Universidad de las Ciencias Informáticas

Facultad 6

Interfaz gráfica de usuario para la configuración del componente Seguridad del marco de trabajo Bosón.

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

Autor

Yaima Zulueta Saladrigas

Tutores

Ing. Claudia Bravo Batista

Ing. Julio Cesar Ocaña Bermúdez

La Habana, Julio 2016

"Año 58 de la Revolución"

Mira que te mando que te esfuerces y seas valiente;
no temas ni desmayes, porque Jehová tu Dios estará
contigo en dondequiera que vayas.

Josué 1:9

Cuando quieras emprender algo,
habrá mucha gente que te dirá que no lo hagas
cuando vean que no te pueden detener,
te dirán como lo tienes q hacer,
y cuando finalmente vean que lo has logrado,
dirán que siempre creyeron en ti.

John C. Maxwell

DECLARACIÓN DE AUTORÍA

Yo, Yaima Zulueta Saladrigas declaro ser la autora principal de la presente tesis titulada: ***Interfaz gráfica de usuario para la configuración del componente Seguridad del marco de trabajo Bosón*** y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente declaración jurada de autoría en La Habana a los ____ días del mes de _____ del año _____.

Autor

_____

Yaima Zulueta Saladrigas

Tutor

_____

Claudia Bravo Batista

Tutor

_____

Julio César Ocaña Bermúdez

AGRADECIMIENTOS

Agradezco a mi Creador, mi Rey, El Todopoderoso e Invencible Jesucristo. ***TODO SE LO DEBO A ÉL****. Mi Dios, quien me da fuerzas y aliento para no desmayar, quien pelea mis batallas cada día, quien me hace fuerte en mi debilidad, quien me levanta cuando caigo, quien me exalta cuando me humillo. A ti mi Señor, que me perdonas y me corriges con amor. GRACIAS*

A mi "mami", la mujer 5 Estrellas en mi vida, mi amiga incondicional. Eres la madre perfecta, aun cuando me reprendes, pero nadie lo hace mejor que tú. Gracias por hacer de mi lo que soy. Gracias por existir.

A Vitico, el esposo de la mujer 5 Estrellas, quien me ha soportado desde mis 9 años. Se dice llamar el jefe y dueño de todo, pero de lo que estoy segura es que Dios no pudo haber puesto mejor padre para mí. Gracias Viti.

Agradezco a Yaiko, mi especial hermano en Cristo, mi novio y futuro esposo, alias "mi amorcito". La pieza que faltaba en mi vida. Gracias por tu amor y dedicación. Por comprenderme en mis días malos y siempre tener la palabra exacta para hacerme ver todo diferente.

Mil agradecimientos a mis tutores Claudia y Julio, por su paciencia, dedicación y apoyo incondicional para realizar este trabajo. Si tuviera que escoger otros tutores, sin dudas les escogería nuevamente.

Gracias también a Castaño, Elennis y Yinet por ser el tribunal perfecto para crecer.

Agradezco a mi oponente Ana María por darme de su tiempo siempre que lo necesité.

Gracias a Mirtha y Rene, a Tony, Sule y Roxy, a mi amigo Raydel, a Dainela, a Zeida, Vladimir, Leticia y Elvira. A todos mis amigos y hermanos en la Fe, que siempre han estado al tanto del proceso de mi tesis. Gracias por sus oraciones.

Quiero agradecer también a mi amiga Susy y a Ángela, dos hermanas esforzadas y valientes. Dios les multiplique siempre lo que han hecho por mí.

A personas importantes como mi papá, Nuria, Evaristo, mi prima Lily, gracias por su apoyo en momentos bien marcados en mi vida.

A todos los profesores que he tenido en el transcurso de mi vida como estudiante, porque han contribuido en mi formación profesional.

A mis amigos y compañeros de aula Sayuri y Pepe, gracias por su amistad, los voy a extrañar mucho.

A todos, Muchas gracias

DEDICATORIA

Dedico este trabajo a una personita muy especial en mi vida: mi hermanita Laura. Quiero que te sea de inspiración y ejemplo para que estudies y te prepares en la vida, como profesional y como mujer sabia, esforzada y valiente. A ti que eres una de las bendiciones más grandes que he recibido de parte de Dios y nuestra madre. Eres a quien amo y cuido cada día aunque no me veas y aunque no te guste. Eres ese pedacito de mí que no me puede faltar.

RESUMEN

La seguridad es un factor fundamental para cualquier sistema, por lo que es necesario que estos posean una política de seguridad adecuada para lograr un buen desempeño de sus objetivos. En la actualidad existen sistemas que gestionan en menor o mayor medida la seguridad. Entre ellos se encuentran los marcos de trabajo, diseñados para ayudar en la construcción de aplicaciones con rapidez mediante la reutilización de componentes y módulos genéricos.

En la Universidad de las Ciencias Informáticas, la Dirección General de Proyectos le asignó al Departamento de Desarrollo de componentes del Centro de Informatización de Entidades (CEIGE) la tarea de crear el marco de trabajo Bosón. Éste es un conjunto de componentes desarrollados sobre Symfony2 que responden a la mayoría de los requisitos tecnológicos de un amplio espectro de aplicaciones. En su estructura incluye varios componentes, entre ellos el componente Seguridad que redefine la seguridad de Symfony2 para gestionar la autenticación y la autorización. Dicho componente gestiona sus funcionalidades a través de comandos o servicios y no cuenta con una interfaz gráfica de usuario (GUI en inglés) que facilite su configuración.

El objetivo de este trabajo es desarrollar la interfaz gráfica de usuario para mejorar la usabilidad del componente Seguridad del marco de trabajo Bosón, cumplir con los estándares web internacionales, así como con los requerimientos de los usuarios.

PALABRAS CLAVE: Bosón, Componente, GUI, Marco de trabajo, Seguridad, Usabilidad

ABSTRACT

Safety is a key factor for any system, so it is necessary that these possess a policy of adequate security to achieve good performance of its objectives. At present there are systems that manage a greater or lesser extent safety. Among them are frameworks designed to help build applications quickly by reusing generic components and modules.

At the University of Information Science, the general direction of projects assigned to the Components Development Department of Center for Entities Informatization (CEIGE in Spanish) the task of creating the Boson framework. This is a set of components developed on Symfony2 that match most of the technological requirements of a wide range of applications. Its structure includes several components, including the Security Component that redefines security Symfony2 to manage authentication and authorization. This component manages its functionality through commands or services and does not have a graphical user interface (GUI) to facilitate configuration.

The aim of this work is to develop the graphical user interface to improve usability of the Security Component of Boson Framework, to comply with international web standards and the requirements of users.

KEYWORDS: Boson, Component, GUI, Framework, Security, Usability

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

En la actualidad en consecuencia con el avance tecnológico se diseñan interfaces gráficas para que las aplicaciones informáticas sean más intuitivas para los usuarios y muestren diseños sencillos y fáciles de administrar. La interfaz gráfica son todos los elementos gráfico que nos ayudan a comunicarnos con un sistema (González, 2004). De esta forma, no se restringe el uso de las aplicaciones solamente a las personas con conocimientos de informática, sino para un amplio grado de usuarios, de manera que sea más fácil la interacción entre el usuario y el ordenador, por lo que surgen las Interfaces Gráficas de Usuario (GUI[1]).

Las GUI permiten representar la información procesada por la computadora de manera visual haciéndola comprensible, didáctica y usable para el usuario. Por tal motivo, se debe lograr que un producto informático tenga un diseño de interfaz gráfica fácil de aprender, de usar, para que el usuario pueda de manera eficiente completar las tareas y conseguir objetivos específicos.

En Cuba, la Universidad de las Ciencias Informáticas (UCI) para organizar la producción, ha instaurado una red de centros de desarrollo de software que tienen como meta fundamental la creación de productos, servicios y soluciones informáticas. En el departamento de Desarrollo de componentes del Centro de Informatización de Gestión de Entidades (CEIGE) perteneciente a la facultad 3, está en desarrollo el marco de trabajo Bosón que materializa los requisitos comunes de las arquitecturas bases para los sistemas de gestión.

Uno de los componentes desarrollados en Bosón es el componente SeguridadBundle que permite proveer las funcionalidades básicas de autenticación y autorización basadas en los estándares propuestos por SAML[2], la implementación de SAML utilizada es SimpleSamlphp. El componente Seguridad permite que se puedan gestionar los usuarios del sistema y su autorización, mediante el paquete de autorización que valida si las credenciales mostradas por el usuario conectado son suficientes para acceder a determinado recurso del proyecto.

Para la gestión de los recursos el componente brinda funcionalidades, las cuales podrán ser accedidas a través del servicio *boson.manager.recursos,* de la misma forma se gestionan los permisos a través del servicio *boson.manager.permisos y las funcionalidades a través de boson.manager.funcionalidades.* Por último, para promover un usuario a un rol específico se emplea la siguiente línea de comando '*bash boson:seguridad:promover-usuario usuario ROLE_ADMIN*' (Amat, 2015).

[1] **GUI**: Por sus siglas en inglés *Graphics User Interface*

[2] **SAML:** Por sus siglas en inglés *Security Assertion Markup Language*, Lenguaje de Marcado para Confirmaciones de Seguridad, es un estándar abierto que define un esquema XML para el intercambio de datos de autenticación y autorización.

Como se describe anteriormente, la gestión de las funcionalidades que provee el componente Seguridad se realiza a través de servicios o por comandos, lo que representa que el tiempo de aprendizaje para interactuar con el componente sea de 24 horas. Aunque internamente el componente funciona de manera correcta, no existe una interfaz gráfica intuitiva que los usuarios puedan utilizar para obtener la información deseada, esto implica que solamente usuarios con conocimientos de informática puedan configurar las funcionalidades.

Después de analizar la problemática, se identifica como **problema a resolver**: ¿Cómo mejorar la usabilidad del componente Seguridad del marco de trabajo Bosón?

Se plantea como **objeto de estudio** el proceso de desarrollo de las GUI en las aplicaciones web centrado su **campo de acción** en el proceso de desarrollo de las GUI del componente Seguridad en el marco de trabajo Bosón.

Para darle solución al problema anterior se plantea como **objetivo general**: Desarrollar la interfaz gráfica de usuario para la configuración del componente Seguridad del marco de trabajo Bosón.

Objetivos específicos

1. Fundamentar la metodología, herramientas y las tecnologías a utilizar en el desarrollo de las GUI.
2. Analizar y diseñar la propuesta de las GUI del componente Seguridad en el marco de trabajo Bosón.
3. Implementar la propuesta de las GUI del componente Seguridad del marco de trabajo Bosón.
4. Validar la solución propuesta.

Para cumplir con estos objetivos se desglosan las siguientes **tareas de la investigación**:

1. Análisis del componente Seguridad en marcos de trabajo existentes de ámbito nacional e internacional para conformar el estado del arte de la investigación.
2. Análisis de tecnologías existentes y tendencias actuales en el diseño de las GUI para obtener características que permitan diseñar la solución propuesta.
3. Selección de la metodología, tecnologías y herramientas para el desarrollo de la solución propuesta.
4. Diseño de las GUI del componente Seguridad del marco de trabajo Bosón para modelar el sistema de manera que soporte los requisitos funcionales y no funcionales.
5. Implementación de las GUI del componente Seguridad del marco de trabajo Bosón para mejorar la usabilidad del sistema y dar solución a los requerimientos.
6. Realización de las pruebas para validar la solución propuesta.
7. Validación de la investigación mediante un diseño pre experimental para validar la variable dependiente usabilidad.

Para dar cumplimiento a las tareas de la investigación se utilizaron los siguientes métodos científicos de investigación.

Métodos teóricos (Ortiz, 2012)

- Histórico – lógico: Este método posibilita conocer los antecedentes y tendencias actuales del desarrollo de las GUI en aplicaciones web y así seleccionar las tecnologías que se ajustan para dar solución de la problemática planteada.
- Analítico – sintético: Este método es utilizado para analizar con profundidad la documentación existente relacionada con el diseño de las GUI en los marcos de trabajo y cómo se comporta y visualiza el componente Seguridad en los mismos, análisis determinante para la usabilidad.

Métodos empíricos (Ortiz, 2012)

- Observación: Este método beneficia la obtención de información al examinar objetivamente el comportamiento actual y en tiempo real del componente Seguridad en el marco de trabajo Bosón en cuanto a la usabilidad del mismo.
- Entrevistas: Se realizaron entrevistas estructuradas basadas en preguntas cerradas a especialistas del Departamento de Desarrollo de componentes del CEIGE. Fueron necesarias para el entendimiento y funcionamiento del componente y permitieron recopilar información nueva y completar la adquirida con la observación.

Estructura de los capítulos de la Tesis

Capítulo I Fundamentación teórica del proceso de desarrollo de interfaces gráficas de usuarios del componente Seguridad: Se presentan los elementos teóricos asociados al dominio del problema que se emplearán a lo largo de la investigación. Se realiza un análisis de las tecnologías que se utilizan para el diseño y creación de las GUI en aplicaciones web. De igual manera se describen, valoran y justifican las tecnologías y herramientas escogidas para la solución, así como la metodología a utilizar para guiar el proceso de desarrollo.

Capítulo II Descripción de las interfaces gráficas de usuario del componente Seguridad: Se exponen los requisitos funcionales y no funcionales definidos para el componente Seguridad del marco de trabajo Bosón. Se realiza la obtención, descripción y validación de los requisitos, el análisis y diseño del componente Seguridad con las GUI, según las disciplinas definidas en la metodología de desarrollo empleada. Se explica la utilización de los patrones arquitectónicos y de diseño aplicados en la solución.

Capítulo III Implementación y validación de las interfaces gráficas de usuarios del componente Seguridad: Comprende todo lo referente a la implementación del sistema. Se presentan los resultados de las pruebas realizadas a las GUI que se crearon para garantizar la usabilidad. Además, se valida mediante un pre experimento la variable dependiente usabilidad definida en el problema a resolver.

CAPÍTULO I FUNDAMENTACIÓN TEÓRICA DEL PROCESO DE DESARROLLO DE INTERFACES GRÁFICAS DE USUARIO DEL COMPONENTE SEGURIDAD.

Ninguna investigación debe tomarse en serio sin una exploración y posterior documentación de la información necesaria para llevarla a cabo, este capítulo es una guía que facilita la comprensión de los principales conceptos asociados al dominio del problema durante el desarrollo de la investigación. Se realiza un estudio sobre las tendencias actuales de las GUI, clasificaciones y características. También se describe la metodología de desarrollo, tecnologías, lenguajes y herramientas a utilizar en la solución propuesta.

1.1 Conceptos fundamentales

1.1.1 Interfaz Gráfica de Usuario

Las GUI surgen dada la necesidad de facilitar el uso de los ordenadores para todo tipo de usuarios, y así ofrecer un entorno visual sencillo. Han llegado a convertirse en una habitual práctica para que la interacción del usuario final con el ordenador sea amigable.

Interfaz gráfica de usuario en términos de informática ha sido definida como un tipo de entorno que representa en la pantalla programas como archivos y opciones por medio de íconos como menús y cuadros de diálogos. El usuario puede seleccionar y activar estas opciones cuando apunta y hace clic con el ratón o, frecuentemente, con el teclado. Un elemento particular (tal como una barra de desplazamiento) trabaja de igual forma para el usuario en todas las aplicaciones, pues la interfaz gráfica de usuario proporciona rutinas de software estándar; las aplicaciones invocan a estas rutinas con los parámetros específicos en lugar de intentar reproducirlas a partir de la nada (Moreno, 2005).

1.1.2 Diseño de interfaces gráficas de usuarios

El diseño elegido para un sistema debe ser el apropiado para la comunidad de usuarios a la que va dirigido y las tareas que van a realizar. Un mismo diseño puede ser apropiado para algunos usuarios y poco apropiado para otros, eficiente para algunas tareas e ineficiente para otras. Lo típico es que los usuarios finales se sientan cómodos y seguros, es decir, que sientan que pueden controlar el sistema.

El objetivo del diseño de las GUI es definir un conjunto de objetos y acciones de interfaz que posibiliten al usuario llevar a cabo todas las tareas definidas de forma que cumplan todos los objetivos de usabilidad definidos por el sistema (Pressman, 2002).

Actualmente, el proceso del desarrollo de una interfaz consta de 4 etapas iterativas. Mientras se disponga de tiempo, se deben de hacer tantos ciclos de mejoramiento como sea posible.

La espiral que se muestra en la Fig. 1 significa que cada una de las tareas anteriores aparecerán más de una vez, en donde a medida que se avanza por la espiral se representará la elaboración adicional de los requisitos y el diseño resultante.

