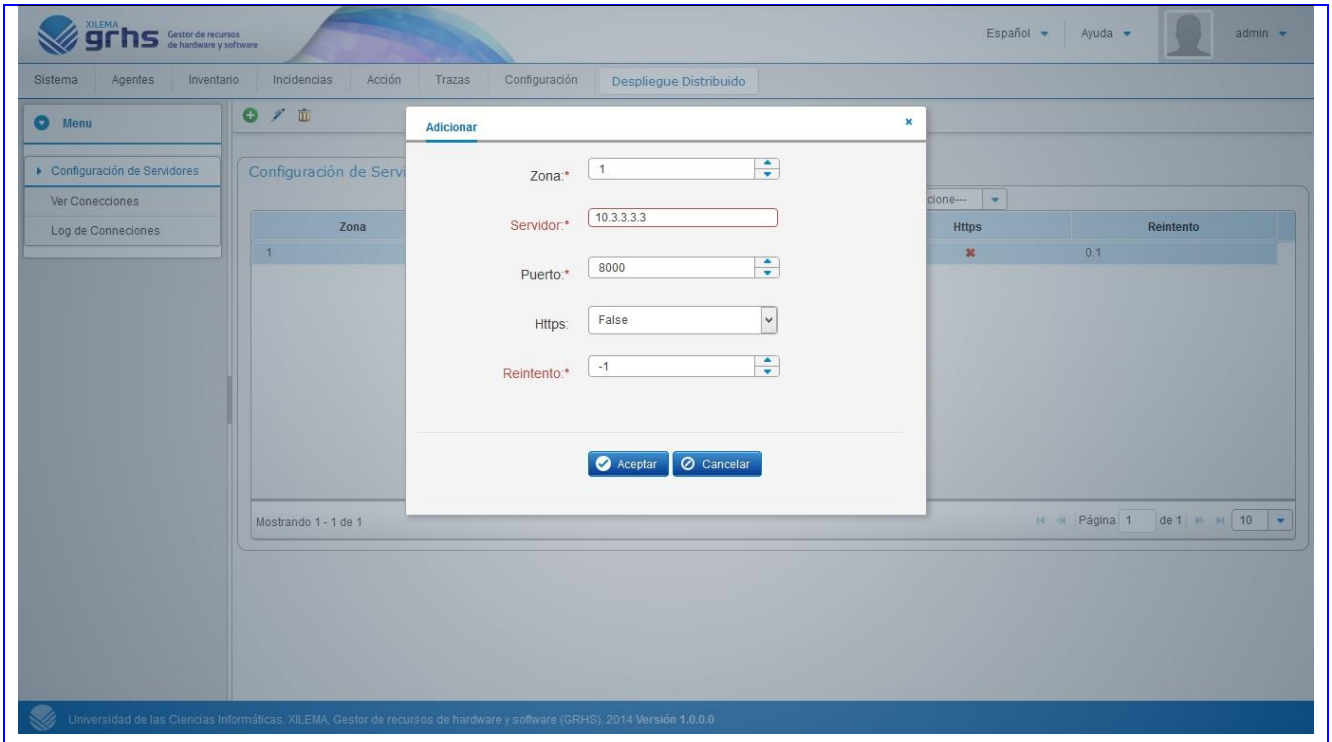
2.6.1. CU 1. Gestionar configuración.

<table>
<tr><td>Objetivo</td><td colspan="2">Gestionar las configuraciones de los servidores.</td></tr>
<tr><td>Actores</td><td colspan="2">Administrador del Sistema establece los parámetros de configuración</td></tr>
<tr><td>Resumen</td><td colspan="2">El caso de uso inicia cuando el administrador desea adicionar, eliminar o modificar las configuraciones de los servidores en el sistema. El sistema muestra los servidores ya existentes. El caso de uso termina cuando se adiciona, elimina o modifican las configuraciones.</td></tr>
<tr><td>Complejidad</td><td colspan="2">Alta.</td></tr>
<tr><td>Prioridad</td><td colspan="2">Alta.</td></tr>
<tr><td colspan="3">Flujo de eventos</td></tr>
<tr><td colspan="3">Flujo básico < Gestionar Configuración ></td></tr>
<tr><td></td><td>Actor</td><td>Sistema</td></tr>
<tr><td>1</td><td>El administrador selecciona lo opción Configuración de Servidores.</td><td>2 El sistema muestra un listado de los servidores ya existentes y las opciones:
✓ Adicionar
✓ Eliminar
✓ Modificar</td></tr>
<tr><td colspan="3">Prototipo de Interfaz</td></tr>
</table>



Sección Adicionar		
1	El administrador selecciona la opción Adicionar.	2 El sistema muestra la interfaz que contiene el formulario Adicionar con los siguientes parámetros: ✓ Zone ✓ Server ✓ Port ✓ HTTPs ✓ Retry (este parámetro se refiere al intento de reenvío en segundos)
3	El administrador introduce los datos solicitados.	4 El sistema verifica que los parámetros sean correctos.

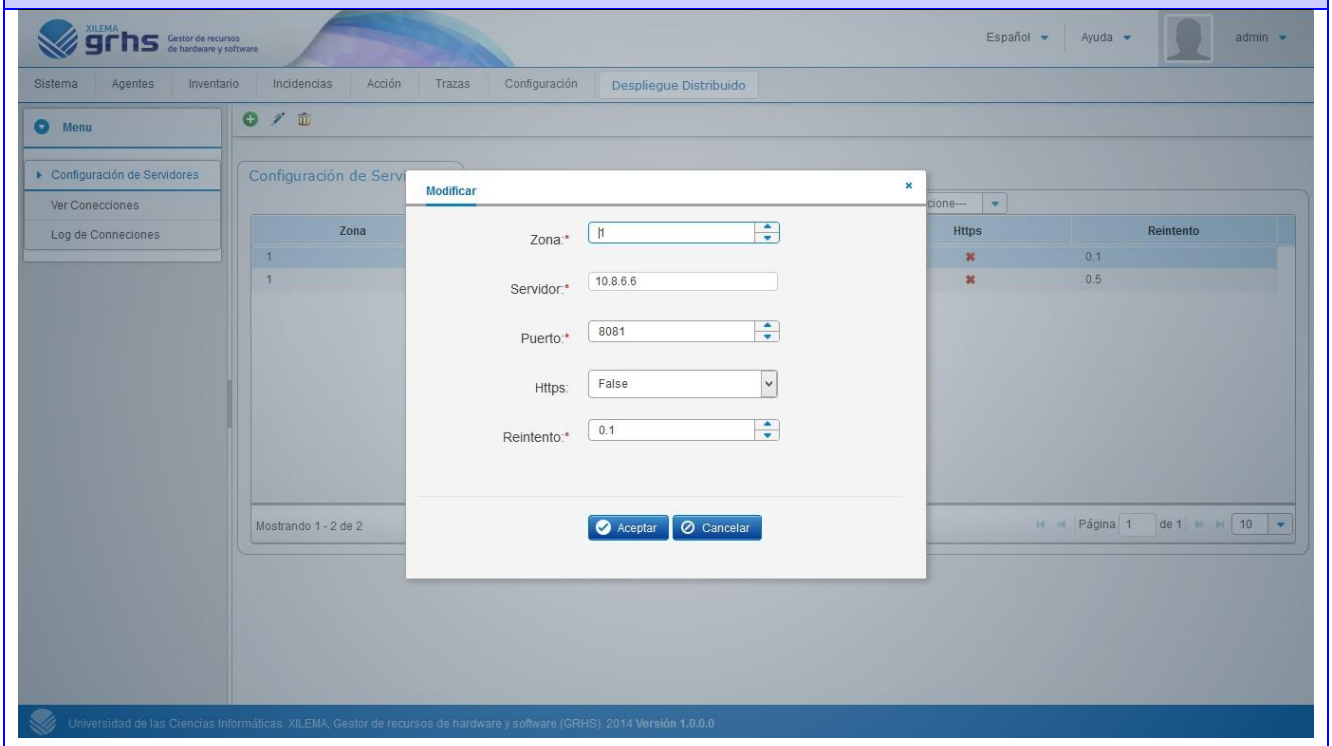
		5 el Sistema guarda la información referente a la configuración mostrando el mensaje “La operación se realizó con éxito”.
Prototipo de Interfaz		
		
Flujos alternos: Datos incorrectos		
1		El sistema muestra los datos incorrectos resaltados en color rojo.
Prototipo de Interfaz		



Sección Modificar		
1	El administrador selecciona el servidor que desea modificar.	2 El sistema muestra los datos del servidor seleccionado que será modificado según los parámetros: ✓ Zone ✓ Server ✓ Port ✓ HTTPs ✓ Retry (este parámetro se refiere al intento de reenvío en segundos)
3	El administrador modifica los datos deseados.	4 El sistema verifica que los datos estén correctos
		5 El sistema actualiza ls modificaciones

