Universidad de las Ciencias Informáticas

Facultad 1

COMPONENTE PARA LA GESTIÓN DE USUARIOS PARA EL SISTEMA XILEMA SMART KEEPER 3.0

Autor:

Nardy Román Padrón

Tutores:

Ing. Yordanka Fuentes Castillo

Ing. Miguel Angel Chávez Alfonso

Declaración de Autoría

Declaro por este medio que yo Nardy Román Padrón, con carnet de identidad 94021904526, soy el único autor del trabajo titulado **"Componente para la gestión de usuarios para el sistema Xilema Smart Keeper"** y autorizo a la Universidad de las Ciencias Informáticas a hacer uso de la misma en su beneficio, así como los derechos patrimoniales de la misma con carácter exclusivo.

Declaro que todo lo anteriormente expuesto se ajusta a la verdad, y asumo la responsabilidad moral y jurídica que se derive de este juramento profesional.

Y para que así conste, firmo la presente declaración de autoría en La Habana a los ___ días del mes de _____ del año 2018.

Autor:

_____

Nardy Román Padrón

Tutores:

_____ _____

Ing. Yordanka Fuentes Castillo Ing. Miguel Angel Chávez Alfonzo

Dedicatoria

A mi abuela, a mi mamá y a mi papá por ser ese pequeño motor impulsor que me han dado la fuerza y el espíritu para seguir adelante. A toda mi familia que siempre ha confiado en mí. A mis primitos que esto le sirva de ejemplo a seguir.

Agradecimientos

A mi familia por apoyarme siempre.

A mis tutores por ser incondicionales conmigo.

A mis verdaderos amigos de adentro y de afuera de la universidad que vivieron buenos y malos momentos conmigo.

A toda persona que de una forma u otra se interesó por mí y me dio un buen consejo para seguir adelante.

Resumen

Debido a las limitaciones de acceso a Internet que posee nuestro país, resulta necesario establecer cuota por usuarios o grupos en las empresas e instituciones que proveen servicios de internet a sus usuarios, con el objetivo de regular dicho acceso. La configuración de cuotas de navegación y la gestión dinámica de usuarios y grupos de navegación eran unas de las funcionalidades más solicitadas por los clientes que desplegaron las antiguas versiones de Xilema Smart Keeper en sus empresas e instituciones; sin embargo, debido a la utilización de funciones en desuso y la incompatibilidad de su framework base con las actuales librerías de php, estas y otras funcionalidades asociadas a la gestión de usuarios han quedado obsoletas. En la presente investigación se desarrolla un componente para la gestión de usuarios, que permite: gestionar usuarios, grupos de navegación y configurarles cuotas de navegación de forma dinámica e importar usuarios de un directorio activo, basadas en funcionalidades desarrolladas sobre las versiones recientes del sistema operativo Ubuntu.

Palabras claves: gestión de usuarios, filtro de contenido web, cuotas de navegación

Índice

Índice de ilustraciones

Índice de tablas

Introducción

En el mundo actual el desarrollo de las tecnologías ha propiciado un intercambio de grandes volúmenes de información a través de la web, la cual se encuentra formada entre otros por servicios y aplicaciones que posibilitan la creación de grupos, comunidades o redes sociales. Las personas que usan y/o trabajan con estos servicios y aplicaciones se les denominan usuarios (Martín, 2014) y al acceder a estos recursos en la web, los usuarios deben dar pruebas de quienes son para así verificar si pueden acceder a determinado tipo de información. Normalmente antes de trabajar en una aplicación web es necesario iniciar una sesión, momento en el que la persona que quiere acceder a dicha aplicación se identifica como uno de los usuarios existentes (Castellar, 2018). Todas estas funciones antes mencionadas y añadiéndole las operaciones básicas de: crear, actualizar, eliminar y mostrar la información con respecto al usuario constituyen la gestión de usuarios. (Significados, 2018). La gestión de usuarios es, hoy en día, un área de entidad propia, en la medida en que la relación de los usuarios con las aplicaciones web es crucial cuando se trata de asegurar un correcto funcionamiento de los servicios puestos a su disposición.

El Departamento Soluciones Informáticas para Internet (SINI) del Centro de Ideoinformática (CIDI) de la Universidad de las Ciencias Informáticas (UCI) tiene entre sus proyectos el desarrollo del sistema Xilema Smart Keeper. El mismo cumple la función de filtro de contenido web orientado a servidores para medianas redes de usuario como las disponibles generalmente en colegios, empresas y bibliotecas. Este sistema es flexible a la política de uso de Internet del lugar donde se aplique y cuenta con una interfaz web fácil e intuitiva para la administración. Mediante la definición de políticas y grupos, posibilita establecer diversos permisos de navegación. Smart Keeper utiliza el servidor proxy Squid para el control de acceso sobre determinados recursos en Internet.

El sistema Xilema Smart Keeper en su versión 2.4 fue desarrollado para hospedarse en servidores con sistema operativo Ubuntu 12.04, el cual fue liberado el 26 de abril del 2012 (Usuario, 2014). Hasta la fecha han sido liberadas nuevas versiones de dicho sistema operativo; llegando hasta la versión 18.04 (ubuntu, 2018), lo que conllevó a que Ubuntu 12.04 perdiera el soporte a partir del 28 de abril del 2017 (Pomeyrol, 2017). Las pruebas de ejecución de Xilema Smart Keeper 2.4 en un sistema operativo superior a Ubuntu 12.04 resultaron fallidas, identificándose incompatibilidades de los sistemas operativos recientes con las librerías y *frameworks* utilizados.

Los clientes que actualmente utilizan Xilema Smart Keeper en su versión 2.4 no pueden actualizar el sistema operativo de sus servidores, ni pueden solicitar nuevas actualizaciones del sistema, lo que puede traer consigo problemas graves de seguridad. Empresas e instituciones cubanas que necesitan controlar el acceso a Internet de sus usuarios no solicitan la instalación de Xilema Smart Keeper 2.4 debido a su gran desactualización, lo que constituye una pérdida de mercado potencial. Debido a estos elementos, el departamento SINI se planteó la tarea de desarrollar la nueva versión 3.0 del sistema Xilema Smart Keeper empleando para su desarrollo tecnologías actuales, garantizando su compatibilidad con las versiones más recientes del sistema operativo Ubuntu.

Al desarrollarse la nueva versión de Smart Keeper ya no será una solución viable el uso del componente de gestión de usuarios de la versión anterior, debido principalmente al uso continuo de funciones en desuso y a la incompatibilidad de su framework base con las actuales librerías de php. Esto trae consigo que el proceso de gestión de usuarios se deba realizar a partir de una configuración manual del fichero principal de configuración de Squid. El proceso de importado de usuarios para la autenticación mediante un servicio de directorio activo LDAP, y las configuraciones de reglas de acceso de los mismos también se debe realizar de forma manual, mediante la utilización de otros sistemas que permitan buscar y listar estos usuarios, y la utilización de archivos en texto plano.

Para realizar una correcta gestión de usuarios en el sistema Xilema Smart Keeper 3.0 un administrador de red necesita conocimientos avanzados sobre configuraciones de Squid y no cuenta con una interfaz interactiva e intuitiva que le permita de forma remota mediante la asistencia de una aplicación web, realizar el proceso de configuración.

La configuración de cuotas de navegación y la gestión dinámica de grupos de navegación eran unas de las funcionalidades más solicitadas por los clientes que desplegaron las antiguas versiones de Xilema Smart Keeper en sus empresas e instituciones, ya que debido a las limitaciones de acceso que posee nuestro país, resulta necesario establecer mecanismos de cuota por usuarios o grupos con el objetivo de regular dicho acceso. Esta característica no la posee el servidor proxy Squid por defecto, por lo que resulta imposible su utilización en la versión actual de Xilema Smart Keeper.

Por lo anteriormente expuesto, se plantea el siguiente problema de investigación: ¿Cómo gestionar los usuarios para la navegación a través de la web en el sistema Xilema Smart Keeper?; definiéndose como **objeto de estudio** la gestión de usuarios, enmarcado en el **campo de acción** la gestión de usuarios para

sistemas de filtrado de contenido web. A partir del problema planteado se propone como **objetivo general** desarrollar un componente para la gestión de usuarios para el sistema Xilema Smart Keeper.

Para darle cumplimiento al mismo se definen los siguientes **objetivos específicos:**

- Construir los referentes teóricos fundamentales que sustentan la investigación relacionados con la gestión de usuarios y los sistemas de filtrado de contenido web.
- Realizar el diseño del componente para la gestión de usuarios para el sistema Xilema Smart Keeper.
- Implementar las funcionalidades del componente para la gestión de usuarios para el sistema Xilema Smart Keeper.
- Validar el componente para la gestión de usuarios y su integración con los restantes módulos existentes del sistema Xilema Smart Keeper.

Para guiar el desarrollo de la solución se plantea como **idea a defender**: el desarrollo de un componente para la gestión de usuarios; permitirá gestionar los usuarios para el filtro de contenido web en el sistema Xilema Smart Keeper.

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

- Estudio de los conceptos asociados al marco teórico de la investigación.
- Caracterización de aplicaciones que permitan gestionar y configurar usuarios.
- Selección de las herramientas, tecnologías y metodología de desarrollo a emplear para la construcción de la solución propuesta.
- Identificación de los requisitos funcionales y no funcionales de la solución.
- Diseño de los requisitos funcionales de la solución.
- Implementación de los requisitos del componente para la gestión de usuarios para el sistema Xilema Smart Keeper.
- Integración del componente desarrollado al sistema Xilema Smart Keeper.
- Validación de la solución aplicando para ello pruebas funcionales, de seguridad, y de integración.

Métodos de investigación científica.

Para el desarrollo del trabajo se debe hacer uso correcto de los métodos investigativos, que permitirán obtener información relacionada al objeto de estudio, para alcanzar este objetivo se propone, la utilización de los métodos de investigación científica que se describen a continuación:

Métodos teóricos:

Permiten tener un conocimiento general sobre el estado actual del tema escogido, su evolución y conceptos generales. Se plantea utilizar los siguientes métodos:

- Histórico-Lógico: Con la utilización de este método se realizará un estudio de las diversas tecnologías que existen actualmente para realizar la selección de las que se van a utilizar de acuerdo a las características propias del sistema a desarrollar.
- Analítico-Sintético: Este método permitirá analizar las teorías y los documentos referentes al objetivo de la investigación, facilitando de esta forma la extracción de los elementos más importantes relacionados con el objeto de estudio. Además, posibilitará construir el camino a seguir, a partir del análisis detallado de cada uno de los documentos previamente mencionados.
- Modelación: Este método permitirá crear abstracciones del objeto de investigación, a través de las que se realizarán los modelos que servirán de base para cada una de las fases del desarrollo de la solución propuesta, tales como el mapa de procesos, los diagramas de procesos del negocio, el modelo del dominio y otros.

Métodos empíricos:

Para enriquecer la investigación se propone utilizar métodos empíricos, que describen y explican las características fenomenológicas del objeto, representan un nivel de la investigación cuyo contenido procede de la experiencia y es sometido a cierta elaboración racional.

- Observación: Con el estudio del mismo se podrá conocer la realidad mediante la percepción directa de los objetos y fenómenos; a través de este método se puede saber la esencia de la problemática definida, lo que ayuda al planteamiento del problema científico, además de permitir conocer el proceso definido como objeto de estudio, lo cual influye a la hora de tener un conocimiento más detallado de lo que se quiere, lo que hace falta hacer y cómo hay que hacerlo.

- Entrevista: La aplicación de este método permitirá identificar los requisitos funcionales y no funcionales que debe cumplir el sistema, estimados por el usuario.

El Trabajo de Diploma consta de introducción, tres capítulos, conclusiones, recomendaciones, referencias bibliográficas, bibliografía y anexos que ampliarán la información que se aporta en la investigación. A continuación, se describe brevemente la estructura del mismo, sintetizando el contenido de cada capítulo.

Capítulo 1. Fundamentación teórica de la investigación.

En el presente capítulo se declara la base conceptual y se tratan los aspectos teóricos que dan sustento a la investigación, tales como: conceptos asociados al dominio del problema, sistemas homólogos vinculados al campo de acción enunciado y selección de las herramientas, tecnologías y metodologías de desarrollo adecuadas a la propuesta de solución.

Capítulo 2. Diseño del componente para la gestión de usuarios para el sistema Xilema Smart Keeper.

En este capítulo se presenta una descripción detallada de los requisitos funcionales y no funcionales del sistema, propiciando una modelación adecuada para la propuesta realizada, determinando así las funcionalidades y características que se conciben para el sistema. Se especifica el patrón arquitectónico y los patrones de diseño que se utilizan y se presenta la propuesta de solución a partir de los artefactos generados como parte de la metodología seleccionada.

Capítulo 3. Implementación y validación del componente para la gestión de usuarios para el sistema Xilema Smart Keeper.

En este capítulo se describe todo el proceso de construcción del sistema y las pruebas realizadas a las funcionalidades del componente en un entorno real, mostrando los resultados obtenidos para la valoración de la solución final.

Al concluir la investigación se espera obtener un componente que permita a los administradores del sistema Xilema Smart Keeper gestionar los usuarios que utilizan el sistema.

Capítulo 1: Fundamentación teórica de la investigación.

En el presente capítulo se abordan temas relacionados con los servidores proxy Squid, el filtro de contenido Dansguardian y la gestión de usuarios. Se realiza un estudio y análisis de los sistemas de filtrado de contenido web, teniendo en cuenta la gestión de usuarios; se analiza de estos sus características y procesos, así como ventajas y desventajas que puedan presentar. Se realiza una investigación de las herramientas, lenguajes, tecnologías y metodologías que se emplearán en el desarrollo del componente para darle solución al problema.

1.1 Servidor proxy Squid

Squid es un proxy de almacenamiento en caché para la web, esto significa que almacena las páginas web en el disco duro del servidor para cuando se vuelva a visitar, que esté almacenada en la cache, la página se leerá del servidor y en vez de acceder de Internet. Squid funciona por defecto por el puerto 3128, aunque se puede configurar para que funcione en otro puerto. Para que pueda controlar el acceso a Internet necesita que el tráfico web sea redireccionado al puerto asignado. Como las peticiones hacia Internet de los equipos de la red local son interceptadas por el servidor proxy este puede ser una tarea de filtrado de acceso, impidiendo aquellos destinos que estén expresamente prohibidos en los archivos de configuración del servicio. Las principales funciones de Squid son: permite el acceso web a máquinas privadas que no están conectadas directamente a Internet; controla el acceso red aplicando reglas; registra el tráfico web desde la red local hacia el exterior; controla el contenido web visitado y descargado; controla la seguridad de la red local ante posibles ataques e intrusiones en el sistema; funciona como una cache de páginas web; guarda en cache las peticiones DNS e implementa una cache para las conexiones fallidas; registra logs (registros) de todas las peticiones fallidas; soporta el protocolo ICP (Protocolo Cache Internet) que permite integrar cachés que colaboran y permiten crear jerarquías de cachés y el intercambio de datos. (Mifsud, 2018)

Como consecuencias de estas funciones, la implantación de un servidor proxy-caché en una red proporciona las siguientes ventajas:

- Reduce los tiempos de respuesta.