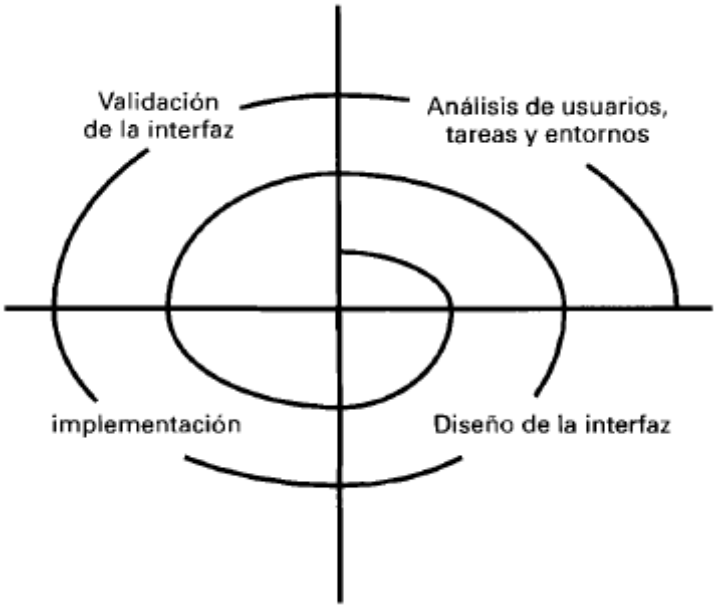


Fig. 1 Proceso de diseño de la interfaz de usuario
Fuente: Ingeniería del Software. Un Enfoque práctico

La actividad de análisis inicial se concentra en el perfil de los usuarios que van a interactuar con el sistema. Se registran el nivel de conocimiento, la comprensión del negocio y la receptividad general del nuevo sistema, y se definen diferentes categorías de usuarios. Una vez definidos los requisitos generales, se analizan, identifican, describen y elaboran las tareas que debe realizar el usuario para conseguir los objetivos. La actividad de implementación comienza normalmente con la creación de un prototipo que permita evaluar los escenarios de utilización. La validación se centra en la habilidad de la interfaz para implementar correctamente todas las tareas del usuario y archivar todos sus requisitos generales; el grado de facilidad de utilización y aprendizaje de la interfaz y la aceptación de la misma por parte del usuario como una herramienta útil en su trabajo.

1.1.3 Usabilidad

El objetivo del análisis de este concepto es caracterizar la usabilidad como elemento de la calidad, junto con sus atributos. Usabilidad se define en el estándar ISO 9241 como *"el grado en el que un producto puede ser utilizado por usuarios específicos para conseguir objetivos específicos con efectividad, eficiencia y satisfacción en un determinado contexto de uso"* (ISO, 1998).

En el estándar ISO/IEC 25010, dentro del cual se identifican características de la calidad del software entre ellas la usabilidad, ésta se define como: *"la capacidad de un producto de software para ser entendido, aprendido, utilizado y atractivo hacia el usuario, cuando se usa bajo condiciones específicas"* (Pupo, 2012).

A partir de las definiciones anteriores se concluye que el funcionamiento correcto y eficiente de un software, como la facilidad de aprender, recordar y usar, son características y cualidades significativas para que un sistema sea usable. Según la referencia de la calidad de la universidad, el Centro Nacional de Calidad de Software (CALISOFT) descompone la usabilidad en los siguientes atributos:

- **Facilidad de aprendizaje**: Indica cuan fácil es aprender las funcionalidades básicas de un sistema como para hacer correctamente la tarea que se desea realizar.
- **Eficiencia**: Se mide por el número de transacciones por unidad de tiempo que el usuario puede realizar cuando usa el sistema. Lo que se busca es la máxima velocidad de realización de tareas del usuario. Cuanto mayor es la usabilidad de un sistema, más rápido es el usuario al utilizarlo, y el trabajo se realiza con mayor rapidez.
- **Recuerdo en el tiempo**: Para usuarios que no utilizan el sistema regularmente es vital ser capaces de usarlo sin tener que aprender cómo funciona cuando comienzan de cero cada vez.
- **Tasa de errores**: Este atributo contribuye de forma negativa a la usabilidad de un sistema. Se refiere al número de errores cometidos por el usuario mientras realiza una determinada tarea. Un buen nivel de usabilidad implica una tasa de errores baja. Los errores reducen la eficiencia y satisfacción del usuario, y pueden verse como un fracaso en la transmisión al usuario del modo de hacer las cosas con el sistema.
- **Satisfacción**: Muestra la impresión subjetiva que el usuario obtiene del sistema.

1.2 Componente Seguridad del Marco de trabajo Bosón

El componente SeguridadBundle del marco de trabajo Bosón es el encargado de proveer las funcionalidades básicas de autenticación y autorización basadas en los estándares propuestos por SAML (Amat, 2015).

1.2.1 Autorización de usuarios

En el componente de Seguridad, el paquete de autorización es el encargado de validar si las credenciales mostradas por el usuario conectado son suficientes para acceder a determinado recurso del proyecto. El modelo utilizado es basado en roles, los que son almacenados en base de datos y asociados a cada uno de los usuarios en una relación de muchos a muchos (Amat, 2015)**.**

1.2.2 Recursos

Para la gestión de los recursos el componente brinda una serie de funcionalidades las cuales podrán ser accedidas a través del servicio *boson.manager.recursos*. La sincronización de las rutas creadas mediante el servicio *boson:seguridad:sincronizar-rutas*, constituye el primer paso para el proceso de autorización (Amat, 2015).

1.2.3 Permisos

La interacción con los permisos se puede hacer a través del servicio *boson.manager.permisos* y *bash boson:seguridad:registrar-permiso* los cuales respectivamente permiten adicionar y registrar un nuevo permiso a partir de la especificación de la firma del recurso que se desea proteger y el nombre del rol al que se le otorgará y/o denegará el acceso. En caso que se desee denegar permisos a un rol sobre determinado recurso solamente se debe especificar el parámetro *–denegar* (Amat, 2015).

1.2.4 Roles

Los roles son los diferentes perfiles que puede adoptar un usuario y están definidos por privilegios o capacidades que al final definirán las tareas que puede realizar el usuario. En Bosón para promover un usuario a un rol especificado se emplea la línea de comando *bash boson:seguridad:promover-usuario usuario ROLE_ADMIN* (Amat, 2015)*.*

1.2.5 Funcionalidades

Por último, las funcionalidades que luego se mostrarán en el Portal se harán a través del servicio *boson.manager.funcionalidades.* (Amat, 2015)

1.3 Análisis de las tecnologías

El objetivo de este sub-epígrafe es conformar el estado del arte de la investigación, donde se seleccionan y comparan marcos de trabajo de ámbito nacional e internacional que incluyen en su arquitectura un componente Seguridad. Se estudia y explica brevemente cómo cada uno gestiona los usuarios, roles y funcionalidades. Además se investiga si alguno tiene un modelo de componente implementado que desarrolle GUI para el componente Seguridad.

1.3.1 Marcos de trabajo PHP

Los marcos de trabajo son herramientas que proveen funcionalidades genéricas las cuales se utilizan en el desarrollo de aplicaciones de manera rápida, fácil, modular y sencilla para ahorrar tiempo y esfuerzo.

En las siguientes tablas se describen diferentes marcos de trabajo implementados en el lenguaje PHP y que incluyen en su arquitectura un componente de Seguridad. Se analizaron el diseño de sus interfaces, así como la manera en que gestionan y configuran la información del componente Seguridad, con el objetivo de adquirir características similares que permitan dar respuesta al objetivo general de la investigación.

Funcionalidad gestionar usuarios del componente Seguridad

Tabla 1. Funcionalidad gestionar usuarios
Elaboración propia

Marcos de trabajo	Descripción	Interfaz gráfica
Sauxe	Brinda servicios para la autorización, gestión de perfil y comprobación de rangos de IP[3]. Cada usuario tendrá un perfil asociado que podrá ser configurado cuando se desee. Tiene implementado 1 tipo de autenticación que utiliza usuario y contraseña.	Si. Ver ***Anexo1***
Bosón	Valida si las credenciales mostradas por el usuario conectado son suficientes para acceder a un determinado recurso del proyecto.	No tiene
Zend framework	Utiliza *Zend_Auth* y *Zend_Acl* para autenticación y autorización de usuarios respectivamente. Son responsables de la configuración de las listas de control de acceso, autentificación de usuarios y manejo de sesiones.	No tiene
Yii	Utiliza *Yii-User*, *Yii-User-Manager* como módulos de gestión de usuarios para las cuentas de los usuarios de registro y de gestión. *yii-user-management* para la administración, registro de usuarios e ingreso a través de perfiles.	No tiene
Codeigniter	*Ion-Auth* permite crear grupos de usuarios y un sistema de autentificación basado en el almacenamiento de las contraseñas y edición de usuarios.	No tiene

[3] **IP**: Por sus siglas en inglés *Internet Protocol*. Es un protocolo no orientado a conexión usado tanto por el origen como por el destino para la comunicación de datos a través de una red de paquetes conmutados.

Funcionalidad gestionar roles del componente Seguridad

Tabla 2. Funcionalidad gestionar rol
Elaboración propia

Marcos de trabajo	Descripción	Interfaz gráfica
Sauxe	Para acceder al sistema los usuarios deben tener al menos un rol en alguna de las entidades del dominio que le fue asignado al crearse. Permite especificar los privilegios que va a tener un rol determinado sobre acciones específicas.	Si. Ver ***Anexo2***
Bosón	Los roles están definidos por privilegios que especifican las tareas que puede realizar el usuario. Para promover un usuario a un rol se emplea la línea de comando *bash boson:seguridad:promover-usuario.*	No tiene
Zend framework	El componente *Zend_Acl* permite la administración de privilegios e implementar una lista de control de acceso. Incluye una implementación básica para Roles y Recursos. Soporta herencia de roles. Permite asignar roles los cuales pueden ser personalizados a través de las clases definidas por el usuario.	No tiene
Yii	El componente *AuthManager* proporciona una API[4] (*CAuthManager::createRole*) para asignar roles a los usuarios. Para esto se llaman individualmente los métodos *CAuthManager::assign* y *CAuthManager::revoke*	No tiene
Codeigniter	El componente *Ion-Auth* además de la autentificación permite crear roles.	No tiene

[4] **API**: por sus siglas en inglés *Application Programming Interface*. Es el conjunto de funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción

Funcionalidad gestionar funcionalidades del componente Seguridad

Tabla 3. Funcionalidad gestionar funcionalidad
Elaboración propia

Marcos de trabajo	Descripción	Interfaz gráfica
Sauxe	El módulo Configurar sistemas permite restringir el acceso a las funcionalidades de cada uno de los sistemas o subsistemas, para ello debe mostrársele al usuario todos los sistemas registrados para que gestione las funcionalidades de cada uno de ellos.	Si. Ver ***Anexo3***
Bosón	Las funcionalidades se gestionan a través del servicio *boson.manager.funcionalidades.*	No tiene
Zend framework	Utiliza *Zend_ACL* para asignar permisos a usuarios o grupos de usuarios.	No tiene
Yii	El componente *AuthManager* proporciona una API para asignar tareas y permisos a un rol determinado: *CAuthManager::createTask* y *CAuthManager::createOperation.*	No tiene
Codeigniter	Se desconoce información para este aspecto.	

Funcionalidad Proveedor de identidad del componente Seguridad

Tabla 4. Funcionalidad Proveedor de identidad
Elaboración propia

Marcos de trabajo	Descripción	Interfaz gráfica
Sauxe	Proporciona el uso del estándar SAML y la implementación de la arquitectura SSO[5].	No tiene
Bosón	Utiliza el estándar SAML 2.0 para la comunicación y el patrón SSO para la autenticación.	No tiene
Zend framework	Utiliza el patrón *Singletón* y utiliza credenciales OpenID para la autenticación.	No tiene
Yii	Utiliza el patrón SSO con el estándar SAML 2.0	No tiene
Codeigniter	Utiliza el patrón SSO con el estándar OAuth.	No tiene

[5] **SSO** por sus siglas en inglés *Single sign-on.*

Funcionalidad Proveedor de servicios del componente Seguridad

Tabla 5. Funcionalidad Proveedor de servicios
Elaboración propia

Marcos de trabajo	Descripción	Interfaz gráfica
Sauxe	Permite la comunicación entre componentes y expresa sus contratos en servicios mediante el formato WSDL[6].	No tiene
Bosón	El componente *Integrator* permite la generación dinámica de servicios REST[7]. Resuelve en un entorno de varias aplicaciones qué servicios pueden satisfacer las dependencias de otras aplicaciones.	No tiene
Zend framework	Proporciona soporte para servicios web que permite a una aplicación hacer uso de servicios ofrecidos por aplicaciones externas. *Zend service* sirve de base para las implementaciones de servicios Web, como SOAP[8] o REST. Utiliza *Zend_Rest_Client* para los servicios web basados en REST.	No tiene
Yii	Yii genera para cada clase una especificación de WSDL. Proporciona *CWebService* y *CWebServiceAction* para simplificar el trabajo de implementación de servicios Web en una aplicación Web.	No tiene
Codeigniter	Utiliza la biblioteca *CodeIgniter Rest Server* para trabajar con REST y *CodeIgniter Rest Client* si deseas consumir servicios. La biblioteca *nuSoap* para la implementación de servicios con SOAP.	No tiene

En el análisis realizado entre los marcos de trabajo seleccionados, se concluye que cada uno gestiona su seguridad de formas diferentes, basándose en requisitos específicos. También se observa que sólo el marco de trabajo Sauxe presenta GUI para la configuración del componente Seguridad, lo que permite obtener características útiles para el diseño de las GUI de marco de trabajo Bosón. Entre ellas se encuentran:

1. Listado en una tabla de los elementos: usuarios, funcionalidades y roles del sistema.
2. Diseño de los botones Adicionar, Aplicar y Cancelar.
3. Diseño de los paneles, en cuanto a las regiones de sus elementos: norte, sur, este y oeste.
4. Atributos que deben contener los usuarios, los roles y las funcionalidades.

[6] **WSDL** por sus siglas en inglés *Web Services Description Language*. Es un formato XML que se utiliza para describir servicios Web.
[7] **REST** por sus siglas en inglés *Representational State Transfer*. Es una arquitectura de servicios Web para sistemas distribuidos.
[8] **SOAP** por sus siglas en inglés Simple Object Access Protocol**.**

1.4 Metodología de desarrollo

Las metodologías y estándares utilizados en un desarrollo de software nos proporcionan las guías para poder conocer todo el camino a recorrer desde antes de empezar la implementación (Chacón, 2006). Las metodologías son un conjunto de procedimientos, técnicas, herramientas y un soporte documental que ayuda a los desarrolladores en la realización aplicaciones.

1.5.1 Metodología de desarrollo para la Actividad productiva de la UCI

La metodología utilizada es la propuesta por la universidad, que es una variación de la metodología AUP[9] en unión con el modelo CMMI[10]-DEV v 1.3, de forma tal que se adapte al ciclo de vida definido para su actividad productiva. Describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de software de negocio mediante la utilización técnicas ágiles (Sánchez, 2015). Se decide utilizar como metodología de desarrollo AUP en su variación UCI ya que es la metodología definida por el proyecto Desarrollo de componentes marco de trabajo Bosón.

1.5.2 Fases

La metodología Variación AUP-UCI propone 3 fases: Inicio, Ejecución y Cierre. El desarrollo de la investigación, se centra en la fase de Ejecución donde se ejecutan las actividades requeridas para desarrollar el software, incluye el ajuste de los planes del proyecto, considera 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.

1.5.3 Disciplinas

Para el ciclo de vida de los proyectos de la UCI se decide tener 7 disciplinas al igual que AUP. La disciplina Modelo de AUP recoge los flujos de trabajo Modelado de negocio, Requisitos y Análisis y diseño en AUP-UCI, se mantiene la disciplina Implementación. En el caso de Prueba, la UCI la descompone en 3 disciplinas: Pruebas Internas, de Liberación y Aceptación. Las restantes 3 disciplinas de AUP asociadas a la parte de gestión, para la variación UCI se cubren con las áreas de procesos que define CMMI-DEV v1.3 para el nivel 2, serían Gestión de la configuración, Planeación de proyecto y Monitoreo y control de proyecto (Sánchez, 2015). Durante el desarrollo de la investigación se utilizaron las disciplinas Requisitos, Análisis y Diseño, Implementación, Pruebas Internas y de Aceptación.

A continuación se describen cada una de las disciplinas que conforman la metodología AUP-UCI.

[9] **AUP** por sus siglas en inglés *Agile Unified Process*. Proceso Unificado Ágil.
[10] **CMM**I por sus siglas en inglés *Capability Maturity Model Integration*.

- **Modelado del Negocio**: Es la disciplina destinada a comprender los procesos de negocio de una organización. Entiende cómo funciona el negocio que se desea informatizar para tener garantías de que el software desarrollado va a cumplir su propósito. Para modelar el negocio se proponen los Casos de Uso del Negocio (CUN), la Descripción de Proceso de Negocio (DPN) y Modelo Conceptual (MC).
- **Requisitos:** Desarrollar un modelo del sistema que se va a construir. Esta disciplina comprende la administración y gestión de los requisitos funcionales y no funcionales.
- **Análisis y Diseño:** En esta disciplina 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. También se modela el sistema y su arquitectura para que soporte todos los requisitos, incluye además los requisitos no funcionales.
- **Implementación:** Se construye el sistema a partir de los resultados del Análisis y Diseño.
- **Pruebas internas:** Se verifica el resultado de la implementación para probar cada instrucción. Se deben desarrollar artefactos.
- **Pruebas de liberación:** Diseñadas y ejecutadas por una entidad certificadora de la calidad externa.
- **Pruebas de aceptación:** Prueba final del despliegue del sistema. Verificar que el software esté listo y puede ser usado por usuarios finales.

