UNIVERSIDAD TÉCNICA PARTICULAR DE LOJA

La Universidad Católica de Loja

ÁREA TÉCNICA

TÍTULO DE INGENIERO EN SISTEMAS INFORMATICOS Y COMPUTACIÓN

Consideraciones de arquitectura de software a nivel de diseño arquitectónico y desarrollo de software para minimizar vulnerabilidades en el proceso de desarrollo de software web basados en OWASP Top Ten 2013, caso de estudio arquitecturas: 3-Layers.

TRABAJO DE TITULACIÓN

AUTOR: Abarca Cabrera, Juan Keyner.

DIRECTOR: Guamán Coronel, Daniel Alejandro, Mgs.

LOJA-ECUADOR

2016

Septiembre, 2016

APROBACIÓN DEL DIRECTOR DEL TRABAJO DE TITULACIÓN

Magister.

Daniel Alejandro Guamán Coronel

DIRECTOR DEL PROYECTO DE TITULACIÓN

De mi consideración:

El presente trabajo de titulación: “Consideraciones de arquitectura de software a nivel de diseño arquitectónico y desarrollo de software para minimizar vulnerabilidades en aplicaciones web basados en OWASP Top Ten 2013, caso de estudio Arquitectura 3-Layers”, realizado por Juan Keyner Abarca Cabrera, ha sido orientado y revisado durante su ejecución, por cuanto se aprueba la presentación del mismo.

Loja, mayo de 2016

f)……

Mgs. Daniel Alejandro Guamán Coronel

CI. 1103777403

DECLARACIÓN DE AUTORÍA Y CESIÓN DE DERECHOS

"Yo, Juan Keyner Abarca Cabrera, declaro ser autor del presente trabajo de titulación: "Consideraciones de Arquitectura de Software a nivel de Diseño Arquitectónico y Desarrollo de Software para minimizar vulnerabilidades en aplicaciones web basados en OWASP Top Ten 2013 caso de estudio 3-Layers", de la Titulación de Sistemas Informáticos y Computación, siendo el Ing. Daniel Alejandro Guamán director del presente trabajo; y eximo expresamente a la Universidad Técnica Particular de Loja y a sus representantes legales de posibles reclamos o acciones legales. Además certifico que las ideas, conceptos, procedimientos y resultados vertidos en el presente trabajo investigativo, son de mi exclusiva responsabilidad.

Adicionalmente declaro conocer y aceptar la disposición del Art. 88 del Estatuto Orgánico de la Universidad Técnica Particular de Loja que en su parte pertinente textualmente dice: "Forman parte del patrimonio de la Universidad la propiedad intelectual de investigaciones, trabajos científicos o técnicos y tesis de grado o trabajos de titulación que se realicen con el apoyo financiero, académico o institucional (operativo) de la Universidad".

f)……..……..

Autor: Juan Keyner Abarca Cabrera

Cédula: 1104417942

DEDICATORIA

Dedico este trabajo a mis padres, hermanos por haberme brindado su apoyo incondicional en todo momento, ya que gracias a ellos he podido culminar mis estudios superiores.

A mi familia como es mi esposa Mariuxi, y mi hija Ailin, por darme animo en los momentos difíciles.

A mis compañeros y amigos de estudio y de trabajo, que de una u otra manera han formado parte, de este aprendizaje.

A mis maestros que me guiaron y me impartieron desinteresadamente sus conocimientos.

AGRADECIMIENTO

Primeramente a Dios por darme la vida y salud.

A mis padres por sus consejos y brindarme su apoyo siempre.

A mi esposa Mariuxi por estar con migo en todo momento.

A mi director de tesis Mgs, Daniel Guamán por el apoyo constante y ser una guía en este trabajo.

A la Universidad Técnica Particular por darme la oportunidad de prepararme profesionalmente como Humanísticamente.