		6 El sistema muestra el mensaje “La operación se realizó con éxito”.

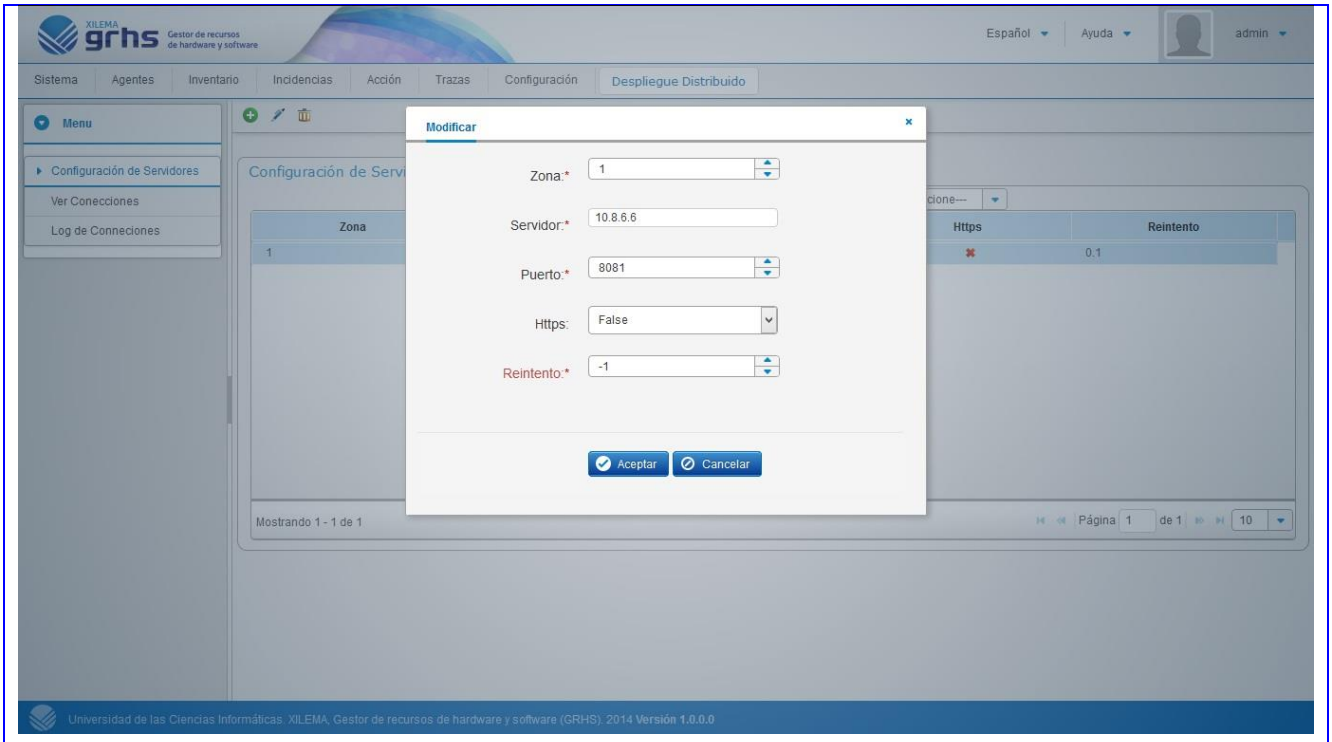
Prototipo de interfaz



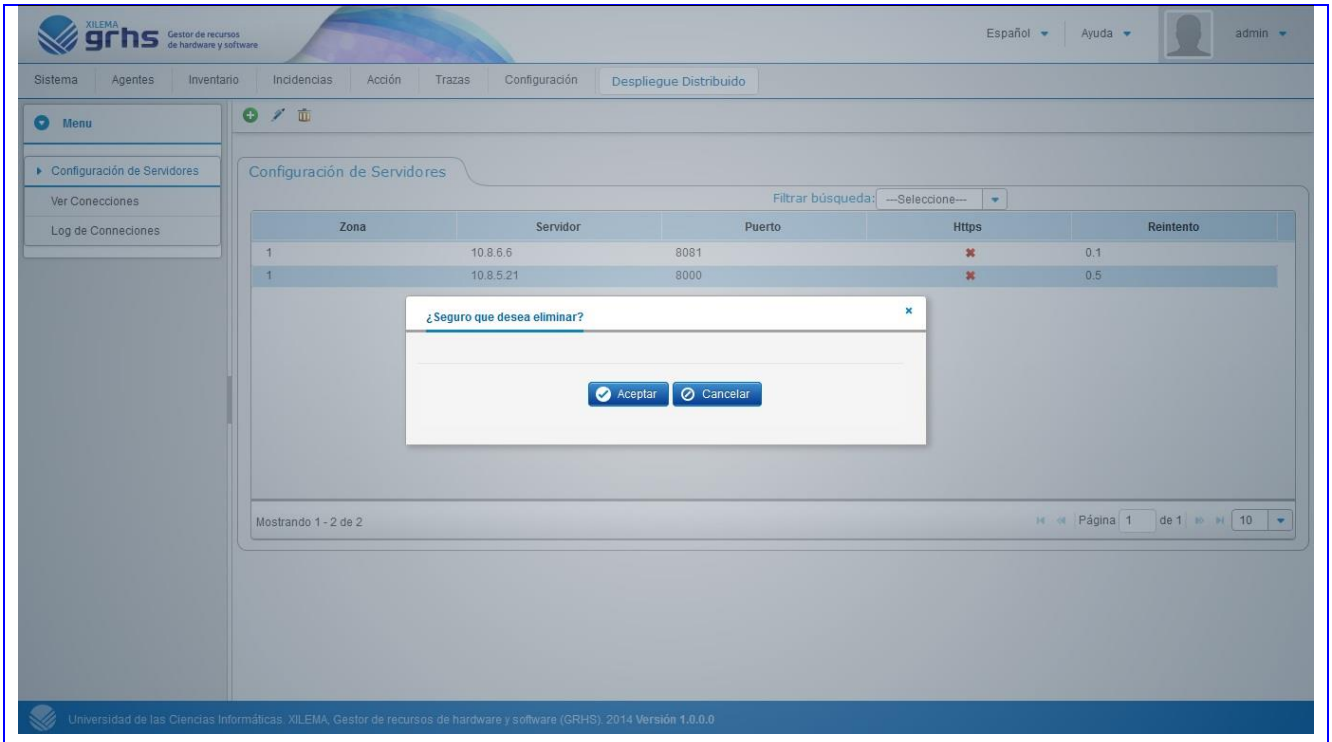
Flujo alterno Datos incorrectos

		1 El sistema muestra los datos incorrectos resaltados en color rojo.

Prototipo de Interfaz



Sección Eliminar		
1	El administrador selecciona el servidor que desea eliminar.	2 El sistema muestra un mensaje de confirmación ¿Seguro que desea eliminar? Y las opciones “Aceptar “y “Cancelar”.
3	El administrador selecciona la opción “Aceptar”.	4El sistema elimina el servidor y muestra el mensaje “La operación se realizó con éxito”.
Prototipo de intefaz		



Flujo Alterno Opción Cancelar		
5	El administrador selecciona la opción “Cancelar”.	6 Regresa a la Interfaz Configuración de Servidores.

2.6.2. CU 2. Replicar Información.

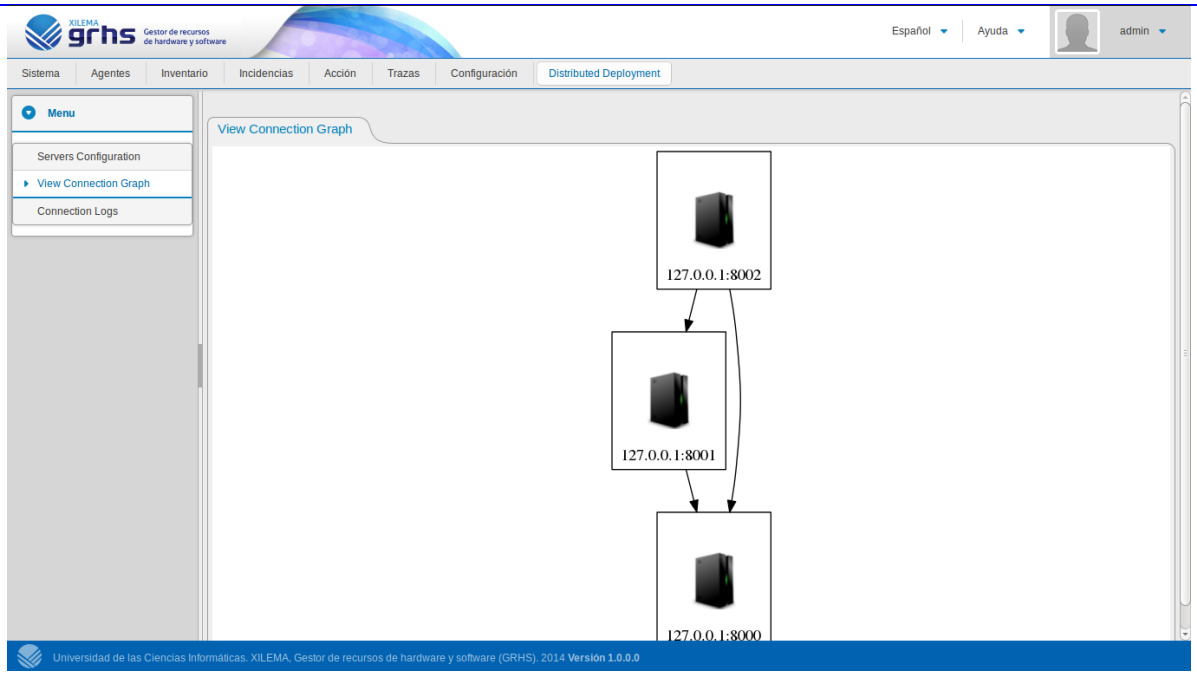
Objetivo	Replicar la información obtenida del (los) cliente (s) o servidor (es) en cuestión hacia el servidor padre.
Actores	Sistema
Resumen	El Sistema realiza la acción replicar información obtenida de uno o varios clientes o servidores.
Complejidad	Alta
Prioridad	Alta

Flujo de eventos		
Flujo básico < Replicar información >		
	Actor	Sistema
1	El cliente envía la información.	2 El servidor recibe información desde el cliente, una vez recibida toda la información realiza la acción replicar información hacia el servidor de una zona superior. El tiempo de volver a intentar la réplica es previamente definido en la gestión de la configuración.
		3 Almacena en el servidor de bases de datos la información que está siendo recibida.
		4 Muestra mensaje referente al estado de la conexión.
		5 Envía la información recibida hacia el servidor.
		6 Se crea una entrada de registro log con el estado del envío.
Evento<Error de conexión>		
1		La información recibida se almacena en la base de datos.
2		Espera el tiempo previamente definido en configuración (atributo Retry).

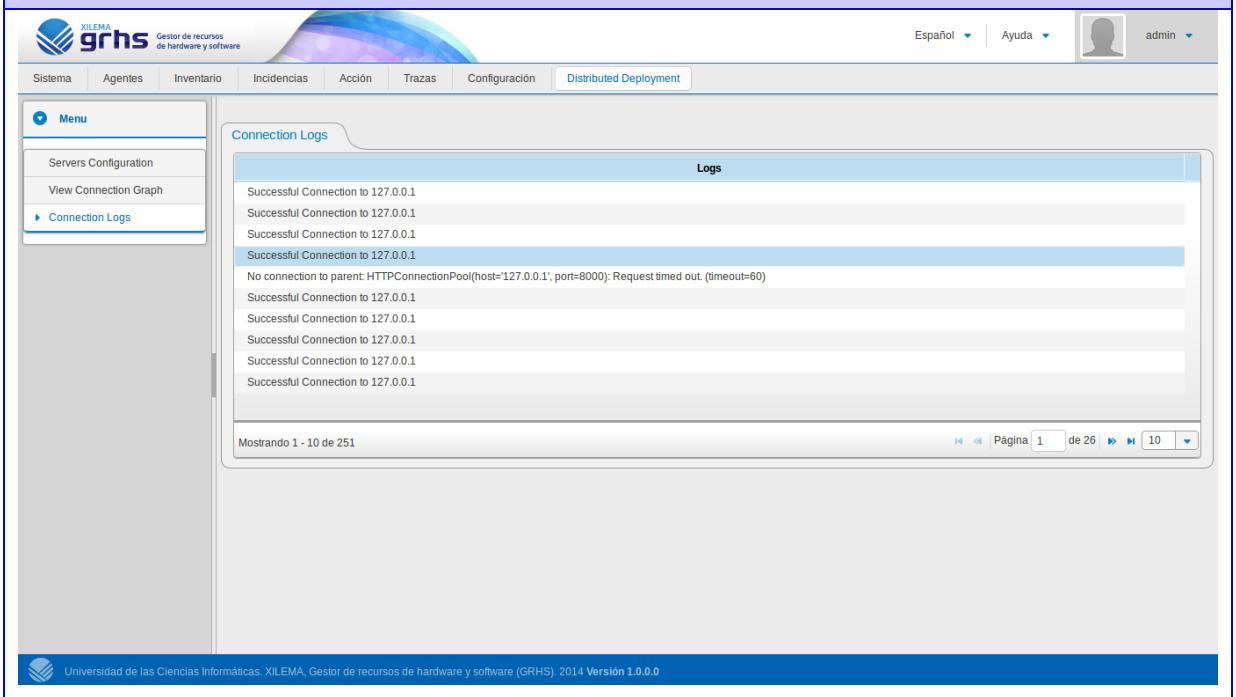
3		Reintenta la conexión desde el paso 2 del caso de uso.

2.6.3. CU 3. Mostrar estructura de los servidores.

Objetivo	Mostrar estructura de los servidores	
Actores	Sistema	
Resumen	El sistema muestra la estructura de los servidores y los caminos definidos por ellos.	
Complejidad	Media	
Prioridad	Alta	
Flujo de eventos		
Flujo básico < Mostrar estructura de los servidores >		
	Actor	Sistema
1	Se selecciona la opción Ver conexiones	2 El sistema muestra la estructura de los servidores y las conexiones que se establecen entre ellos.
3	Se selecciona la opción Log de Conexiones	4 El sistema muestra los logs de conexión
Prototipo de Interfaz Ver Conexiones		



Prototipo de interfaz Log de Conexión



2.7. Diagrama de clases del diseño

Un diagrama de Clases representa las clases que serán utilizadas dentro del sistema y las relaciones que existen entre ellas. Facilita visualizar las relaciones entre las clases que involucran el sistema, las cuales pueden ser asociativas, de herencia, de uso y de convencimiento. Un diagrama de clases está compuesto por los siguientes elementos: Clase: atributos, métodos y visibilidad. Relaciones: Herencia, Composición, Agregación, Asociación y Uso (34).

Los atributos de una clase no deberían ser manipulables directamente por el resto de objetos. Por lo que existen los niveles de encapsulamiento, los cuales son (35):

Niveles de encapsulamiento:

(-)Privado: es el más fuerte. Esta parte es totalmente invisible desde fuera de la clase (excepto para clases friends en terminología C++).