1.5.4 Escenarios

En la metodología AUP-UCI existen tres formas de encapsular los requisitos por lo que surgen cuatro escenarios para modelar el sistema en los proyectos. El escenario 1 aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan que puedan modelar una serie de interacciones entre los trabajadores del negocio/actores del sistema (usuario). El escenario 2 se recomienda para proyectos donde el objetivo primario es la gestión y presentación de información. No es necesario incluir las responsabilidades de las personas que ejecutan las actividades en el negocio, por lo que se modela exclusivamente los conceptos fundamentales del mismo. El escenario 3 es conveniente si se desea representar gran cantidad de niveles de detalles y las relaciones entre los procesos identificados. Es el escenario apropiado para proyectos que obtengan un negocio con procesos muy complejos.

Por último existe un escenario 4 que corresponde a los proyectos que no modelan negocio, por lo que solo pueden modelar el sistema con Historia de Usuarios (HU). Se recomienda en proyectos no muy extensos, ya que una HU no debe poseer demasiada información. El departamento de Desarrollo de componentes del CEIGE se encuentra dentro de este escenario, donde el cliente estará siempre asociado al equipo de desarrollo para convenir los detalles de los requisitos y así poder implementarlos, probarlos y validarlos.

1.5 Herramientas y tecnologías

Para el desarrollo de las GUI del componente Seguridad del marco de trabajo Bosón se definen un conjunto de herramientas, tecnologías y lenguajes a utilizar teniendo en cuenta las características del proyecto y el entorno de despliegue. A continuación se describen cada una de ellas.

1.5.1 Herramienta CASE: Visual Paradigm 8.0

Las herramientas CASE[11] son herramientas individuales para ayudar al desarrollador de software o administrador de proyecto durante una o más fases del desarrollo de software. Visual Paradigm es una herramienta que propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, recorre el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación. Ha sido concebida para soportar el ciclo de vida completo del proceso de desarrollo del software a través de la representación de todo tipo de diagramas.

1.5.2 Entorno integrado de desarrollo: Phpstorm 9

Es un editor de código inteligente que proporciona un excelente soporte para PHP, HTML, JavaScript, CSS y muchos otros idiomas. Proporciona la mejor finalización de código, refactorizaciones, sobre la marcha de la prevención de errores. Realiza tareas rutinarias desde el IDE[12], gracias a la integración de sistemas de control de versiones, el apoyo a la implementación remota, bases de datos / SQL, herramientas de línea de comandos y muchas otras herramientas. Es multiplataforma y se incluyen las tecnologías de vanguardia como HTML5, CSS y JavaScript (JetBrains, 2016).

1.5.3 Sistema Gestor de Base de Datos: PostgreSQL 9.2.4

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD[13] y con su código fuente disponible libremente. Es el sistema de gestión de bases de datos de código abierto más potente del mercado. Utiliza un modelo cliente/servidor y usa multiprocesos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando. Estabilidad, potencia, robustez, facilidad de administración e implementación de estándares han sido las características que más se han tenido en cuenta durante su desarrollo. Funciona muy bien con grandes cantidades de datos y una alta concurrencia de usuarios accediendo a la vez al sistema (Guerrero, 2010).

[11] **CASE**: por sus siglas en inglés ***C**omputer **A**ided **S**oftware **E**ngineering*
[12] **IDE**: por sus siglas en inglés *Intregrated Development Environment*
[13] **BSD**: por sus siglas en inglés *Berkeley Software Distribution.*

1.5.4 Sistema de Control de Versiones: Subversion (SVN)

Subversion es un sistema de control de versiones centralizado y de código abierto. Se caracteriza por su fiabilidad como un refugio seguro para los datos valiosos; la sencillez de su modelo y su uso; y su capacidad para apoyar las necesidades de una amplia variedad de usuarios y proyectos, desde particulares a las operaciones empresariales a gran escala (The Apache Software Foundation, 2016).

Subversion maneja ficheros y directorios, y los cambios hechos a ellos, con el tiempo. Esto le permite recuperar versiones antiguas de sus datos o examinar la historia de cómo cambiaron.

Puede operar a través de las redes, lo que permite que sea utilizado por personas en diferentes equipos. A cierto nivel, la capacidad de que varias personas puedan modificar y administrar el mismo conjunto de datos de sus respectivas ubicaciones fomenta la colaboración. Se puede progresar más rápidamente y sin un único conducto por el cual deban pasar todas las modificaciones (Collins-Sussman, 2005).

1.5.5 Servidor Web: Apache

El servidor web *Apache* es probablemente el más utilizado en la actualidad. Es un servidor de código abierto, potente y ofrece un servicio estable y sencillo de mantener y configurar. Se ejecuta en varios sistemas operativos. Permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor. Se determinó utilizar el servidor Apache, pues es el servidor web escogido por excelencia. La calidad de servicios, robustez y estabilidad hacen que los usuarios y servidores reiteren su confianza y renueven la elección de este servicio.

1.5.6 Symfony2.3

Symfony es un completo marco de trabajo desarrollado totalmente con PHP5. 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. Automatiza las tareas más comunes y permite al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. Ha sido probado en numerosos proyectos reales y se utiliza en sitios web de comercio electrónico de primer nivel. 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 (Potencier, 2008)

Symfony2 es su versión más reciente, y supone un cambio radical tanto en arquitectura interna como en filosofía de trabajo respecto a sus versiones anteriores. Ha sido ideado para exprimir al límite todas las nuevas características de PHP 5.3 y por eso es uno de los marcos de trabajo PHP con mejor rendimiento y que más ideas incorpora del resto de los de su tipo, incluso de aquellos que no están programados con PHP. Su arquitectura interna está completamente desacoplada, lo que permite reemplazar o eliminar fácilmente aquellas partes que no encajan en el proyecto (Eguiluz, 2013).

1.5.7 Lenguajes de Desarrollo del lado del cliente

HTML5

Es la última versión de HTML y está diseñada para ser utilizable por todos los desarrolladores de Open Web. Se trata de una nueva versión de HTML, con nuevos elementos, atributos y comportamientos. Contiene un conjunto más amplio de tecnologías que permite a los sitios web y a las aplicaciones ser más diversas y de gran alcance. Proporciona una mayor optimización de la velocidad y un mejor uso del hardware. Permite a las páginas web almacenar datos localmente en el lado del cliente y operar sin conexión de manera más eficiente (Mozilla Developer Network, 2016).

AngularJS

Para el desarrollo de las GUI se seleccionó AngularJs, que se encuentra definido por el departamento de Desarrollo de componentes en el documento Vista de Desarrollo Tecnológico. AngularJs es un marco de trabajo de código abierto con programación del lado del cliente. Su objetivo central es crear aplicaciones web dinámicas fundamentalmente aquellas de una sola página. Implementa el patrón de diseño MVC[14] lo que hace que el desarrollo y las pruebas sean más fáciles. Contiene un conjunto de bibliotecas útiles para el desarrollo de aplicaciones web y se enfoca en la optimización y sencillez de uso para los desarrolladores. Trabaja con directivas HTML que permite incluir código avanzado de manera simple. No genera componentes gráficos o CSS[15] por lo que se tiene total libertad para crear la aplicación en los términos que desees, lo que lo hace liviano y eficiente. Incluye la posibilidad de hacer vinculaciones entre objetos en el código con objetos visuales.

CCS3

Para los estilos CSS de la página se utilizará CSS3, ya que permite separar el contenido de la página de su presentación, lo cual mejora la accesibilidad de la página desde disímiles dispositivos y facilita su mantenimiento y perfeccionamiento. La novedad más importante que aporta CSS3 consiste en la incorporación de nuevos mecanismos para mantener un mayor control sobre el estilo con el que se muestran los elementos de las páginas, sin tener que recurrir a trucos, que a menudo complican el código de las web (Álvarez, 2008).

JavaScript

A veces abreviado como JS, es un lenguaje ligero orientado a objetos con funciones de primera clase, más conocido como el lenguaje de *script* para páginas web, pero también usado en muchos entornos sin navegador, tales como node.js o Apache CouchDB**.** Es un lenguaje multi-paradigma, basado en prototipos, dinámico, soporta estilos de programación funcional, orientado a objetos e imperativo.

[14] **MVC:** Modelo Vista Controlador, por sus siglas en inglés *Model View Controller.*
[15] **CSS**: Hojas de estilo en cascada, por las siglas en inglés de ***C****ascade* ***S****tyle* ***S****heet*

Es soportado por la mayoría de los navegadores como Internet Explorer, Netscape, Mozilla Firefox y Google Chrome (Mozilla Developer Network, 2015).

1.5.8 Lenguajes de Desarrollo del lado del servidor

PHP v5.4

Es interpretado y completamente orientado al desarrollo de aplicaciones web dinámicas. Es gratuito, fácil de usar y aprender, portable, de código abierto y multiplataforma. Presenta interfaces para una gran cantidad de sistemas de base de datos diferentes, así como bibliotecas incorporadas para muchas tareas web habituales.

Provee diferentes niveles de seguridad los cuales se pueden configurar desde el archivo .ini y se integra muy bien junto a otro software, especialmente bajo ambiente Unix. Utiliza su propio sistema de administración de recursos y dispone de un sofisticado método de manejo de variables, que lo hace un sistema robusto y estable. (Mariño, 2008)

1.6 Conclusiones del capítulo

El análisis de las tecnologías existentes aportó un mayor conocimiento para el desarrollo de la solución. Características como el formato de los botones, la estructura en los menús, las tablas y paneles de las GUI del componente Seguridad del marco de trabajo Sauxe permitió obtener ideas para el diseño de la solución. Además, se concluye que ningún marco de trabajo analizado a nivel internacional presenta GUI implementadas para el componente Seguridad.

El análisis de la metodología permitió definir las etapas y disciplinas a utilizar en la investigación. No se realizó modelado de negocio porque no se tiene un negocio definido en el proyecto, ya que su objetivo principal es el desarrollo de componentes para obtener una plataforma base para las aplicaciones web y facilitar el trabajo a los desarrolladores.

El análisis de las herramientas definidas por el proyecto Desarrollo de componentes para el marco de trabajo Bosón permitió conocer sus características y las ventajas de su utilización.

CAPÍTULO II DESCRIPCIÓN DE LAS INTERFACES GRÁFICAS DE USUARIO DEL COMPONENTE SEGURIDAD

En el presente capítulo se hace referencia a los principales elementos que componen la propuesta de solución. Se identifican y describen los requisitos funcionales y no funcionales. Se exponen los artefactos que genera la metodología de desarrollo seleccionada. Se hace una descripción de la arquitectura y estructura del componente, patrón arquitectónico y patrón de diseño de la solución. Finalmente se muestra el diagrama de clases del diseño que representan las clases y las relaciones entre ellas y el diagrama de despliegue que diseña como quedará desplegado el sistema.

La investigación tiene como propósito principal desarrollar las GUI del componente Seguridad del marco de trabajo Bosón. Tienen como objetivo fundamental mejorar la usabilidad del componente y garantizar la seguridad de la información. El diseño debe estar orientado al tipo de usuario que va a interactuar con el sistema

2.1 Requisitos de la propuesta

Los requisitos de un sistema son las cualidades o condiciones que el propio sistema debe cumplir. Surgen de las necesidades del cliente y delimitan cómo quiere el cliente que se comporte el sistema, qué información tiene que manejar y cómo la debe procesar y presentar.

Para la captura de los requisitos se usaron varias técnicas. La extracción de requisitos desde la información proporcionada por el cliente y las entrevistas con el mismo, fueron técnicas de utilidad para obtener los requisitos del sistema. Esto facilitó recopilar información y reunir datos actuales del proyecto que recogen la idea de lo que necesitan.

Los requisitos se agruparon según su funcionalidad, en dependencia del tipo de datos que manejan y procesos que van a cubrir. Se agrupan aplicando el patrón de Casos de uso CRUD Completo para los casos de uso Gestionar usuario, Gestionar rol y Gestionar funcionalidad. El CRUD Completo se aplica para administrar la información (CRUD Información), permite agrupar las operaciones, tales como adicionar, listar, modificar y eliminar. En el caso de la solución propuesta, no se modela el negocio a través de los Casos de Uso del Sistema ya que el proyecto se encuentra dentro del escenario 4 que propone la metodología seleccionada, donde solo se puede modelar el sistema a través de HU.

Los requisitos se clasifican en funcionales y no funcionales. Seguidamente, se presentan los identificados para el desarrollo de la solución propuesta según su clasificación.

2.1.1 Requisitos funcionales

Los requisitos funcionales son los que describen los procesos a los que se destina el sistema y afectan directamente su funcionalidad. A continuación se muestra una lista de los 22 requisitos funcionales identificados durante la fase de captura de requisitos.

GESTIONAR USUARIO

RF1. Adicionar usuario

RF2. Asignar rol

RF3. Cambiar contraseña

RF4. Modificar usuario

RF5. Buscar usuario

RF6. Listar usuario

RF7. Eliminar usuario

RF8. Activar o Desactivar usuario

GESTIONAR ROL

RF9. Adicionar rol

RF10. Modificar rol

RF11. Buscar rol

RF12. Listar roles

RF13. Eliminar rol

RF14. Asignar permisos a un rol

RF15. Asignar roles a un rol

GESTIONAR FUNCIONALIDAD

RF16. Adicionar funcionalidad

RF17. Modificar funcionalidad

RF18. Buscar funcionalidad

RF19. Listar funcionalidad

RF20. Eliminar funcionalidad

RF21. Configurar proveedor de identidad

RF22. Configurar proveedor de servicios

2.1.2 Requisitos no funcionales

Los requisitos no funcionales recogen todos los requisitos del sistema que no representan la funcionalidad principal del mismo, sino que fijan condiciones para realizar dicha funcionalidad.

- **Soporte**

RNF1. Mantenimientos al código donde se inserten comentarios que tributen a la documentación del desarrollo del proyecto.

- **Software**

Se requiere para las PC clientes

RNF2. Navegador Mozilla Firefox 40.0 o superior.

RNF3. Sistemas operativos GNU/Linux, Windows XP o superior.

Se requiere para las PC servidoras

RNF4. Sistema operativo Linux en cualquiera de sus distribuciones.

RNF5. Servidor web Apache en su versión 2.2 o superior.

RNF6. PHP 5.4 y extensiones **mcrypt**, **json y ldap.**

RNF7. PostgreSQL en su versión 8.3 o superior, MySql en su versión 5.5, como Sistemas Gestor de Base de Datos.

- **Requerimientos de Hardware**

En el caso de las PC clientes donde se ejecutará la aplicación, se requiere de las siguientes condiciones:

RNF8. Procesador Pentium IV 2x2 caché.

RNF9. Velocidad de procesamiento a 1 GHz y 1 GB de RAM.

En dependencia de la funcionalidad concebida para cada uno de los servidores, son las condiciones que se requiere que deban cumplir:

Servidor web

RNF 10. Requiere un procesador Pentium IV 2X2 caché, velocidad del procesador de 1GHz, memoria de 1GB de RAM.

Servidor de datos

RNF 11. Requiere un procesador I3, velocidad del procesador de 1 GHz, memoria de 1 GB de RAM.

RNF 12. Almacenamiento en disco con una capacidad igual o superior a los 10GB (como mínimo).

- **Seguridad**

RNF 13. Control de acceso al sistema que se realiza mediante la identificación, autenticación y autorización de usuarios.

RNF 14. Políticas de usuarios y contraseñas donde se deben utilizar al menos 8 caracteres para crear la clave. Se recomienda emplear en una misma contraseña dígitos, letras y caracteres especiales y que las letras mayúsculas y minúsculas se alternen aleatoriamente.

Utilizar signos de puntuación si el sistema lo permite. En este caso de incluir otros caracteres que no sean alfa-numéricos en la contraseña, hay que comprobar primero si el sistema permite dicha elección y cuáles son los permitidos. Dentro de ese consejo se incluiría utilizar símbolos como: ! " # $ % & ' ( ) * + , - . / : ; < = > ? @ [\ ] ^ _ ` { | } ~.

RNF 15. Control de las acciones de los usuarios. Se tiene en cuenta que los usuarios en dependencia del rol que tengan asignado tendrán acceso a las funcionalidades y acciones del sistema.

RNF 16. Gestión de los roles de los usuarios. Garantiza que cada usuario autenticado en el sistema, tenga un rol asignado, para delimitar los niveles de acceso a la aplicación. Sea un rol de administrador, de usuario común o de invitado.

2.2 Descripción de requisitos funcionales a través de Historias de usuario

En el capítulo anterior se definió la metodología para guiar el proceso de desarrollo, donde se describen para la disciplina de Requisitos los cuatros escenarios para modelar el sistema en los proyectos.

Se señala además que el departamento de Desarrollo de componentes del CEIGE se encuentra dentro del escenario 4 donde para modelar el sistema solo puede ser con HU.

2.3.1 Historias de usuario

Una HU es una representación de un requisito escrito en una o dos frases donde se utiliza el lenguaje común del usuario. Son una forma rápida de administrar los requisitos de los usuarios sin tener que elaborar gran cantidad de documentos formales y sin requerir de mucho tiempo para administrarlos. Permiten responder rápidamente a los requisitos cambiantes.