ÍNDICE DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

El presente trabajo de titulación se enfoca en el análisis y prevención de las vulnerabilidades Broken Authentication Session Management y Unvalidated Redirects and Forwards que se encuentran listadas dentro de OWASP, este tipo de vulnerabilidades pueden estar presentes en las aplicaciones web poniendo en riesgo la integridad, disponibilidad y confidencialidad de los datos. Para realizar el análisis se construye un prototipo bajo el esquema del estilo arquitectónico 3-Layers en el cual se aplicó técnicas y recomendaciones de OWASP, además sobre él se implementa el framework de seguridad Apache Shiro, patrones de diseño FACADE y DAO, con la finalidad de prevenir las vulnerabilidades antes mencionadas y de esta manera garantizar la seguridad en cuanto a autenticación y autorización de los usuarios finales.

Como parte final del trabajo, se validó el prototipo con herramientas especializadas de forma automática y manual, dicha validación se realizó a nivel de diseño arquitectónico, escritura de código con lo que se pretende evaluar la calidad del software.

PALABRAS CLAVES: 3-Layers, Apache Shiro, Aplicación Web, Arquitectura de Software, Estilo, Framework, OWASP Top Ten, Vulnerabilidades.

ABSTRACT

This research Project was focused on the analysis and prevention of vulnerabilities named “Broken Authentication Session Management” and “Unvalidated Redirects” and Forwards that are listed in The OWASP, this kind of vulnerabilities may be present in web applications, putting in high risk the integrity , availability and confidentiality of data;

To do the analysis I used a prototype under the scheme of architectural style 3-Layers in which I applied techniques and recommendations OWASP, also I implemented in it the security framework named Apache Shiro, design patterns like FACADE and DAO, with the main objective of preventing vulnerabilities mention above and thus ensure safety in terms of authentication and authorization of end users.

As the last phase of the reseach project, the prototype was validated with tools designed for evaluation and in a more personal way by the developer, I did this both at the level of architectural design, and writing code to check the quality of the software that was validated.

KEY WORDS: 3-Layers, Apache Shiro, Software Architecture, OWASP Top Ten, Framework, Style, Vulnerabilities, Web Application.

INTRODUCCIÓN

Las aplicaciones de software que se despliegan en la web, ofrecen beneficios como accesibilidad, portabilidad y disponibilidad; este despliegue trae consigo riesgos de seguridad tales como robo de identidad de usuario, autorización insuficiente, recuperación fraudulenta de contraseñas, entre otras. Para evitar dichos riesgos de seguridad organizaciones como OWASP recomiendan la implementación de métodos, procedimientos y estándares de seguridad a nivel de autenticación y autorización desde el punto de vista de escritura de código.

Por lo general, la mayoría de los riesgos de seguridad ocurren por aspectos como escritura defectuosa de código, ausencia de estándares y patrones de diseño, inadecuada selección de un estilo arquitectónico, entre otros. Por este motivo es necesario que desde las primeras etapas de desarrollo de software se considere y se implemente mecanismos de seguridad con el fin de proteger los datos e información de las organizaciones.

En el presente trabajo se aplican recomendaciones y estándares que permitan garantizar la seguridad en las aplicaciones web; para lo cual se toma como referencia las especificaciones dadas por OWASP y se trabaja en la mitigación de dos vulnerabilidades que pueden ocurrir en este tipo de aplicaciones: 1) Broken Authentication Sesion Management y 2) Unvalidate Redirects and Forwards; mismas que se encuentran catalogadas como la tercera y décima posición dentro del top ten de OWASP.

Para su validación se construyó un prototipo basado en el estilo arquitectónico 3-Layers con el propósito de analizar el uso y aplicación de estándares, métodos y técnicas para proporcionar seguridad, ésta validación se la realiza a nivel de diseño y codificación con el fin de verificar correcto uso e implementación de patrones de diseño, frameworks de seguridad, entre otros.