Si la página web que se solicita está en el cache del servidor, dicha página se sirve sin necesidad de acceder de nuevo al servidor original, con lo cual se ahorra tiempo.

- Disminuye el tráfico en la red y el consumo de ancho de banda.

Si la página web está almacenada en la cache del servidor, la petición no sale de la red local y no será necesario hacer uso de la línea exterior consiguiendo así un ahorro en la utilización del ancho de banda.

- Cortafuegos:

Cuando se utiliza un servidor proxy-caché este comunica con el exterior, y puede funcionar como cortafuego, lo cual aumentara la seguridad del usuario respecto a la información a la que se acceda.

- Filtrado de servicios:

Es posible configurar el servidor proxy-caché dejando solo disponible aquellos servicios que se consideren necesarios, impidiendo la utilización del resto. Aunque Squid puede realizar una tarea de filtrado de acceso, este no es un filtro de contenido web (Mifsud, 2018).

1.2 Dansguardian filtro de contenido web.

Dansguardian es un filtro directo que se ubica entre el cliente web y el servidor proxy Squid, acepta conexiones en el puerto 8080 y se conecta a Squid en el puerto 3128. Este filtro de contenido web es una herramienta de código abierto desarrollada en C++ y permite una configuración flexible adaptándose a las necesidades del usuario. El mecanismo es el siguiente: Los clientes mediante sus navegadores web hacen peticiones que son recibidas por Dansguardian y solo son redireccionado al servidor proxy Squid aquellas que superan la fase de filtrado como: coincidencias de textos en la página web, filtrado de imágenes, cadenas de textos en la URLs (Mifsud, 2018).

1.3 Gestión de usuarios.

La gestión de los usuarios es, hoy en día, un área de entidad propia, en la medida en que la relación de éstos con los sistemas es crucial a la hora de asegurar un correcto funcionamiento de los servicios puestos a su disposición.

La seguridad de los datos de un sistema de gestión pasa por mantener un riguroso control de acceso a la información. Mediante la concesión de permisos se puede garantizar esa seguridad para controlar qué usuarios deben tener acceso a parte o a toda la información y quiénes no. Se pueden asignar permisos sólo para acceder a la información o permisos para acceder a la información y modificarla. (Moreno Boiza, 2012).

1.4 Análisis del estudio a los sistemas homólogos

1.4.1Sistemas Internacionales.

OpenDNS. Sistema de filtrado de contenido web.

OpenDNS es un sistema de filtrado de contenido web, que puede ser utilizado en varias plataformas, ofrece un servicio gratuito de filtrado de contenido web que se puede configurar para que funcione con el acceso a Internet. Fue desarrollado por la empresa de igual nombre fundada por David Ulevitch en noviembre de 2005. Dicho sistema tiene licencia gratuita, pero el código fuente es privativo; funciona con independencia del navegador web utilizado, permite activar/desactivar y configurar el filtro una sola vez en todos los navegadores del centro, la lista de páginas bloqueadas es actualizada y revisada constantemente.

Funcionamiento de OpenDNS:

Para activar el filtro de contenidos en un ordenador del centro es necesario asignarle manualmente estas direcciones IP: **208.67.222.222** y **208.67.220.220**. como servidor de DNS en la configuración de la tarjeta de red o bien configurar el servidor DHCP de direcciones IP del centro para que lo haga de forma automática cuando un ordenador se inicia y solicita una dirección IP con que conectarse a la intranet local. Es necesario registrar la IP de la red local para que OpenDNS filtre todas las peticiones que procedan de la misma. Es la IP externa que tiene el router que permite acceder a Internet desde la red local. De router hacia dentro, cada ordenador del centro tiene una IP local distinta, pero todas las peticiones de páginas web se realizarán a través del router y todas ellas llevarán asociadas la misma IP externa del router (OpenDNS, 2018).

Salix Networks. Sistema de filtrado de contenido Web

Salix Networks es un sistema de filtrado de contenido web desarrollado por la compañía mexicana de igual nombre en el año 2014. Dicho sistema permite bloquear el contenido no deseado de páginas Web, restringe el acceso a páginas de entretenimiento, compras, y páginas de chat entre otras, también bloquea los mensajeros instantáneos, deniega y evita los programas de descarga, inclusive llamados p2p como, por ejemplo: música, videos, la mayoría con contenido pornográfico, programas piratas y un alto contenido de virus. Presenta una licencia privativa de pago anual para uso de su servicio, además puede ser utilizado en varias plataformas (Networks, 2014).

1.4.2 Sistemas nacionales.

Sistema Xilema Smart Keeper versión 2.4

Xilema Smart Keeper 2.4 es un filtro de contenido web que permite aplicar las políticas de uso aceptable de Internet (AUP por sus siglas en inglés) en una institución. Está diseñado para interactuar como una lista de control de acceso (ACL por sus siglas en inglés) del servidor proxy Squid; cuenta fundamentalmente con una base de datos de URL clasificadas que rigen el acceso de los usuarios, así como una interfaz que facilita las tareas de administración. Su arquitectura en forma de plugin permite añadir nuevas características sin necesidad de modificar las actuales y el desacoplamiento entre sus componentes permite la distribución de los mismos en la red (Usuario, 2014).

Valoración de los sistemas analizados.

Los sistemas anteriormente estudiados no cumplen con el objetivo de la presente investigación, ni contienen una solución que se pudiera aplicar a la nueva versión del sistema Xilema Smart Keeper. Debido a que ninguno permite gestionar usuarios con respecto a la asignación de cuotas para la navegación en la web, de tal forma que: este proceso se pueda realizar de manera dinámica. Además, en el caso de los sistemas internacionales, no se puede acceder al código, porque tienen licencia privativa. Sin embargo, el análisis realizado posibilitó la identificación de nuevas funcionalidades como: requisitos funcionales, sincronizar, importar y filtrar usuarios del directorio activo LDAP.

1.5 Metodologías de desarrollo de software.

Las metodologías de desarrollo de software se pueden clasificar en: tradicionales y ágiles. Aquellas metodologías centradas en la definición detallada de los procesos y tareas a realizar, herramientas a utilizar y una extensa documentación, reciben el apelativo de metodologías tradicionales o robustas. Este tipo de tecnología es más eficaz en la medida en que el proyecto a realizar sea mayor, pues requiere de una gran organización.

Las metodologías ágiles hacen énfasis en la comunicación con el cliente y no exigen mucha documentación técnica. La comunicación en las metodologías ágiles se realiza cara a cara entre los miembros del proyecto y los clientes, evitando el trabajo de documentación fijado por las metodologías tradicionales. Están basadas en el trabajo organizado en equipos para la continua revisión y tratamiento de los productos de software alcanzados en cada iteración (Oliveros, 2014).

De entre estas se ha ajustado la metodología ágil AUP a una variación que se adapta al ciclo de vida definido para la actividad productiva en la UCI. Esta variación llamada AUP-UCI cuenta con 3 fases las cuales se describen como:

Inicio: Es en este punto del proyecto donde se llevan a cabo las actividades relacionadas con la planeación del mismo. En esta fase se definen el alcance del proyecto, las estimaciones de tiempo, el esfuerzo y costo y se decide si el proyecto será ejecutado o no; todo esto a través de un estudio al cliente.

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

Cierre: En esta fase se analizan los resultados del proyecto, se comprueba su correcta ejecución y se realizan las actividades formales de cierre del proyecto.

Las disciplinas se llevan a cabo de manera sistemática con objeto de desarrollar, validar, y entregar un software que responda a las necesidades de sus clientes. Estas disciplinas son:

1. Modelado de negocio: En esta disciplina se comprende cómo funciona el negocio que se desea informatizar como garantía de que el software desarrollado cumpla con el propósito planteado.

2. Requisitos: La tarea de esta disciplina es desarrollar un modelo del sistema que se va a construir. En la misma se comprenden la administración y gestión de los requisitos funcionales y no funcionales del producto.
3. Diseño: Para esta parte, de ser necesario, los requisitos pueden ser refinados y estructurados para conseguir una comprensión más precisa de estos, y una descripción que sea fácil de mantener y ayude a la estructuración del sistema. Además, en esta disciplina se modela el sistema y su forma para que soporte todos los requisitos.
4. Implementación: En la implementación se construye el sistema a partir de los resultados del análisis y el diseño.
5. Prueba interna: En esta disciplina se verifica el resultado de la implementación probando cada construcción, incluyendo tanto las construcciones internas como intermedias, así como las versiones finales a ser liberadas. Se deben desarrollar artefactos de prueba como: diseños de casos de prueba, listas de chequeo y de ser posibles componentes de prueba ejecutables para automatizar las pruebas.
6. Pruebas de liberación: Pruebas diseñadas y ejecutadas por una entidad certificadora de la calidad externa, a todos los entregables de los proyectos antes de ser entregados al cliente para su aceptación.
7. Pruebas de aceptación: Es la prueba final antes del despliegue del sistema. Su objetivo es verificar que el software está listo y que puede ser usado por usuarios finales para ejecutar aquellas funciones y tareas para las cuales el software fue construido.

Por último, para la parte de gestión y soporte se definen por las áreas de procesos de CMMIDEV v1.3 para el nivel 2, estas serían CM (Gestión de la configuración), PP (Planeación de proyecto) y PMC (Monitoreo y control de proyecto).

A partir de que el Modelado de negocio propone tres variantes a utilizar en los proyectos (Casos de uso del negocio (CUN), Descripción de proceso de negocio (DPN) o Modelo conceptual (MC)) y existen tres formas de encapsular los requisitos (Casos de uso del sistema (CUS), Historias de usuarios (HU), Descripción de requisitos por proceso (DRP)), surgen cuatro escenarios para modelar el sistema en los proyectos los cuales son:

- Proyectos que modelen el negocio con CUN solo pueden modelar el sistema con CUS.

CUN + MC =CUS

- Proyectos que modelen el negocio con MC solo pueden modelar el sistema con CUS.

MC = CUS

- Proyectos que modelen el negocio con DPN solo pueden modelar el sistema con DRP.

DPN + MC = DRP

- Proyectos que no modelen negocio solo pueden modelar el sistema con HU.

1.5.1 Metodología de desarrollo de software a utilizar.

La selección de la metodología a utilizar, fue determinada por el hecho de que la solución a desarrollar es un componente perteneciente a la nueva versión del sistema Xilema Smart Keeper; por tanto, para garantizar una perfecta integración se decide utilizar la Metodología AUP con variación para la UCI que es con la cual será desarrollado la nueva versión del sistema. Dicha metodología se adapta al ciclo de vida definida para la actividad productiva de la UCI, la cual tiene en cuenta los siguientes aspectos:

- Describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de software en la UCI, utilizando a su vez las buenas prácticas del modelo CMMI.
- Es una metodología ágil por tanto se adapta al trabajo de equipos de pequeño formato.
- Permite la gestión de cambios ágiles sin la necesidad de generar excesiva documentación.
- Es la metodología utilizada en el proyecto Smart Keeper, al cual se integrará el componente final.
- Permite la descripción simplificada del proceso de desarrollo de software acortando los tiempos de desarrollo de una aplicación.
- De esta metodología se decide usar el cuarto escenario para la disciplina de requisitos pues el mismo es recomendado para proyectos donde el objetivo primario es la gestión y presentación de información.

1.6 Ambiente de desarrollo.

Se realiza una investigación de las herramientas, lenguajes, tecnologías y metodologías que se emplearán en el desarrollo del componente para darle solución al problema

1.6.1 Lenguajes para el desarrollo.

HTML 5

HTML es un lenguaje de computadora ideado para permitir la creación de sitios web. Estos sitios web pueden ser vistos por cualquiera quien se encuentre conectado a Internet. Es relativamente fácil de aprender siendo su contenido básico de fácil acceso para la mayoría de las personas; y bastante poderoso en lo que permite crear. Se encuentra en constante revisión y evolución para cumplir con las demandas y requisitos de la creciente audiencia de Internet bajo la dirección del W3C (World Wide Web Consortium), la organización encargada del diseñando y manteniendo el lenguaje.

La definición de HTML es *HyperText Markup Language* (Lenguaje de marcado de hipertexto).

• *HyperText* es el método por el cual se mueve en la web - haciendo clic en *textos* especiales llamados hipervínculos los cuales conectan con la siguiente página. El **híper** sólo significa que no es lineal, esto significa que pueden ir a cualquier lugar en Internet siempre que lo desee solo con hacer clic en los enlaces, no hay un orden establecido para hacer las cosas.

• *Markup* es lo que las etiquetas HTML hacen con el texto dentro de ellas. Lo marcan como un cierto tipo de texto (ej. cursiva).

• HTML es un lenguaje, ya que tiene códigos de palabras y sintaxis como cualquier otro idioma (GONZÁLEZ, 2016).

CSS 3

El *Cascading Style Sheets* (CSS) es un lenguaje de diseño estándar para la web. La clave de su invención por parte de *Bert Bos* y *Hakon Wium Lie*, es que CSS separa la presentación de la estructura subyacente y la semántica. CSS saca las instrucciones visuales de HTML y se adhiere lo suficiente para presentar el sitio en la forma de un navegador gráfico tradicional (SEBASTIÁN, 2015).

JavaScript 1.11.2

JavaScript es un lenguaje de programación interpretado con capacidades orientadas a objetos (OO). Sintácticamente se asemeja a C, C ++ y Java, con construcciones de programación como la instrucción ***if***, el ciclo ***while*** y el operador ***&&***. La similitud termina con este parecido sintáctico, sin embargo, es un lenguaje de tipo suelto, lo que significa que las variables no necesitan tener un tipo específico. Los objetos en *JavaScript* asignan nombres de propiedades a valores de propiedad arbitrarios. De esta forma, se parecen más a las tablas hash o a las matrices asociativas (en Perl) que a las estructuras (en C) u objetos (en C ++ o Java). Su mecanismo de herencia OO está basado en prototipos como el del lenguaje poco conocido Self.

El núcleo del lenguaje JavaScript admite números, cadenas y valores booleanos como tipos de datos primitivos. También incluye soporte integrado para objetos de matriz, fecha y expresión regular. La versión integrada de este ejecuta scripts incrustados en páginas web HTML. Comúnmente se llama client-side JavaScript (del lado del cliente) para enfatizar que los scripts son ejecutados por la computadora cliente en lugar del servidor web (VIGOUROUX, 2015).

PHP 5.6

PHP es un popular lenguaje multiplataforma y del lado del servidor integrado en HTML. En el nivel más básico, PHP puede hacer cualquier cosa que cualquier otro programa CGI (*Common Gateway Interface*) pueda hacer, como recopilar datos de formularios, generar contenido de páginas dinámicas o enviar y recibir cookies. PHP se distribuye gratis bajo la licencia GNU y se usa en decenas de miles de sitios web de Internet en todo el mundo (PHP, 2017).