Se realizaron 22 HU. La tabla 7 muestra la HU número 30 Asignar rol. En los anexos se encuentran el resto de las HU identificadas.

Tabla 6. Historia de usuario. Asignar rol

Numero: 30	**Nombre del requisito:** Asignar rol
Programador: Yaima Zulueta Saladrigas	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 16 horas
Riesgo en Desarrollo: Alta	**Tiempo Real:** N/A
Descripción: Este requisito se encarga de asignar un rol a un usuario que exista en el sistema. Luego de adicionado un usuario se selecciona y se presiona el botón Asignar rol, se listan los roles que existen en el sistema y se selecciona el rol que se desea asignar al usuario. Un usuario puede tener asignado más de un rol, por lo que se puede realizar más de una selección. Mediante la asignación usuario-rol se permite al usuario el acceso a los recursos del sistema puesto que el rol es quien tiene permiso sobre los recursos.	
 **Prototipo de interfaz gráfica de usuario**	

2.3 Descripción de la arquitectura

La arquitectura de software de un sistema es la estructura de las estructuras del sistema, la cual comprende los componentes del software, las propiedades de esos componentes visibles externamente, y las relaciones entre ellos. Una de las definiciones más completas es la formulada por la IEEE 1471-2000: "*La Arquitectura de Software es la organización fundamental de un sistema, encarnada en sus componentes, las relaciones entre ellos, el ambiente y los principios que orientan su diseño y evolución*" (IEEE 1471, 2000). La modelación de la arquitectura que se presenta en la Fig.2 refleja el estilo arquitectónico en capas de los componentes del marco de trabajo Bosón.

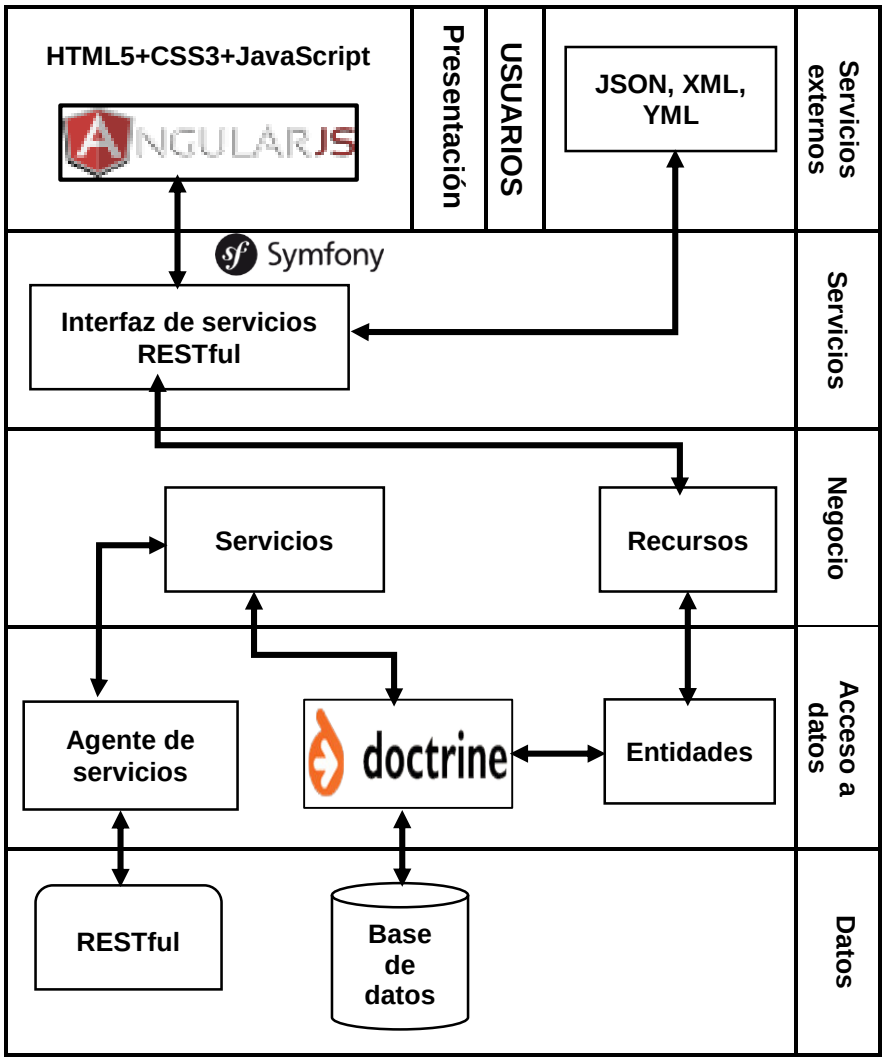


Fig. 2. Modelo de capas del componente Seguridad del marco de trabajo Bosón

Fuente: Arquitectura de Software Vista de Integración Proyecto marco de trabajo Bosón

Elaboración propia

Las capas representan los elementos que conforman a un componente, y las flechas representan el flujo de datos de un elemento a otro. La estructura de cada componente está conformada básicamente por cinco capas: *presentación, servicios, negocio, acceso a datos y datos.* Cada uno de los niveles solamente puede hacer uso de los componentes de niveles adyacentes. Esta restricción posibilita la optimización y el refinamiento de un nivel sin afectar los otros (Calás, y otros, 2015).

A continuación se realiza una breve descripción de cada una de las capas.

Presentación y sistemas externos

En esta capa se propone el uso del marco de trabajo AngularJS que garantiza que la lógica de presentación no necesite conocer cómo se procesa la información en el servidor. Incluye a los navegadores, que se encargan de interpretar todo el código programado en JavaScript, HTLM5, CSS3 y mostrar a los usuarios la interfaz resultante. Se encuentra un componente que permite que los sistemas puedan acceder y consumir los recursos que son públicos en algunos de los formatos estándares para este tipo de comunicación (XML, JSON y YML) (Calás, y otros, 2015).

Servicios

Esta capa es el punto de entrada hacia la lógica de negocio programada en el servidor. El único componente es la Interfaz de Servicios RESTful, que tiene como objetivo comunicarse bidireccionalmente con los recursos definidos en la capa de negocio (Calás, y otros, 2015).

Negocio

Esta capa implementa las funcionalidades principales del sistema y encapsula la lógica de negocio. En los componentes del sistema, esta capa es la que incluye todos los servicios que responden a los requisitos funcionales de la aplicación. Cuando se refiere a servicios, se habla de los propios de Symfony 2 (Calás, y otros, 2015).

Acceso a datos

Esta capa provee acceso a la información que almacena el sistema en sus bases de datos, y permite también acceder a los datos expuestos por otros sistemas (Calás, y otros, 2015).

Datos

En esta última capa se encuentran las fuentes de datos de la cual se persiste o extrae información. En la arquitectura se propone utilizar las bases de datos relacionales soportadas por Doctrine (Oracle, MySQL, PostgreSQL, SQL Server) (Calás, y otros, 2015).

2.3.1 Arquitectura Cliente-Servidor.

La tecnología Cliente-Servidor es el procesamiento cooperativo de la información por medio de un conjunto de procesadores, en el cual múltiples clientes, distribuidos geográficamente, solicitan requerimientos a uno o más servidores centrales como podemos apreciar en la Fig. 3.

Desde el punto de vista funcional, se puede definir la tecnología Cliente-Servidor como una arquitectura distribuida que permite a los usuarios finales obtener acceso a la información de forma transparente aún en entornos multiplataforma. Se trata pues, de la arquitectura más extendida en la realización de Sistemas Distribuidos (TIC, 2010).

De una forma más simple también se puede decir que la arquitectura Cliente-Servidor se encuentra dentro de la clasificación del estilo llamada y retorno**,** y es un modelo de aplicación distribuida en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores, y los demandantes, llamados clientes (Campo, Jimmy, 2008).

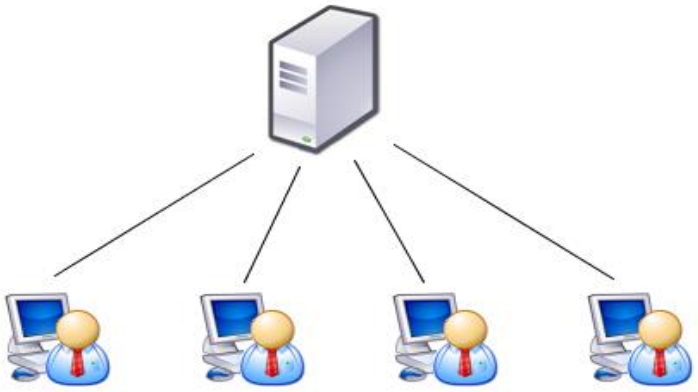


Fig. 3. Arquitectura cliente-servidor
Fuente: (Morales, 2014)

2.4 Patrón Arquitectónico

Los patrones arquitectónicos son los que definen la estructura de un sistema software y a su vez se componen de subsistemas con sus responsabilidades. También tienen una serie de directivas para organizar los componentes del mismo sistema, con el objetivo de facilitar la tarea del diseño.

2.4.1 Modelo Vista Controlador

El patrón conocido como Modelo Vista Controlador (MVC) separa el modelado del dominio, la presentación y las acciones basadas en datos ingresados por el usuario en tres clases diferentes (Ocaña Bermudez, 2014):

- **Modelo**: El modelo administra el comportamiento y los datos del dominio de aplicación, responde a requerimientos de información sobre su estado (usualmente formulados desde la vista) y responde a instrucciones de cambiar el estado (habitualmente desde el controlador).
- **Vista:** Maneja la visualización de la información.
- **Controlador**: Interpreta las acciones del ratón y el teclado, informando al modelo y/o a la vista para que cambien según resulte apropiado.

Tanto la vista como el controlador dependen del modelo, el cual no depende de las otras clases. Esta separación permite construir y probar el modelo independientemente de la representación visual.

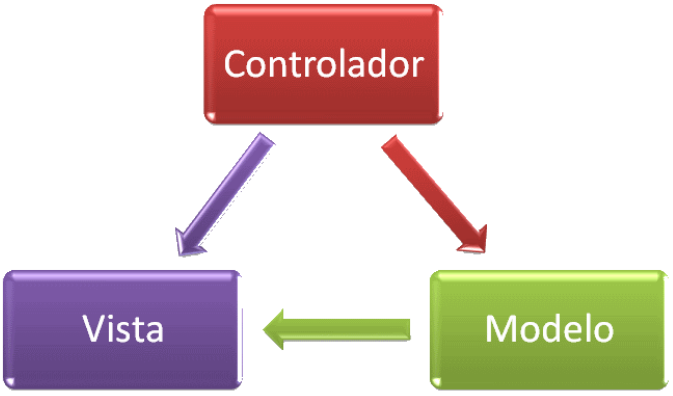


Fig. 4. Funcionamiento del patrón arquitectónico MVC
Fuente: Elaboración propia

El marco de trabajo Symfony propone el patrón MVC para la implementación de las aplicaciones, por lo cual todos los sistemas desarrollados sobre este marco de trabajo están obligados a utilizar este patrón arquitectónico.

2.5 Modelo de Diseño

El Modelo de diseño ha sido definido como *"una abstracción del Modelo de Implementación y su código fuente, el cual fundamentalmente se emplea para representar y documentar su diseño"* (Gómez Baryolo, 2010). Puede contener: diagramas, clases, paquetes, interfaces, entre otros, que son utilizados como entrada esencial en las actividades relacionadas a la implementación. A continuación se muestran los elementos que se tuvieron en cuenta para el diseño de la solución, así como los resultados y artefactos generados durante esta disciplina del proceso de desarrollo.

2.5.1 Diagrama de clases del diseño con estereotipos web

Los diagramas de clases son representaciones estáticas bidimensionales de la estructura del sistema a describir. Están compuestas por clases utilizadas en el sistema, interfaces y relaciones. Cuando uno de sus componentes tiene un significado especial, se le asigna una etiqueta que lo caracteriza la cual recibe el nombre de estereotipo.

En la Fig.5 se muestra el diagrama de clases del diseño del componente Seguridad del marco de trabajo Bosón correspondiente a la funcionalidad Gestionar rol. Se pueden identificar las clases y sus relaciones entre sí. Este escenario contiene varios formularios que envían un <<submit>> a la *server page* para que sean procesados los pedidos. A través de la clase *AppKernel* donde se encuentran registrados los bundles de Symfony, se envía la información a la controladora *RolController* que solicita los datos en las tablas de las bases de datos que se representan como entidades en el diagrama.

La página principal del componente Seguridad utiliza para construir sus interfaces ficheros CSS, las bibliotecas de AngularJs y las controladoras de este marco de trabajo: *rolCtrl, rolUpdateCtrl, roleHierarchyCtrl, rolCreateCtrl, assignPermisosCtrl y rolDeleteCtrl.*

En los anexos se encuentran el resto de los diagramas de clases del diseño que corresponden a las demás funcionalidades que integran el componente Seguridad.

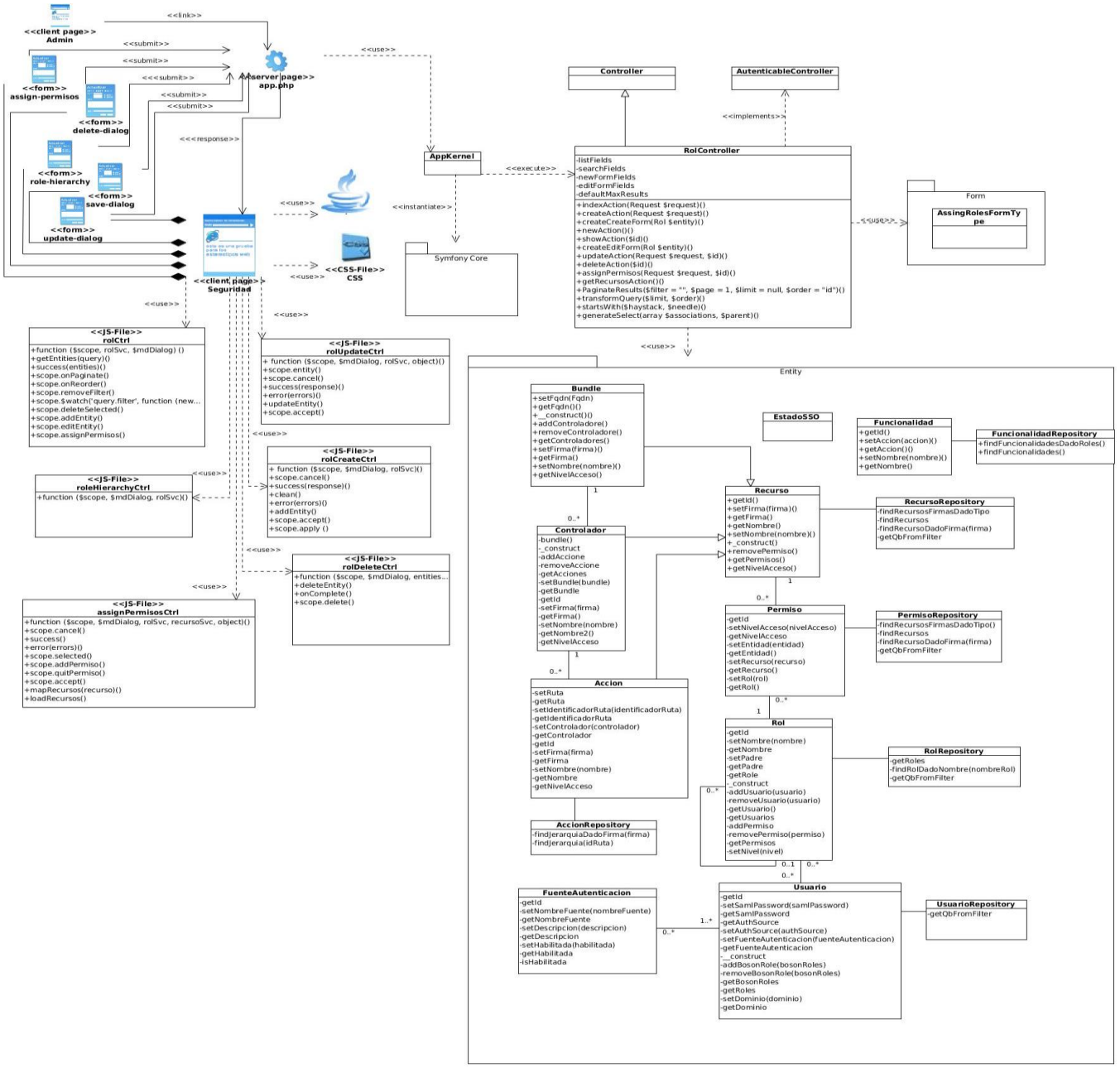


Fig. 5. Diagrama de clase del diseño Gestionar rol
Fuente: Proyecto Bosón

2.6 Patrones de diseño

El uso de patrones es considerado como una buena práctica en el desarrollo de software. Un patrón de diseño es una buena práctica documentada o solución, que se ha aplicado con éxito en múltiples ambientes para erradicar problemas comunes de diseño de software, con una probada efectividad y con características de reutilización (Fowller, 2002).