El presente trabajo de fin de titulación contiene cinco capítulos; en el capítulo uno, se realiza la conceptualización de términos referente a arquitectura de software, estilos y patrones arquitectónicos, 3-Layers, recomendaciones de OWASP, para tener una visión global del tema de estudio. En el capítulo dos, se presenta la propuesta de solución del prototipo basado en el estilo arquitectónico 3-Lyers y los patrones de diseño, framework de seguridad y tecnologías que se van a utilizar para el desarrollo del mismo, en el capítulo tres se menciona el diseño arquitectónico del prototipo especificando las medidas y técnicas de seguridad que se implementará. En el capítulo cuatro se detalla la implementación del prototipo, detallando los diagramas de paquetes y clases que se implementó; así como de la codificación y

configuración del framework de seguridad Apache Shiro con las técnicas de seguridad de OWASP para minimizar las vulnerabilidades antes expuestas. Finalmente, en el capítulo cinco, se detallan los resultados de las pruebas y validaciones al prototipo para verificar la calidad del software tanto a nivel de diseño, codificación y seguridad.

GLOSARIO

Arquitectura de software.- es una estructura que sirve como guía en la construcción de un software la cual se la define en la fase inicial del proyecto.

3-Layers.- Estilo arquitectónico de software estructurado en capas, que se utiliza para construir aplicaciones web.

Software.- Es un conjunto de programas, funciones que permiten realizar distintas operaciones en un equipo informático.

Vulnerabilidad.- Se entiende como vulnerabilidad a una debilidad en una determinada área, la cual puede ser explotada por un atacante de software.

OWASP.- Organización sin fines de lucro encargada de proveer de información sobre vulnerabilidades informáticas.

Framework.- Es un marco de trabajo que permite construir un software de forma ordenada y estructurada.

Apache Shiro.- Framework de seguridad que permite implementar seguridades en las aplicaciones web java.

Autenticación.- Método para identificar la identidad de un usuario sea quien dice ser, permitir o denegar el ingreso al Sistema.

Autorización.- Método que autoriza el acceso a los módulos del sistema a los usuarios de acuerdo a los roles y privilegios asignados.

Broken Authentication Sesion Management.- Vulnerabilidad de perdida de autenticación y gestión de sesiones que se encuentra presente en aplicaciones informáticas.

Unvalidate Redirects and Forwards.- Vulnerabilidad de reenvíos y redirecciones no válidas, que se producen en una aplicación web.

Datos.- Es una representación de un suceso, pueden ser representados a través de una letra o números.

Servidor de aplicaciones.- Es equipo o un software en donde se ejecutan una o varias aplicaciones web.

Base de datos.- Es un repositorio de datos, donde son almacenados de una forma estructurada y ordenada.

Seguridad.- Es una disciplina de diseño e implementación de métodos y técnicas que permiten proteger datos sensibles de una aplicación informática.

Java.- Lenguaje de programación en donde se puede diseñar, construir aplicaciones informáticas.

Glassfish.- Es un tipo de servidor de aplicaciones Java que permite ejecutar múltiples aplicaciones.

HTTPS.- Protocolo de trasporte seguro de Hipertexto.

Sniffer.- Es una aplicación que captura los datos de usuario que se transportan por una red.

CAPITULO I

ESTADO DEL ARTE

1.1. Arquitectura de software

1.1.1. Definición.

La arquitectura de software se puede definir como la “estructura o estructuras del sistema, que comprenden los componentes del software, las propiedades externamente visibles de esos componentes, y las relaciones entre ellos” (Gardazi & Shahid, 2009).

Según Bass, Clements, & Kazman (2003) la arquitectura de software sirve como una guía, cuando se desarrolla un software este crece y hace difícil que una sola persona pueda diseñar, planificar y entender el mismo.

La arquitectura de software se convierte en un factor indispensable en cuanto a la planificación, organización y diseño, con el fin que los miembros del equipo de desarrollo de sistemas no tengan problemas en la construcción del software.

1.1.2. Características de la arquitectura de software.

La arquitectura de software tiene las siguientes características (Tortosa, 2006):

- ✓ Permite definir las propiedades del software.
- ✓ Exige llevar una buena organización y control del proyecto de software.
- ✓ Permite analizar y evaluar el software durante su construcción.
- ✓ Representación de alto nivel de la estructura del sistema.
- ✓ Puede incluir los patrones que supervisan la composición de sus componentes.