1.6.2 Framework de desarrollo.

Symfony 3.4

Symfony es un completo framework diseñado para optimizar el desarrollo de las aplicaciones web. El mismo separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. Symfony está desarrollado completamente con PHP 5 (symfony, 2017).

Ha sido probado en numerosos proyectos reales y se utiliza en sitios web de comercio electrónico de primer nivel. Symfony es compatible con la mayoría de gestores de bases de datos, como MySQL, PostgreSQL, Oracle Y SQL Server De Microsoft. Se Puede ejecutar tanto en plataformas *nix (Unix, Linux, etc.) como en plataformas Windows.

(symfony, 2017)

Características de Symfony:

- Fácil de instalar y configurar en la mayoría de plataformas.
- Independiente del sistema gestor de bases de datos.
- Mayormente sencillo de usar y a la vez adaptable a casos complejos.
- El desarrollador solo debe configurar aquello que no es convencional.
- Sigue la mayoría de las mejores prácticas y patrones de diseño para la web.
- Adaptable a las políticas y arquitecturas propias de cada empresa, y estable como para desarrollar a largo plazo.
- Código fácil de leer que incluye comentarios de *phpDocumentor* y que permite un mantenimiento muy sencillo.
- Fácil de extender, lo que permite su integración con librerías desarrolladas por terceros.

Para la selección de esta versión de Symfony se tuvo en cuenta que dicha versión 3.4 es la más estable actualmente para el desarrollo de proyectos, es una versión que tiene soporte a largo plazo, hasta noviembre del 2020.Además proporciona una gran cantidad de herramientas para depurar aplicaciones como por ejemplo el comando *debut: form,* para depurar formularios. Esto significa que al ejecutar este comando en la consola de Symfony se muestran todos los *form types* que incluye Symfony, todos los que define la aplicación, las extensiones de formulario y los *guessers*.

1.6.3 Entorno de desarrollo integrado

NetBeans 8.2

NetBeans IDE es un entorno de desarrollo visual de código abierto para aplicaciones programadas mediante Java, uno de los lenguajes de programación más poderosos del momento. Su aprendizaje se ha convertido en fundamental para quienes están interesados en el desarrollo de aplicaciones multiplataforma. Mediante NetBeans es posible diseñar aplicaciones con solo arrastrar y soltar objetos sobre la interfaz de un formulario. La programación mediante NetBeans se realiza a través de componentes de software modulares, también llamados módulos. NetBeans pone a disposición de los usuarios decenas de módulos a través de su página web, que podrás integrar en él para conseguir mejores aplicaciones. (Netbeans, 2016)

Para la selección de este entorno de desarrollo integrado se tuvo en cuenta las características antes mencionadas y que: la versión 8.2 es la última de NetBeans, es el entorno de desarrollo integrado utilizado en el Centro de Ideoinformática (CIDI).

1.6.4 Servidor de Bases de Datos.

Una de las tareas más comunes de las aplicaciones web dinámicas es la persistencia y lectura de la información en las bases de datos. El *framework Symfony* seleccionado como marco de trabajo para el desarrollo de la actual propuesta de solución integra *Doctrine*, esta es una biblioteca que se utiliza para simplificar las operaciones que se realizan en las bases de datos, posee controladores que permiten la compatibilidad con servidores de bases de datos de código abierto como es el caso de *MySQL* (PACHECO, 2013)

My SQL 5.7

Es un SGBD relacional, su diseño multihilo le permite soportar una gran carga de datos de forma eficiente. *MySQL* es un gestor de base de datos de código abierto ofrecido por Oracle. Su utilización está enfocada en aplicaciones web dinámicas escritas en *PHP* por la optimización de consultas sencillas y su compatibilidad con el servidor web *Apache*. Este se caracteriza por su nivel de seguridad, es multiplataforma, admite hasta 32 índices por tablas, la variedad de tipos de datos para las columnas y la utilización de *triggers* (Oracle, 2017)

1.6.5 Servidor web.

Para el hospedaje del sistema se emplea el servidor web *Apache* 2.2, es un software de código abierto desarrollado por la empresa *Apache Software Fundation* en 1996. Es un servidor web flexible, rápido y eficiente, altamente configurable por su diseño modular, esta característica permite ampliar

considerablemente sus capacidades pues existe un repositorio extenso completamente gratuito de extensiones y módulos. Es compatible con el lenguaje de programación *PHP,* comparten muchas de sus características (Apache, 2017)

1.6.6 Herramienta CASE.

Una herramienta CASE permite organizar y manejar la información de un proyecto informático. Las herramientas CASE de apoyo son esenciales para la gestión de configuraciones y pueden ser combinadas para crear entornos de trabajo de este tipo que soporten todas las actividades (Sommerville, 2011). Está destinada a aumentar la productividad en el desarrollo de software reduciendo el costo en términos de tiempo y dinero.

Visual Paradigm 8.0.

Visual Paradigm es una herramienta que soporta el ciclo de vida completo del desarrollo de software. Ayuda a la construcción de aplicaciones de calidad, permitiendo el modelado de los diagramas de clases, código inverso, generar código desde diagramas y generar documentación.

Sus principales características son:

- Posee licencia gratuita y comercial.
- Soporta aplicaciones web.
- Resulta fácil de instalar y actualizar.
- Soporta notación UML 2.x.
- Permite importaciones de *Rational Rose*.
- Permite generación de código e ingeniería inversa a la vez de los lenguajes: Java, C++, CORBA, IDL, PHP, XML *Schema*, Ada y *Python*.
- Soporta la generación de código en: C#, VB .NET, *Object Definition Language* (ODL), *Flash Action Script*, *Delphi*, *Perl*, *Objective-C* y *Ruby.*
- Soporta ingeniería inversa para Java *class*, .NET (dll, exe), JDBC y ficheros mapeados de *Hibernate*.

1.7 Conclusiones del capítulo.

El análisis de los elementos que componen al servidor proxy Squid, así como los elementos y estructuras que componen al filtro de contenido web Dansguardian y la definición de gestión de usuarios permitió identificar que se puede unificar el funcionamiento del servidor proxy Squid, con el filtro de contenido web Dansguardian como fundamento principal para implementar la gestión de usuarios.

A partir del análisis de las soluciones existentes para filtrar contenidos web se identificó que no existe un sistema que tenga implementado una gestión de usuarios para la configuración de cuotas de navegación de forma dinámica que se pueda integrar con los módulos del sistema Xilema Smart Keeper, por lo que se decide el desarrollo de un componente para la gestión de usuarios, teniendo en cuenta algunos requisitos identificados en las herramientas estudiadas.

El estudio de las herramientas y tecnologías utilizadas garantizan un correcto nivel de integración con los módulos del sistema Xilema Smart Keeper 3.0.

Capítulo 2: Diseño del componente para la gestión de usuarios para el sistema Xilema Smart Keeper.

Las características del componente deben quedar claramente identificadas y redactadas para que el software quede con la calidad que requiere. Por esta razón en el presente capítulo se realiza una descripción de la solución propuesta para la implementación del componente para la gestión de usuarios para el sistema Xilema Smart Keeper. Para ello se utilizan los artefactos de información propuestos por la metodología AUP-UCI seleccionada en el capítulo anterior.

2.2 Características del componente.

El componente para la gestión de usuarios para el sistema Xilema Smart Keeper permitirá la administración de los usuarios estableciendo las principales configuraciones generales y avanzadas. Brinda la posibilidad de realizar la autenticación de los usuarios, permitiendo seleccionar el origen de los datos de usuarios a validar como son: nuevos usuarios o usuarios LDAP (se encuentran localizados en un servicio de directorio activo LDAP de la institución). Permite agrupar y administrar (gestionar, listar y filtrar) usuarios en grupos de navegación, además de realizar acciones como: reiniciar cuota, habilitar o deshabilitar navegación a Internet. Posibilita importar y sincronizar a los usuarios LDAP de forma dinámica.

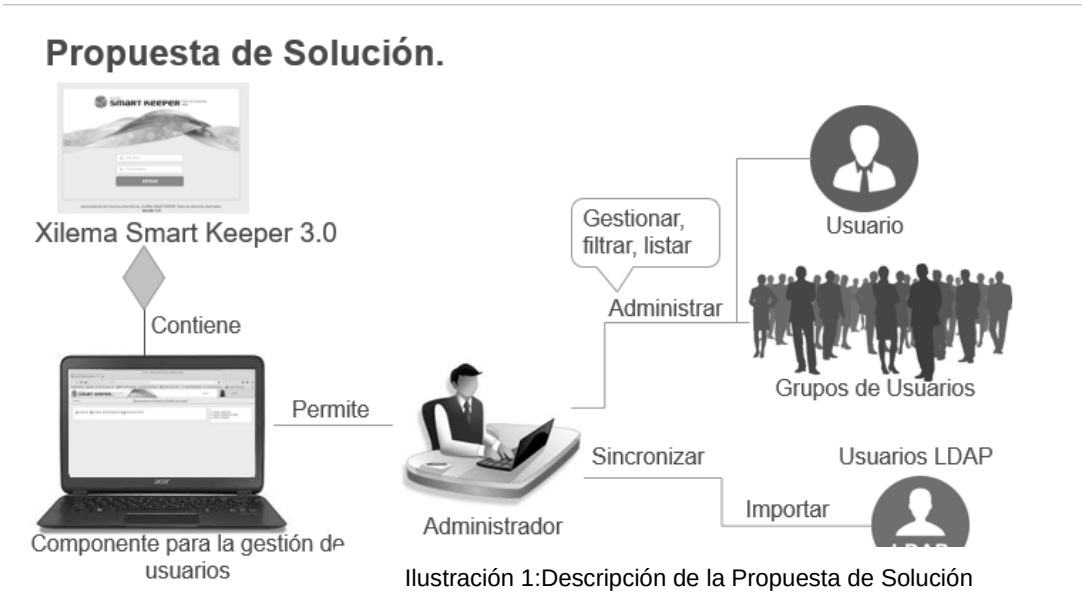


Ilustración 1:Descripción de la Propuesta de Solución

2.3 Especificación de los requisitos de Software.

2.3.1 Requisitos Funcionales.

Los requisitos funcionales especifican comportamientos particulares de un sistema, son características requeridas del sistema que expresan una capacidad de acción del mismo (Pressman, 2010). A continuación, se listan los requerimientos funcionales que el componente para la gestión de usuarios para el sistema Xilema Smart Keeper debe cumplir y la prioridad que se establece teniendo en cuenta las necesidades estimadas por el cliente. Todos los requisitos son desarrollados por el programador del componente, entiéndase por programador al autor de este documento.

Nº	**Nombre**	**Prioridad**
RF1	Autenticar usuario	Alta
RF2	Insertar usuarios	Alta
RF3	Editar usuarios	Media
RF4	Ver detalles de un usuario	Baja
RF5	Eliminar usuario	Media
RF6	Eliminar varios usuarios	Media
RF7	Listar usuario	Media
RF8	Filtrar usuarios	Baja
RF9	Reiniciar cuota de un usuario	Media
RF10	Reiniciar cuota de un grupo de navegación	Media
RF11	Adicionar un grupo de navegación	Alta
RF12	Editar un grupo de navegación	Media

RF13	Eliminar un grupo de navegación	Media
RF14	Eliminar grupos de navegación	Media
RF15	Listar grupos de navegación	Media
RF16	Cambiar cuota de un grupo de navegación	Media
RF17	Habilitar navegación a un grupo de navegación	Alta
RF18	Deshabilitar navegación a un grupo de navegación	Alta
RF19	Filtrar grupos de navegación	Media
RF20	Importar un usuario del directorio activo	Alta
RF21	Importar usuarios del directorio activo	Alta
RF22	Sincronizar los usuarios del directorio activo	Alta
RF23	Filtrar usuarios del directorio activo	Media

Tabla 1. Requisitos Funcionales del Componente para la gestión de usuarios para el sistema Xilema Smart Keeper.

2.3.2 Requisitos No Funcionales.

Son restricciones de los servicios o funciones ofrecidas por el sistema. Incluyen restricciones de tiempo, de desarrollo y estándares. A menudo se aplican al sistema en su totalidad (Pressman, 2010). A continuación, se presentan los requisitos no funcionales del Componente para la gestión de usuarios para el sistema Xilema Smart Keeper.

Usabilidad:

- El componente debe mantener las pautas de diseño que establece la estrategia marcaria de la UCI para productos comerciales.
- Debe ser intuitivo, agrupando los requisitos funcionales por responsabilidad.

Hardware:

- El ordenador donde se ejecute el componente debe tener una memoria RAM de 2 GB, además de poseer un procesador familia INTEL u otro con una velocidad mínima de 2.10 GHz y la capacidad del disco duro debe ser de 500Mb como mínimo.
- El disco duro del servidor de base de datos debe tener una capacidad de almacenamiento de 80 GB como mínimo.

Software:

- El procesador donde se ejecute el componente debe tener instalado GNU/Linux como sistema operativo, se recomiendan la distribución Ubuntu versión 16.04.
- El ordenador donde se ejecute el componente requiere de un servidor de base de datos *MySQL* 5.7.

Seguridad:

- La información manejada en el componente debe ser accesible solo por un usuario con rol administrador.
- Los datos sensibles como contraseñas almacenadas en Bases de Datos deben estar encriptadas.

2.4 Historias de Usuarios.

La metodología *AUP*-UCI, en su escenario cuatro (4) para modelar el sistema en la disciplina Requisitos, genera como artefacto a las Historias de Usuarios (HU) (SÁNCHEZ, 2015), estas se utilizan para especificar los requisitos de software en las aplicaciones. Las HU se descomponen en tareas de programación y describen las características que el sistema debe cumplir, están escritas en un formato legible por el cliente, sin necesidad de sintaxis técnicas (Ordoñez, 2016)

A continuación, se muestran dos (2) HU de las quince (15) que fueron generadas pertenecientes a los RF 1, RF 2, RF 3, RF 4 y RF5, el resto pueden encontrarlas en el Anexo 1 de la presente investigación.

Tabla 2 : HU_1 Autenticar usuarios.

Número: 01	**Nombre del requisito:** Autenticar usuario
Programador: Nardy Román Padrón	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 3
Riesgo en Desarrollo: Alta	**Tiempo Real:** 2

Descripción: Permitirá autenticar un cliente.