(~) Package: Sólo es visible dentro del mismo package (paquete).

(#) Los atributos/operaciones **protegidos** están visibles para las clases friends y para las clases derivadas de la original.

(+) Los atributos/operaciones **públicos** son visibles a otras clases (cuando se trata de atributos se está transgrediendo el principio de encapsulamiento).

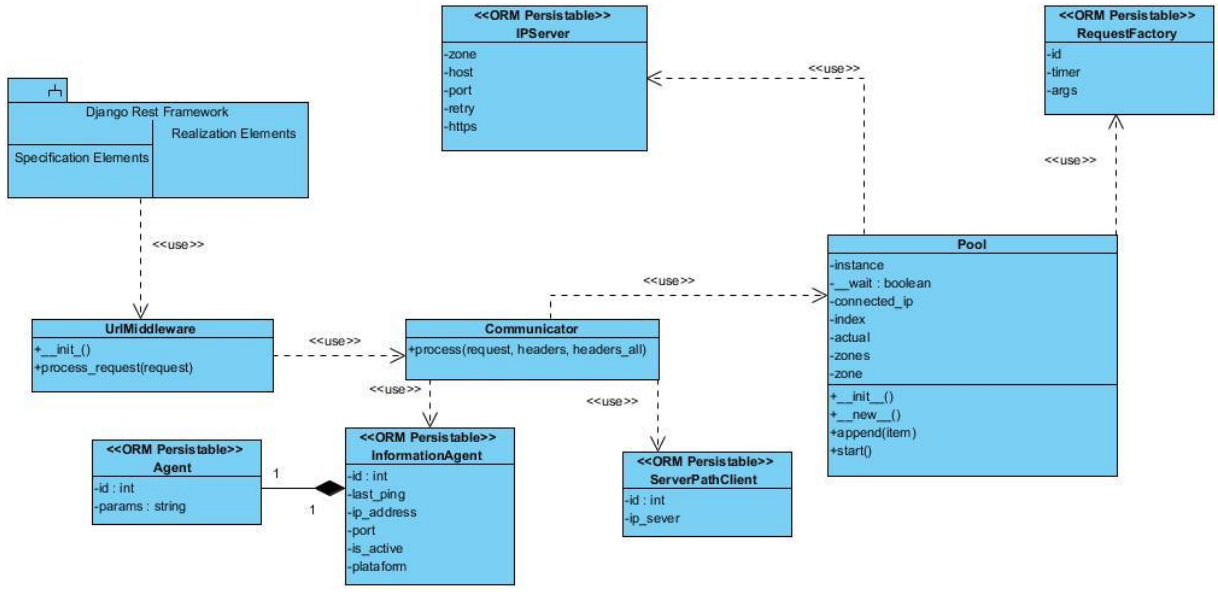


Figura 11: Diagrama de Clases del Diseño Replicar Información

La figura 11 muestra el Diagrama de Clases del Diseño del caso de uso Replicar Información, donde el paquete Django Rest Framework posee las vistas y las clases controladoras de las que hereda la clase UrlMiddleware. La clase UrlMiddleware es el intermediario de Django que procesa la petición (request), esta es enviada a la clase Communicator la cual procesa la información que es utilizada por la clase Pool, encargada de enviarla al próximo servidor.

La tabla Agent almacena la información del agente, el id generado con “uid”23 un algoritmo que permite crear un id único que identificará al agente dentro de toda la jerarquía del sistema en todos los niveles.

La tabla InformationAgent almacena la información del colector (agente), el id generado con uid un algoritmo que permite crear un id único que identificará al colector dentro de toda la jerarquía del sistema en todos los niveles. Además, en esta tabla se almacena toda la información necesaria para garantizar la comunicación con el agente.

La tabla ServerPathClient almacena el id de los agentes conectados a un servidor por medio de un segundo servidor, o sea, almacena el IP del servidor del cual proviene la información del agente que está llegando a un servidor x de la jerarquía. El campo id es un campo auto numérico.

La tabla RequestFactory almacena la información que debe ser enviada de un servidor a su padre cuando ocurre un problema de conectividad. En cada entrada almacena un id auto numérico y los datos en json de la información a enviar para que posteriormente cuando se reestablezca la conexión, el servidor automáticamente envíe toda la información pendiente y elimina las entradas de la tabla a medida que se comprueba el correcto envío de los datos.

La tabla IPServer almacena todos los datos referentes a los servidores a los que se va a conectar.

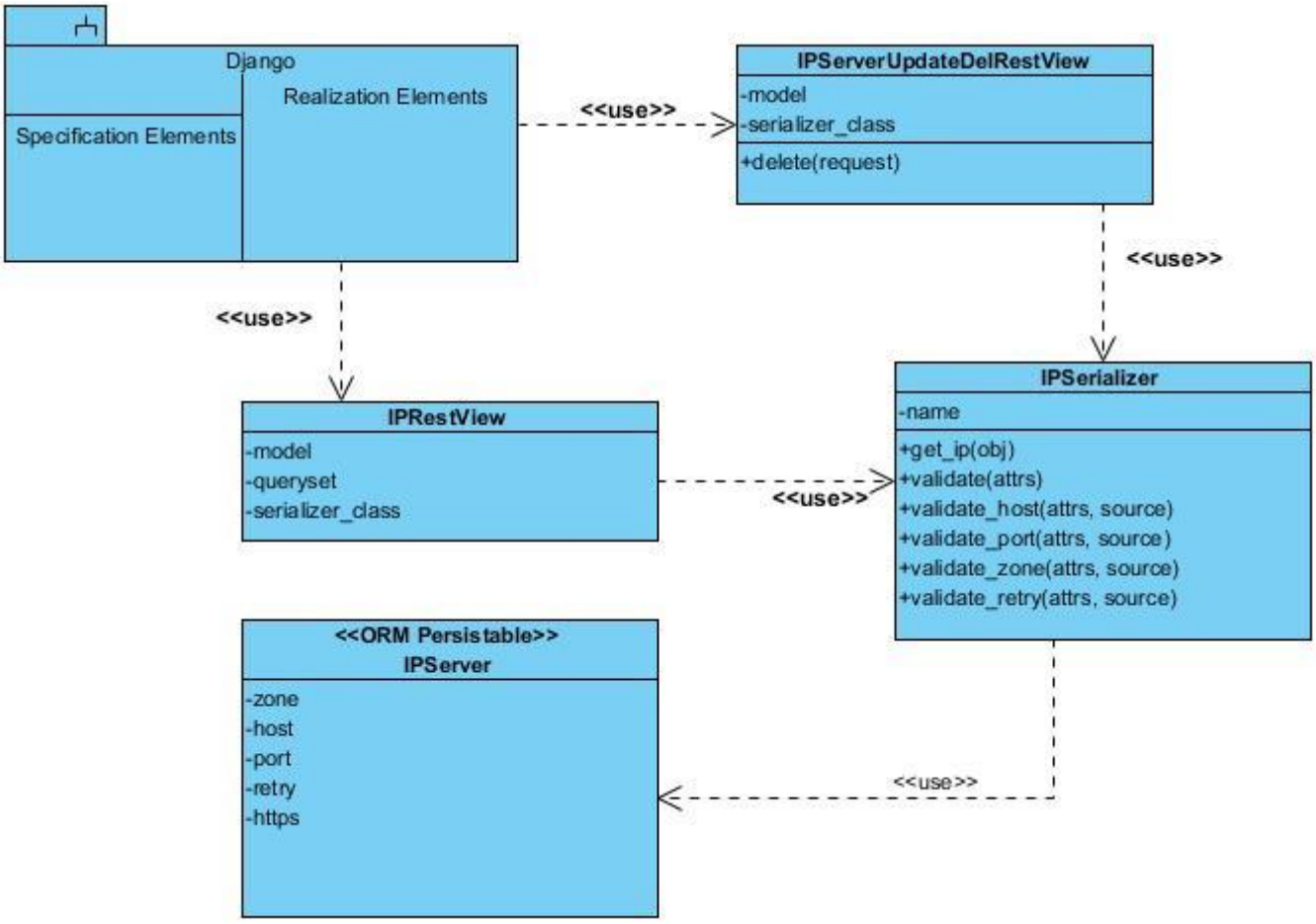


Figura 12: Diagrama de Clases de Diseño Gestionar Configuración

La figura 12 muestra el Diagrama de Clases del Diseño del caso de uso Gestionar Configuración donde el paquete Django posee las vistas y las clases controladoras de las cuales heredan las clases IPRestView, que se encarga de obtener los datos de la base de datos e IPServerUpdateDelRestView que se encarga de actualizar los datos en la base de datos cuando se adiciona modifica o elimina un servidor. Las dos clases anteriores utilizan la Clase IPSerializer donde se validan los datos que han de entrar a la base de datos.

La taba de la base de datos IPServer, almacena todos los datos referentes a los servidores a los que se va a conectar.

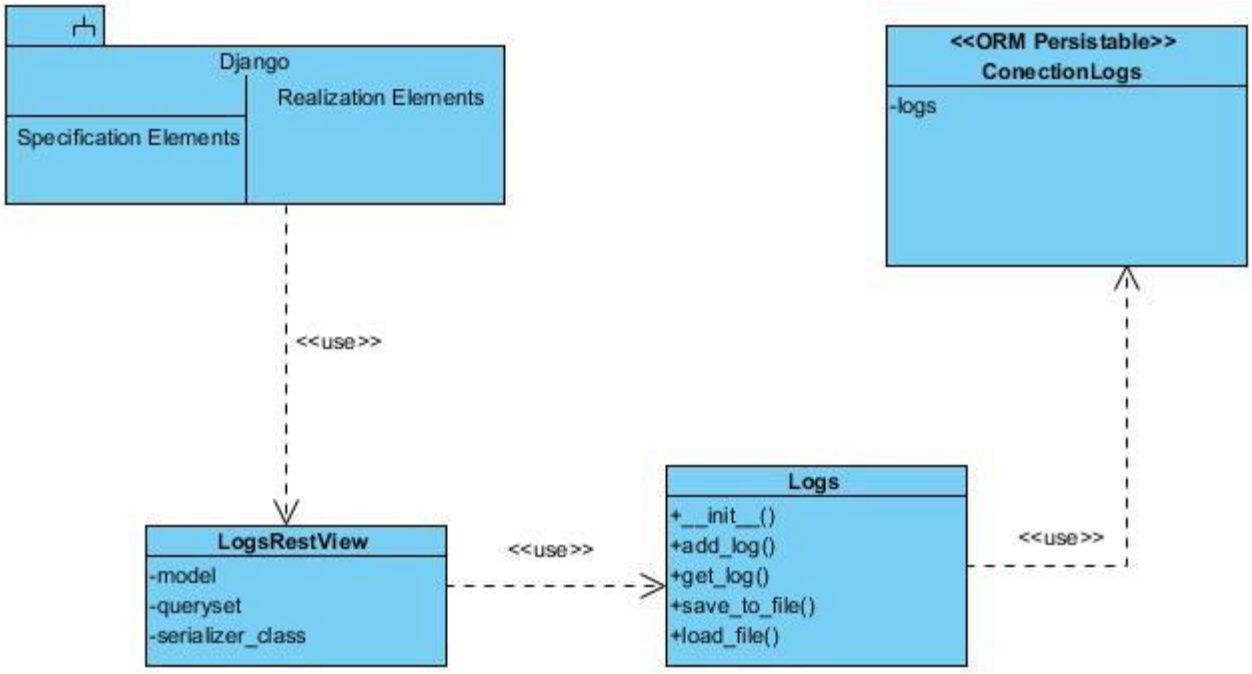


Figura 13: Diagrama de Clases del Diseño Mostrar Logs de Conexión

La figura 13 muestra el Diagrama de Clases del Diseño Mostrar Logs de Comunicación donde el paquete Django posee las vistas y las clases controladoras de que hereda la clase LogsRestView que se encarga de obtener los valores almacenados en la base de datos. Esta es usada por la clase Logs que me permite adicionar los logs, obtenerlos, salvarlos en un archivo físico y leer el archivo físico, este a su vez utiliza la taba de la base de datos ConnectionLogs. La tabla antes mencionada se encarga de obtener los logs de la base de datos.