1.1.3. Calidad de la arquitectura de software.

La calidad de la arquitectura de software es el conjunto de propiedades inherentes a una entidad las cuales permiten juzgar y cuantificar el valor que se le dan a las mismas; de esta manera la calidad es subjetiva y circunstancial: Es subjetiva porque depende de los atributos elegidos para medirla y es circunstancial porque el conjunto de atributos seleccionados puede variar en situaciones diferentes (Dávila & Melendez, 2006).

Según las normas ISO ISO/IEC 9126 describe la calidad de una arquitectura de software en las siguientes características:

- ✓ **Funcionalidad.-** Es la capacidad del producto de software para proporcionar funciones adecuadas que satisfacen las necesidades expuestas, esto se da cuando se usan condiciones específicas que solicita el cliente. La funcionalidad evalúa el conjunto de

características y capacidades del programa, para aquello debemos tener en cuenta los siguientes parámetros:

- Adecuación.
- Exactitud.
- Tolerancia a fallos.

✓ **Eficiencia.-** La eficiencia en software se mide en los recursos utilizados en cuanto a calidad para producir un software confiable y el mismo sea accesible para los usuarios en cuanto a costos**;** para lograr una eficiencia adecuada se debe evaluar lo siguiente:

- Eficiencia en ejecución.
- Eficiencia en almacenamiento.
- Almacenamiento necesario.
- Utilización de recursos.

✓ **Integridad.-** Es un atributo importante dentro de un software ya que controla el acceso de los usuarios o a los datos y de esta manera proteger y garantizar la información de un sistema. La integridad toma en cuenta los siguientes aspectos:

- Control de accesos.
- Facilidad de auditoria.
- Seguridad.

✓ **Facilidad de Uso.-** El software debe ser fácil e intuitivo con el fin que el usuario lo entienda e interprete al mismo en cuanto: Al ingreso de información, mensajes de alertas y la salida de información. Para aquello se debe tener en cuenta los siguientes parámetros:

- Facilidad de Operación.
- Facilidad de Comunicación.
- Facilidad de aprendizaje.
- Formación.

1.1.4. Importancia del uso de una arquitectura de software.

El uso de la arquitectura de software en todo el ciclo de construcción del software es fundamental para que el proyecto no fracase; según (Pressman, n.d.) Existen tres razones porque es importante una arquitectura de software en los distintos proyectos:

✓ Permite la comunicación entre todas las partes del desarrollo de un sistema.

✓ Constituye un modelo relativamente pequeño e intelectualmente comprensible de cómo está estructurado el sistema y como trabajan sus componentes de forma conjunta.

- ✓ Destaca las decisiones iniciales relacionadas con el diseño, las cuales tendrán un impacto importante en todo el trabajo de la ingeniería de software, y el éxito de todo el proyecto.

La arquitectura de software es la parte fundamental dentro del ciclo de desarrollo del software en el desarrollo del sistema; la cual se la debe definir en etapas tempranas del desarrollo, ayuda a satisfacer los atributos de calidad como el desempeño, seguridad, integridad y eficiencia sirviendo como una guía en el desarrollo del proyecto.

1.2. Estilos Arquitectónicos

1.2.1. Definición.

Según Buschmann, Meunie, Rohnert, Sommerlad, & Stal (2001), los estilos arquitectónicos son la base en la estructura de un sistema de software, proporcionan soluciones a los distintos problemas que se presentan en un sistema. Asociando métodos que especifican la forma de implementarlo, cuando usarlo y su impacto.

Fielding (2000) menciona que un estilo arquitectónico está formado por condiciones arquitectónicas que limita los roles, funciones de los elementos y las relaciones que se encuentran permitidas entre ellos, todo esto se da dentro de cualquier arquitectura que se ajuste al estilo.