Los usuarios con rol Administrador que se hayan registrado con anterioridad en el sistema se podrán autenticar, para ello deberán llenar los siguientes campos:

- Nombre de usuario (Obligatorio. Campo de texto. Permiten los caracteres: (a-z, A-Z, 0-9). Debe comenzar con (a-z, A-Z). Longitud entre 5 y 50.)
- Contraseña (Obligatorio. Campo de texto. Permite todos los caracteres y con longitud entre 6 y 50)

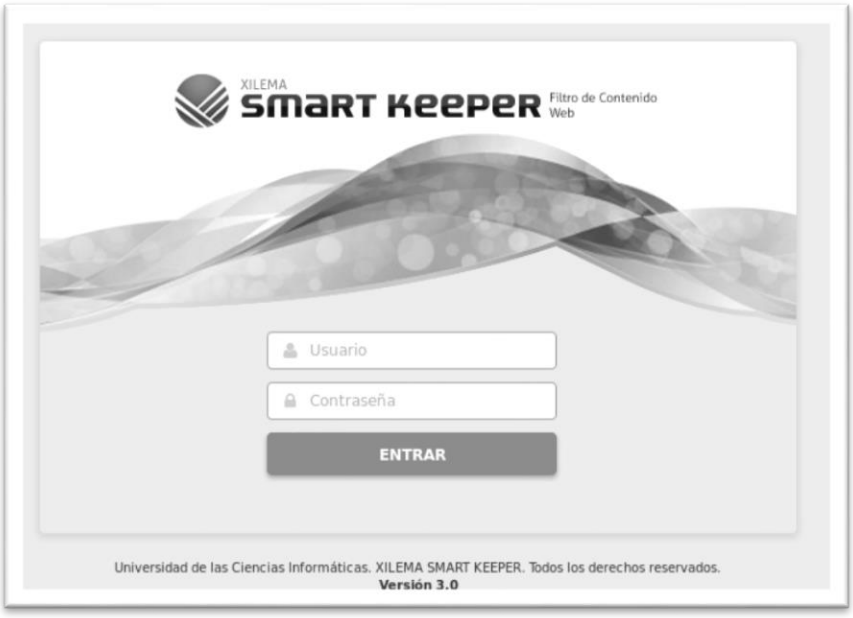
Observaciones: 1. El usuario debe estar registrado en la base de datos del sistema. 2. Si el usuario introduce la información de forma correcta, el sistema emite un mensaje notificando que se ha autenticado satisfactoriamente el usuario. 3. Si el usuario introduce la información de forma incorrecta, el sistema emite un mensaje notificando el error. 4. Si el usuario introduce la información dejando campos obligatorios vacíos, el sistema emite un mensaje indicándole que los campos obligatorios deben llenarse.
Interfaz del sistema: - 

Tabla 3:HU_2 Administrar usuarios.

Número: 02	**Nombre del requisito:** Administrar usuarios
Programador: Nardy Román Padrón	**Iteración Asignada:** 2

Prioridad: Alta	**Tiempo Estimado:** 8
Riesgo en Desarrollo: Que no se haiga insertado el usuario correctamente.	**Tiempo Real:** 6

Descripción:

Permitirá insertar, modificar, eliminar, ver detalles.

Los usuarios con rol Administrador podrán administrar los contenidos de usuarios en el sistema realizando alguna de las siguientes tareas:

1. **Insertar nuevos usuarios.**
2. **Modificar usuarios.**
3. **Eliminar usuarios.**
4. **Ver detalles de usuarios.**

Para realizar las tareas deberán llenar los siguientes campos.

- Nombre(s) (Obligatorio. Campo de texto. Longitud entre 2 y 255 caracteres. Permite todos los caracteres. No puede contener solo caracteres especiales y/o números.)
- Apellido(s) (Obligatorio. Campo de texto. Longitud entre 2 y 255 caracteres. Permite todos los caracteres. No puede contener solo caracteres especiales y/o números.)
- Correo (Obligatorio. Campo de texto. Longitud máxima 255 caracteres. Permite direcciones de correo estructuralmente válidas (Estructura: usuario@subdominios.dominio)
- Contraseña (Obligatorio. Campo de texto. Permite todos los caracteres y con longitud entre 6 y 50)

Observaciones:

-

Interfaz del sistema:


USUARIOS Adicionar
PERFIL
GRUPOS, PERMISOS, NAVEGACION
Nombre(s)
Apellidos
Usuario
Contraseña
Debe contener mas de 6 caracteres
Correo Electónico
Ej:usuario@institucion.cu
GUARDAR USUARIO
CANCELAR

-

2.5 Arquitectura de Software.

Para el desarrollo del componente para la gestión de usuarios para el sistema Xilema Smart Keeper y para el apoyo a la toma de decisiones se empleará como marco de trabajo Symfony que utiliza el patrón arquitectónico Modelo-Vista-Controlador (MVC).

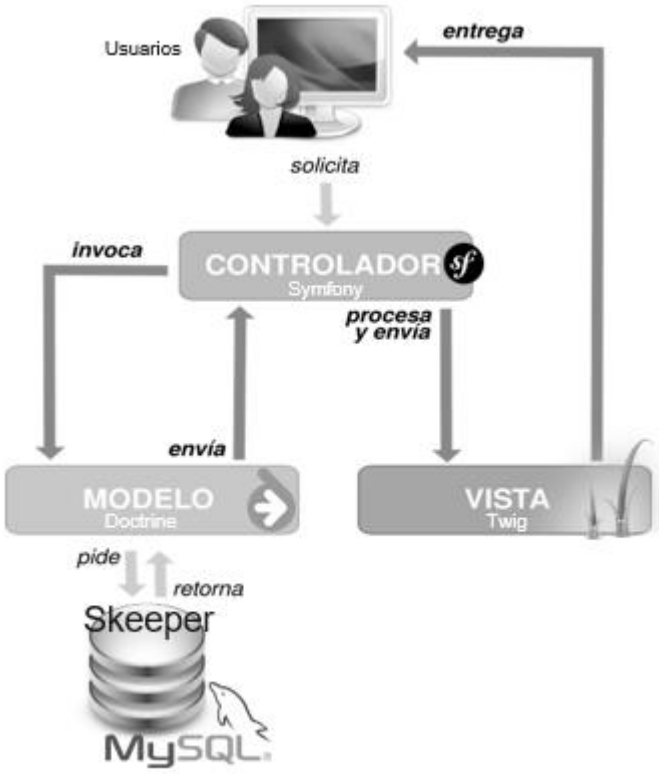


Ilustración 2:Arquitectura Modelo-Vista-Controlador de Symfony en base a la propuesta de solución

La arquitectura MVC separa la lógica de negocio (el modelo) y la presentación (la vista) por lo que se consigue un mantenimiento más sencillo de las aplicaciones. Si por ejemplo una misma aplicación debe ejecutarse tanto en un navegador estándar como un navegador de un dispositivo móvil, solamente es necesario crear una vista nueva para cada dispositivo; manteniendo el controlador y el modelo original. El controlador se encarga de aislar al modelo y a la vista de los detalles del protocolo utilizado para las peticiones (HTTP, consola de comandos, email, etc.). El modelo se encarga de la abstracción de la lógica relacionada con los datos, haciendo que la vista y las acciones sean independientes de, por ejemplo, el tipo de gestor de bases de datos utilizado por la aplicación. (symfony, 2017).

- **Modelo:** Representa la información con la que trabaja el sistema, su lógica de negocio. El modelo está representado en los archivos del directorio: *Entity*.
- **Vista:** Transforma el modelo en una página web, lo que permite al usuario interactuar con este. Las vistas del componente están agrupadas en el directorio: *app/Resources/view.*
- **Controlador:** Se encarga de procesar las interacciones del usuario, y realiza los cambios apropiados en el modelo o la vista. Las clases controladoras de la propuesta de solución están situadas en el directorio: Controller.

En el controlador se encuentran las acciones que permiten administrar y configurar los usuarios, los grupos de navegación, los usuarios importados de un directorio activo y contienen la lógica de la aplicación. Estas acciones utilizan el modelo y envían las variables a la vista, que es la encargada de originar las páginas que son mostradas como resultado de las acciones, desarrolladas en el controlador. En el modelo se encuentra la clase relacionada con los usuarios para el acceso y la manipulación de los datos, realizado mediante objetos. El controlador es la capa intermediaria entre la Vista y el Modelo.

2.6 Patrones utilizados en el desarrollo de la aplicación.

Un patrón de diseño describe un problema que ocurre frecuentemente en el campo de la construcción de software y su respectiva solución; puede ser empleado muchas veces, en diferentes contextos, sin tener que duplicar el diseño. Se trata de un elemento de diseño que puede ser reutilizado, el mismo identifica: clases, instancias, roles, colaboraciones y la distribución de responsabilidades. (SIERRA, 2017).

Los Patrones Generales de Software para Asignar Responsabilidades (GRASP del inglés General Responsibility Assignment Software Patterns) tienen una importante utilidad en el diseño de una aplicación, al igual que los Gang-of-Four o Pandilla de los Cuatro (GoF por sus siglas en inglés). A continuación, se muestra una selección de estos patrones los cuales serán utilizados durante el diseño del componente:

2.6.1 Patrones generales de software para la Asignación de Responsabilidades (GRASP).

Los GRASP son patrones generales de software para asignación de responsabilidades a objetos. Aunque se considera que más que patrones propiamente dichos, son una serie de "buenas prácticas" de aplicación recomendable en el diseño de software. Constituyen un apoyo para la enseñanza que ayuda a entender el

diseño de objeto esencial y aplica el razonamiento para el diseño de una forma sistemática, racional y aplicable (Visconti, y otros, 2012).

- Experto

Este patrón sugiere que se debe asignar las responsabilidades a aquellos objetos que disponen de la información para hacerlo. En el Componente este patrón se evidencia en las clases UsuarioController, GruposNavegacionController y UserLDAPController, que son las clases que tienen acceso a todas las entidades necesarias para generar la información, por lo tanto, son las expertas en información.

- Creador

Se aplica para la asignación de responsabilidades a las clases relacionadas con la creación de objetos, de forma tal que una instancia de un objeto sólo pueda ser creada por el objeto que contiene la información necesaria para ello. En este caso el patrón se refleja en las clases encargada de crear las entidades de usuario, que será utilizado posteriormente por UsuarioController y GruposNavegacionController.

- Bajo Acoplamiento

El acoplamiento es una medida de la fuerza en que una clase está conectada a otras, que la conoce y recurre a ellas. El objetivo de este patrón consiste en mantener un bajo nivel de dependencia de otros elementos, por lo que constituye un principio que debe estar presente en todas las decisiones de diseño con lo que se reduce el impacto de los cambios. Su utilización se evidencia en que las clases controladoras del sistema no se relacionan entre sí, lo que disminuye las dependencias entre las mismas.

- Alta Cohesión

La cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Se puede afirmar que cada una de las clases del sistema tiene alta cohesión, de manera que estas poseen la característica de tener las responsabilidades estrechamente relacionadas. En el componente, este patrón se manifiesta en la clase controladora UsuarioController, colabora y delega responsabilidades a la clase UsuarioType para la creación de formularios relacionados con la entidad Usuario.

- Controlador

Para este caso se hace necesario conocer que un evento del sistema es una operación que se realiza en este, generada por un usuario externo. Un controlador, es un objeto de interfaz no destinada al usuario que se encarga de manejar un evento del sistema. En el componente, los eventos generados por el usuario son redirigidos a una clase controladora que realiza las operaciones solicitadas

2.6.2 Patrones Gang of Four (GOF)

El catálogo de patrones más famoso es el contenido en el libro “Design Patterns: Elements of Reusable ObjectOriented Software”, también conocido como: El libro GOF (Gang-Of-Four Book). Según este documento, estos patrones se clasifican según su propósito en creacionales, estructurales y de composición, mientras que respecto a su ámbito se clasifican en clases y objetos:

- Estructural:
- Decorador (Decorator)

Forma parte de la familia de patrones denominados estructurales. Este tipo de patrones describen como las clases y objetos pueden ser combinados para formar grandes estructuras y proporcionar nuevas funcionalidades. Decorador permite modificar, retirar o agregar responsabilidades a un objeto dinámicamente. La gran ventaja es que permite extender objetos incluso en situaciones donde la extensión vía herencia no es necesaria. Adicionalmente ayuda a conservar el principio de Abierto/Cerrado, en donde se dicta que cada entidad debe estar abierta a extensión, pero cerrada a modificación. Este patrón se evidencia en la entidad usuario ya que mediante anotaciones (comentarios) en los atributos de la clase se puede especificar la columna de la base de datos en la cual será guardado este atributo como se muestra en el siguiente ejemplo donde la anotación @ORM\Column realiza dicha función:

```
/**
 * @var string
 * @ORM\Column (name="username", type="string", length=255)
*/
 private $username
```

- Controlador Frontal

Es uno de los patrones de diseño de Symfony más conocido. Es una sección de código que atiende todas las solicitudes en la aplicación y devuelve una respuesta al navegador. Además de que este controlador se utiliza para direccionar todas las solicitudes, es el encargado de iniciar el núcleo del sistema lo que le permite decorar el mismo con características adicionales. Uno de los principales problemas que evita es la duplicación de código, porque define un método comando en una clase abstracta, que luego se podrá redefinir en las otras clases según las particularidades que se necesiten, con el uso de decoradores. Se evidencia en los url donde todos parten del nodo app_dev.php

ejemplo de uno de estos casos es *http://localhost/SmartKeeper/web/app_dev.php.*

- Comportamiento:

Comando (Command): Es un patrón de diseño de comportamiento de objetos. Manipular y encapsular las peticiones de los usuarios y enviarlas a un objeto encargado de darle respuesta. Su uso es apropiado cuando lo fundamental en la relación entre una petición y la acción que la satisface es la flexibilidad (AspAlliance, 2013).

Método Plantilla (Template Method): Define una estructura algorítmica en la súper clase, delegando la implementación a las subclases. Es decir, define una serie de pasos, en donde los pasos serán redefinidos en las subclases (polimorfismo). Un ejemplo del uso de este patrón en Symfony2 se ve reflejado en las plantillas twig. Estas permiten la herencia entre clases, así como la redefinición de los métodos a fin de lograr el diseño deseado (AspAlliance, 2013).

Otros:

Registro (Registry): Este patrón es útil para los desarrolladores en la Programación Orientada a Objetos. Es un medio sencillo y eficiente de compartir datos y objetos en la aplicación sin la necesidad de preocuparse por conservar numerosos parámetros o hacer uso de variables globales. Se aplica en la clase de configuración que es la encargada de guardar todas las variables globales del sistema (AspAlliance, 2013).

2.7 Diagramas de clases del Diseño

Un Diagrama de Clases del Diseño (DCD) muestra la especificación de las clases de una aplicación, sus asociaciones, atributos y métodos, interfaces, navegabilidad y dependencias. Las clases de diseño de los DCD definen entidades y no conceptos del mundo real (2016). Se generaron un total de trece (13) DCD, de ellos se exponen (2) correspondientes a las HU_1 y HU_2, el resto pueden ser consultados en los anexos.