2.5.1 Patrones Generales de Asignación de Responsabilidades

Son una ayuda de aprendizaje que permiten al desarrollador entender lo esencial del diseño de objetos y aplicar el razonamiento del mismo de una forma metódica, racional y explicable. Este enfoque se entiende en el uso de los principios del diseño basado en patrones para la asignación de responsabilidades a las clases y objetos de una aplicación (Larman, 2004).

Experto

Propone la asignación de responsabilidad a la clase que cuenta con la información necesaria para crear una instancia o implementar un método. Su uso se enmarca en el mapeo y abstracción de la base de datos. Las clases generadas poseen un grupo de funcionalidades que facilitan el acceso y la manipulación de los datos de las entidades persistentes en la base de datos (Larman, 2004) .En la propuesta de solución este patrón se utilizó en la clase entidad Usuario contenedora de la información de los usuarios, en la clase entidad Rol contenedora de la información de roles y la clase entidad Funcionalidad contenedora de la información de las funcionalidades.

Creador

Asigna responsabilidades relacionadas con la creación de objetos. Es utilizado en los controladores, en ellos se encuentran las acciones definidas para el sistema (Larman, 2004). En la implementación de las acciones se crean los objetos de las clases que presentan las entidades y se evidencia que las clases controladoras son creadoras de dichas entidades. En la solución se muestra en la implementación de la clase FuncionalidadController que crea las instancias de la clase entidad Funcionalidad, en la clase RolController que crea las instancias de la clase entidad Rol y en la clase UsuarioController que crea las instancias de la clase entidad Usuario.

Controlador

Sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es la que recibe los datos del usuario y la que los envía a las distintas clases según el método llamado. Sugiere que la lógica de negocio debe estar separada de la capa de presentación, esto para aumentar la reutilización de código y a la vez tener un mayor control *(Ocaña Bermudez, 2014).* En la solución se muestra en la implementación de las clases ConfiguracionController, FuncionalidadController, RolController y UsuarioController.

Bajo acoplamiento

Es la idea de tener las clases lo menos ligadas entre sí que se pueda. De tal forma que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de las clases. Así fortalece la reutilización, y disminuye la dependencia entre las clases (Ocaña Bermudez, 2014). En la solución se muestra en la implementación de las clases FuncionalidadController, RolController, UsuarioController que tienen poca dependencia con las restantes clases, es decir, si se elimina una de ellas no se afectan las restantes.

Alta Cohesión

Se utiliza para mantener manejable la complejidad, asignando responsabilidades de manera que la información que almacena una clase sea coherente y esté relacionada (Delgado, 2016).

Symfony permite la organización del trabajo en cuanto a la estructura del proyecto y la asignación de responsabilidades con una alta cohesión. Una clase con alta cohesión, compartirá la responsabilidad de una operación, con otras clases. En la propuesta de solución este patrón se utilizó en las clases entidades Usuario, Funcionalidad y Rol porque tienen la responsabilidad de almacenar únicamente la información referente a los usuarios, las funcionalidades y los roles.

2.5.2 Patrones GoF (Gang of Four)

Los patrones GoF son una serie de posibles soluciones a problemas que suelen ser comunes en el desarrollo del software, se dividen en tres grandes categorías: creacionales, estructurales y de comportamiento (Guerrero, y otros, 2013). Los creacionales abstraen el proceso de instanciación y hacen al sistema independiente de las creaciones de objetos. Los de comportamiento se centran en algoritmos y asignación de responsabilidades entre objetos y no sólo determinan patrones de clases y objetos, sino también patrones de comunicación entre los mismos (Gamma E., 2012).

Patrón creacional Fábrica (*Factory*)

Involucra algún tipo de factoría o fábrica de objetos. Los objetos de fabricación tienen la función de crear instancias de otras clases. Además tienen la responsabilidad y el conocimiento necesario para encapsular la forma en que se crean determinados tipos de objetos en una aplicación (Aguilera Pérez, 2015). Este patrón se emplea para crear objetos de tipo formulario. En la solución se evidencia en la implementación de la clase *UsuarioController.php* la cual se relaciona con el formulario *AssingRolesFormType.php*.

Patrón estructural Fachada (*Facade*)

Este patrón está motivado por la necesidad de estructurar un entorno de programación y reducir su complejidad con la división en subsistemas, de manera que minimiza las comunicaciones y dependencias entre estos.

Su principal ventaja consiste en que para modificar las clases de los subsistemas, sólo hay que realizar cambios en la interfaz/fachada, y los clientes pueden permanecer ajenos a ello, o sea los clientes no necesitan conocer las clases que hay tras de la interfaz (Larman, 2004). En la solución se evidencia en la implementación de los formularios configFormType y AssingRolesFormType.

Patrón estructural Decorador (*Decorator*)

Se encarga de añadir dinámicamente funcionalidades a un objeto. Es aplicado a las vistas donde el contenido de la plantilla se integra en el diseño, de manera que este contiene el código HTML y los elementos comunes a todas las páginas, mientras las plantillas se encargan de mostrar el resultado de las acciones (Fowller, 2002). En la solución se evidencia en la plantilla *base.html.twig*. Este archivo almacena el código HTML que es común en muchas de las páginas de la aplicación para no tener que repetirlo en otra página. El contenido de la misma se integra a las plantillas heredadas, por ejemplo la plantilla *index.html.twig*.

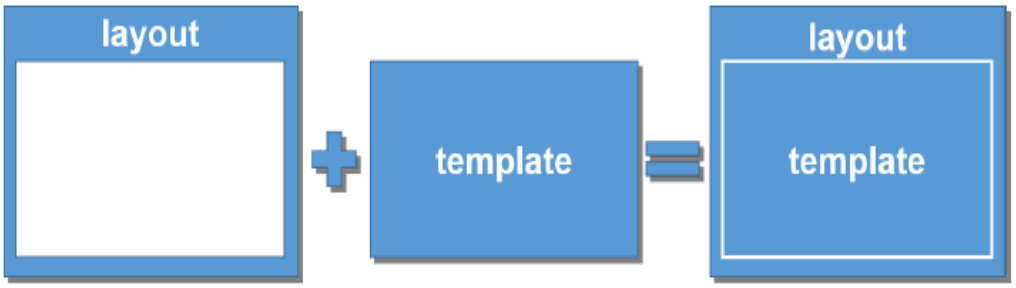


Fig. 7. Patrón Decorador
Fuente: (Potencier, 2008)

2.7 Conclusiones del capítulo

La identificación y descripción de los requisitos funcionales mediante las HU, facilitó el posterior diseño de la solución enfatizando en las capacidades o condiciones que el componente Seguridad del marco de trabajo Bosón debe cumplir. También se pudo definir la prioridad y los riesgos de implementación de las GUI.

La utilización de patrones arquitectónicos brinda solidez a la arquitectura, garantizando que la estructura propuesta rija de forma correcta el posterior diseño.

La fundamentación de los patrones de diseño utilizados permitió especificar la estructura y comportamiento de las clases, lo que garantiza una solución que atiende los niveles de reutilización.

El diseño de las GUI del componente Seguridad a través de la realización del diagrama de clases del díselo, permitió representar las clases que componen la solución y la relación entre ellas, de forma tal que establece las bases para la implementación de la propuesta de solución.

CAPÍTULO III IMPLEMENTACIÓN Y VALIDACIÓN DE LAS INTERFACES GRÁFICAS DE USUARIO DEL COMPONENTE SEGURIDAD

El presente capítulo se refiere a la disciplina de implementación donde se construye el sistema. Se describen las principales clases y los estándares de codificación utilizados para garantizar un buen entendimiento del código. Se muestran los resultados de las pruebas realizadas a la interfaz gráfica del componente Seguridad para validar que los requisitos identificados fueron implementados correctamente. Finalmente se valida la usabilidad como variable dependiente de la investigación.

3.1 Implementación

La implementación parte del resultado del análisis y diseño, se construye el diagrama de componentes donde se implementa el sistema en términos de componentes como ficheros de código binario, código fuente, scripts, ejecutables, entre otros. Su importancia se debe a que se obtiene como consecuencia un sistema ejecutable, siendo uno de los principales objetivos en el desarrollo de software.

3.2 Estándares de codificación

Los estándares de codificación son pautas de programación que no están enfocadas a la lógica del programa, sino a su estructura y apariencia física para facilitar la lectura, comprensión y mantenimiento del código (ERP, 2008). Para el desarrollo de la solución se utilizaron algunos de los estándares de codificación y normas propuestos como parte de la línea de la arquitectura determinada.

3.2.1 Visión general

Los archivos deben utilizar solamente las etiquetas <?php y <?= .

Los archivos deben emplear solamente la codificación UTF-8[16] sin BOM[17] para el código PHP.

Los archivos deberían declarar cualquier estructura (clases, funciones, constantes, etc.) o realizar partes de la lógica de negocio (por ejemplo, generar una salida, cambio de configuración ini, etc.) pero no deberían hacer las dos cosas.

Los nombres de las clases deben declararse en notación *StudlyCaps*[18].

Las constantes de las clases deben declararse en mayúsculas con guiones bajos como espacio *constante_de_clase*.

[16] **UTF-8:** por sus siglas en inglés 8-bit *Unicode Transformation Format*. Formato de codificación de caracteres.

[17] BOM: por sus siglas en inglés *Byte Order Mark* es un carácter Unicode que indica el orden de los bytes de un fichero de texto

[18] ***StudlyCaps*** es una forma de notación de texto que sigue el patrón de palabras en minúscula sin espacios y con la primera letra de cada palabra en mayúscula

Los nombres de los métodos deben declararse en notación *camelCase*[19].

Las constantes de las clases deben declararse siempre en mayúsculas y separadas por guiones bajos.

3.2.2 Métodos

La visibilidad debe ser declarada en todos los métodos.

Los nombres de los métodos no deberían usar un guion bajo como prefijo para indicar si son privados o protegidos.

Los nombres de métodos no deben estar declarados con un espacio después del nombre del método. La llave de apertura debe situarse en su propia línea, y la llave de cierre debe ir en la línea siguiente al cuerpo del método. No debe haber ningún espacio después del paréntesis de apertura, y no debe haber ningún espacio antes del paréntesis de cierre.

Cuando se realice una llamada a un método o a una función, no debe haber un espacio entre el nombre del método o la función y el paréntesis de apertura, no debe haber un espacio después del paréntesis de apertura, y no debe haber un espacio antes del paréntesis de cierre.

3.2.3 Estilo de codificación

El código debe usar 4 espacios como indentación, no tabuladores.

Las llaves de apertura de las clases deben ir en la línea siguiente, y las llaves de cierre deben ir en la línea siguiente al cuerpo de la clase.

Las llaves de apertura de los métodos deben ir en la línea siguiente, y las llaves de cierre deben ir en la línea siguiente al cuerpo del método.

Las palabras clave de las estructuras de control deben tener un espacio después de ellas, las llamadas a los métodos y las funciones no deben tenerlo.

Las llaves de apertura de las estructuras de control deben estar en la misma línea, y las de cierre deben ir en la línea siguiente al cuerpo.

Los paréntesis de apertura en las estructuras de control no deben tener un espacio después de ellos, y los paréntesis de cierre no deben tener un espacio antes de ellos.

3.2.4 Palabras claves

Las palabras clave de PHP deben estar en minúsculas.

Las constantes de PHP *true*, *false* y *null* deben estar en minúsculas.

Las palabras clave *extends* e *implements* deben declararse en la misma línea del nombre de la clase.

[19] ***camelCase*** es una forma de notación de texto que sigue el patrón de palabras en minúscula sin espacios y con la primera letra de cada palabra en mayúsculas exceptuando la primera palabra.

Ejemplo:

class NombreDeClase ***extends*** *ClasePadre implements*

Cuando estén presentes las declaraciones *abstract* y *final* deben preceder a la declaración de visibilidad. Cuando esté presente la declaración static, debe ir después de la declaración de visibilidad.

Ejemplos:

abstract *protected function zim();*

final *public* ***static*** *function bar()*

3.3 Modelo de datos

El modelo de datos permite el almacenamiento de la información de forma coherente y organizada Los objetos de datos definidos durante el análisis de requisitos del software son modelados utilizando diagramas de entidad-relación. Éste es un modelo de datos basado en una percepción del mundo real que consiste en un conjunto de objetos básicos llamados entidades y relaciones entre estos objetos, implementándose en forma gráfica a través del diagrama entidad-relación.

A continuación se reutiliza el modelo de datos elaborado por el proyecto Bosón. En la Fig.8 se muestra el diagrama entidad-relación correspondiente al componente Seguridad del marco de trabajo Bosón.

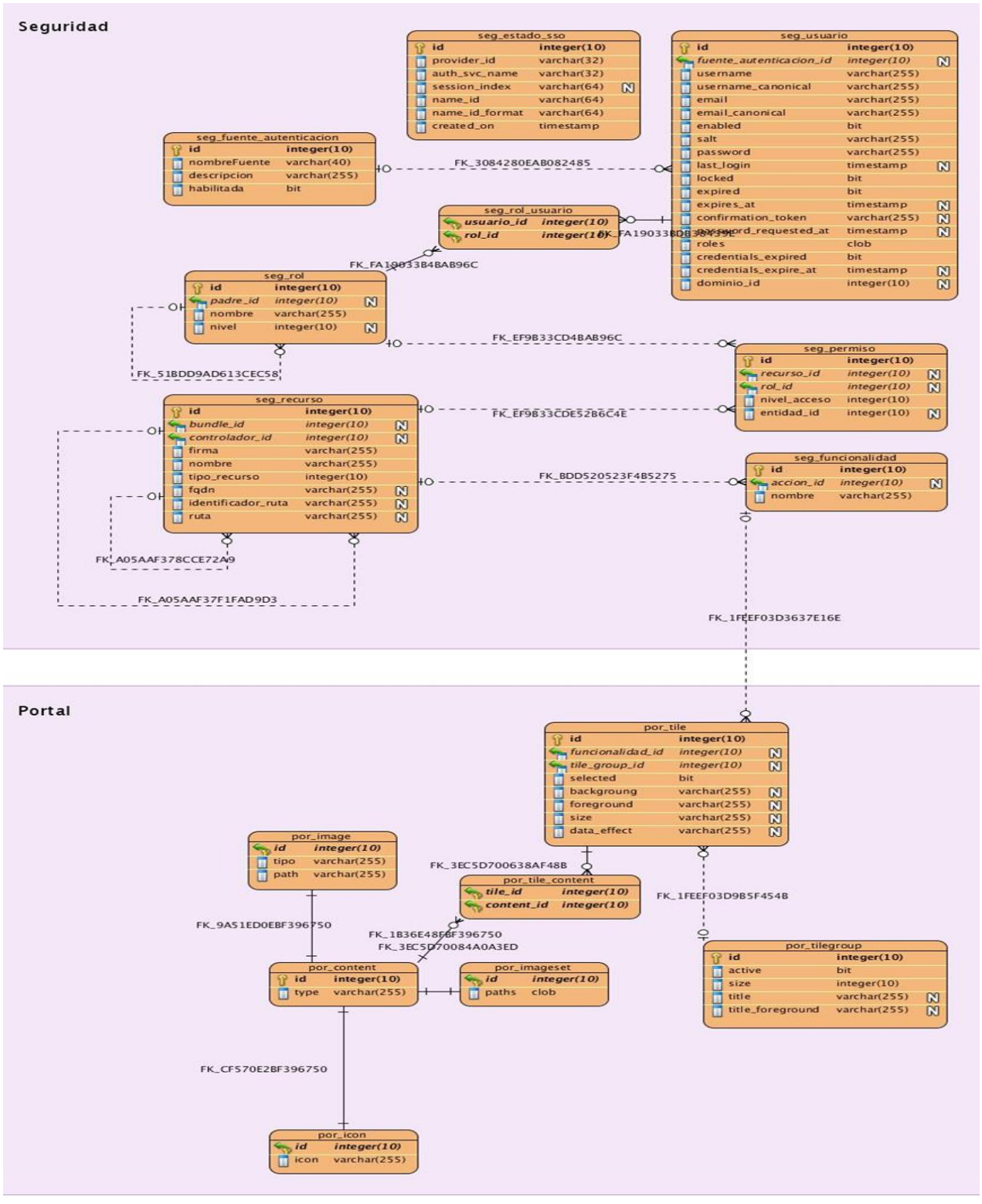


Fig. 6. Diagrama Entidad Relación del componente Seguridad del marco de trabajo Bosón
Fuente: Proyecto Bosón

3.4 Diagrama de componentes

Un diagrama de componentes muestra la organización y las dependencias entre un conjunto de componentes. Normalmente contienen: componentes, interfaces, relaciones de dependencia, generalización, asociaciones y realización, paquetes o subsistemas e instancias de algunas clases (Ferré Grau, y otros). A continuación la Fig.9 muestra el diagrama de componente correspondiente al componente Seguridad del marco de trabajo Bosón, el cual contiene componentes, interfaces y relaciones entre ellos, incluye además el paquete Manager donde agrupa los elementos que lo componen.