1.2.2. Tipos.

Existen varios tipos de estilos arquitectónicos diferenciados por sus características, descritas a continuación:

1.2.2.1. Componentes.

Permite reutilizar piezas de código pre-elaborado que permiten realizar diversas tareas, conllevando a diversos beneficios como mejoras a la calidad, la reducción del ciclo de desarrollo y el mayor retorno sobre la inversión (Terreros, 2010). Entre principales características se mencionan (Peláez Juan, 2009):

- ✓ El estilo arquitectónico basado en componentes está diseñado para aplicaciones que se encuentran conformadas por componentes individuales.
- ✓ Permite dividir el software en componentes lógicos que constan con interfaces bien definidas.
- ✓ Permite dar un primer diseño de los componentes, realizan la comunicación a través de interfaces que contienen métodos y eventos.

1.2.2.2. Flujo de datos.

Esta arquitectura es aplicada cuando los datos de entrada son trasformados a través de una serie de componentes computacionales, y se basa en un patrón de tuberías y filtros, entre las principales características tenemos:

- ✓ Cada filtro trabaja de manera independiente de los componentes que se encuentran situados antes o después de ella.
- ✓ Los datos de salida están en un formato específico (Wolfgang , s.f.).

1.2.2.3. Centrados en datos.

En el centro de esta arquitectura se encuentra un conjunto de datos al que otros componentes acceden con frecuencia para actualizar, añadir, borrar o modificar los datos. El software de lado del cliente accede a los datos centrales, es decir accede a los datos de una forma independiente a cualquier cambio en estos o las acciones del cliente. El estilo arquitectónico centrado en datos tiene las siguientes características:

- ✓ Los componentes pueden realizar operaciones sobre los datos.
- ✓ El usuario accede a los datos de una forma independiente a las operaciones que se pueden hacer (Rosenblum, 2007).

1.2.2.4. Arquitecturas de llamada y retorno.

Permite al arquitecto de software diseñar una estructura de un sistema de una manera relativamente fácil de modificar y ajustar a escala. La arquitectura de llamada y retorno consta con dos subdivisiones fundamentales que son (Scientist, 2008):

- ✓ **Arquitectura de programa principal**: Clasifica la programación y descompone las funciones en una jerarquía de control donde un programa principal llama a un determinado número de componentes.
- ✓ **Arquitectura de llamada de procedimiento remoto:** Los componentes de una arquitectura de programa principal/subprograma, están distribuidos entre varias computadoras conectadas entre sí a través de una red.

1.2. Patrones

Los patrones de software permiten resolver problemas que se presentan en el desarrollo de un proyecto, por esta razón se debe que elegir adecuadamente el tipo de patrón a utilizar, cada tipo de patrón se asocia con un determinado problema, es por esto se puede utilizar varios patrones en un proyecto de software.

Al hablar de un patrón que ayuda a resolver problemas, debemos de tener en cuenta la estructura del mismo, un patrón arquitectónico se define mediante un esquema en el cual denota las reglas que establecen relaciones en un contexto de un problema dado, por lo cual el esquema está conformado por tres elementos: Contexto, Problema y Solución, como se muestra en la Figura 1, el cual captura la esencia de un patrón independientemente de su dominio.