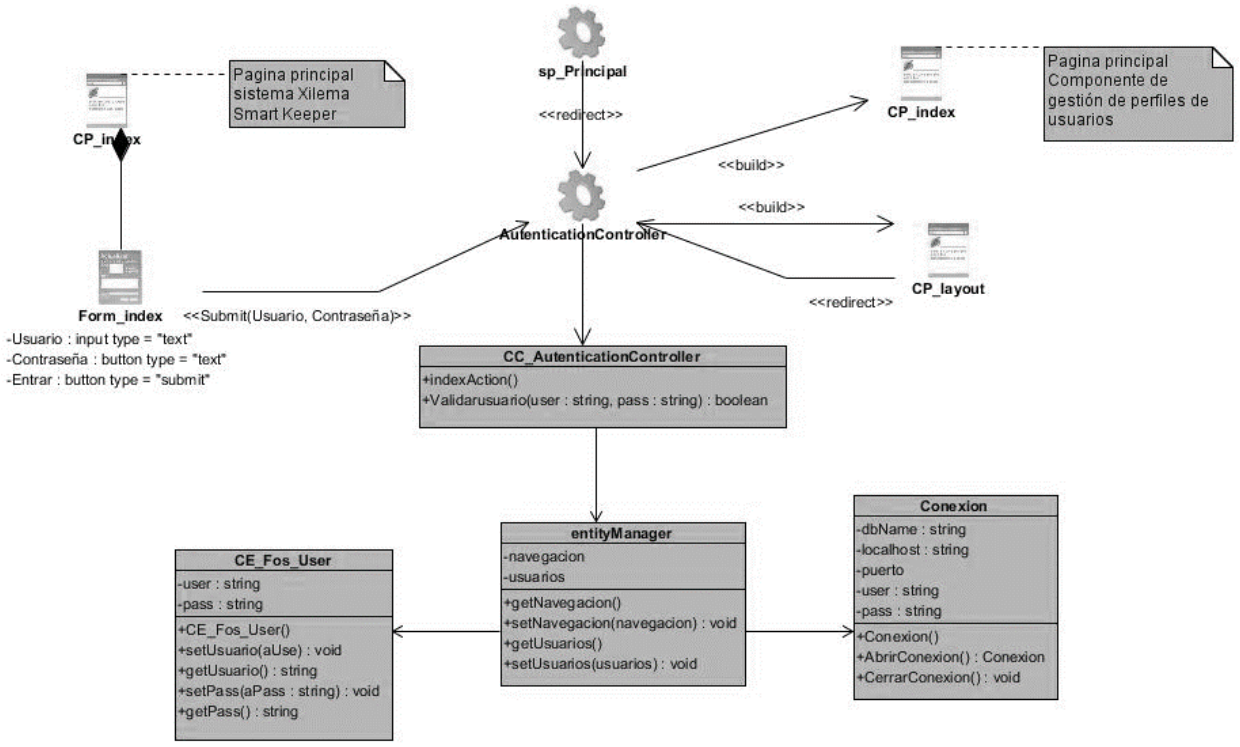


Ilustración 3:Diagrama de clases del diseño HU Autenticar usuario.

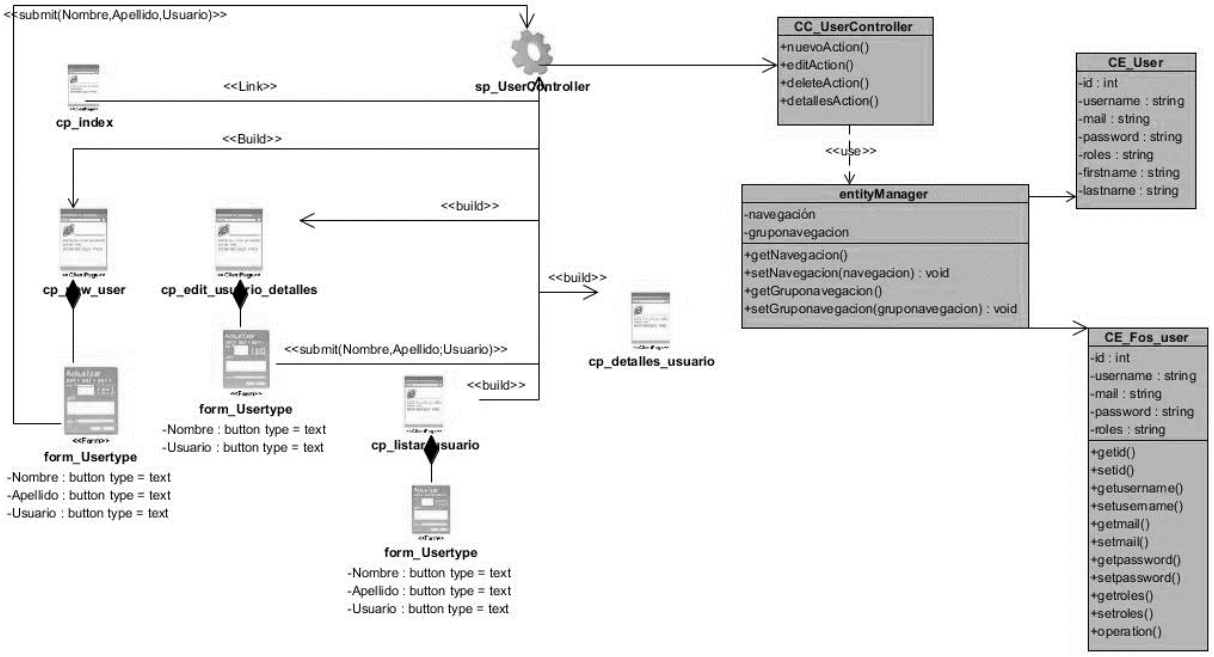


Ilustración 4: Diagrama de Clases del Diseño HU Administrar usuario.

2.8 Diagramas de secuencias.

Los diagramas de secuencia (DS) en el UML se utilizan principalmente para modelar las interacciones entre los actores y los objetos en un sistema, así como las interacciones entre los objetos en sí. (SOMMERVILLE, 2011). Estas interacciones son descritas en una especie de formato de cerca o muro. Estos diagramas muestran cómo los objetos se comunican entre sí en una secuencia de tiempo y para ello contienen los ciclos de vida y los mensajes que se envían entre ellos ordenados secuencialmente. A continuación, se muestran dos (2) relacionados con los RF 2 y RF 5.

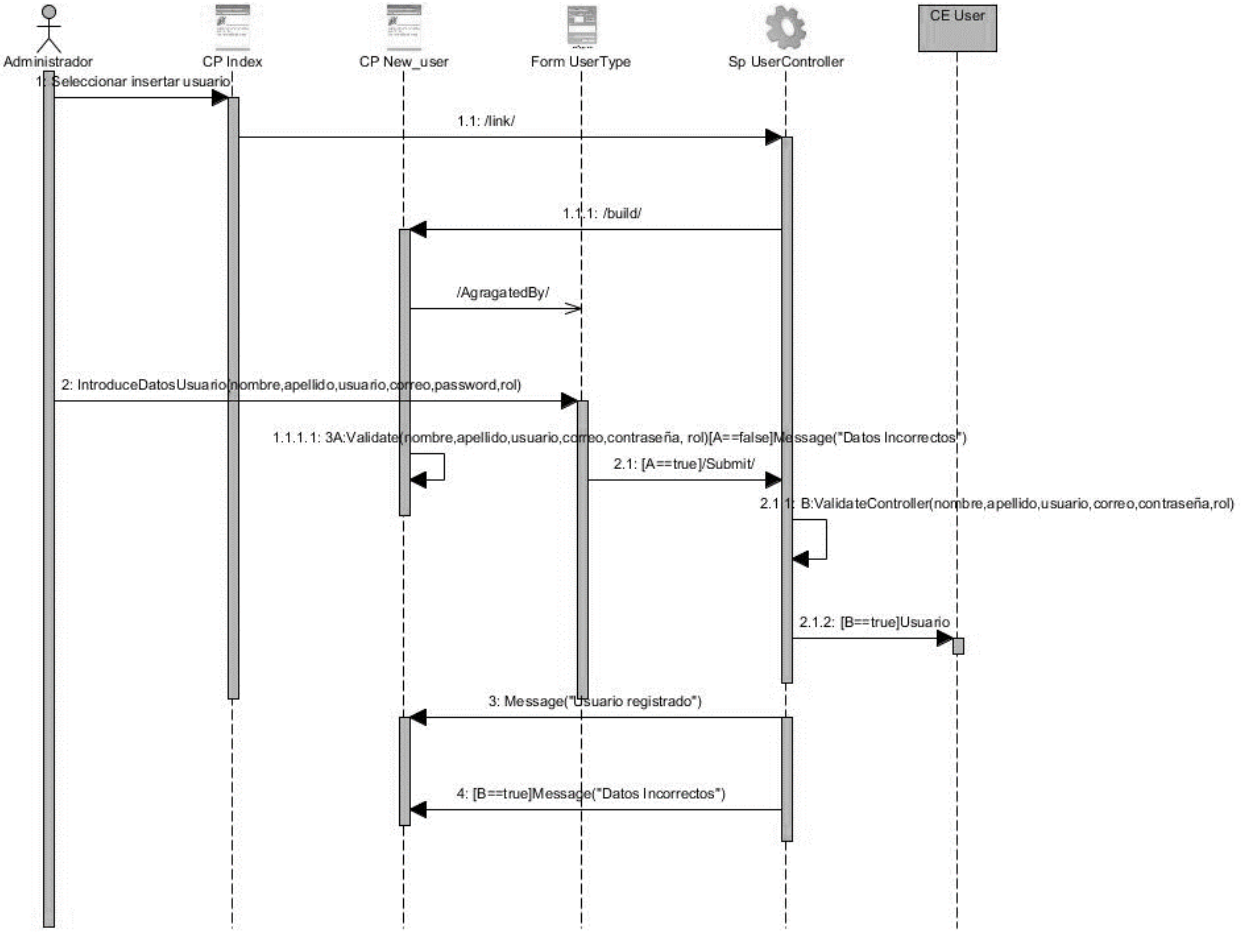
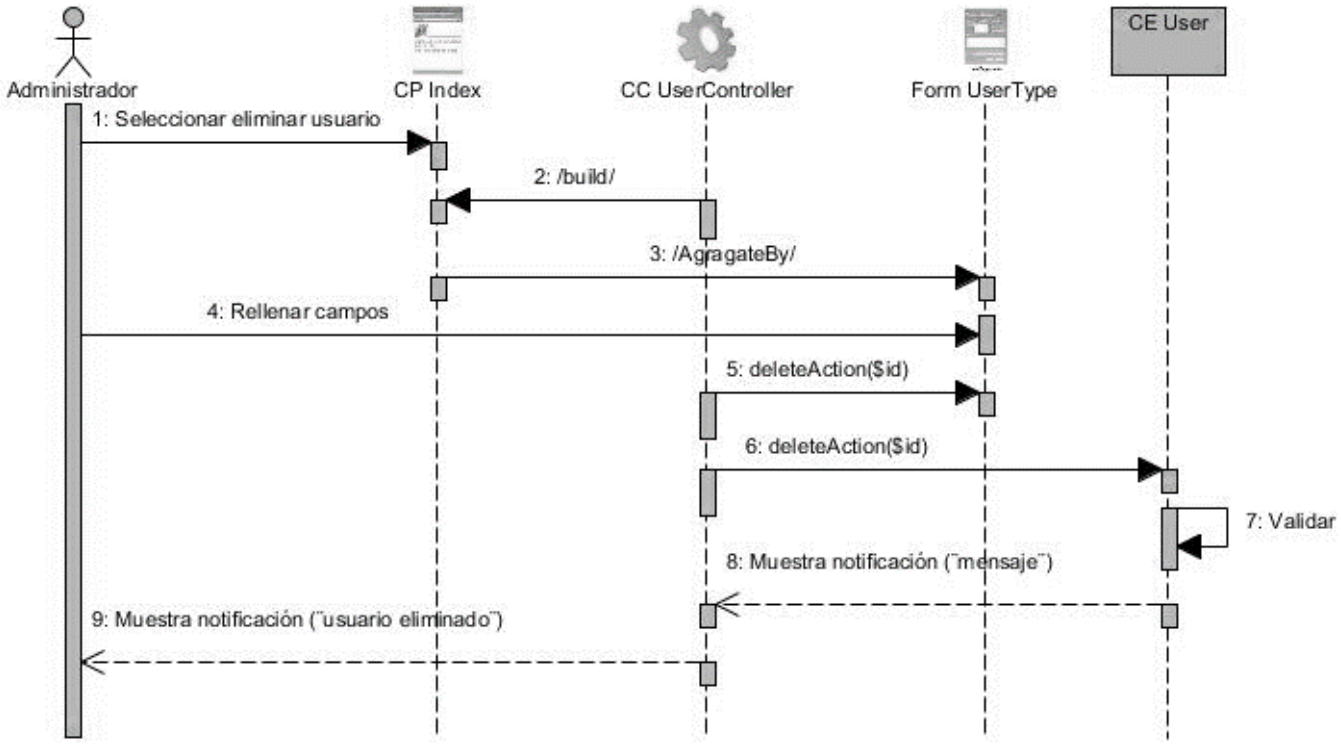


Ilustración 5: Diagrama de Secuencia Insertar usuario

Fuente: Elaboración propia.



2.9 Modelo de datos

Un modelo de datos es un sistema formal y abstracto que permite describir los datos de acuerdo con reglas y convenios predefinidos o podríamos decir que es un conjunto de conceptos que permiten describir, a distintos niveles de abstracción, la estructura de una base de datos (Redondo, 2016).

- Principales características de un modelo de datos:
- Independencia lógica y física de los datos.
- Redundancia mínima.
- Respaldo y recuperación.
- Seguridad de acceso y auditoria.
- Integridad de los datos.
- Consultas complejas optimizadas.

Fos_user: Registra la información de los usuarios del sistema. De este se registran el atributo que lo identifica (id), el usuario(*username*), correo electrónico(*email*), semilla para la encriptación de la contraseña(*salt*), contraseña(*password*), el role(roles), nombre (*firstname*), apellido(lastname).

Navegación: Almacena las características de los usuarios con respecto a la navegación. De estos se registran el identificador(id), la navegación (con acceso Internet o sin acceso), cuota (está basada en *megabyte*).

Grupo de Navegación: Almacena las características de los grupos de navegación. De estos se registran el identificador(id), nombre, descripción, cuota, día de reinicio de la cuota de navegación.

Fos_use_navegacion: Evidencia la relación mucho a mucho entre las tablas Fos_user y Navegación. Registra el identificador de dichas tablas.

Fos_user_grupo_navegación: Evidencia la relación mucho a mucho entre las tablas Fos_user y Grupo Navegación. Registra el identificador de dichas tablas.