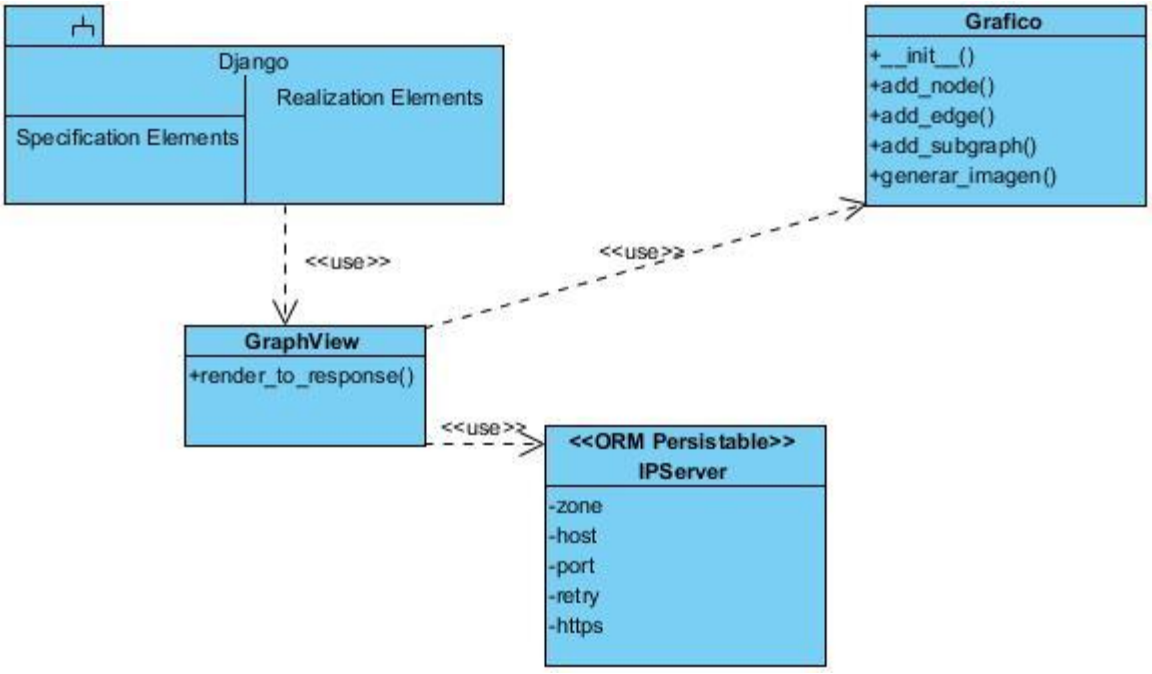


Figura 14: Diagrama de Clases del Diseño Mostrar Gráfico

La figura 14 muestra el Diagrama de Clases del Diseño Mostrar Gráfico donde el paquete Django posee las vistas y las clases controladoras de la que hereda la clase GraphView que contiene el método que genera el grafo. Esta utiliza las clases IPServer que almacena todos los datos referentes a los servidores a los que se va a conectar y Grafico que permite adicionar un nodo, adicionar borde, adicionar subgrafo y generar la imagen del grafo obtenido.

2.8. Modelo de Datos

El diseño de bases de datos es el proceso de analizar un problema para el que se necesita una solución, desarrollando el modelo lógico de los datos disponibles para la misma. Agrupando los datos en tablas relacionadas para así almacenar, manejar y recuperar dichos datos referentes a la solución. El modelo relacional se centra en esta idea: la organización de los datos en colecciones de tablas bidimensionales llamadas relaciones que reflejan las entidades existentes y las relaciones entre ellas (36).

Una base de datos correctamente diseñada permite obtener acceso a información exacta y actualizada. Puesto que un diseño correcto es esencial para lograr los objetivos fijados para la base de datos, parece lógico emplear el tiempo que sea necesario en aprender los principios de un buen diseño ya que, en ese caso,

es mucho más probable que la base de datos termine adaptándose a sus necesidades y pueda modificarse fácilmente (36).

La figura 15 muestra el modelo entidad relación (diagrama físico) e mismo solo muestra las principales tablas utilizadas en la solución. Las dos tablas señaladas en color azul son las tablas que fueron modificadas. Además se añadieron las tablas ipserver y connectionlogs.

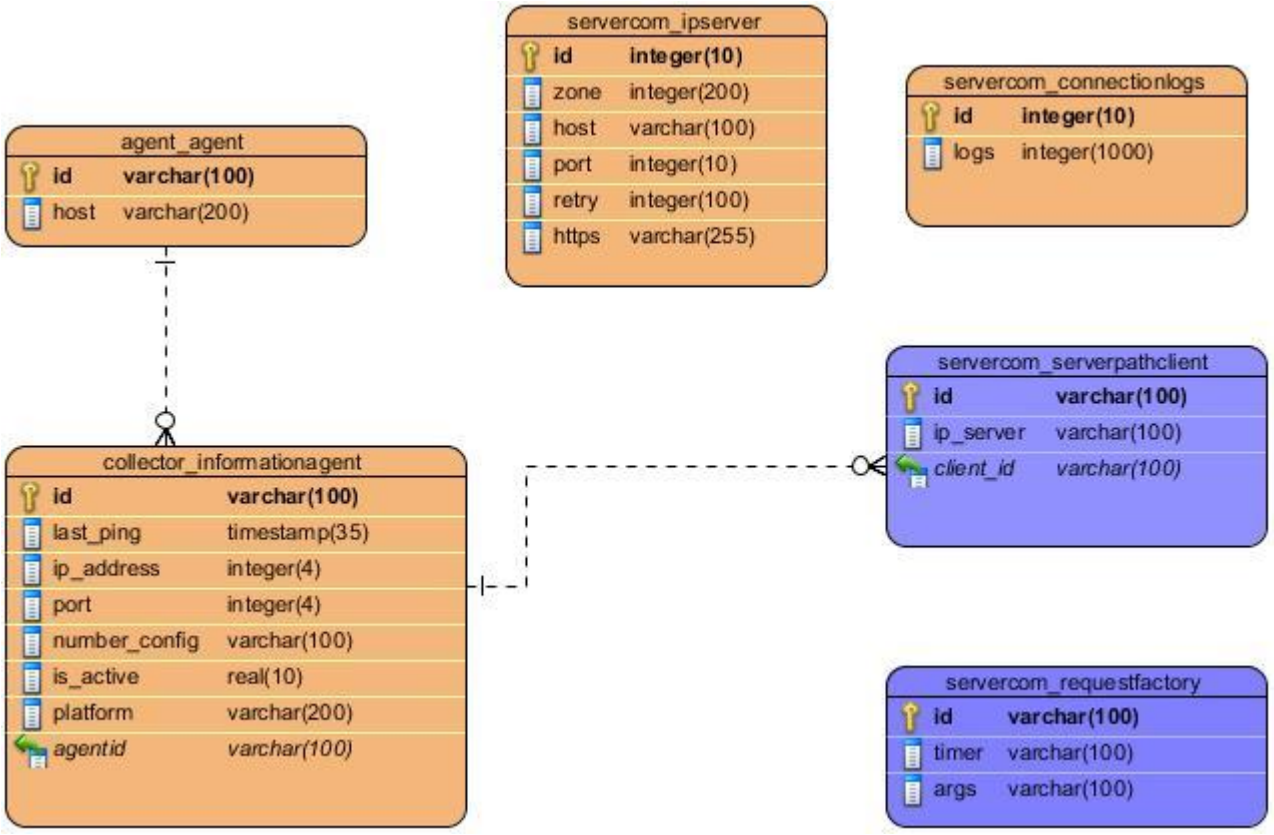


Figura 15: Modelo Entidad Relación de las principales tablas del Sistema

El diagrama de base de datos está compuesto por 131 tablas. Para la implementación de la solución dos tablas fueron agregadas por la presente solución al sistema XILEMA GRHS v1.0.

La tabla servercom_ipserver almacena todos los datos referentes a los servidores a los que se va a conectar.

La tabla servercom_connectionlogs se encarga de obtener los logs de la base de datos.

2.9. Diagrama de Secuencia

A continuación se muestran los diagramas de secuencia realizados en el proceso de implementación.