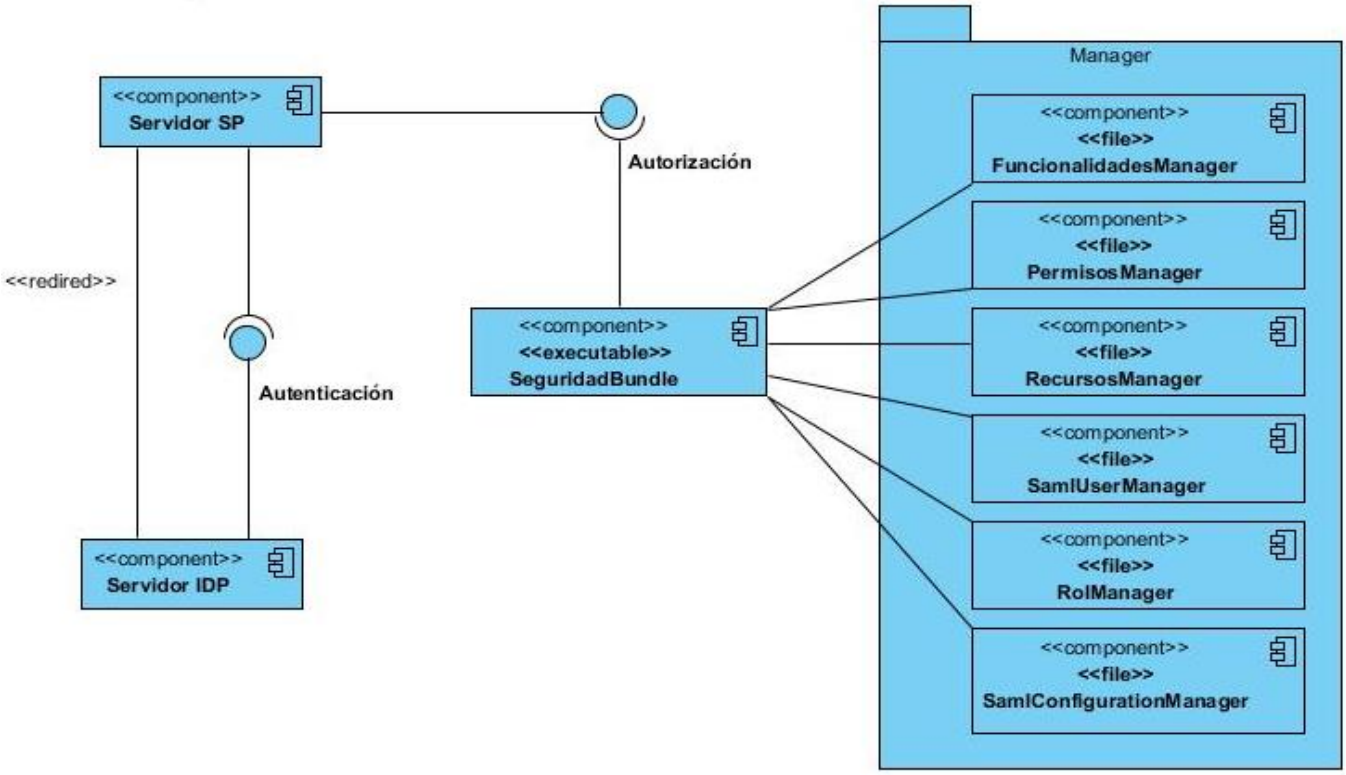


Fig. 7. Diagrama del componente Seguridad del marco de trabajo Bosón
Fuente: Elaboración propia

3.5 Diagrama de despliegue

Los diagramas de despliegue muestran las relaciones físicas de los distintos nodos que componen un sistema. En la Fig.10 se representa el diagrama de despliegue del módulo con tres nodos interconectados, PC cliente, Servidor web y Servidor base de datos. En la PC cliente se hará uso de la aplicación que mediante el protocolo HTTPS se comunicará con el componente que estará ubicado en el Servidor web y que proporcionará una interfaz para garantizar el punto de acceso al componente y obtener los datos del Servidor base de datos a través del protocolo TCP/IP.

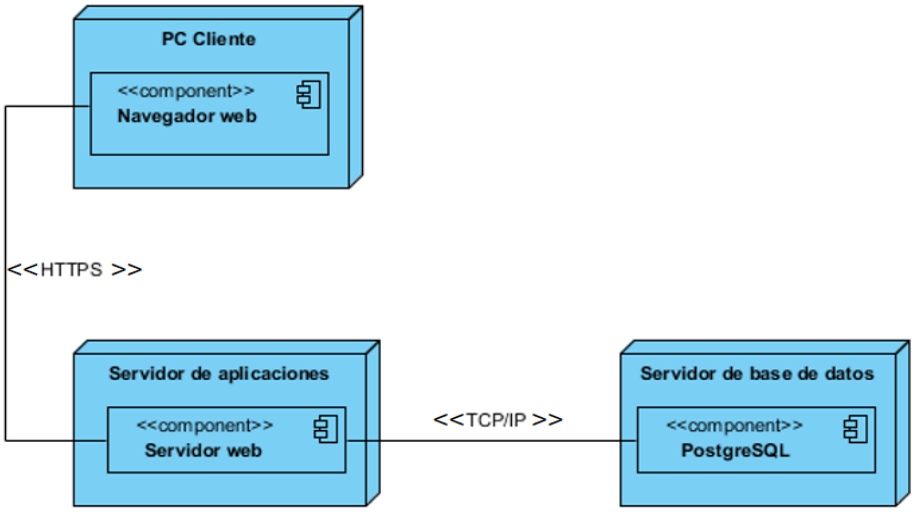


Fig. 8. Diagrama de despliegue
Fuente: Elaboración propia

3.6 Pruebas de software

Las pruebas del software según Pressman *"son un elemento crítico para la garantía de calidad del software y representa una previsión final de las especificaciones, del diseño y de la codificación ".* (Pressman, 2002). Afirma que además de detectar errores, las pruebas tienen como objetivo medir el grado en que el software cumple con los requerimientos, y expone que hay dos enfoques principales, las pruebas de caja blanca y las pruebas de caja negra, donde en ambos casos se intenta encontrar el mayor número de errores con la menor cantidad de esfuerzo y tiempo (Pressman, 2002).

3.6.1 Pruebas de caja negra

Las pruebas de caja negra, también denominadas prueba de comportamiento, se centran en los requisitos funcionales del software. Permite al ingeniero del software obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa (Pressman, 2002).

Intenta encontrar errores de las siguientes categorías:

- Funciones incorrectas o ausentes
- Errores de interfaz
- Errores en estructuras de datos o en accesos a bases de datos externas
- Errores de rendimiento

3.6.2 Partición equivalente

La partición equivalente es un método de prueba de caja negra que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba (Pressman, 2002).

Los casos de prueba especifican una forma de probar el sistema, incluyen las entradas con las que se ha de probar, las condiciones bajo las que ha de probarse, así como los resultados esperados (Gómez Baryolo, 2010). El diseño de casos de prueba para la partición equivalente se basa en una evaluación de las clases de equivalencia para una condición de entrada. Típicamente, una condición de entrada es un valor numérico específico, un rango de valores, un conjunto de valores relacionados o una condición lógica (Pressman, 2002).

3.7.1 Ejecución de los casos de pruebas

Para comprobar el correcto funcionamiento de las GUI desarrolladas, se realizaron 22 diseños de casos de prueba (registrados en la siguiente dirección del sistema Gestor de Documentos Administrativos: excriba, http://excriba.prod.uci.cu/page/site/centroceige/documentlibrary#filter=path|%2F06%2520Gesti%25F3n%2520de%2520Proyectos%2F0602%2520Expedientes%2520de%2520Proyectos%2FProyectos%25204.0%2FEds.4.0.Arquitectura%2520de%2520Referencia%2520de%2520PHPBoson_Fase%25202%2F1ingenieria%2F1.3implementacion_y_pruebas%2Fpruebas%2FComponente%2520Seguridad|&pag=1, para la versión 4.0 del expediente de proyecto: Eds.4.0.Arquitectura de Referencia de PHP-Boson_Fase 2.).

A continuación se muestra un ejemplo de diseño de caso de prueba para el escenario Eliminar usuario.

Condiciones de ejecución				
El usuario debe estar autenticado en el sistema.				
El usuario debe seleccionar el componente Seguridad.				
El usuario debe seleccionar la funcionalidad Usuarios.				
Debe haberse adicionado al menos un usuario al sistema.				
SC Eliminar usuario				
Escenario	**Descripción**	**Usuario**	**Respuesta del sistema**	**Flujo central**
EC 1.1 Eliminar usuario.	Se elimina un usuario de los que existen en el sistema.	V arquitecto	El sistema elimina un usuario y muestra el mensaje: "El usuario ha sido eliminado satisfactoriamente".	- Se selecciona un usuario de los listados en la interfaz. - Se presiona el botón **Eliminar** en la parte superior derecha de la interfaz. - Se muestra un mensaje de confirmación **"¿Está seguro que desea eliminar el elemento seleccionado?"**. - Presiona el botón **Aceptar**. - Se muestra un mensaje indicando el éxito de la operación.
EC 1.2 Presionar botón **Cancelar.**	Se cancela la operación de eliminar un usuario.	V analista	El sistema cancela la acción.	- Se selecciona un usuario de los listados en la interfaz. - Se presiona el botón **Eliminar** en la parte superior derecha de la interfaz. - Se muestra un mensaje de confirmación **"¿Está seguro que desea eliminar el elemento seleccionado?"**. - Presiona el botón **Cancelar** para abortar la operación.

Fig. 9. Diseño de caso de prueba. Eliminar usuario

A continuación, en la Fig.11 se muestra una gráfica generada a partir de las no conformidades detectadas en una primera iteración (Anexo 30). Para una primera iteración de pruebas se detectaron 2 no conformidades en la documentación y 9 no conformidades de la aplicación. En la segunda iteración se detectaron solamente 3 no conformidades de aplicación, quedando para una tercera iteración 0 no conformidades.

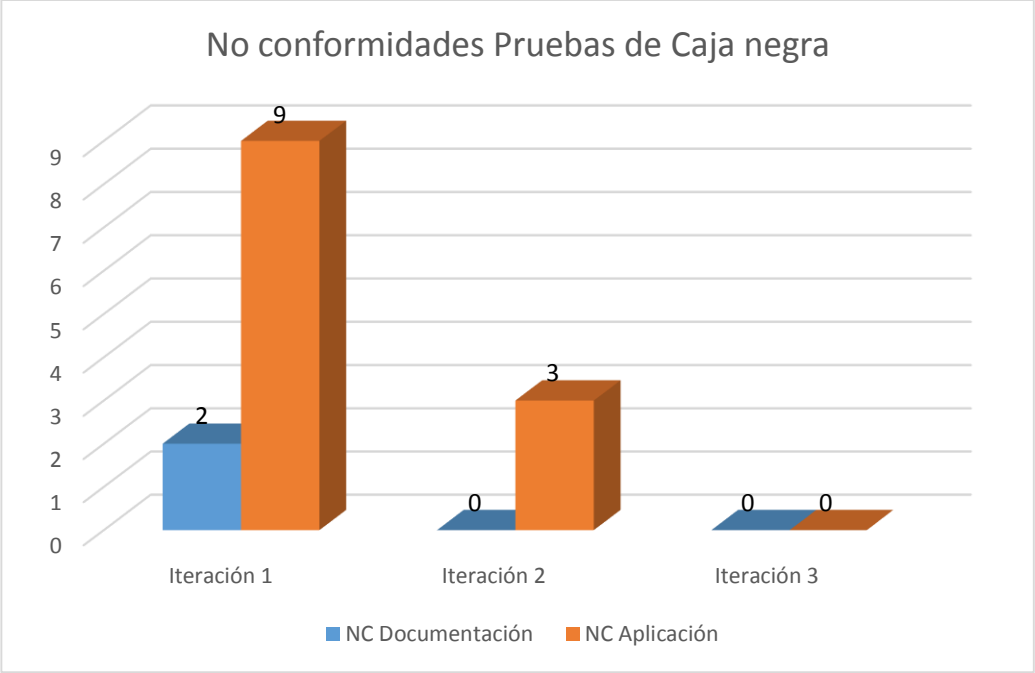


Fig. 10 Resultados de las pruebas de caja negra

3.6.1 Prueba de integración

Según Pressman, la prueba de integración es una técnica sistemática para construir la estructura del programa mientras que, al mismo tiempo, se llevan a cabo pruebas para detectar errores asociados con la interacción (Pressman, 2002). Entre los tipos de estrategias de integración incremental propuesta por Pressman, se selecciona la prueba de integración descendente, porque se ajusta a la solución propuesta, ya que se integra el componente Seguridad de forma descendente al marco de trabajo Bosón que es el componente de control principal, en el fichero AppKernel de Symfony.

Pruebas de integración descendente

La prueba de integración descendente es un planteamiento incremental a la construcción de la estructura de programas. Se integran los módulos moviéndose hacia abajo por la jerarquía de control, comenzando por el componente de control principal. Los componentes subordinados (subordinados de cualquier modo) al módulo de control principal se van incorporando en la estructura, bien de forma primero-en-profundidad, o bien de forma primero-en-anchura (Pressman, 2002).

Ejecución de la prueba de integración del componente Seguridad

En el marco de trabajo Bosón, el componente Seguridad se integra de forma descendente mediante el Composer. Se agrega a su composer.json la llave:

"boson/seguridad-bundle": "latest"

Seguidamente se registra el componente como un bundle de Symfony en el AppKernel.

```
public function registerBundles() {
$bundles = array(
                    // ... new UCIBosonSeguridadBundleSeguridadBundle(),

)}
```

3.7 Validación de la investigación

La validación de la investigación se hizo mediante un diseño pre experimental, para validar la variable dependiente de la investigación usabilidad, a través de una lista de Chequeo Usabilidad de Sitios Web propuesta por el departamento de CALISOFT de la UCI. Se probaron cada una de las GUI del componente Seguridad del marco de trabajo Bosón, teniendo en cuenta los siguientes indicadores que se evalúan en dicha lista:

1. Visibilidad del sistema
2. Lenguaje común entre sistema y usuario
3. Libertad y control por parte del usuario
4. Consistencia y estándares
5. Estética y diseño minimalista
6. Prevención de errores
7. Ayuda a los usuarios a reconocer, diagnosticar y recuperarse de los errores
8. Ayuda y documentación
9. Flexibilidad y eficiencia de uso

3.7.1 Listas de chequeo

El uso de las listas de chequeo agiliza el proceso de pruebas y constituyen una guía básica y única para el probador en la revisión de los artefactos y un apoyo en la ejecución de evaluaciones estáticas. Garantizan una mayor calidad en los artefactos de apoyo a los sistemas desarrollados por la UCI y establecen orden de revisión por subtítulos y acápites que coinciden con la organización del artefacto a evaluar.

En el Anexo 31, podemos ver la estructura de la lista de chequeo que se utilizó por el departamento de calidad del CEIGE para realizar las pruebas de usabilidad. Esta lista de chequeo tiene un carácter flexible donde pueden surgir modificaciones e inclusiones.

Ha sido confeccionada para guiar a desarrolladores, especialistas y expertos técnicos en la verificación de los productos desarrollados por la UCI en cuanto de los elementos establecidos en las pruebas de usabilidad.

3.8 Resultados

En el proceso de liberación se probaron las interfaces gráficas de usuario del componente Seguridad del marco de trabajo Bosón. Se realizaron 2 iteraciones de revisiones por el Grupo de Aseguramiento de la Calidad del CEIGE. La Fig.13 ofrece detalles de los resultados obtenidos durante la fase de pruebas de usabilidad y la cantidad de no conformidades detectadas por cada iteración.

Se puede observar como a medida que aumentan las iteraciones disminuyen las no conformidades hasta no quedar ninguna, esto demuestra que el componente está listo para ser utilizado. En los anexos se puede observar el acta de liberación emitida por el departamento de calidad del centro.

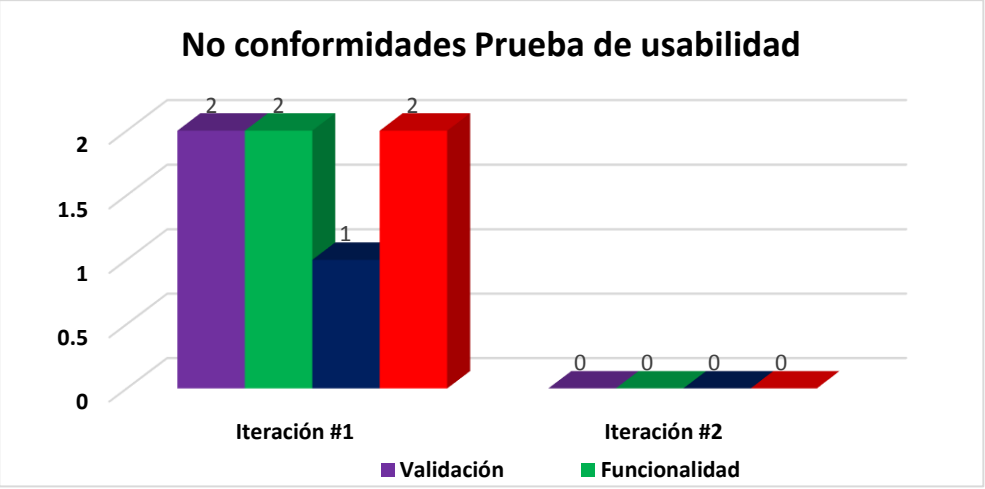


Fig. 11. Resultado de las pruebas de usabilidad

3.9 Conclusiones del capítulo

La implementación de las interfaces gráficas de usuario del componente Seguridad del marco de trabajo Bosón, ha permitido modelar el componente para representar las relaciones con los restantes componentes del marco de trabajo Bosón. Además, visualiza la información del componente Seguridad que el usuario debe configurar. La realización de las pruebas de caja negra ha permitido detectar y corregir los errores de las interfaces gráficas de usuarios. La realización de las pruebas de integración ha permitido que el componente Seguridad se pueda integrar al marco de trabajo Bosón con las interfaces graficas de usuarios, funcionado como un todo. La validación de la investigación mediante las listas de chequeo propuestas por CALISOFT, ha permitido corroborar que con el desarrollo de las GUI se mejora la usabilidad del componente Seguridad del marco de trabajo Bosón.