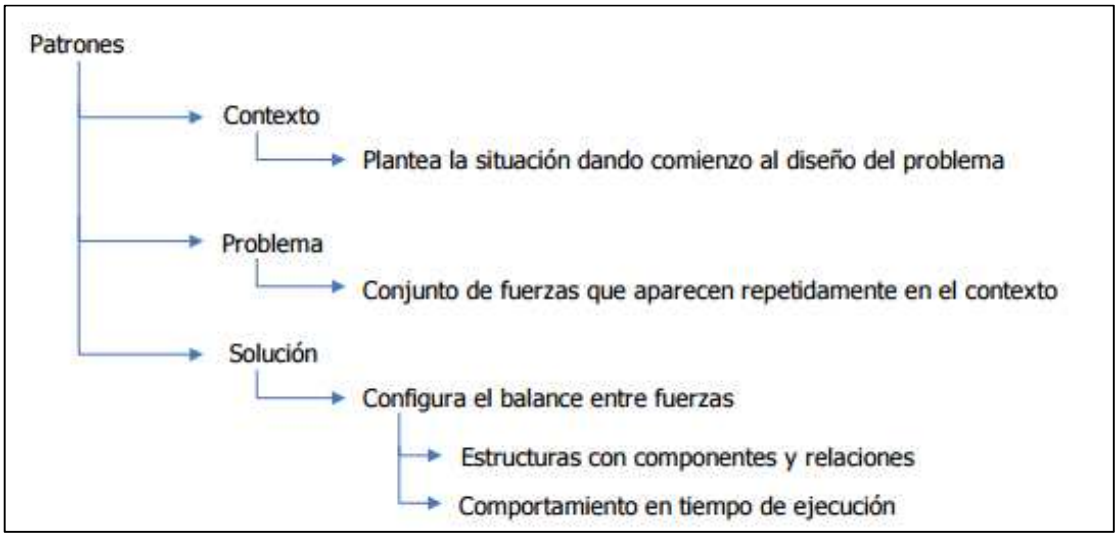

Figura 1. Elementos de un patrón.
Fuente: Autor.
Elaboración: Autor

1.2.1. Patrones Arquitectónicos

1.2.1.1. Definición.

Según Adriana Venete, (2011) Los patrones arquitectónicos representan el nivel más alto dentro del sistema y expresan un esquema de la estructura fundamental de la organización para sistemas de software. Proporcionan subsistemas previamente definidos especificando responsabilidades, reglas y guías con el fin de facilitar la comunicación. Cada patrón ayuda a lograr una propiedad específica del sistema de una forma global como es la adaptabilidad de la interfaz de usuario.

A un patrón arquitectónico se lo representa como una plantilla en donde se debe describir todos los requisitos de la arquitectura de software, permitiendo a la organización exprese de forma clara las responsabilidades que tiene cada elemento que se encuentra inmerso en el desarrollo de un software

1.2.1.2. Tipos de patrones arquitectónicos

En la Tabla 1 se describen los diferentes patrones de diseño con su descripción y sus características.

Tabla 1. Clasificación de patrones arquitectónicos

Patrón Arquitectónico	Descripción	Características
Capas	Las operaciones se divide en tres elementos denominadas capas, con un reparto claro de sus funciones: una capa para la presentación otra para el cálculo (donde se encuentra modelado el negocio) y otra para el almacenamiento (persistencia); una capa solamente tiene relación con la siguiente.	✓ Asigna roles y responsabilidades a cada capa. ✓ Permite la independencia de las capas lógicas; logrando que el sistema sea flexible a cambios. ✓ Las capas lógicas de la aplicación no necesitan estar en diferentes equipos. ✓ Permiten redistribuir el trabajo por capas.
Tubos y Filtros	Permite acoplar componentes mediante dispositivos de forma que cualquier proceso sea ejecutado en secuencia. Y el resultado se transporta por tuberías y filtros.	✓ Cada componente tiene un conjunto de entradas y un conjunto de salidas. ✓ Los componentes se conocen como filtros y son independientes. ✓ Los conectores se comportan como conductores ráfagas, transmitiendo salidas de un componente hacia entradas de otro.
Pizarra	En una arquitectura modelo pizarra, se llama pizarra a una estructura de datos global donde se almacena toda la información. Las fuentes de datos producen cambios en la	✓ Las fuentes del conocimiento no pueden comunicarse entre sí.

	pizarra que llevan incrementalmente a la solución del problema. Las comunicaciones e interacciones entre las fuentes de datos se producen solamente a través de la pizarra.	✓ Permiten aplicar razonamiento en varios niveles. ✓ Permite un control dinámico de las actividades.
Broker	Es una arquitectura que puede ser usada para estructurar sistemas de Software distribuido con componentes desacoplados que interactúan por invocaciones a servicios remotos. Un componente broker es responsable de coordinar la comunicación, como el reenvío de solicitudes, así como también la transmisión de resultados y excepciones	✓ Reduce la complejidad en el desarrollo de las aplicaciones distribuidas ✓ Ofrecen una ruta para la integración de las tecnologías en la distribución de los objetos. ✓ Extienden los modelos de objetos desde aplicaciones individuales hasta aplicaciones distribuidas.
Modelo Vista Controlador	Patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos: Modelo, Vista y controlador	✓ Separar los datos de la representación visual de los mismos. ✓ Es fácil agregar múltiples representaciones de los mismos datos o información. ✓ Permite agregar nuevos tipos de datos según sea requerido por la aplicación. ✓ Crea independencia de funcionamiento. ✓ Facilita el mantenimiento en caso de errores.