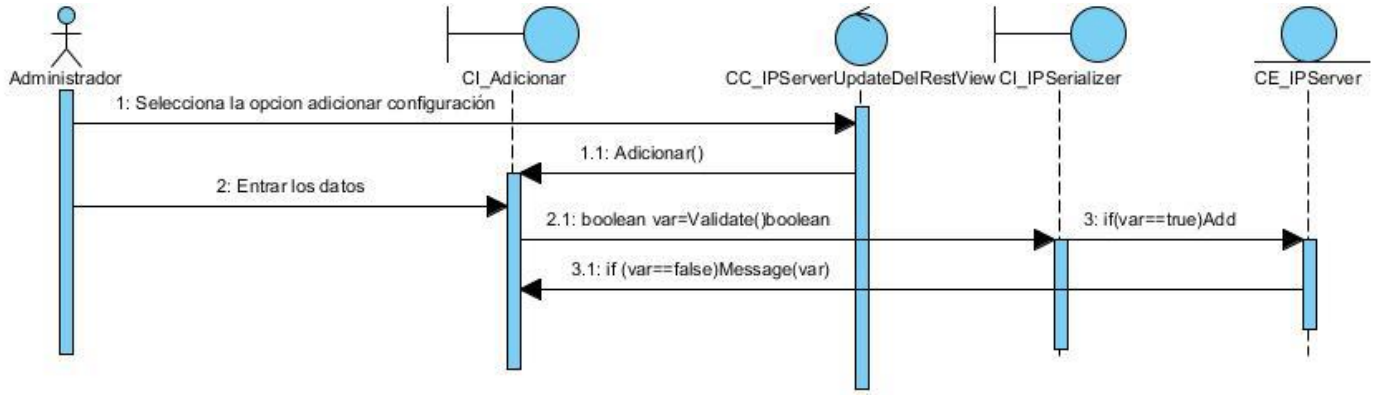


Figura 16: Diagrama de Secuencia Adicionar Configuración

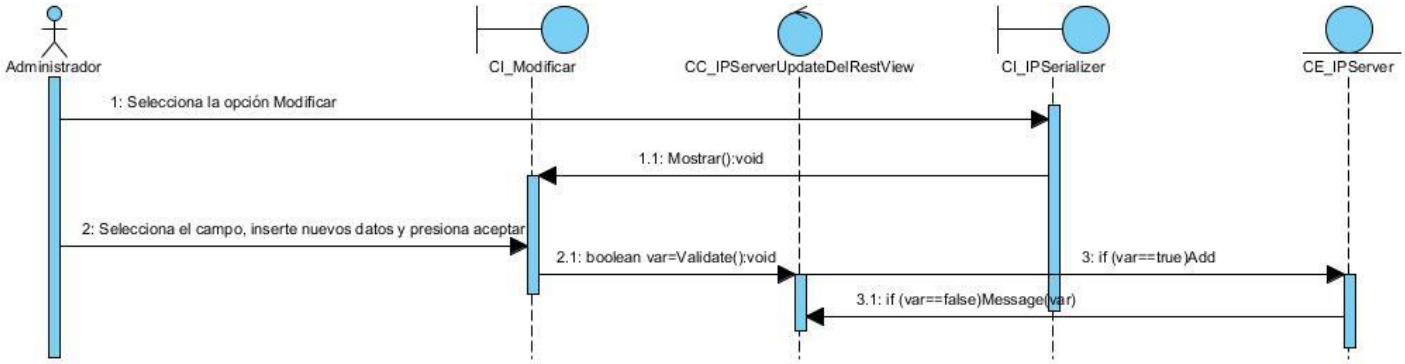


Figura 17: Diagrama de Secuencia Modificar Configuración

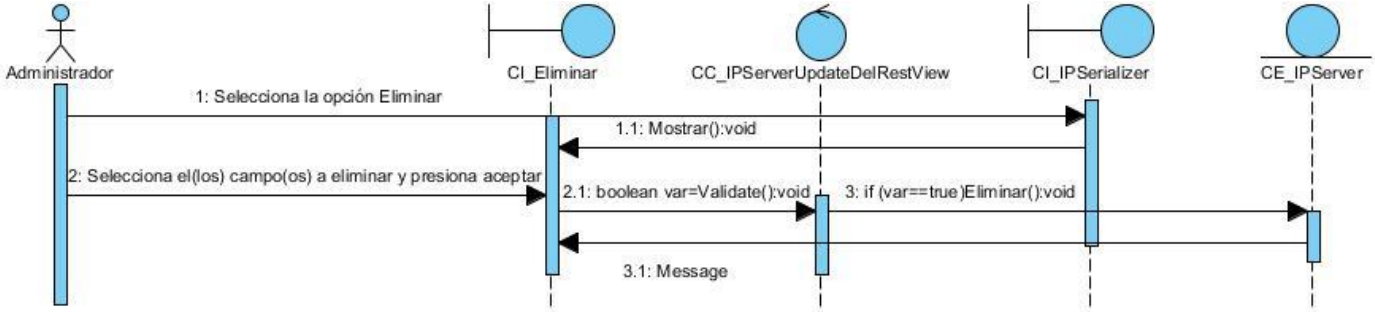


Figura 18: Diagrama de Secuencia Eliminar Configuración

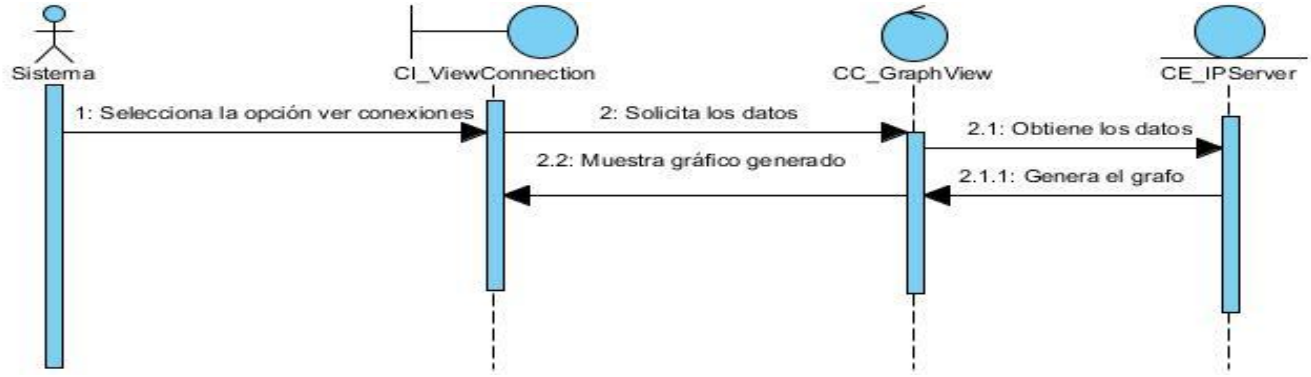


Figura 19: Diagrama de Secuencia Mostrar Gráfico

2.10. Conclusiones parciales

Durante el desarrollo del capítulo se realizó una descripción del negocio de la solución, se efectuó un análisis de las características del sistema que se presenta, lo cual contribuye a la documentación y correcta implementación de la aplicación. Se definieron los requisitos, tanto funcionales, como no funcionales y la representación del diagrama de negocio que hará posible el comienzo de la implementación del sistema. Para un mayor entendimiento de lo que debe realizar el sistema se realizaron las descripciones de los casos de uso para que el usuario que interactúe con el sistema no posea ninguna dificultad con su funcionamiento. Se conformaron además los diagramas de clases del diseño correspondientes a la metodología escogida, para mejor comprensión de la propuesta de solución

.

CAPÍTULO 3. ARQUITECTURA, IMPLEMENTACIÓN Y PRUEBAS

3.1. Introducción

El presente capítulo se enfoca en la definición de la arquitectura a utilizar la implementación del sistema y las pruebas que validen su correcto funcionamiento. Se realizarán diagrama de componentes y diagrama de secuencia. Se analizará y fundamentará la arquitectura del sistema, así como todo el marco de trabajo del desarrollo de la aplicación. Se especifican los patrones de diseño utilizados en la solución.

3.2. Arquitectura del sistema

La solución centrará la implementación en el servidor (Gserver), el cual utiliza el framework Django y su estructura está basada en la arquitectura del mismo.

Django basa su arquitectura en el patrón MVT el cual es una especificación de MVC. Permitiendo separar la lógica de negocio de las vistas de usuario y el acceso a datos. A grandes rasgos, esta es la manera en que se dividen la M, la V y la C en Django: (26)

1. M, el acceso a datos, lo gestiona la capa de base de datos de Django.

2. V, la parte que escoge qué datos mostrar y la manera de hacerlo, la controlan las vistas y las plantillas.

3. C, la parte que delega a una vista dependiendo de la entrada del usuario, la implementa el propio framework atendiendo a las urls configuradas programadas por el usuario e invocando a la función Python apropiada para una URL dada.

Ya que la "C" la maneja el propio framework y lo más novedoso de Django sucede en los modelos, plantillas y vistas, a Django se le ha definido como framework MVT. En el patrón de desarrollo MVT: (37)

1. La "M" se refiere al modelo, la capa de acceso de datos. Esta capa contiene todo lo referido a los datos: cómo acceder a ellos, cómo validarlos, qué comportamiento tienen y las relaciones entre ellos.

2. La "T" se refiere a las plantillas (Templates en inglés), la capa de presentación. Esta capa contiene las decisiones relacionadas con la presentación: cómo debería mostrarse el contenido en una página Web u otro tipo de documento.

3. La "V" se refiere a la vista, la capa de la lógica de negocio. Esta capa contiene el acceso al modelo y delega en la(s) plantilla(s) apropiada(s). Puede pensar en ella como el puente entre los modelos y las plantillas.

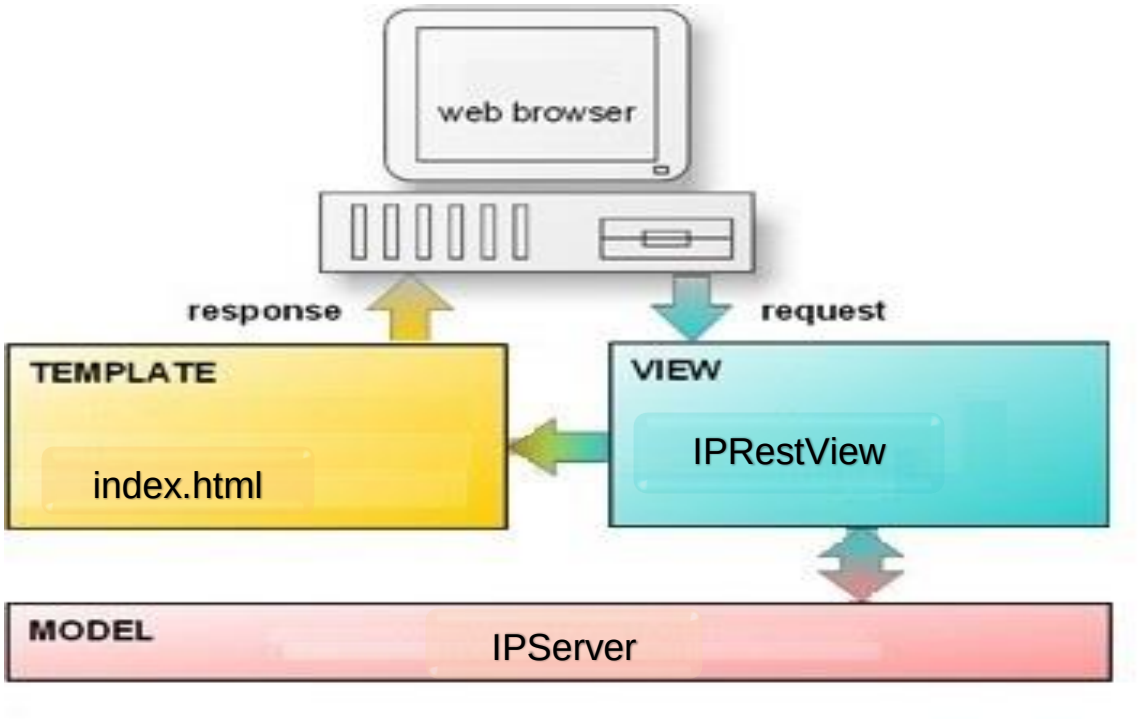


Figura 20: Modelo Vista Plantilla

En esta especificación del patrón modelo vista controlador el framework Django utiliza el componente modelo tal y como lo define MVC, tal como se videncia en la Figura 20. En esta figura se ejemplifica cómo funciona el patrón MVT .El buscador envía la petición al servidor la cual es procesada primeramente por el repartidor de url que se encarga de ejecutar la vista correspondiente, en esta se acceden a los datos a través de los modelos y se envía una plantilla como respuesta hacia el buscador. En este proceso existe un intermediario llamado middleware que se ejecuta antes de que la petición sea procesada por el repartidor de url. El framework middleware de Django, es un conjunto de acoples dentro del procesamiento de petición/respuesta de Django. Es un sistema de “plug-in” liviano y de bajo nivel capaz de alterar de forma global tanto la entrada como la salida de Django. Cada componente middleware es responsable de hacer alguna función específica (38). Django viene con algunos middleware incorporados para lidiar con problemas comunes, los cuales evidencian a continuación. **Middleware de soporte para autenticación**. Este middleware permite el soporte para

autenticación. Agrega el atributo request.user, que representa el usuario actual registrado, a todo objeto HttpRequest que se recibe. Otro de ellos es **Middleware "Common**. Este middleware agrega algunas conveniencias pues prohíbe el acceso a los agentes de usuario especificados en la configuración. Realiza re-escritura de URL basado en las configuraciones. **Middleware de soporte para sesiones**. Este middleware habilita el soporte para sesiones. Middleware "X-View. Este middleware envía cabeceras HTTP X-View personalizadas a peticiones HEAD que provienen de direcciones IP definidas en la configuración INTERNAL_IPS (38).

La siguiente imagen muestra la comunicación establecida para el envío de la información entre un Gserver y otro a través de los middleware. A un Gserver le se le realiza una petición la cual es procesada antes del repartidor de URL por los middlewares, estos se encargan de procesarla y enviar la información al siguiente Gserver.

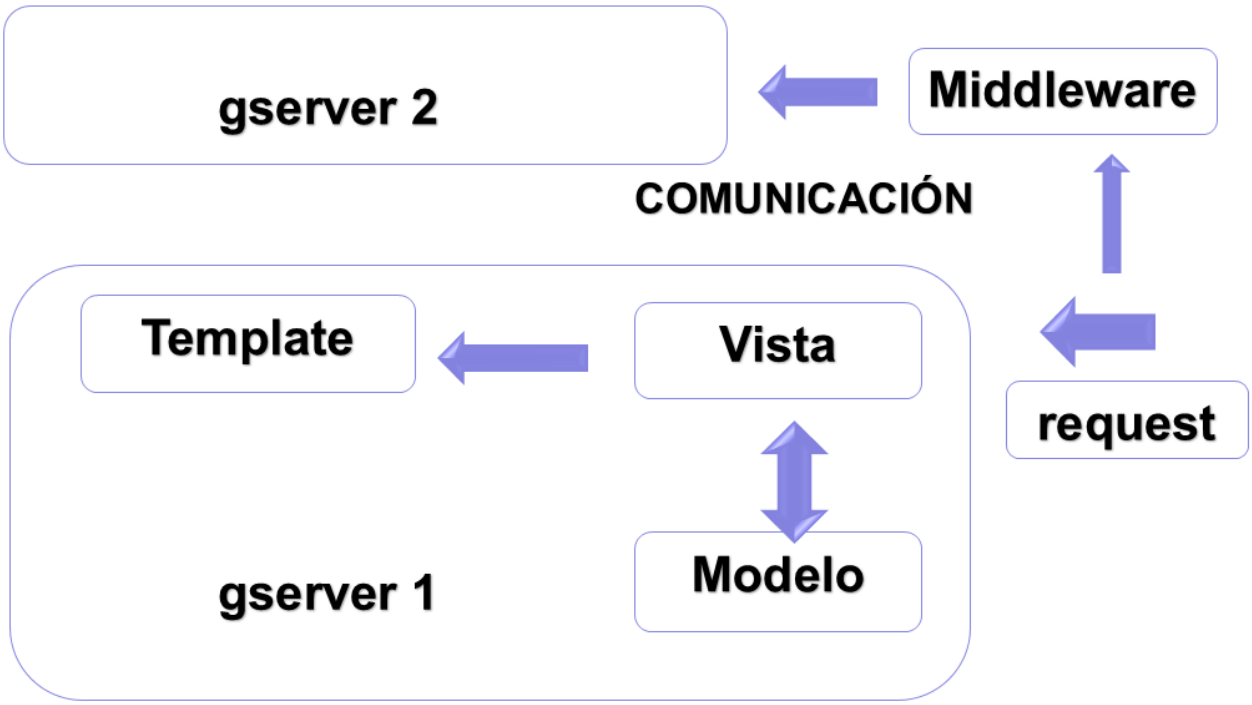


Figura 21: Comunicación entre un Gserver y otro utilizando middleware.