CONCLUSIONES GENERALES

Durante el desarrollo de la investigación se concluye lo siguiente:

El análisis de la metodología, herramientas y las tecnologías a utilizar, permitió guiar el proceso de desarrollo de las interfaces gráficas, definir los requisitos que representan las funcionalidades básicas de la aplicación y diseñar las interfaces de usuarios para su posterior desarrollo. Además se pudo representar las clases y las relaciones entre las mismas, definir el modelo de datos y la relación entre las entidades donde será almacenada la información a configurar en las GUI. Por último representar el componente Seguridad y la relación con los restantes componentes de la aplicación.

El análisis de los marcos de trabajo implementados en el lenguaje PHP, facilitó la obtención de características particulares para diferentes escenarios, teniéndolo en cuenta en el desarrollo de la solución propuesta.

La implementación de las GUI para el componente Seguridad del marco de trabajo Bosón, se ejecutaron sobre las herramientas y tecnologías libres, guiado por los modelos y estándares más utilizados para componentes de este tipo.

La validación de la solución fue guiada por casos de pruebas, las cuales se realizaron por la línea de calidad del CEIGE. Con esto se pudo verificar que se cumplieron todos los requisitos funcionales identificados hasta finalmente ser liberado en la segunda iteración.

El resultado de este trabajo demostró que con el desarrollo de las GUI para la configuración del componente Seguridad del marco de trabajo Bosón, mejoró la usabilidad del mismo, dándole solución al problema planteado.

RECOMENDACIONES

Una vez concluido el desarrollo de las GUI y cumplidos los objetivos trazados, se recomienda:

1. Elaborar un manual técnico para la configuración del componente.

REFERENCIAS BIBLIOGRÁFICAS

Aguilera Pérez, Félix Daniel. 2015. Sistema para la gestión de información académica en el Departamento Secretaría Docente de la UEB Centro de Preparación de Operarios “Oscar Lucero Moya". 2015.

Álvarez, Miguel Ángel. 2008. Introducción a CSS 3. [En línea] 9 de Junio de 2008. [Citado el: 18 de Noviembre de 2015.] http://www.desarrolloweb.com/articulos/introduccion-css3.html.

Amat, Daniel Arturo Casals. 2015. https://codecomunidades.prod.uci.cu. [En línea] 11 de Octubre de 2015. [Citado el: 3 de Noviembre de 2015.] https://codecomunidades.prod.uci.cu/Boson/seguridadbundle.

Calás, Abraham, Saria Preval, Katia y Suárez Riquenes, Ricardo E. 2015. *Arquitectura de Referencia para PHP.* 2015.

CALISOFT. *Proyecto de Investigación y Desarrollo II.* Universidad de las Ciencias Informáticas : s.n.

Campo, Jimmy. 2008. Sistemas de Información - Arquitectura Cliente-Servidor. [En línea] 2008. [Citado el: 7 de Diciembre de 2015.] http://www.slideshare.net/jcampo/cliente-servidor-307243.

Chacón, Rueda. 2006. Aplicación de la metodología RUP para el desarrollo rápido de aplicaciones basado en el estándar J2EE. [En línea] Marzo de 2006. http://biblioteca.usac.edu.gt/tesis/08/08_0308_CS.pdf.

Collins-Sussman, Ben. 2005. Control de versiones con Subversion. [En línea] 2005. [Citado el: 21 de Junio de 2016.] http://svnbook.red-bean.com/index.es.html.

Delgado, Anabel Ponce. 2016. 5, s.l. : Serie Científica de la Universidad de las Ciencias Informáticas, 2016, Vol. 9. ISSN: 2306-2495.

Eguiluz, Javier. 2013. Desarrollo web ágil con Symfony2. 2013.

ERP, Proyecto. 2008. *Normas y estándares de Codificación del ERP.Línea de Arquitectura.* La Habana : Universidad de las Ciencias Informáticas, 2008.

Estándares de codificación. Proyectos con el marco de trabajo Bosón del CEIGE.

Ferré Grau, Xavier y Sánchez Segura, María Isabel. Desarrollo Orientado a Objetos con UML. [En línea] [Citado el: 27 de Abril de 2016.] http://cursos.sigop.net/publico/informatica/varios/20111223 UML/PDF/UML Total.pdf.

Fowller, Martín. 2002. *Pattern Enterprise Application Arquitecture.* s.l. : Addison Wesley, 2002. ISBN: 0-321-12742-0.

Gamma E., Helm R., Johnson R.,. 2012. *Design Patterns: Elements of Reusable Object-Oriented Software.* s.l. : Addison-Wesley, 2012. ISBN: 0-201-63361-2.

Gómez Baryolo, Oiner. 2010. Solución informatica de autorización en entornos multientidad y multisistema. 2010.

González, Lizbeth Luna. 2004. Revista Digital Universitaria. [En línea] 10 de Agosto de 2004. [Citado el: 25 de 11 de 2015.] http://www.revista.unam.mx/vol.5/num7/art44/ago_art44.pdf. ISSN:1067-6079.

Guerrero, Carlos, Suarz, Johanna M y Gutierrez, Luz E. 2013. *Patrones de Diseño GoF (The Gang of Four) en el contexto de Procesos de Desarrollo de Aplicaciones Orientadas a la Web. Vol. 24.* 2013. DOI 10.4067/S0718-07642013000300012.

Guerrero, Rafael Martinez. 2010. PostgreSQL-es. [En línea] 2 de Octubre de 2010. [Citado el: 6 de Diciembre de 2015.] http://www.postgresql.org.es/sobre_postgresql.

IEEE 1471. 2000. Arquitectura. [En línea] 2000. [Citado el: 7 de Diciembre de 2015.] http://www.sei.cmu.edu/architecture/definitions.html.

ISO, 9241-11. 1998. *Ergonomic requirements for office work with visual display terminals.* 1998. ISO.

JetBrains. 2016. www.jetbrains.com. [En línea] 2016. [Citado el: 21 de Junio de 2016.] http://www.jetbrains.com/phpstorm/features/.

Larman, Craig. 2004. *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* Mexico : s.n. PRENTICE HAL, 2004. ISBN: 970-1 7-0261-1.

Mariño, Carlos Vázquez. 2008. Programacion en PHP5. Nivel Básico. 2008.

Morales, Nelibel. 2014. http://sistemasditribuidos.blogspot.com. [En línea] 12 de Mayo de 2014. [Citado el: 15 de Marzo de 2016.] http://sistemasditribuidos.blogspot.com/2014/05/cliente-servidor.html.

Moreno, Luciano. 2005. Componentes de una interfaz web. *Desarrolloweb.com.* [En línea] 2005. www.desarrolloweb.com/articulos/2171.php.

Mozilla Developer Network. 2016. [En línea] 12 de Mayo de 2016. [Citado el: 21 de Junio de 2015.] https://developer.mozilla.org/es/docs/HTML/HTML5.

—. 2015. [En línea] 2015. [Citado el: 18 de Noviembre de 2015.] https://developer.mozilla.org/es/docs/Web/JavaScript.

Ocaña Bermudez, Julio Cesar . 2014. *Arquitectura de Software. Vista de Integración.* 2014.

Ortiz, Dr. Emilio. 2012. www.moebio.uchile.cl. [En línea] 8 de Febrero de 2012. [Citado el: 29 de Junio de 2016.] http://www.redalyc.org/pdf/706/70600907.pdf. ISSN 0717-554X.

Potencier, F.a.Z., Francois. 2008. The Definitive Guide to Symfony. 2008.

Pressman, Roger S. 2002. *Ingeniería del Software. Un enfoque práctico.* 2002. ISBN 0-07-709677-0.

Pupo, Ing. Iliannis. 2012. *Procedimiento para el aseguramiento y evaluación de la usabilidad basado en patrones arquitectónicamente sensibles para los sistemas de gestión del Centro de Informatización de Gestión de Entidades.* La Habana : s.n., 2012.

2016. symfony.com. [En línea] 2016. [Citado el: 29 de Junio de 2016.] http://symfony.com/legacy/doc/jobeet/1_4/es/04?orm=Doctrine.

The Apache Software Foundation. 2016. SUBVERSION. [En línea] 2016. [Citado el: 21 de Junio de 2016.] https://subversion.apache.org/.

TIC, Oposiciones. 2010. Introducción a la Arquitectura Cliente-Servidor. [En línea] 2010. [Citado el: 7 de Diciembre de 2015.] http://oposicionestic.blogspot.com/2011/06/arquitectura-cliente-servidor.html.

UCI. 2015. *Metodología UCI.* 2015.

GLOSARIO

COMPONENTE: unidad de composición de aplicaciones de software, que posee un conjunto de interfaces y un conjunto de requisitos, que puede ser desarrollado e incorporado al sistema y compuesto con otros componentes de forma independiente.

BUNDLE: Los bundles son la parte más importante de Symfony2. Permiten utilizar funcionalidades construidas por terceros o empaquetar tus propias funcionalidades para distribuirlas y reutilizarlas en otros proyectos. Son un conjunto estructurado de archivos que se encuentran en un directorio y que implementan una sola característica.

ESTANDARIZACIÓN: es la redacción y solo aprobación de normas que se establecen para garantizar el acoplamiento de elementos construidos independientemente

IDENTACIÓN: mover un bloque de texto hacia la derecha con espacios o tabuladores, para así separarlo del margen izquierdo. Separación con espacios en blanco entre los diferentes elementos que componen las líneas de código.

IEEE: es una asociación técnico-profesional mundial dedicada a la estandarización, entre otras cosas.

ISO: La Organización Internacional de Normalización es el organismo encargado de promover el desarrollo de normas internacionales de fabricación (tanto de productos como de servicios), comercio y comunicación para todas las ramas industriales a excepción de la eléctrica y la electrónica.

RETROCOMPATIBILIDAD: La retrocompatibilidad o compatibilidad regresiva indica la capacidad de una aplicación informática para utilizar datos creados con versiones anteriores de ella misma, que bien permite abrirlos o incluso guardarlos con compatibilidad.

REST*ful*: se usa para describir cualquier interfaz web simple que utiliza XML y HTTP, sin las abstracciones adicionales de los protocolos basados en patrones de intercambio de mensajes.

PRE EXPERIMENTOS son aquellos en los cuales no existe un grupo de control (patrón o testigo) para comparar. Reducen las posibilidades de manipular las variables independientes y las conclusiones son extraídas por la variación de la variable dependiente en relación con su historia anterior.

ANEXOS

Anexo1

ENCUESTA

Nombre y apellidos: _____

Institución a la que pertenece: _____

Cargo actual: _____

Calificación profesional, grado científico o académico:

Profesor: _____.

Licenciado: _____.

Especialista: _____.

Máster: _____.

Doctor: _____.

Años de experiencia en el cargo: _____.

Con el propósito de validar la investigación propuesta del trabajo de diploma ***Interfaz gráfica de usuario para la configuración del componente Seguridad del marco de trabajo Bosón*** que se construye sobre el marco de trabajo Bosón se propone la siguiente encuesta donde se desea conocer su valoración crítica referente al comportamiento del componente en cuanto a la usabilidad del mismo.

Marque con una ***X*** como convenga a las siguientes preguntas según su valoración.

1. ¿Cómo evalúa la actual configuración del componente Seguridad del marco de trabajo Bosón?
   Bien ___ Regular ___ Mal ___

2. ¿Cómo valora el proceso de interacción con el sistema?
   Bien ___ Regular ___ Mal ___

3. ¿Hay demasiados errores durante la operación con el sistema?
   Sí ___ No ___

4. ¿Cree que debe memorizar y escribir muchos comandos para realizar alguna tarea?

Sí ___ No ___

5. ¿Son fáciles de aprender y memorizar?
   Sí ___ No ___

6. ¿El sistema es fácil de usar?
   Sí ___ No ___

7. ¿Cree que usuarios sin experiencia aprenderían a hacer uso del sistema rápidamente?
   Sí ___ No ___

8. ¿Cree dificil realizar las tareas asignadas dentro del sistema?
   Sí ___ No ___

9. ¿Siente satisfacción al realizar las tareas cuando hace uso del sistema?
   Sí ___ No ___

10. ¿Es fácil de recordar como ejecutar una tarea?
    Sí ___ No ___

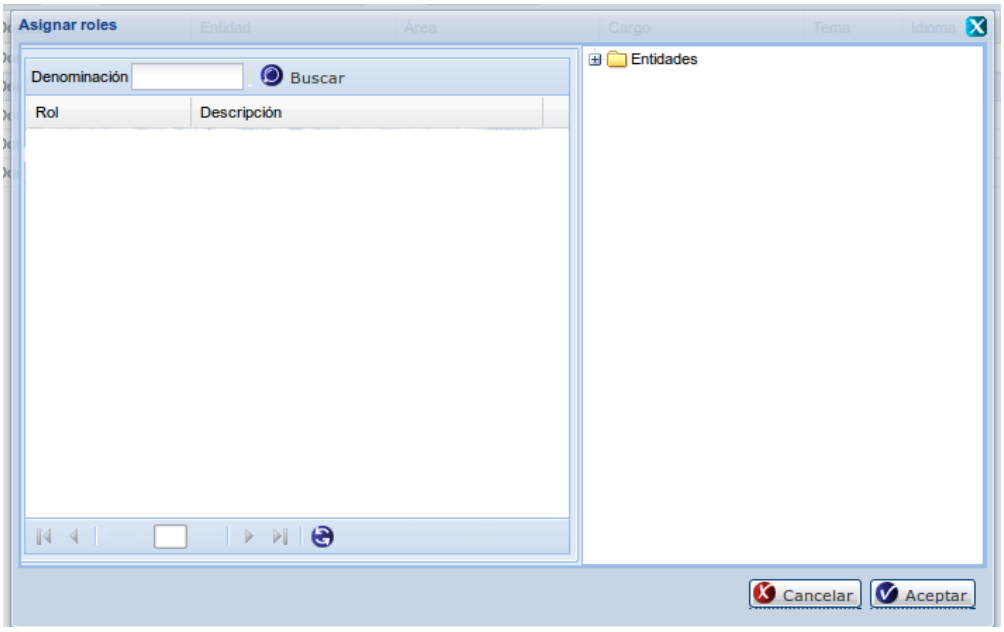
11. ¿Cree que la creación de una interfaz gráfica de usuario intuitiva pueda mejorar la situación existente?
    Sí ___ No ___

Se dice que un sistema tiene buena usabilidad cuando presenta una interfaz gráfica amigable y sencilla y cuando el usuario no necesita ser experto para poder utilizar el sistema.