UserLDAP: Registra las características de los usuarios obtenidos de un servicio de directorio activo LDAP**.**

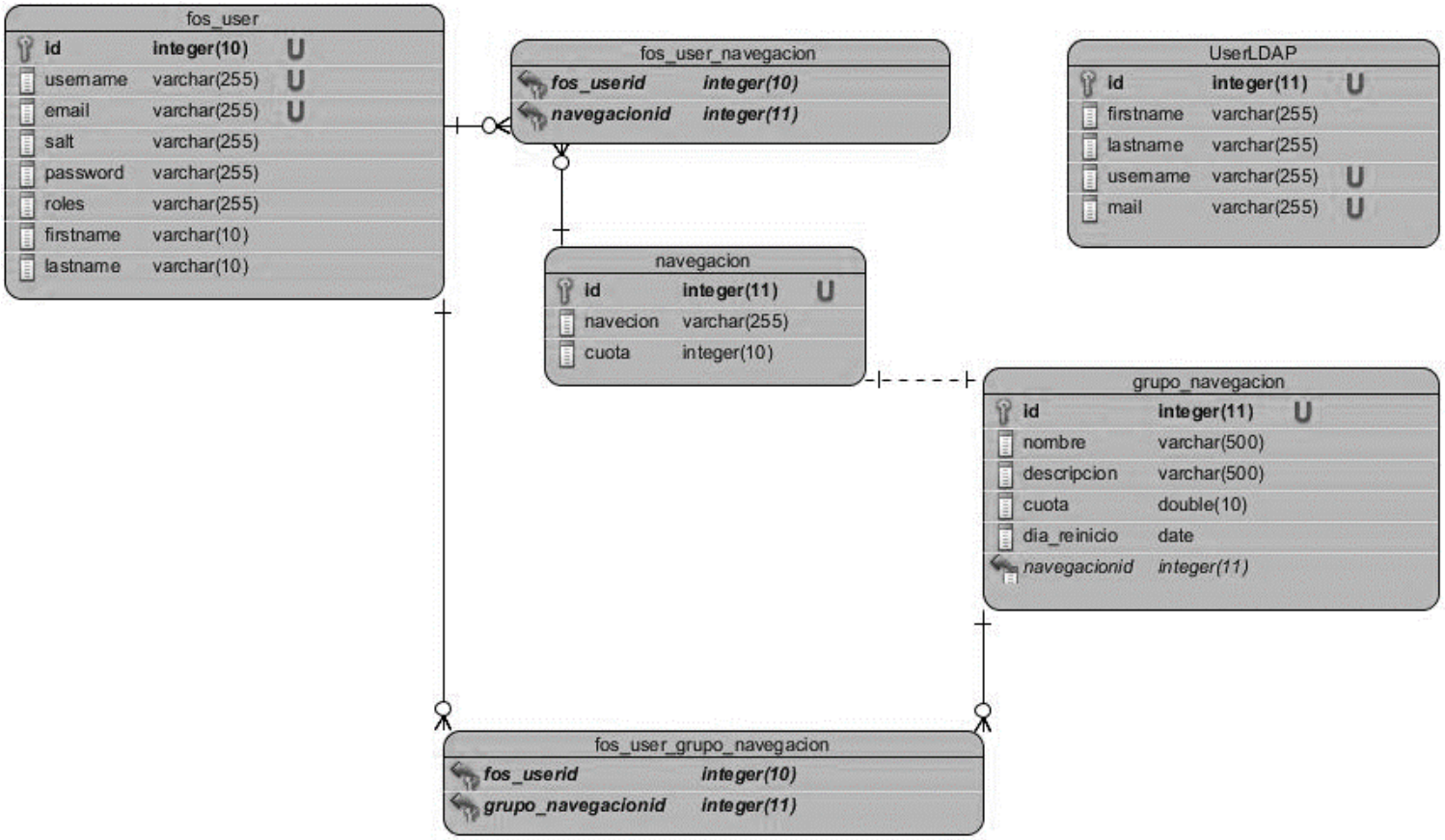


Ilustración 6: Modelo de Datos de la propuesta de solución

Fuente: Elaboración propia.

2.10 Modelo de despliegue.

El diagrama de despliegue se utiliza para mostrar la estructura física del sistema, incluyendo las relaciones entre el hardware y el software que se despliega, estas relaciones son representadas por los protocolos de comunicación que se utilizan para acceder a cada uno (SPARXSYSTEMS, 2014). En la siguiente figura puede visualizarse el diagrama de despliegue definido para la solución propuesta.

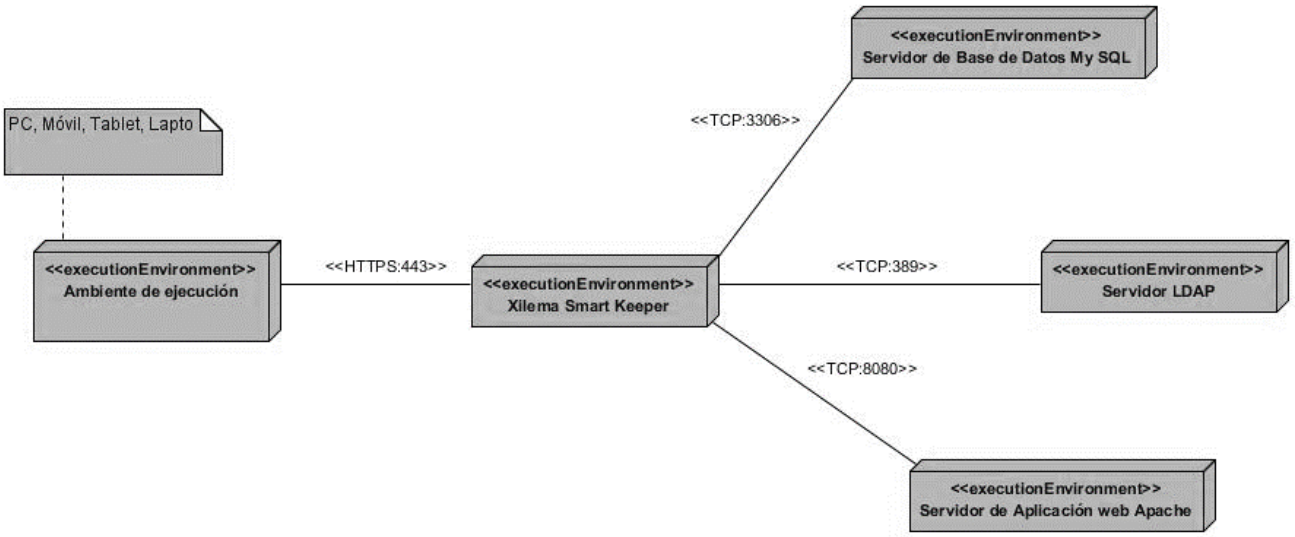


Ilustración 7:Diagrama de Despliegue

En este diagrama se define el nodo PC Cliente desde donde se originan las peticiones mediante el protocolo HTTPS a un servidor de aplicación web, donde se encuentra alojado el sistema Xilema Smart Keeper, el sistema mantendrá una comunicación con el servidor de base de datos MySQL a través del protocolo TCP, al servidor de directorio LDAP por el protocolo TCP y al servidor web Apache a través del protocolo TCP.

2.11 Conclusiones del capítulo.

- Con la especificación de los requisitos funcionales y no funcionales del sistema, se logró una mejor comprensión, de los resultados que se pretenden obtener de una manera precisa y sirven de guía para la implementación del sistema.
- La representación y descripción de los artefactos generados garantizó un mejor entendimiento de los flujos de trabajos presentes en el proceso de generar.
- La utilización de los patrones de diseño permitió identificar aspectos importantes de la estructura del diseño del componente propuesto, lo que garantizó una mayor organización e hizo el código más legible.
- La elaboración del diagrama de despliegue permitió identificar la disposición física de los artefactos del componente para la gestión de usuarios para el sistema Xilema Smart Keeper.

Capítulo 3: Implementación y validación del componente para la gestión de usuarios para el sistema Xilema Smart Keeper.

Con el objetivo de asegurar que el componente propuesto funciona de acuerdo a los requisitos definidos y materializar en forma de componentes la arquitectura, diagramas y artefactos generados en la fase de diseño, en el presente capítulo se describe la etapa de implementación y codificación del componente. Se documentan los resultados obtenidos de las estrategias de pruebas, utilizadas para garantizar la calidad del software durante la etapa de validación.

3.1 Diagrama de componente

El diagrama de componentes proporciona una visión física de la construcción del sistema. Muestra la organización de los componentes software, sus interfaces y las dependencias entre ellos (RAMOS Cardozzo, 2016). Basándose en la arquitectura de software que propone el marco de trabajo Symfony, seleccionado en el capítulo anterior. La ilustración 5 muestra el diagrama de componente para la gestión de usuarios para el sistema Xilema Smart Keeper a continuación, se describen los elementos que componen el diagrama.

Componentes				Descripción
Xilema Smart Keeper	Componente para la gestión de usuarios	Modelo	User.php	Clase entidad que almacena los datos de los usuarios para realizar la gestión.
			GrupoNavegacion.php	Clase entidad que almacena los datos de los grupos de navegación relacionado con los usuarios
			UserLDAP.php	Clase entidad que almacena los datos de los usuarios LDAP

		Controlador	UserController.php	Clase controladora que contiene las acciones relacionadas con los usuarios y sus características. Realiza acciones generales de gestión (insertar, eliminar, mostrar, editar, listar y buscar).	
			GrupoNavegacionController.php	Clase controladora que gestiona los grupos de navegación, así como su administración y configuración.	
			UserLDAPController.php	Clase controladora encargada de sincronizar, actualizar e importar los usuarios de un directorio LDAP.	
			Formularios	UserType.php	Clase responsable de construir los formularios relacionados con los usuarios.
				GrupoNavegacionType.php	Clase responsable de construir los formularios relacionados con los

					grupos de navegación
		Vista	conf_user	index.html.twig	Muestra todas las acciones a realizar en el componente para la gestión de usuarios
				new_user.html.twig	Permite registrar un usuario en el sistema a través de un formulario
				edit_user.html.twig	Permite editar un usuario a través de un formulario.
				delete_user.html.twig	Permite eliminar un usuario
				listar_user.html.twig	Permite listar los usuarios y sus

					características
				filter_user.html.twig	Permite filtrar los usuarios
			conf_grupoNavegación	nuevo_grupo_navegacion.html.twig	Permite registrar un grupo de navegación en el sistema a través de un formulario
				edit_grupo_navegacion.html.twig	Permite editar un grupo de navegación a través de un formulario.
				eliminar_grupo_navegacion.html.twig	Permite eliminar un grupo de navegación a través de un formulario.

				listar_grupo_navegacion.html.twig	Permite listar los grupos de navegación y sus características
				filtrar_grupo_navegacion	Permite filtrar los grupos de navegación
			conf_userLdap	actualizar.html.twig	Muestra si se actualizaron los usuarios en el directorio LDAP
				importar.html.twig	Muestra si se importaron los usuarios desde el directorio LDAP hacia la base de datos del componente
				sincronizar.html.twig	Muestra si se sincronizaro

					n los usuarios en el directorio LDAP con la base de datos del componente
		routing.yml		Contiene el mapa de rutas *URL* que se enlazan a las acciones de los controladores del componente.	
		libssh2		Librería *PHP* que permite realizar conexiones remotas a través del protocolo *SSH.*	
	config.yml			Contiene las configuraciones generales del sistema Xilema Smart Keeper.	

Tabla 4:Descripción de los componentes del Diagrama de Componentes

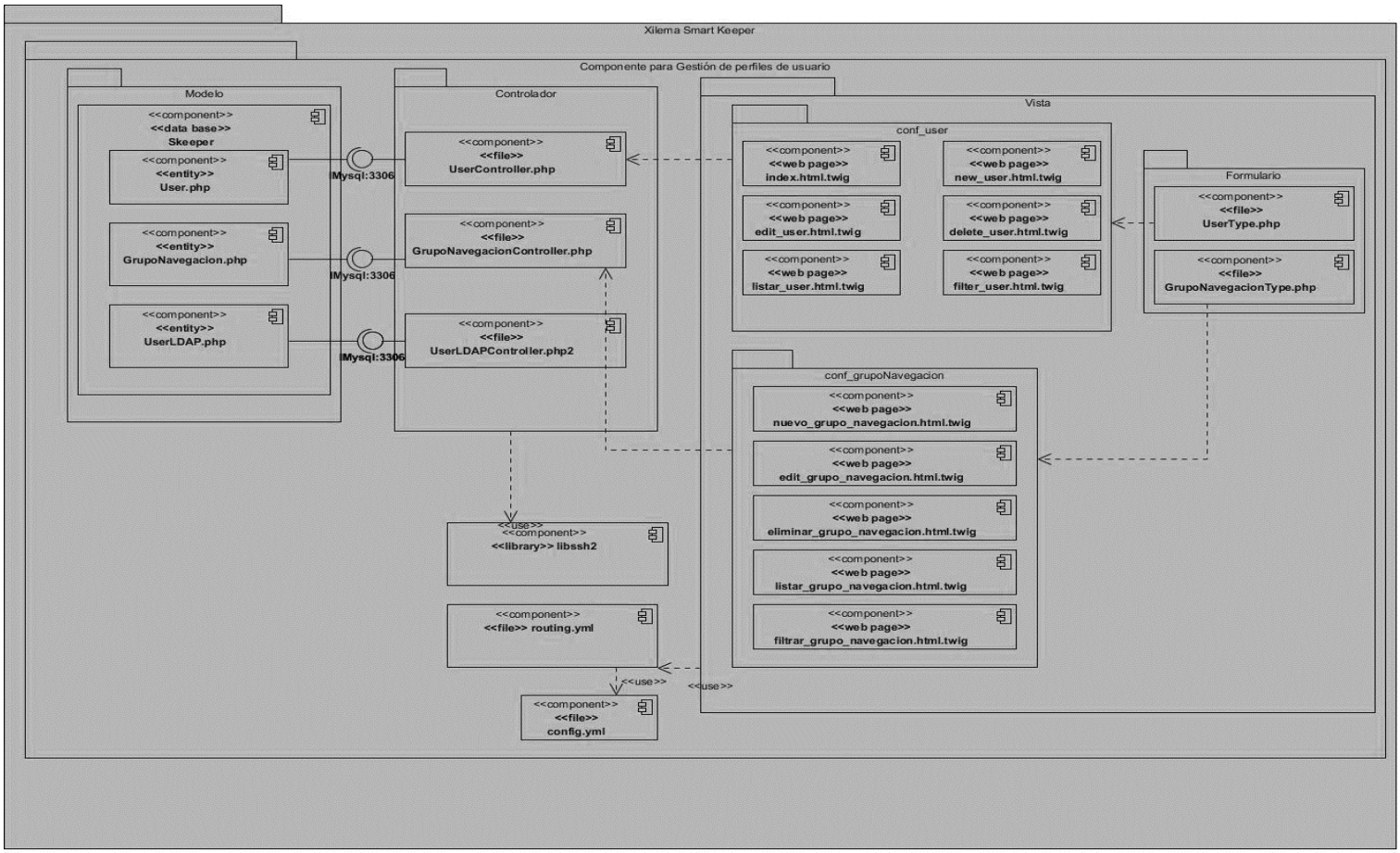


Ilustración 8:Diagrama de Componente

3.2 Estándar de Codificación.

Los estándares de codificación son un conjunto de convenciones para lograr uniformidad en el código fuente de un software. Permiten una mayor comprensión, modificación, calidad y mantenibilidad del código generado independientemente de su autor. A continuación de definen los estándares de codificación utilizados durante la implementación del componente, basándose en los utilizados por el lenguaje PHP para el marco de trabajo Symfony.