3.3. Patrones de diseño utilizados

Patrones de diseño: Aquellos que expresan esquemas para definir estructuras de diseño (o sus relaciones) con las que construir sistemas software. (39)

Los patrones GRASP[7] describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. (39)

Patrón Experto: Es el encargado de asignar una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad. (39) . Se refleja en las clases modelo de Django que son expertas en la información de la base de datos.

Patrón Creador: Es el encargado de asignarle a la clase B la responsabilidad de crear una instancia de clase A. (39)**.** En la presente solución se evidencia en la clase UrlMiddleware.

Patrón Bajo Acoplamiento: Es el encargado de asignar una responsabilidad para mantener bajo acoplamiento. El grado de acoplamiento no puede considerarse aisladamente de otros principios como Experto y Alta Cohesión. Sin embargo, es un factor a considerar cuando se intente mejorar el diseño. (39). Se refleja en la clase GraphView la cual utiliza solamente dos clases Grafico e IPServer.

Patrón Alta Cohesión: Es el encargado de asignar una responsabilidad de modo que la cohesión siga siendo alta. La cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase (39). Se refleja en la clase Communicator.

Se utilizaron también patrones del Gang of Four (GoF):

Creacional:

- **Singleton:** El patrón singleton consiste en crear una única instancia de un objeto para una aplicación y la creación de un mecanismo de acceso global a dicha instancia. El uso del patrón se evidencia en el

[7] GRASP es un acrónimo que significa General Responsibility Asignment Software Patterns (patrones generales de software para asignar responsabilidades).

manejador de replicación, en el cual se usa una única instancia del hilo que corre en segundo plano procesando la cola de peticiones. Este patrón se refleja en la Clase Pool.

3.4. Diagrama de Componentes

Se representa como un grafo de componentes software unidos por medio de relaciones de dependencia (compilación, ejecución), pudiendo mostrarse las interfaces que estos soporten. Son utilizados para estructurar el modelo de implementación dando una vista de alto nivel en términos de subsistemas de implementación y mostrar las relaciones entre estos (40).

Se utilizan para:

•Mostrar las dependencias de compilación de los ficheros de código.

•Mostrar las relaciones de derivación entre ficheros de código fuente y ficheros que son resultado de la compilación.

•Mostrar dependencias entre los elementos de implementación y los correspondientes elementos de diseño que son implementados.

Componente: Parte modular de un sistema, desplegable y reemplazable que encapsula implementación y un conjunto de interfaces y proporciona la realización de los mismos. Un componente típicamente contiene clases y puede ser implementado por uno o más artefactos (ficheros ejecutables, binarios, etc.) (40).

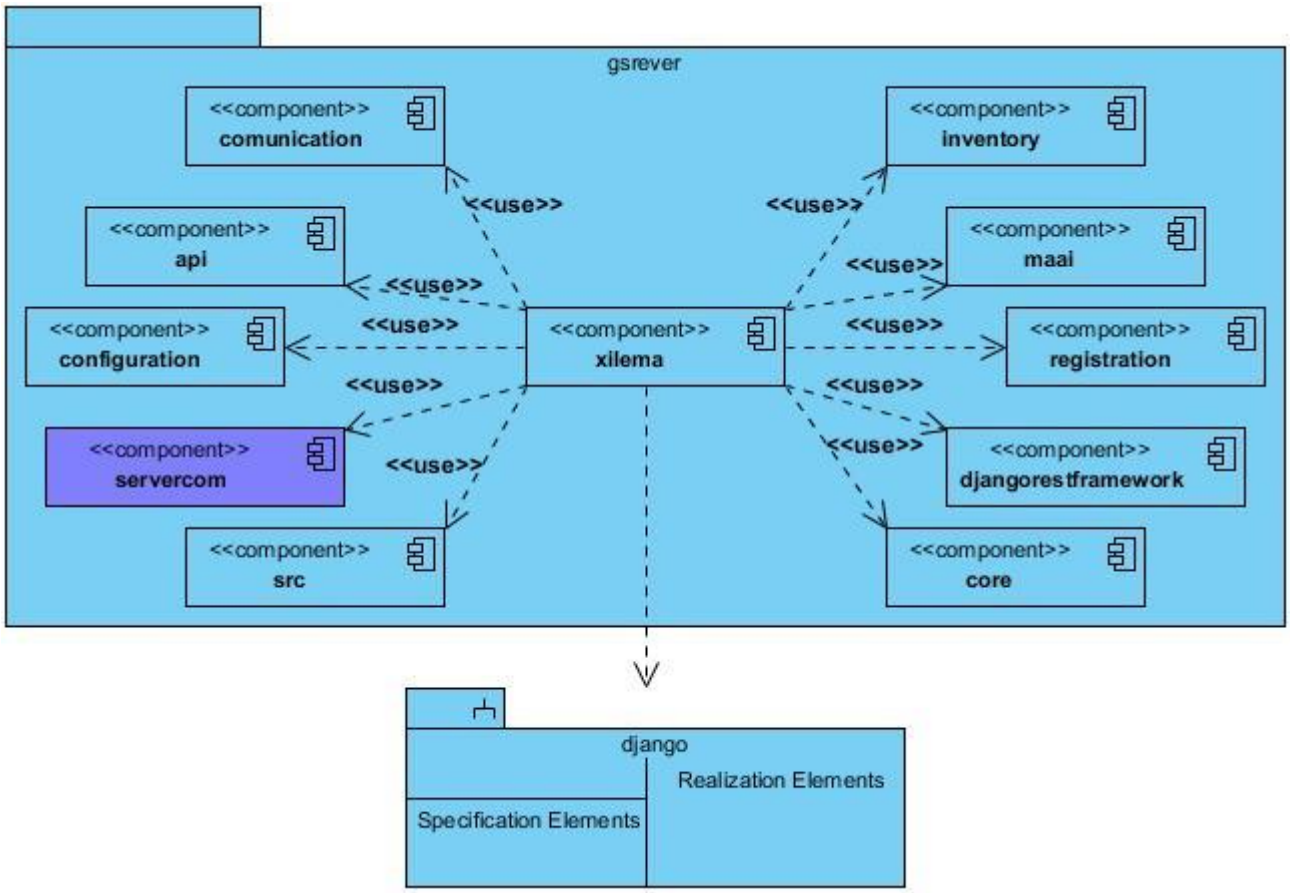


Figura 22: Diagrama de Despliegue

La figura 22 muestra el diagrama con todos los componentes de Gserver, entre los cuales el componente servercom ha sido resaltado este es el componente que contiene la implementación de la presente solución. El resto de los componentes realizan funciones específicas como se describe a continuación:

- comunication: contiene las implementaciones que permiten la comunicación cliente - servidor.
- api: contiene implementaciones por medio de las cuales el administrador puede consultar información del servidor a través de la interfaz web Gadmin.
- configuration: gestiona todas las configuraciones.
- src: es el componente donde se encuentra implementada la interfaz web del sistema Gadmin.
- inventory: contiene las implementaciones encargadas de almacenar la información que llega desde los clientes en la base de datos.

- maai: es el componente de acciones ante incidencias, encargado de realizar las acciones configuradas por el administrador cuando los clientes reportan incidencias en las estaciones de trabajo. Entre las posibles acciones se encuentra la notificación vía correo al administrador.
- registration: contiene las implementaciones mediante las cuales Gserver registra la información de los agentes que están activos, su dirección IP, puerto de escucha, identificador del medio básico, etc.
- djangorestframework: contiene funcionalidades que se reutilizan en los demás componentes haciendo posible el uso de servicios web de tipo REST con Django.
- core: contiene las implementaciones sobre las cuales está implementado el núcleo del servidor.

3.5. Diagrama de Despliegue

Indica la situación física de los componentes lógicos desarrollados. Es decir se sitúa el software en el hardware que lo contiene. Cada Hardware se representa como un nodo. Un nodo se representa como un cubo, un nodo es un elemento donde se ejecutan los componentes, representan el despliegue físico de estos componentes que representan algún tipo de recurso computacional (capacidad de memoria y procesamiento) (40).

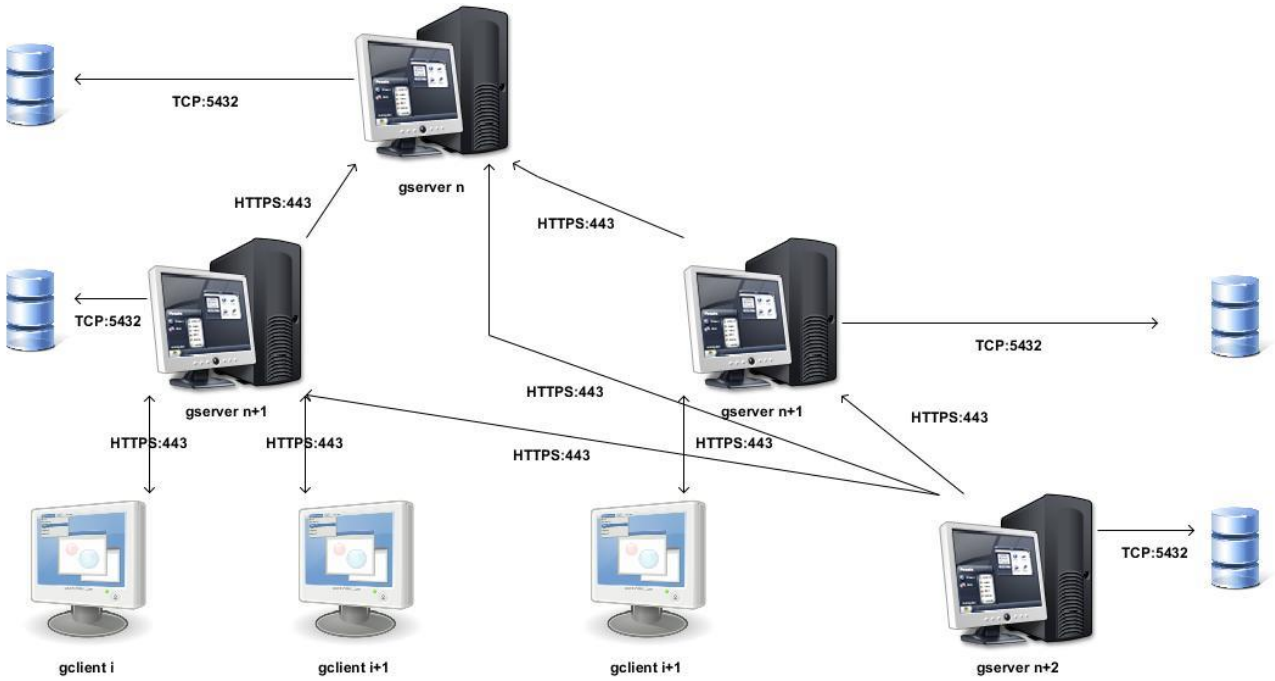


Figura 23: Diagrama de Despliegue

El sistema trabajará con varios servidores conectados de forma de grafo dirigido no cíclico relacionados con sus bases de datos correspondientes, realizándose la comunicación con la base de datos a través del protocolo TCP por el puerto 5432, teniendo a su vez varios clientes conectados a ellos. La comunicación se establecerá haciendo uso del protocolo HTTPS por el puerto 443 esto será entre servidores y servidor cliente.