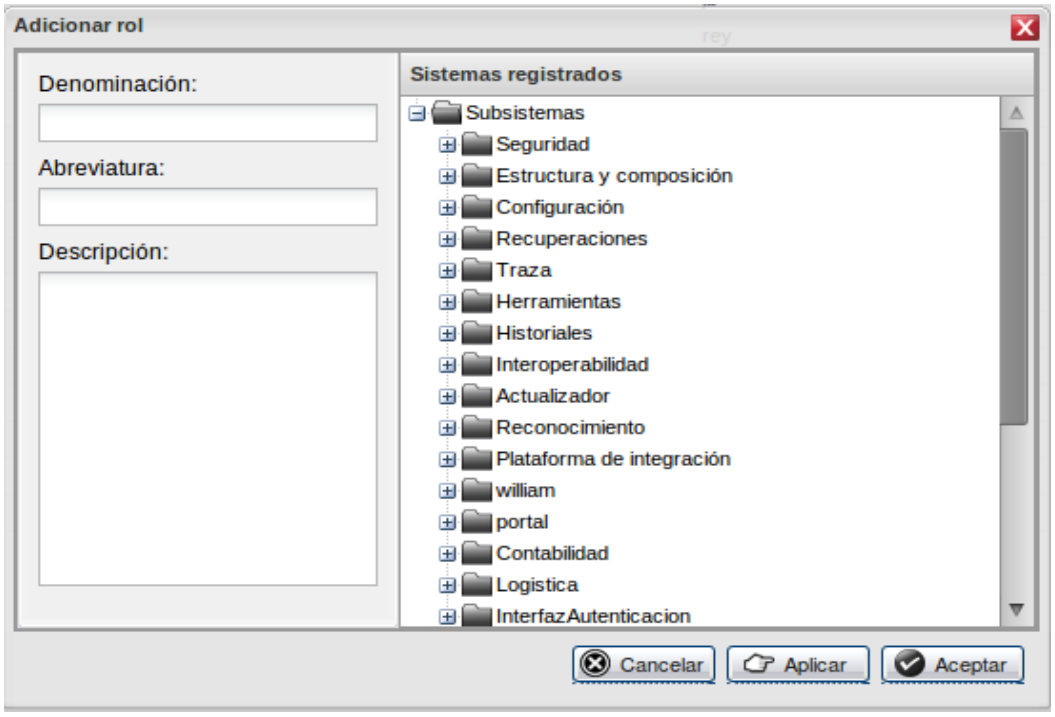
12. ¿Cómo considera el nivel de usabilidad del componente Seguridad?
    Bien ___ Regular ___ Mal ___

Anexo 2 ***Interfaz gráfica de usuario Adicionar usuario del Subsistema Seguridad de Acaxia***

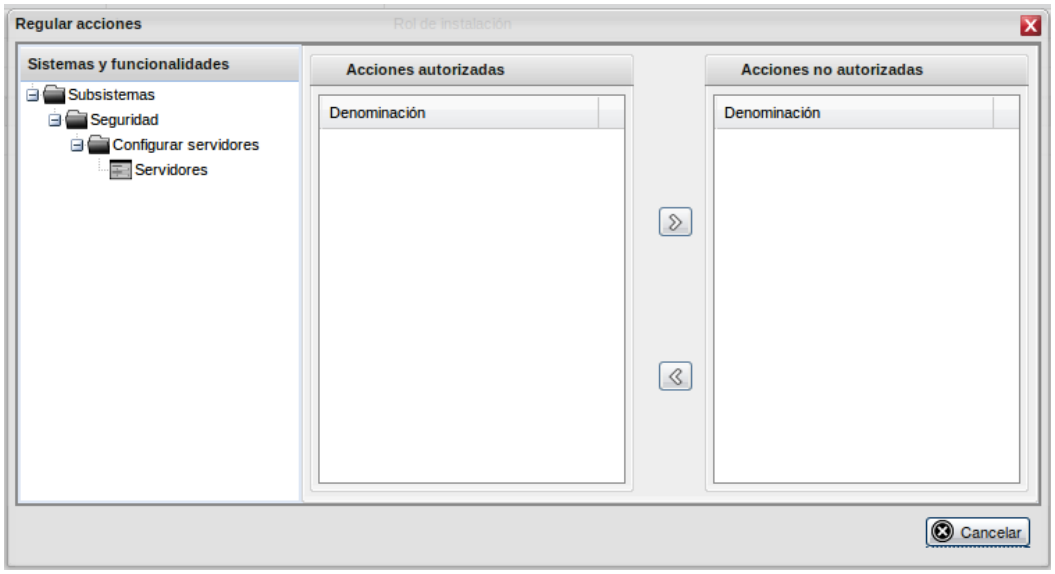
Anexo 3 ***Interfaz gráfica de usuario Asignar roles del Subsistema Seguridad de Acaxia***



Anexo 4 ***Interfaz gráfica de usuario Adicionar rol del Subsistema Seguridad de Acaxia***



Anexo 5 ***Interfaz gráfica de usuario Regular acciones del Subsistema Seguridad de Acaxia***



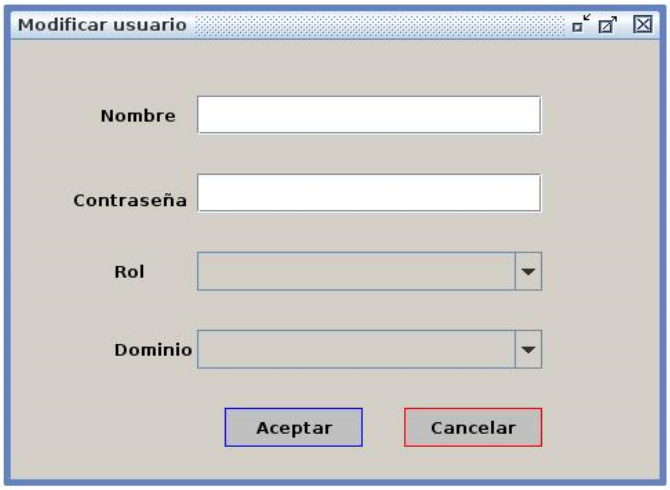
Anexo 6 Historia de usuario. Modificar usuario

Tabla 7. Historia de usuario. Modificar usuario

Numero: 26	**Nombre del requisito:** Modificar usuario
Programador: Yaima Zulueta Saladrigas	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 16 horas
Riesgo en Desarrollo: Alta	**Tiempo Real:** N/A

Descripción: Este requisito se encarga de permitir modificar un usuario de los existentes en el sistema. Al levantar la interfaz los datos que posee el usuario se mostrarán para que sean modificados. Cuando se modifica un usuario el sistema efectúa varias validaciones para comprobar que el mismo es correcto. En caso que la información brindada sea incorrecta mostrará un mensaje indicándolo, si los datos son correctos mostrará un mensaje que indique el éxito de la operación. Los campos a modificar son:

- Nombre que contiene la denominación del usuario.
- Contraseña donde se especifica la nueva contraseña con la cual accederá al sistema.
- Rol donde se establece el o rol(es) que será asignado al usuario el cual posee los permisos sobre las funcionalidades del sistema.
- Dominio establece el dominio de áreas al que tendrá acceso el usuario.



Prototipo de interfaz gráfica de usuario

Anexo 7 Historia de usuario. Eliminar usuario

Tabla 8. Historia de usuario. Eliminar usuario

Numero: 27	**Nombre del requisito:** Eliminar usuario
Programador: Yaima Zulueta Saladrigas	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 16 horas
Riesgo en Desarrollo: Alta	**Tiempo Real:** N/A
Descripción: Este requisito se encarga de eliminar un usuario de los existentes en el sistema. Para eliminar un usuario se debe seleccionar en la lista de usuario el deseado. Al presionar el botón Eliminar el sistema mostrará un mensaje de confirmación, al aceptar la eliminación se mostrará un mensaje que indique el éxito de la operación.	

Anexo 8 Historia de usuario. Buscar usuario

Tabla 9. Historia de usuario. Buscar usuario

Numero: 28	**Nombre del requisito:** Buscar usuario
Programador: Yaima Zulueta Saladrigas	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 16 horas
Riesgo en Desarrollo: Alta	**Tiempo Real:** N/A
Descripción: Este requisito se encarga de buscar un usuario de los existentes en el sistema. Para buscar un usuario se deben introducir los criterios de búsquedas establecidos por el usuario. El sistema mostrará todos los usuarios que cumplan con el criterio establecido. En el caso que no exista ninguna coincidencia el sistema mostrará un mensaje indicándolo.	

Anexo 9 Historia de usuario. Mostrar usuario

Tabla 10. Historia de usuario. Mostrar usuario

Numero: 29	**Nombre del requisito:** Mostrar usuario	
Programador: Yaima Zulueta Saladrigas		**Iteración Asignada:** 1
Prioridad: Alta		**Tiempo Estimado:** 16 horas
Riesgo en Desarrollo: Alta		**Tiempo Real:** N/A
Descripción: Este requisito se encarga de mostrar un listado de los usuarios que existen en el sistema. Esto permitirá que la información sea consultada de una forma más fácil y que se puedan realizar las acciones de modificación, búsqueda y eliminación de los mismos.		

Anexo 10 Historia de usuario. Adicionar usuario

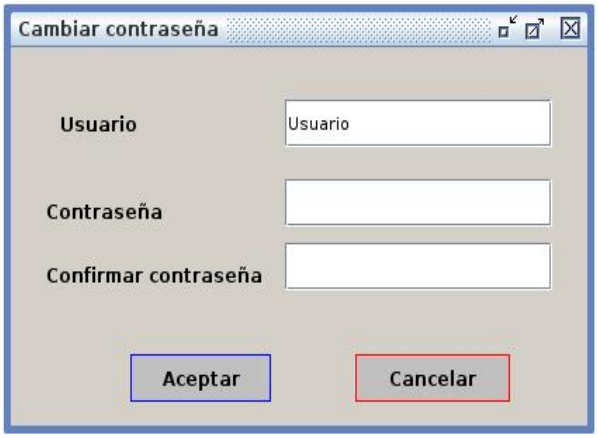
Tabla 11. Historia de usuario. Adicionar usuario

Numero: 25	**Nombre del requisito:** Adicionar usuario
Programador: Yaima Zulueta Saladrigas	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 16 horas
Riesgo en Desarrollo: Alta	**Tiempo Real:** N/A
Descripción: Este requisito se encarga adicionar un nuevo usuario al sistema, con el cual se podrá acceder al sistema. Cuando se inserta un usuario el sistema efectúa varias validaciones para comprobar que el mismo es correcto. En caso que la información brindada sea incorrecta mostrará un mensaje indicándolo, si los datos son correctos mostrará un mensaje que indique el éxito de la operación. Los campos a insertar son: - Nombre que contiene la denominación del usuario. - Contraseña donde se especifica la contraseña con la cual accederá al sistema. - Rol donde se establece el o rol(es) que será asignado al usuario el cual posee los permisos sobre las funcionalidades del sistema. - Dominio establece el dominio de áreas al que tendrá acceso el usuario.	
Observaciones: No debe permitir adicionar usuarios con la misma denominación.	

Prototipo de interfaz gráfica de usuario

Anexo 11 Historia de usuario. Cambiar contraseña

Tabla 12. Historia de usuario. Cambiar contraseña

Numero: 31	**Nombre del requisito:** Cambiar contraseña
Programador: Yaima Zulueta Saladrigas	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 16 horas
Riesgo en Desarrollo: Alta	**Tiempo Real:** N/A
Descripción: Este requisito se encarga cambiar la contraseña de un usuario que se halla adicionado con anterioridad en el sistema. Para cambiar la contraseña se debe de seleccionar un usuario de los listados y se presiona el botón Cambiar contraseña. Se carga una interfaz con el nombre del usuario y los campos para insertar la nueva contraseña y la confirmación de la misma. En caso que no coincidan ambas contraseñas proporcionadas se muestra un mensaje que indique al usuario que debe corregir la información brindada en estos campos. En caso contrario se muestra un mensaje que indica el éxito de la operación.	
	

Prototipo de interfaz gráfica de usuario

Anexo 12 Historia de usuario. Activar o desactivar usuario

Tabla 13. Historia de usuario. Activar o desactivar usuario

Número: 32	**Nombre del requisito:** Activar o desactivar usuario	
Programador: Yaima Zulueta Saladrigas		**Iteración Asignada:** 1
Prioridad: Alta		**Tiempo Estimado:** 16 horas
Riesgo en Desarrollo: Alta.		**Tiempo Real:** N/A
Descripción: Este requisito se encarga de activar o desactivar un usuario que se halla insertado en el sistema. Para ello se selecciona uno o varios usuarios de los listados, y se presiona la propiedad Activar/Desactivar mostrada en la parte derecha de cada fila de los usuario listados. Cuando esté activo se mostrará la propiedad de color verde, en caso contrario de color rojo.		

Anexo 13 Historia de usuario. Adicionar rol

Tabla 14. Historia de usuario. Adicionar rol

Número: 33	**Nombre del requisito:** Adicionar rol
Programador: Yaima Zulueta Saladrigas	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 16 horas
Riesgo en Desarrollo: Alta.	**Tiempo Real:** N/A
Descripción: Este requisito se encarga de adicionar un nuevo rol para ser asignado a un usuario. De un rol se debe conocer el nombre y el nivel. Luego de insertado los campos el sistema efectuará las validaciones persistentes, en caso que existan errores muestra un mensaje que indique los campos incorrectos. Si toda la información brindada es correcta se muestra un mensaje que indica el éxito de la operación.	

Prototipo de interfaz gráfica de usuario

Anexo 14 Historia de usuario. Modificar rol

Tabla 15. Historia de usuario. Modificar rol

Número: 34	**Nombre del requisito:** Modificar rol
Programador: Yaima Zulueta Saladrigas	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 16 horas
Riesgo en Desarrollo: Alta.	**Tiempo Real:** N/A
Descripción: Este requisito se encarga de modificar un rol de los existentes en el sistema. Se le podrán modificar al rol el nombre y el nivel. Cuando se modifica en caso que los valores sean incorrectos el sistema mostrará un mensaje que diga que existen valores erróneos, en caso contrario mostrará un mensaje que indique el éxito de la operación.	
Observaciones: La denominación del rol no debe coincidir con ninguna de las existentes en el sistema.	

Prototipo de interfaz gráfica de usuario

Anexo 15 Historia de usuario. Eliminar rol

Tabla 16. Historia de usuario. Eliminar rol

Número: 35	**Nombre del requisito:** Eliminar rol
Programador: Yaima Zulueta Saladrigas	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 16 horas
Riesgo en Desarrollo: Alta.	**Tiempo Real:** N/A
Descripción: Este requisito se encarga de liminar un rol de los existentes en base de datos. Para eliminar un rol se selecciona el deseado de los listados, se presiona el botón Eliminar. El sistema solicita confirmación, en caso que se acepte se muestra un mensaje que indica el éxito de la operación.	

Anexo 16 Historia de usuario. Buscar rol

Tabla 17. Historia de usuario. Buscar rol

Número: 36	**Nombre del requisito:** Buscar rol
Programador: Yaima Zulueta Saladrigas	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 16 horas
Riesgo en Desarrollo: Alta.	**Tiempo Real:** N/A
Descripción: Este requisito se encarga de buscar un rol en el sistema. Cuando el usuario inserte el criterio de búsqueda establecido, el sistema mostrará un listado de los roles que coincidan con dicho criterio. En el caso que no exista coincidencias mostrará un listado vacío.	

Anexo 17 Historia de usuario. Listar roles

Tabla 18. Historia de usuario. Listar roles

Número: 37	**Nombre del requisito:** Listar roles	
Programador: Yaima Zulueta Saladrigas		**Iteración Asignada:** 1
Prioridad: Alta		**Tiempo Estimado:** 16 horas
Riesgo en Desarrollo: Alta.		**Tiempo Real:** N/A
Descripción: Este requisito se encarga de listar los roles que existan en el sistema. Esto permitirá al usuario poder consultar el número de los roles que existen así como poder ejecutar las acciones de modificación, eliminación y búsqueda de los mismos.		

Anexo 18 Historia de usuario. Asignar permisos a un rol

Tabla 19. Historia de usuario. Asignar permisos a un rol

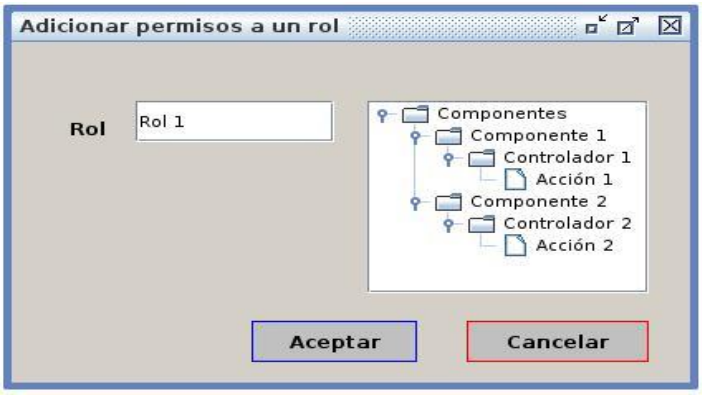
Número: 38	**Nombre del requisito:** Asignar permisos a un rol
Programador: Yaima Zulueta Saladrigas	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 16 horas
Riesgo en Desarrollo: Alta.	**Tiempo Real:** N/A

Descripción: Este requisito se encarga de asignar los recursos a las que un rol tiene permiso. Para ello en el listado de roles se selecciona el rol deseado y presiona el botón Asignar permisos. Luego se muestra una interfaz con los recursos que serían los bundle, controladores y acciones. Las categorías de asignación serían:

- Bundle: acceso total a todos los controladores y acciones contenidos dentro del mismo.
- Controlador: solo tendría acceso el rol en el bundle especificado al controlador seleccionado y todas las acciones que contiene el mismo.
- Acción: solo tendría acceso el rol al bundle especificado al controlador seleccionado y la acción seleccionada.

Esto implica el nivel restrictivo del acceso a los recursos del sistema.

El requisito se encarga de modificar un rol de los existentes en el sistema. Se le podrán modificar al rol el nombre y el nivel. Cuando se modifica en caso que los valores sean incorrectos el sistema mostrará un mensaje para advertir que existen valores erróneos, en caso contrario mostrará un mensaje que indique el éxito de la operación.



Prototipo de interfaz gráfica de usuario

Anexo 19 Historia de usuario. Asignar roles a rol

Tabla 20. Historia de usuario. Asignar roles

Número: 39	**Nombre del requisito:** Asignar roles a rol	
Programador: Yaima Zulueta Saladrigas		**Iteración Asignada:** 1
Prioridad: Alta		**Tiempo Estimado:** 16 horas
Riesgo en Desarrollo: Alta.		**Tiempo Real:** N/A