Tabla 5:Estándares de codificación

Tipo de Estándar	**Descripción**
Estructura	Añade un solo espacio después de cada delimitador coma
	Añade un solo espacio alrededor de los operadores (==, &&, ...)
	Añade una coma después de cada elemento del arreglo en un arreglo multilínea, incluso después del último
	Usa llaves para indicar la estructura del cuerpo de control, independientemente del número de declaraciones que contenga
Convenciones de nomenclatura	Utiliza mayúsculas intercaladas —sin guiones bajos— en nombres de variable, función, método o argumentos
	Usa guiones bajos para nombres de opción y nombres de parámetro
	Utiliza espacios de nombres para todas las clases
Documentación	Añade bloques *PHPDoc* a todas las clases, métodos y funciones
	Omite la etiqueta *@return* si el método no devuelve nada
	Las anotaciones *@package* y *@subpackage* no se utilizan
Licencia	Symfony se distribuye bajo la licencia MIT, y el bloque de la licencia tiene que estar presente en la parte superior de todos los archivos PHP, antes del espacio de nombres.

3.3 Estrategias de pruebas

El método de prueba que se decide aplicar al sistema es el método de Caja Negra, con el fin de estudiar la especificación de las funciones, la entrada y la salida para poder derivar los casos de prueba, definiendo como algo fundamental el probar todas las posibles entradas y salidas del sistema.

3.3.1 Pruebas Funcionales

Las pruebas funcionales tienen como objetivo validar que las funcionalidades implementadas cumplan con las especificaciones de los requisitos definidos. Para este tipo de prueba, el autor selecciona la técnica de Caja Negra, que según (Pressman, 2010) permite derivar un conjunto de condiciones de entrada, que revisarán por completo todos los requisitos funcionales de una aplicación con el objetivo encontrar errores de funciones incorrectas o ausentes, errores de interfaz, de comportamiento o rendimiento. Para aplicar las pruebas funcionales al componente se diseñaron doce (12) casos de pruebas (CP). A continuación, se muestran dos (2) casos de pruebas correspondiente a los requisitos funcionales uno (1) y dos (2). Los restantes casos de pruebas aparecen en los anexos.

Nº	Nombre del Campo	Clasificación	Valor Nulo	Descripción
1	Nombre	Campo de texto	No	Longitud mínima 4 , longitud máxima 50
2	Apellido	Campo de texto	No	Longitud mínima 4 , longitud máxima 50
3	Usuario	Campo de texto	No	Longitud mínima 4 , longitud máxima 50
4	Correo electrónico	Campo de texto	No	Longitud mínima 4 , longitud máxima 255, permite direcciones de correo estructuralmente validas (usuario@subdominios.dominios)
5	Contraseña	Campo compuesto(campo de texto y campo numérico)	No	Longitud mínima 6 , longitud máxima 50, permite todos los caracteres.
6	Roles	Campo de selección	No	Puede Seleccionar entre Administrador o usuario

Tabla 6:Descripción de las variables para el caso de Prueba 1

Caso de Prueba 1:SC RF2 Insertar Usuario									
Condiciones de Ejecución : El usuario no debe existir en el sistema									
Escenario	Descripción	1	2	3	4	5	6	Respuesta del sistema	Flujo Central
EC 1.1 Insertar usuarios. De forma correcta	El componente inserta un usuario de forma correcta	1	V	V	V	V	V	Muestra un mensaje de confirmación en la parte superior central. “El usuario se ha registrado con éxito”	En el menú superior del sistema: 1-El administrador accede a la opción Gestión. 2-Selecciona la opción Usuario 3-Selecciona la opción Usuarios 4-Selecciona la opción Adicionar 5-Introduce los nuevos valores y
		nardy	roman	nroman	nroman@estudiantes.uci.cu	*****	administrador		

									presiona el botón Guardar.
EC 1.2 Insertar usuarios dejando algún campo vacio.	El componente no inserta un usuario	V	V	I	I	V	V	Muestra un mensaje , “estos campos son obligatorios”	

Tabla 7:Caso de Prueba 1 Requisito funcional 2

No	Nombre del Campo	Clasificación	Valor Nulo	Descripción
1	Usuario	Campo de texto	No	Longitud mínima 4 , longitud máxima 50
2	Contraseña	Campo compuesto(Campo de texto y campo numérico)	No	Longitud mínima 6 , longitud máxima 50, permite todos los caracteres

Tabla 8:Descripción de las variables para el Caso de Prueba 2

Caso de Prueba 2 SC RF1_Autenticar Usuario					
Condiciones de Ejecución: El usuario debe estar autenticado como administrador en el sistema					
Escenario	Descripción	1	2	Respuesta del Sistema	Flujo Central
EC 2.1 Autenticar usuario de forma correcta	El componente autentica de forma correcta al usuario registrado como administrador.	V nroman	V Eldmonio	Accede al componente	1-Introducir el usuario y la contraseña en sus respectivos campos de texto. 2-Seleccionar la acción de entrar.

EC 2.2 Autenticar usuarios con campos incorrectos o vacios	El componente no autentica con campos incorrectos o vacios	I	I	Muestra un mensaje de error en la parte superior izquierda"Error de usuario o contraseñar. Vuelva a intentarlo "	

Tabla 9:Caso de Prueba 2 Requisito Funcional 1

Resultado de las pruebas funcionales.

La ejecución de las pruebas funcionales al componente dio como resultado en una primera iteración un total de diez (10) no conformidades, divididas en cuatro (4) de ortografía y seis (6) de funcionalidad. Las no conformidades encontradas fueron resueltas en su totalidad dando paso a la siguiente iteración. En la segunda iteración se encontraron siete (7) de funcionalidad, las cuales fueron resueltas. En la tercera iteración no se encontraron no conformidades. Las ejecuciones de estas pruebas son expuestas en la siguiente gráfica.

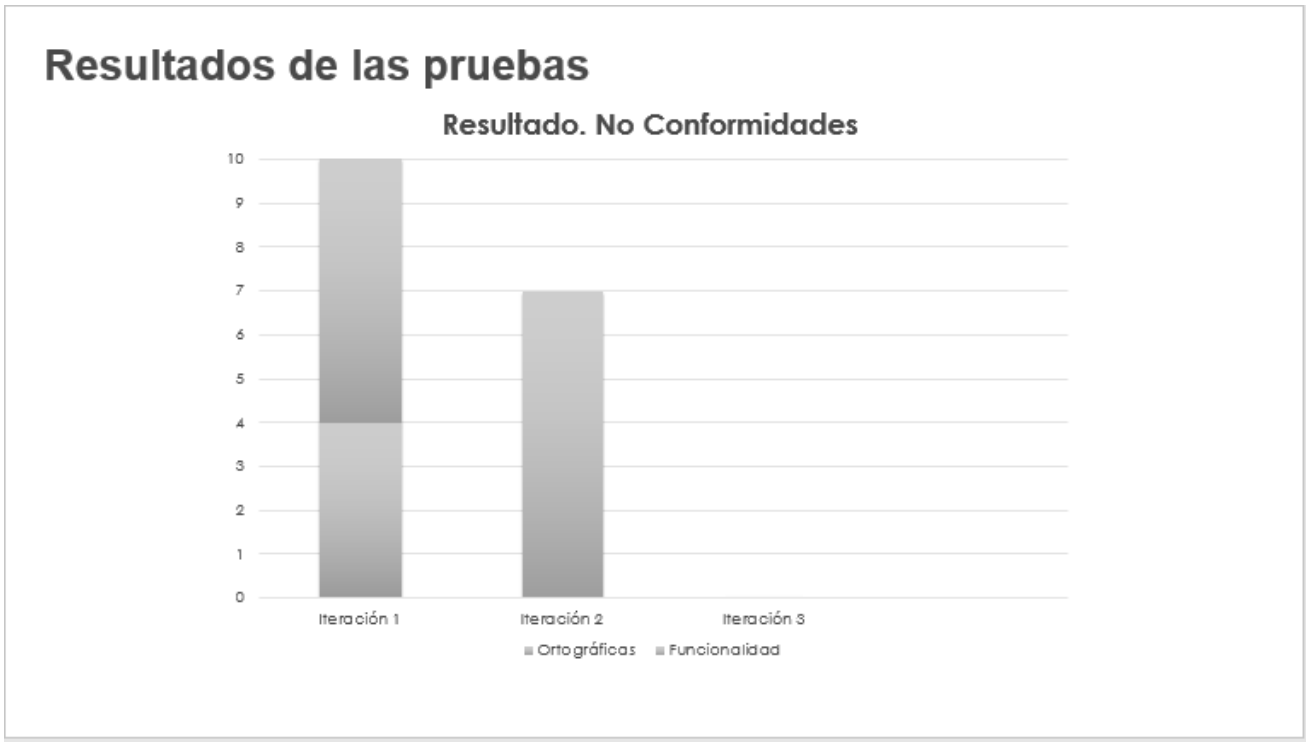


Ilustración 9: Resultados de las Pruebas Funcionales

Las no conformidades funcionales detectadas están relacionadas con la respuesta del sistema ante la introducción de valores incorrectos en los formularios del segundo CP, donde no se muestra el mensaje de error especificado. La presencia de esta no conformidad se debe a que los métodos de las clases controladoras que responden a estas acciones no enviaban un mensaje de respuesta a la vista por errores de validación, para resolverlo se incluyeron los mensajes antes mencionados al código de fuente.

3.3.2 Pruebas de Seguridad

Las pruebas de seguridad miden la confidencialidad, integridad y disponibilidad de la información, con el objetivo de identificar amenazas, riesgos y la probabilidad de enfrentarse a ataques informáticos que puedan descubrir y explotar las vulnerabilidades de la aplicación (Seguridad, 2016). Para realizar las pruebas de seguridad al componente, se utiliza la herramienta Acunetix Web Vulnerability Scanner 3.12, en una primera iteración, la herramienta realiza veinte y cinco (25) ataques informáticos, de los que se generan doce (12) alertas de vulnerabilidad, divididas en ocho (8) de nivel medio y cuatro (4) de carácter informacional. Las

alertas identificadas fueron corregidas en la primera iteración, para una segunda iteración, se realizan veinte (20) ataques y no se generan nuevas alertas.

Las alertas de nivel medio estuvieron relacionadas con doce (12) mensajes de error provocados por el modo de ejecución DEV para el desarrollo de aplicaciones en el marco de trabajo Symfony, de los que se puede obtener información sensible de los registros DEBUG del componente. Se identificaron dos (2) formularios HTML vulnerables a la falsificación de petición en sitios cruzados (CSFR, por sus siglas en inglés). Los formularios fueron comprobados de forma manual como recomienda la herramienta Acunetix y se identificó que no pueden ser utilizados por el atacante para alterar el sistema ni obtener información, se denomina estas vulnerabilidades como falsos positivos (Quality, 2016).

3.3.3 Pruebas de Integración

Las pruebas de integración aseguran que los distintos componentes de un software interactúan entre sí de forma correcta. Con el objetivo de validar la compatibilidad y el funcionamiento de las distintas interfaces que componen al Componente para la gestión de usuarios para el sistema Xilema Smart Keeper se toman acciones relacionadas con:

- La correcta validación de la sincronización de usuarios de un directorio activo LDAP. Caso de Prueba 3 RF 22: Sincronizar los usuarios del directorio activo.
- Verificación de la conexión entre el componente y la base de datos en *MySQL.*
- Validación de las rutas de acceso, en el *routing* principal del sistema Xilema Smart Keeper.
- Utilización de los *layout* del Bundle en el que se definen las vistas principales del sistema Xilema Smart Keeper (AdminBundle).

Mediante la ejecución de las pruebas de integración se logró identificar la operación conjunta de cada uno de los módulos pertenecientes al sistema Xilema Smart Keeper, junto al componente de gestión de usuarios, todos integrados dentro del bundle (AdminBundle) del sistema Xilema Smart Keeper y *layout* del bundle en el que se definen las vistas principales de dicho sistema (admin), haciendo énfasis en la interacción entre estos y comprobando el correcto funcionamiento de los mismos.

Caso de Prueba 3: SC RF22_Sincronizar los usuarios del directorio activo LDAP.			
Condiciones de ejecución: El usuario debe estar autenticado en el sistema.			
Escenario	**Descripción**	**Respuesta del sistema**	**Flujo Central**
EC 3.1. Sincronizar los usuarios del directorio activo LDAP	El sistema comprueba la conexión al directorio activo LDAP, luego muestra todos los usuarios encontrados en una lista	Muestra un mensaje de confirmación en la parte superior central: ”Sincronizados Correctamente”	En el menú superior del sistema: 1- El usuario accede a la opción Gestión. 2- El usuario selecciona la opción Usuario. 3- El usuario selecciona la opción Usuarios LDAP en la parte superior izquierda del panel principal. 4- El usuario selecciona la opción Sincronizar. 5- El sistema muestra un listado con todos los usuarios sincronizados para importarlos al componente

Tabla 10:Caso de Prueba 3 Requisito Funcional 22

En la aplicación de estas pruebas no se identificaron no conformidades, lo que valida que existe una correcta integración de las funcionalidades internas del Componente.

3.4. Conclusiones del capítulo

- El componente desarrollado es capaz de realizar las funcionalidades siguientes: gestionar los usuarios, grupos de navegación, configurar cuotas de navegación e importar usuarios de un directorio activo de forma dinámica; además permite filtrar y listar los resultados de los usuarios y grupos.
- La estrategia de prueba permitió validar el correcto funcionamiento del componente, una respuesta adecuada ante peticiones de gran carga y estrés, así como validar las funcionalidades ante códigos mal intencionados que puedan afectar la seguridad del mismo.
- El componente se integra correctamente con el resto de los módulos del sistema Xilema Smart Keeper.

Conclusiones

- Las bases teóricas de la investigación propiciaron establecer los elementos necesarios para solucionar el problema y asegurar que se realice una gestión de usuarios cumpliendo con las necesidades planteadas por los clientes.
- El diseño de la propuesta de solución, el modelo conceptual y la elaboración de los artefactos generados durante la fase de características de la propuesta de solución, sirvieron para implementar las funcionalidades que fueron identificadas.
- La ejecución de las pruebas funcionales y las pruebas de integración demostraron que el componente puede realizar todas sus funcionalidades sin errores y es capaz de integrarse a los módulos del sistema Xilema Smart Keeper.