3.6. Pruebas de Software

La prueba es un proceso de ejecución de un programa con la intención de descubrir errores, donde un buen caso de prueba es aquel que tiene una alta probabilidad de mostrar un error no descubierto hasta entonces esta prueba tiene éxito si descubre un error no detectado hasta entonces. El objetivo de la prueba es diseñar pruebas que saquen a la luz diferentes clases de errores con la menor cantidad de tiempo y espacio. La prueba no puede asegurar la ausencia de defectos; solo pueden demostrar que existen defectos en el software. El flujo de pruebas le presta servicios a los demás flujos. Su principal objetivo es evaluar o valorar la calidad del producto a través de buscar y documentar errores, validar el cumplimiento de requerimientos, validar el desempeño y dar una indicación de calidad (41).

3.6.1. Métodos de Pruebas: Caja Negra

Pruebas de Caja Negra:

Son las pruebas que se llevan a cabo sobre la interfaz del software, por lo que los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce una salida correcta, así como que la integridad de la información externa se mantiene. Esta prueba examina algunos aspectos del modelo fundamentalmente del sistema sin tener mucho en cuenta la estructura interna del software. Tiene como objetivo demostrar que las funciones del software son operativas, que las entradas se aceptan de forma adecuada y se produce un resultado correcto y que la integridad de la información externa se mantiene (no se ve el código) (41).

3.6.1.1. Técnicas de Prueba

Técnica de la Partición de Equivalencia: esta técnica divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software (42).

- Permite examinar los valores válidos e inválidos de las entradas existentes en el software.

- Descubre de forma inmediata una clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico.

Técnica del Análisis de Valores Límites: esta Técnica prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables (42).

Técnica de Grafos de Causa-Efecto: es una técnica que permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones (42).

Los autores de la presente investigación deciden realizar la Técnica de la Partición de Equivalencia a las funcionalidades del software implementadas.

3.7. Resultado de las Pruebas

Se realizaron las técnicas pertinentes, del método de caja negra arrojando resultados satisfactorios. A continuación se muestra la tabla referente al caso de uso Gestionar Configuración.

Caso de uso Gestionar Configuración.

Descripción general:

El sistema debe permitir gestionar la configuración de los servidores, es decir, adicionar, modificar y eliminar un servidor.

Condiciones de Ejecución:

El administrador debe estar autenticado.

1. Secciones a probar en el Caso de Uso:

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo central
SC1: Adicionar Servidor	EC 1.1 Adicionar servidor correctamente	El sistema debe mostrar una interfaz que posibilita insertar los	1. Se selecciona la opción Adicionar. 2. Se muestra la interfaz Adicionar con los campos a insertar (Zona, Servidor, Puerto, Https y Reintento). 3. Se insertan los datos solicitados. 4. Se oprime el botón crear.

		datos de un servidor.	5. El sistema comprueba que los datos sean correctos, y muestra el mensaje “La operación se realizó con éxito”.
	EC 1.2: Adicionar Servidor dejando campos vacíos.	El sistema debe mostrar un mensaje informando que existen campos vacíos.	1. Se selecciona la opción Adicionar. 2. Se muestra la interfaz Adicionar con los campos a insertar (Zona, Servidor, Puerto, Https y Reintento). 3. Se deja uno o más campos vacíos. 4. Se oprime el botón Aceptar. 5. El sistema marca los campos vacíos.
	EC 1.2: Adicionar Servidor con datos incorrectos.	El sistema debe mostrar un mensaje indicando que los valores insertados no son correctos.	1. Se selecciona la opción Adicionar. 2. Se muestra la interfaz Adicionar con los campos a insertar (Zona, Servidor, Puerto, Https y Reintento). 3. Se insertan uno o más campos con valores incorrectos. 4. Se oprime el botón Aceptar. 5. El sistema muestra un mensaje indicando que existen campos con valores incorrectos.
SC 2: Modificar Servidor	EC 2.1: Modificar Servidor correctamente.	El sistema debe mostrar una interfaz que posibilita	1. Se selecciona la opción Modificar.

		actualizar los datos de un servidor.	2. Se muestra la interfaz Modificar con los campos a actualizar (Zona, Servidor, Puerto, Https y Reintento). 3. Se insertan los datos solicitados. 4. Se oprime el botón Aceptar. 5. El sistema comprueba que los datos sean correctos, y muestra el mensaje “La operación se realizó con éxito”.
	EC 2.2: Modificar Servidor dejando campos vacíos.	El sistema debe mostrar un mensaje informando que existen campos vacíos.	1. Se selecciona la opción Modificar. 2. Se muestra la interfaz Modificar con los campos a actualizar (Zona, Servidor, Puerto, Https y Reintento). 3. Se deja uno o más campos vacíos. 4. Se oprime el botón Aceptar. 5. El sistema marca los campos vacíos.
	EC 2.3 Modificar Servidor insertando campos con valores incorrectos.	El sistema debe mostrar un mensaje indicando que los valores insertados no son correctos.	1. Se selecciona la opción Modificar. 2. Se muestra la interfaz Modificar con los campos a actualizar (Zona, Servidor, Puerto, Https y Reintento). 3. Se insertan uno o más campos con valores incorrectos. 4. Se oprime el botón Aceptar.

			5. El sistema muestra un mensaje indicando que existen campos con valores incorrectos.
SC 3: Eliminar Servidor	EC 3.1 Eliminar Servidor	El sistema mostrará un mensaje de confirmación informando que el indicador ha sido eliminado correctamente.	1. Se selecciona la opción Eliminar. 2. Se oprime el botón Aceptar. 3. El sistema elimina el servidor y muestra un mensaje de confirmación, el Servidor ha sido eliminado correctamente.

Id ES	Escenario	Zona	Servidor	Puerto	Https	Reintento	Respuesta del sistema	Resultado de la prueba
EC 1.1	Adicionar Servidor correctamente	(V) 1	(V) 10.8.3.20	(V) 8000	(V) False	(V) 1	El sistema muestra un mensaje: “La operación se realizó con éxito”.	Se obtuvo el resultado esperado en dicha prueba.
EC 1.2	Adicionar Servidor dejando	(I) “”	(V) 10.8.5.20	(V) 8001	(V) False	(V) 0.5	El sistema indica el campo vacío.	Se obtuvo el resultado esperado en dicha prueba.

	campos vacíos.	(V) 1	(I) “”	(V) 8001	(V) False	(V) 0.5	El sistema indica el campo vacío.	Se obtuvo el resultado esperado en dicha prueba.
		(V) 1	(V) 10.8.3.20	(I) “”	(V) False	(V) 1	El sistema indica el campo vacío.	Se obtuvo el resultado esperado en dicha prueba.
		(V) 1	(V) 10.8.3.20	(V) 8000	(I) “”	(V) 1	El sistema indica que el campo Https debe ser selecciona do	Se obtuvo el resultado esperado en dicha prueba.
		(V) 1	(V) 10.8.3.20	(V) 8000	(V) False	(I) “”	El sistema indica el campo vacío.	Se obtuvo el resultado esperado en dicha prueba.
EC 1.3	Adicionar campos con valores incorrecto s	(I) -1	(V) 10.8.3.20	(V) 8000	(V) False	(V) 1.2	El sistema indica que el campo debe ser un número positivo entero	Se obtuvo el resultado esperado en dicha prueba.
		(V)	(I)	(V)	(V)	(V)	El sistema indica que	Se obtuvo el resultado

		1	1.8.2.1.3	8000	False	1	el campo no tiene la estructura correspondiente.	esperado en dicha prueba.
		(V) 1	(V) 10.8.3.20	(V) -	(V) False	(V) 1	El sistema indica que el campo debe ser un número positivo entero	Se obtuvo el resultado esperado en dicha prueba.
		(V) 1	(V) 10.8.3.20	(V) 8000	(I) “”	(V) 1	El sistema indica que el campo debe ser seleccionado.	Se obtuvo el resultado esperado en dicha prueba.
		(V) 1	(V) 10.8.3.20	(V) 8000	(V) True	(I) -1	El sistema indica que el campo debe ser un número real positivo.	Se obtuvo el resultado esperado en dicha prueba.
EC 2.1	Modificar Servidor	(V) 1	(V) 10.8.3.21	(V) 8000	(V) False	(V) 1	El sistema muestra un	Se obtuvo el resultado

	correctamente						mensaje: “La operación se realizó con éxito”.	esperado en dicha prueba.
EC 2.2	Modificar Servidor dejando campos vacíos.	(I) “”	(V) 10.8.5.20	(V) 8001	(V) False	(V) 0.5	El sistema indica el campo vacío.	Se obtuvo el resultado esperado en dicha prueba.
		(V) 1	(I) “”	(V) 8001	(V) False	(V) 0.5	El sistema indica el campo vacío.	Se obtuvo el resultado esperado en dicha prueba.
		(V) 1	(V) 10.8.3.20	(I) “”	(V) False	(V) 1	El sistema indica el campo vacío.	Se obtuvo el resultado esperado en dicha prueba.
		(V) 1	(V) 10.8.3.20	(V) 8000	(I) “”	(V) 1	El sistema indica que el campo Https debe ser	Se obtuvo el resultado esperado en dicha prueba.

							selecciona do	
		(V) 1	(V) 10.8.3.20	(V) 8000	(V) False	(I) “”	El sistema indica el campo vacío.	Se obtuvo el resultado esperado en dicha prueba.
EC 2.3	Modificar campos con valores incorrecto s	(I) -1	(V) 10.8.3.20	(V) 8000	(V) False	(V) 1.2	El sistema indica que el campo debe ser un número positivo entero	Se obtuvo el resultado esperado en dicha prueba.
		(V) 1	(I) 1.8.2.1.3	(V) 8000	(V) False	(V) 1	El sistema indica que el campo no tiene la estructura correspon diente.	Se obtuvo el resultado esperado en dicha prueba.
		(V) 1	(V) 10.8.3.20	(V) -8000	(V) False	(V) 1	El sistema indica que el campo debe ser un número positivo entero	Se obtuvo el resultado esperado en dicha prueba.

		(V) 1	(V) 10.8.3.20	(V) 8000	(I) “”	(V) 1	El sistema indica que el campo debe ser selecciona do.	Se obtuvo el resultado esperado en dicha prueba.
		(V) 1	(V) 10.8.3.20	(V) 8000	(V) True	(I) -1	El sistema indica que el campo debe ser un número real positivo.	Se obtuvo el resultado esperado en dicha prueba.

3.8. Conclusiones parciales

En el presente capítulo se definió la arquitectura en la que se basa la propuesta de solución lo cual permitió implementar el módulo de manera escalable y extensible. Además se identificaron los patrones de diseños utilizados haciendo robusta la solución planteada. Se generaron los artefactos correspondientes a la metodología seleccionada obteniendo más documentación para el proyecto al que pertenece la solución. Se realizaron además las pruebas evidenciando la realización de un sistema implementado actualmente sin errores.

CONCLUSIONES

La presente investigación permitió obtener una solución a las dificultades presentes en la estructura jerárquica del despliegue de servidores GRHS, por lo que se concluye que:

- El estudio de sistemas que permiten el despliegue distribuido de información no se pueden integrar al sistema GRHS. Como estructura que posibilita el envío de información por varios caminos se selecciona grafo dirigido no cíclico.
- El estudio de las herramientas y tecnologías permitieron el desarrollo de un sistema robusto, escalable y flexible. La utilización de la metodología de desarrollo permitió una mejor comprensión del mismo generando la documentación necesaria para el proyecto.
- Se realizaron las pruebas funcionales, asegurando de esta forma que el módulo implementado no presenta errores.