Recomendaciones

Para futuras investigaciones se recomiendan las siguientes acciones:

- Desarrollar técnicas que permitan al componente editar directamente las características de los usuarios que se encuentran dentro de un directorio activo LDAP.

Referencias

MARTÍN, Adriana Elizabeth, et al. Identificación, desarrollo y uso de soluciones web centradas en el usuario. En *XVI Workshop de Investigadores en Ciencias de la Computación*. 2014.

Ordoñez Taco & Betancourt Cabezas, M. Á., C. L. (2016). Desarrollo e implementación del sistema de aplicaciones empresariales oracle con J2EE utilizando weblogic, autenticación de usuarios a aplicaciones con LDAP y gestión de versiones desplegadas en emerge-soft (Bachelor's thesis, Quito: UCE)

Instituto Puig Castellar (2018). Gestión de usuarios [Accedido el 18 de mayo del 2018]. Disponibles en https://elpuig.xeill.net/Members/vcarceler/c1/didactica/apuntes/ud4/na1

Significados (2018). Gestión [Accedido el 18 de mayo del 2018]. Disponibles en https://www.significados.com/gestion/

Moreno Boiza, Vanesa, Universidad de Carlos III, Madrid, Análisis y diseño de una plataforma web para un sistema de gestión de usuarios, [Accedido el 18 mayo 2018]. Disponibles en http://hdl.handle.net/10016/16046

OpenDNS Oficial 2018. Sistema de filtrado de contenido web [Accedido el 18 mayo 2018]. Disponible en: https:///www.opendns.com

Salix Networks Oficial 2014. Sistema de filtrado de contenido web [Accedido el 18 mayo 2018]. Disponible en https://www.salixnetworks.com/filtrado_web.html

UNIVERSIDAD de las Ciencias Informáticas. **Manual de usuario** Smart Keeper. La Habana: Universidad de las Ciencias Informáticas, 2014.

Ubuntu Oficial 2018 Canonical Ltd. Ubuntu and Canonical are registered trademarks of Canonical Ltd

Pomeyrol, J, 2017 Disponible en : http://wwww.muylinux.com/ubuntu/

Squid: servidor proxy-caché | Observatorio Tecnológico. [En Linea]. [Accedido el 7 marzo 2018].Disponible en: http://recursostic.educacion.es/observatorio/Web/ca/software/servidores/589-elviramifsud.

Carlos Beriso Cardaso, 2011. Proyecto fin de carrera. Universidad Rey Juan Carlos. España. Disponible en:

https://eciencia.urjc.es/bitstream/handle/10115/5538/Memoria%20PFC%20Carlos%20Beriso.pdf?sequence=1&isAllowed=y

Universidad Agraria de la Habana. Sistema integrado de gestión del capital humano. La Habana. Disponible en: http://www.gecyt.cu/redcapitalhumano/ponencias/p_68.pdf

OLIVEROS, Alejandro; DANYANS, Fernando J.; MASTROPIETRO, Matías L. Prácticas de Ingeniería de Requerimientos en el desarrollo de aplicaciones Web. En *WER*. 2014.

GONZÁLEZ, Enrique. ¿Qué es y para qué sirve HTML? *El lenguaje más importante para crear páginas webs. HTML tags (CU00704B). Septiembre*, 2016.

SEBASTIÁN, Francisco Javier Izquierdo. *Desarrollo Web del Grupo FluIng adaptado a los nuevos estándares HTML5 y CSS3*. 2015. Tesis Doctoral.

VIGOUROUX, Christian. *Aprender a desarrollar con JavaScript*. Ediciones ENI, 2015.

SYMFONY. *What is Symfony*. Symfony para programadores [En línea]. (2017). [Fecha de consulta: 8 de marzo de 2018]. Disponible en: http://symfony.com/what-is-symfony

NetBeans. [Fecha de consulta 18 de mayo de 2018]. Disponibles en https://netbeans.org/

PACHECO, Nacho. Bases de datos y Doctrine [En línea]. 22 de abril de 2013. [Fecha de consulta: 14 de marzo de 2018]. Disponible en: http://gitnacho.github.io/symfony-docs-es/book/doctrine.html

ORACLE. Getting Started with MySQL. 2017. Disponible en: https://dev.mysql.com/doc/mysql-getting-started/en/.

APACHE*. What is the Apache HTTP Server Project? Apache Software Fundation, 2017. Disponible en: https://httpd.apache.org/ABOUT_APACHE.html*

SOMMERVILLE, Ian. Ingeniería del software. 7ª ed. Madrid: Pearson Educación S.A, 2011. ISBN: 84-7829-074-5.

PRESSMAN, Roger S. Ingeniería del Software. Un enfoque práctico. 7ª ed. México, D.F: *The MacGraw-Hill Companies* Inc., 2010. ISBN: 978-607-15-0314-5.

Universidad Nacional Abierta y a Distancia UNAD. Disponible en: https://www.unad.edu.co/

REDONDO, Miguel J. González. *Estudio con modelos de datos para la automatización en redes eléctricas inteligentes*. 2016. Tesis Doctoral. Universidad de Córdoba.

SPARXSYSTEMS. Diagrama de Despliegue UML 2. [En línea] Sparx Systems-Tutorial UML 2-Diagrama de Despliegue, 2014. [Citado el: 15 de febrero de 2018] Disponible en: http://www.sparxsystems.com.ar/resources/tutorial/uml2_deploymentdiagram.html.

QUALITY, Pruebas de Seguridad. V&V *Quality*, Colombia. 2016. Disponible en: http://vyvquality.com/pruebas-seguridad/

Bibliografía

APACHE. What is the Apache HTTP Server Project? Apache Software Fundation, 2017. Disponible en: https://httpd.apache.org/ABOUT_APACHE.html.

ARIAS, Ángel. Bases de Datos con MySQL. 2015. ISBN: 978-1515194392

BOOCH, G., RUMBAUGH, J. and JACOBSON. El Lenguaje Unificado de Modelado. S.l.: Addison Wesley., [no date].

Burbeck, Steve. 1992. Application programming in Smalltalk-80: How to use Model-View-Controller (MVC).

CARRAZANA Galán, Deilis. Programa de mejora: Procedimiento para pruebas de seguridad [En línea]. La Habana: Universidad de las Ciencias Informáticas, (2 de marzo de 2017). [Fecha de consulta: 20 de marzo de 2018]. Disponible en: http://mejoras.prod.uci.cu/proceso_desarrollo_produccion/guidances/whitepapers/resources/Procedimiento%20de%20Pruebas%20de%20Seguridad.pdf.

Carlos Beriso Cardaso, 2011. Proyecto fin de carrera. Universidad Rey Juan Carlos. España. Disponible en: https://eciencia.urjc.es/bitstream/handle/10115/5538/Memoria%20PFC%20Carlos%20Beriso.pdf?sequence=1&isAllowed=y

Diccionario de Informática e Internet de Microsoft, Cuarta edición. [s.l], McGraw-Hill/Interamericana

El Lenguaje Unificado de Modelado. BOOCH, G., RUMBAUGH, J. and JACOBSON

Equiluz, Javier. 2011. Desarrollo Web Ágil con Symfony2

Fernández Cayo León, 2016

GONZÁLEZ, Matías; SOSA, Hernán; MARTIN, Adriana Elba. Sistemas de comunicación no verbales. Informes Científicos-Técnicos UNPA, 2014, vol. 6, no 2, p. 30-56.

GONZÁLEZ, Enrique. ¿Qué es y para qué sirve HTML? El lenguaje más importante para crear páginas webs. HTML tags (CU00704B). Septiembre, 2016.

Instituto Puig Castellar (2018). Gestión de usuarios [Accedido el 18 de mayo del 2018]. Disponibles en https://elpuig.xeill.net/Members/vcarceler/c1/didactica/apuntes/ud4/na1.com

JETBRAIN. Enjoy Productive PHP. 2017. Disponible en: https://www.jetbrains.com/phpstorm/

JQuery, 2013. Página Oficial de JQuery. [En línea]. 2013 [Citado el: 7 de marzo de 2018.] http://www.jquery.com

Larman, Craig. 1999. UML y Patrones. Introducción al Análisis y Diseño Orientado a Objetos. ISBN 970-170261-1. México: Addison Wesley.

MARTÍN, Adriana Elizabeth, et al. Identificación, desarrollo y uso de soluciones web centradas en el usuario. En XVI Workshop de Investigadores en Ciencias de la Computación. 2014.

Moreno Boiza, Vanesa, Universidad de Carlos III, Madrid, Análisis y diseño de una plataforma web para un sistema de gestión de usuarios, [Accedido el 18 mayo 2018]. Disponibles en http://hdl.handle.net/10016/16046

NetBeans. [Fecha de consulta 18 de mayo de 2018]. Disponibles en https://netbeans.org.com/

OpenDNS Oficial 2018. Sistema de filtrado de contenido web [Accedido el 18 mayo 2018]. Disponible en: https:///www.opendns.com

ORACLE. Getting Started with MySQL. 2017. Disponible en: https://dev.mysql.com/doc/mysql-getting-started/en/.

ORDOÑEZ, Hugo. Business Processes as a Strategy to Improve Requirements Elicitation in Extreme Programming. Colombia, Popayán: Memorias del VII Congreso Iberoamericano de Telemática CITA, 2015.

Ordoñez Taco & Betancourt Cabezas, M. Á., C. L. (2016). Desarrollo e implementación del sistema de aplicaciones empresariales oracle con J2EE utilizando weblogic, autenticación de usuarios a aplicaciones con LDAP y gestión de versiones desplegadas en emerge-soft (Bachelor's thesis, Quito: UCE).

OLIVEROS, Alejandro; DANYANS, Fernando J.; MASTROPIETRO, Matías L. Prácticas de Ingeniería de Requerimientos en el desarrollo de aplicaciones Web. En WER. 2014.

PACHECO, Nacho. Bases de datos y Doctrine [En línea]. 22 de abril de 2013. [Fecha de consulta: 14 de marzo de 2018]. Disponible en: http://gitnacho.github.io/symfony-docs-es/book/doctrine.html

Patrón Modelo-Vista-Controlador. Yenisleidy Fernández Romero1, Yanette Díaz González2. Disponible en

PÉREZ-WIESNER, MATEO, et al. El fenómeno de las redes sociales= The social networks phenomen: evolución y perfil del usuario= evolution and user profile. 2014.

PHP. ¿Qué es PHP? 7 de noviembre de 2017. Disponible en: http://php.net/manual/es/intro-whatis.php

PHP-FIG. PHP Framework Interop Group. 2018. Disponible en: https://www.php-fig.org/psr/

Pomeyrol, J, 2017 Disponible en : http://wwww.muylinux.com/ubuntu/

PRESSMAN, Roger S. Ingeniería del Software. Un enfoque práctico. 7ª ed. México, D.F: The MacGraw-Hill Companies Inc., 2010. ISBN: 978-607-15-0314-5.

Salix Networks Oficial 2014. Sistema de filtrado de contenido web [Accedido el 18 mayo 2018]. Disponible en https://www.salixnetworks.com/filtrado_web.html

QUALITY, Pruebas de Seguridad. V&V Quality, Colombia. 2016. Disponible en: http://vyvquality.com/pruebas-seguridad/

RAMOS Cardozzo, Daniel. Curso de Ingeniería de Software. IT Campus Academy, 2015.

RAMOS Cardozzo, Daniel. Desarrollo de Software: Estimación Requisitos y Análisis. 2ª ed. IT Campus Academy, 2016. ISBN: 9781530088614.

REAL ACADEMIA ESPAÑOLA Y. ASOCIACIÓN. ACADEMIAS DE LA LENGUA ESPAÑOLA (2014). Diccionario de la Lengua Española. Madrid: Santillana, 2016.

REDONDO, Miguel J. González. Estudio con modelos de datos para la automatización en redes eléctricas inteligentes. 2016. Tesis Doctoral. Universidad de Córdoba.

Sampieri, Roberto. Metodología de la investigación. 6ª ed. Punta Santa FE. ISBN: 978-1-4562-2396-0

SEBASTIÁN, Francisco Javier Izquierdo. Desarrollo Web del Grupo FluIng adaptado a los nuevos estándares HTML5 y CSS3. 2015. Tesis Doctoral.

SIERRA, F., et al. Estudio y análisis de los framework en php basados en el modelo vista controlador para el desarrollo de software orientado a la web. Revista Investigación y Desarrollo en TIC, 2017, vol. 4, no 2.

Significados (2018). Gestión [Accedido el 18 de mayo del 2018]. Disponibles en https://www.significados.com/gestion/

SOMMERVILLE, Ian. Ingeniería del software. 7ª ed. Madrid: Pearson Educación S.A, 2011. ISBN: 84-7829-074-5.

SPARXSYSTEMS. Diagrama de Despliegue UML 2. [En línea] Sparx Systems-Tutorial UML 2-Diagrama de Despliegue, 2014. [Citado el: 15 de febrero de 2018] Disponible en: http://www.sparxsystems.com.ar/resources/tutorial/uml2_deploymentdiagram.html.

Squid: servidor proxy-caché | Observatorio Tecnológico. [En Linea]. [Accedido el 7 marzo 2018]. Disponible en: http://recursostic.educacion.es/observatorio/Web/ca/software/servidores/589-elviramifsud.

SYMFONY. What is Symfony. Symfony para programadores [En línea]. (2017). [Fecha de consulta: 8 de marzo de 2018]. Disponible en: http://symfony.com/what-is-symfony

Ubuntu Oficial 2018 Canonical Ltd. Ubuntu and Canonical are registered trademarks of Canonical Ltd

UNAD. Lenguaje Unificado de Modelado UML. Diagrama de Clases de Diseño [En línea]. Universidad Nacional Abierta y a Distancia, (2016). [Fecha de consulta: 1 de marzo de 2018]. Disponible en: http://stadium.unad.edu.co/ovas/10596_9836/diagrama_de_clases_de_diseo.html

UNIVERSIDAD de las Ciencias Informáticas. Manual de usuario Smart Keeper. La Habana: Universidad de las Ciencias Informáticas, 2014.

Universidad Agraria de la Habana. Sistema integrado de gestión del capital humano. La Habana. Disponible en: http://www.gecyt.cu/redcapitalhumano/ponencias/p_68.pdf

Universidad Nacional Abierta y a Distancia UNAD. Disponible en: https://www.unad.edu.co/

VÁSQUEZ, Augusto Cortez; FERNÁNDEZ, Cayo León. Aprendizaje de perfiles de usuario web para modelizar interfaces adaptativas. Theorēma (Lima, Segunda época, En línea), 2016, no 3, p. 155-164.

Visconti, Marcello; Astudillo, Hernán. 2012. Fundamentos de Ingeniería de Software. Universidad Técnica Federico Santa María

VIGOUROUX, Christian. Aprender a desarrollar con JavaScript. Ediciones ENI, 2015.