RECOMENDACIONES

- Elaborar una funcionalidad que permita mostrar el estado de los servidores (encendido y apagado).
- Optimizar las rutas de conexión entre servidores, teniendo en cuenta la carga de peticiones.

REFERENCIAS BIBLIOGRÁFICAS

1. **Informática Moderna. Informática Moderna. [En línea] [Citado el: 6 de noviembre de 2014.] http://www.informaticamoderna.com/Servidor.htm..**

2. Definicion de. [En línea] [Citado el: 6 de noviembre de 2014.] http://www.definicion.de/servidor/..

3. Wiki GRHS. [En línea] [Citado el: 27 de abril de 2015.] http://10.128.50.236/grhs-doc/index.php/Modelos.

4. Wiki GRHS. Wiki GRHS. [En línea] [Citado el: 24 de abril de 2015.] http://10.128.50.236/grhs-doc/index.php/Conceptos.

5. GestioPolis. [En línea] [Citado el: 7 de noviembre de 2014.] http://www.gestiopolis.com/economia/metodos-y-tecnicas-de-investigacion.htm..

6. Scribd. [En línea] [Citado el: 7 de noviembre de 2014.] http://es.scribd.com/doc/21229743/METODOS-EMPIRICOS.

7. Definion ABC. [En línea] [Citado el: 12 de noviembre de 2014.] http://www.definicionabc.com/general/desplegar.php..

8. Word Reference. [En línea] [Citado el: 12 de noviembre de 2014.] http://www.wordreference.com/es/en/frames.aspx?es=distribuido..

9. Microsoft System Center. Microsoft System Center. [En línea] [Citado el: 15 de diciembre de 2014.] https://technet.microsoft.com/es-es/library/ff399610.aspx.

10. Academia. [En línea] [Citado el: 13 de diciembre de 2014.] http://www.academia.edu/3590242/HYDRA_plataforma_para_despliegue_y_administraci%C3%B3n_remota_de_sistemas_heterog%C3%A9neos_en_red.

11. NI VeriStand. [En línea] [Citado el: 14 de noviembre de 2014.] http://www.ni.com/white-paper/11060/es/.

12. Prophet. [En línea] [Citado el: 17 de noviembre de 2014.] http://www.networks.imdea.org/es/investigacion/proyectos/prophet.

13. Prophet2. [En línea] 2013. [Citado el: 17 de noviembre de 2014.] http://www.networks.imdea.org/es/actualidad/noticias/2013/prophet-un-proyecto-aumentar-fiabilidad-redes-nuestra-sociedad.

14. P2P. [En línea] http://aprenderinternet.about.com/od/Multimedia/a/Que-Son-Los-Programas-P2p-Y-Como-Funcionan.htm..

15. Harbour, Michael González. grafos. [En línea] 4 de noviembre de 2009. [Citado el: 20 de noviembre de 2014.] http://www.ctr.unican.es/asignaturas/eda/cap4-grafos-2en1.pdf.

16. Cantador, Iván, Acuña, Silvia Teresita y Donrroso, José R. estructura de grafos. [En línea] [Citado el: 21 de noviembre de 2014.] http://ir.ii.uam.es/ivan/eda/eda-tema05-4.pdf.

17. Aprende Redes. [En línea] [Citado el: 28 de abril de 2015.] http://aprenderedes.com/2006/11/protocolo-de-arbol-de-extensionstp/.

18. Cisco Redes. [En línea] [Citado el: 28 de abril de 2015.] http://ciscoccna3.blogspot.com/2012/07/stp-spanning-tree-protocol.html.

19. Material Curso de Comunicacion de Datos. [En línea] [Citado el: 28 de abril de 2015.] https://sites.google.com/site/uccomunicaciondedatos1g1/home/material-curso-comunicacion-de-datos-i/contenido-del-curso/capitulo-4-principios-de-conmutacion-y-enrutamiento.

20. Pérez, Isaías Carrillo. Metodología de Desarrollo de Software.

21. Figuerola, Norberto. BPMN vs UML. Buenos Aires : s.n., 2014.

22. Sommerville, Ian. Ingeniería de software.

23. Herramientas Case. [En línea] 2 de Abril de 2011. [Citado el: 25 de noviembre de 2014.] http://fds-herramientascase.blogspot.com/..

24. León, Eduardo. visual paradigm. [En línea] [Citado el: 28 de noviembre de 2014.] http://www.visual-paradigm.com/.

25. Duque, Raúl Gonzáles. Python para todos. España : s.n.

26. The Django Book. [En línea] 2014. [Citado el: 1 de diciembre de 2014.] http://www.djangobook.com/..

27. Curso sobre. [En línea] [Citado el: 12 de diciembre de 2014.] http://curso-sobre.berlios.de/introsobre/2.0.1/sobre.html/eclipse.html..

28. Appcelerator. [En línea] [Citado el: 12 de diciembre de 2014.] http://pydev.org/..

29. Martín, Ramos. McGraw-Hill Interamericana de España. [En línea] [Citado el: 2 de diciembre de 2014.] http://www.mcgraw-hill.es/bcv/guide/capitulo/8448148797.pdf.

30. Martinez, Rafael.. PostgreSQL-es. [En línea] 2 de 10 de 2010. [Citado el: 2 de diciembre de 2014.] http://www.postgresql.org.es/sobre_postgresql..

31. Ingeniería de Requisitos: conceptos. Madrid : s.n., 2005.

32. Quiroga, Juan Pablo. [En línea] [Citado el: 24 de febrero de 2015.] https://sistemas.uniandes.edu.co/~csof5101/dokuwiki/lib/exe/fetch.php?media=principal:csof5101-requerimientos.pdf.

33. González., Dra. Anaisa Hernández. Scribd. [En línea] [Citado el: 15 de diciembre de 2014.] http://es.scribd.com/doc/13500172/actividad2-diagrama-de-casos-de-uso-del-negocio-y-del-sistema..

34. Olivares, Freddy Egdamar Paez. Diseño UML. [En línea] 22 de mayo de 2009. [Citado el: 15 de mayo de 2015.] http://egdamar877.blogspot.com/2009/05/expocicion.html.

35. isoft 1. [En línea] [Citado el: 15 de mayo de 2015.] http://www-2.dc.uba.ar/materias/isoft1/is1-2005_2/apuntes/SlidesDC.pdf.

36. Office. [En línea] [Citado el: 12 de enero de 2015.] https://support.office.com/es-mx/article/Conceptos-b%C3%A1sicos-del-dise%C3%B1o-de-una-base-de-datos-1eade2bf-e3a0-41b5-aee6-d2331f158280?ui=es-ES&rs=es-MX&ad=MX.

37. Django en español. [En línea] [Citado el: 1 de diciembre de 2014.] http://django.es/.

38. Django. [En línea] [Citado el: 27 de mayo de 2015.] http://django-book.mkaufmann.com.ar/chapter15.html.

39. arqu. [En línea] [Citado el: 12 de marzo de 2015.] 1. http://www.sei.cmu.edu/architecture/definitions.html.

40. Orozco, Adrián Fernández. EVA. Diagramas. [En línea] [Citado el: 20 de mayo de 2015.] http://eva.uci.cu.

41. Pressman, Roger S. Ingeniería del Software.Un enfoque Práctico. 5ta edición . Madrid : McGraw-Hill/ Interamericana de España. S.A.U., 2002.

42. EVA. [En línea] [Citado el: 22 de mayo de 2015.] http://eva.uci.cu/file.php/158/Documentos/Recursos_bibliograficos/Libros_y_articulos_UD_2/Comun/Material_de_caja_b_y_caja_n.pdf.

43. IBM Knowledge Center. [En línea] [Citado el: 16 de noviembre de 2014.] www.ibm.com.

44. Informatica Hoy. [En línea] [Citado el: 18 de noviembre de 2014.] http://www.informatica-hoy.com.ar/aprender-informatica/Que-es-DMZ.php.

45. nc microsof. [En línea] [Citado el: 22 de noviembre de 2014.] http://msdn.microsoft.com/es-es/architecture/default.aspx.

46. micr. [En línea] [Citado el: 22 de noviembre de 2014.] https://msdn.microsoft.com/es-es/library/bb384398.aspx.

47. RUP2. [En línea] [Citado el: 24 de noviembre de 2014.] http://eumed.net.

48. Ortega, Raúl Jiménez.. Seminario: Python+Django. . 2013.

49. researshgate. [En línea] [Citado el: 3 de abril de 2015.] http://www.researchgate.net/publication/265126450_Pster.

50. Definicion de. [En línea] [Citado el: 11 de febrero de 2015.] http://www.definicion.de/grafos/.

51. P.Letelier. [En línea] [Citado el: 20 de mayo de 2015.] https://pid.dsic.upv.es.

52. EVA. [En línea] [Citado el: 22 de mayo de 2015.] http://eva.uci.cu/file.php/158/Documentos/Recursos_didacticos/Presentaciones_digitales_UD_2/Procedimiento_para_pruebas_de_caja_negra.ppt.

ANEXOS

Anexo 1

Caso de uso Mostrar Conexiones.

Descripción general: El sistema debe permitir mostrar las conexiones.

Condiciones de Ejecución:

El sistema debe estar funcionando.

1. Secciones a probar en el Caso de Uso:

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo central
SC1: Mostrar Conexiones	EC 1.1 Mostrar Conexiones	El sistema debe mostrar una interfaz con un gráfico en el que se muestran las conexiones entre los servidores.	1. Se selecciona la opción Ver Conexiones 2. Se muestra la interfaz con el grafico de las conexiones entre servidores configurados

Anexo 2

Id ES	Escenario	Respuesta del sistema	Resultado de la prueba

EC 1.1	Mostrar Conexiones	El sistema muestra el grafico de las conexiones entre servidores.	Se obtuvo el resultado esperado en dicha prueba.

GLOSARIO DE TÉRMINOS

Despliegue: es la acción de desdoblar, extender aquello que se encuentra plegado o cerrado. En la informática, se hace referencia a la fase final del desarrollo de software en la que se empaqueta el producto para su posterior distribución y puesta en funcionamiento en su entorno de ejecución final.

Despliegue: a instancias de la Informática se conoce como lista de a aquel programa o rutina compuesto por una serie de instrucciones para gráficos y que es ejecutada mediante el procesador; las instrucciones definirán una determinada imagen de salida. Dicha imagen será interpretada a partir de la ejecución de las instrucciones de la lista de despliegue.

Distribuido: es la acción de dividir algo entre varios elementos, designando lo que a cada uno corresponde, según voluntad, conveniencia, regla o derecho. Dar a algo su oportuna colocación o el destino conveniente.

Despliegue distribuido: es el acto de desdoblar un conjunto de elementos entidades o programas desde el punto de vista informático que están relacionados entre sí empleando una estructura de grafo, donde el nodo principal es el servidor central y cualquiera de los demás nodos puede ser el origen del grafo y cada nodo puede generar un camino diferente para llevar la información necesaria a su destino. Posibilitando que la información llegue por diferentes caminos sin tener en cuenta los niveles de jerarquía que puedan existir en una determinada entidad.

Grafo: se puede definir como un objeto unitario de naturaleza abstracta que abarca a las grafías que componen una letra. La palabra tiene origen griego y significa “imagen” o “dibujo”.

Grafo: para las ciencias de la computación y la matemática, es una representación gráfica de diversos puntos que se conocen como nodos o vértices, los cuales se encuentran unidos a través de líneas que reciben el nombre de aristas.

Réplica: es una copia completa de los datos, la cual posee todas las propiedades del mismo.

Información: está constituida por un grupo de datos ya supervisados y ordenados, que sirven para construir un mensaje.

Réplica de información: no es más que la copia completa de los datos ya supervisados y ordenados para construir un mensaje.