Fuente: (Gastón, 2008; Buschmann et al., 2001).
Elaboración: Autor.

1.2.2. Patrones de diseño de software

1.2.2.1. Definición.

Los patrones de diseño brindan soluciones probadas para los problemas comunes que se encuentran al desarrollar una aplicación web. Cuando se construye una aplicación web siempre se la diseña como única y esto no es tan cierto, siempre existen partes comunes con otros sistemas como: el acceso a datos, reglas de negocio, módulos de seguridad entre otros; es por estos motivos que en lugar diseñar y crear un propio patrón de diseño, se puede dar solución a los problemas que se está abordando haciendo uso de un patrón probado y utilizado por otros programadores (Charlascylon, 2014).

Los patrones de diseño pueden acelerar el proceso de desarrollo de un sistema, se considera como un documento que permite definir una estructura de clases, las cuales permiten dar soluciones y con la reutilización de los mismos ayudan a prevenir problemas en el software (Pavlova, 2010).

1.2.2.2. Objetivos de los patrones de diseño.

Los principales objetivos que cumplen los patrones de diseño dentro de los proyectos de software son:

- ✓ Permiten la reducción de tiempos de construcción de un software.
- ✓ Facilita el mantenimiento del sistema.
- ✓ Aumenta la eficiencia en resolver problemas en el diseño de un software.
- ✓ Asegura la consistencia durante el desarrollo del sistema.
- ✓ Aumenta la fiabilidad.
- ✓ Protege la inversión en el desarrollo de un proyecto de software.

1.2.2.3. Clasificación de Patrones de Diseño

Los patrones de diseño ofrecen soluciones generales, estos son documentados en un formato que no requiere características específicas vinculadas a un problema particular; a continuación los describimos.

- ✓ **Patrones Estructurales**.- Los patrones de diseño estructurales están enfocados en la gestión de forma en la que las clases y los objetos se combinan para dar lugar a estructuras más complejas.
- ✓ **Patrones de Comportamiento**.- Son utilizados para resolver problemas relacionados con el comportamiento de la aplicación en tiempo de ejecución. Nos permiten definir

la comunicación entre los objetos del sistema y el flujo de la información; a continuación se describen los patrones que pertenecen a este grupo (Guerrero, 2013).

- ✓ **Patrones Creacionales**.- Los patrones creacionales permiten instanciar y crear objetos, y a su vez ser encapsulada y abstraída. Logrando así la independencia con el sistema.

En la Tabla 2 se muestra los patrones de diseño organizados de acuerdo al tipo de patrón que pertenecen, una breve descripción y su objetivo.

Tabla 2. Clasificación de los patrones de diseño

Tipo	Patrón	Descripción	Objetivo
Creación	Abstract Factory	Declara una interfaz para las operaciones que crean objetos de productos abstractos.	✓ Define un método en una clase de cual se crean instancias de otras clases. ✓ Proporciona una interfaz para crear familias de objetos relacionados o que dependen entre sí, sin especificar sus clases concretas.
	Factory Method	Crea una instancia de varias clases derivadas.	✓ La clase principal delega a las subclases la creación de los objetos. ✓ Una clase no puede prever la clase de objetos que tiene que crear.
	Prototipo	Un ejemplo inicializado completamente para ser copiada o clonada.	✓ Permite crear y manipular copias de otros objetos. ✓ Permite evitar las subclases de un objeto creador como hace el patrón Abstract Factory.

Estructurales	Adapter	Convertir la interfaz de una clase en la interfaz que el cliente espera.	✓ Permite que clases trabajen juntas. Se los utiliza para transformar una interfaz.
	Bridge	Desvincula una abstracción de su implementación, de manera que ambas puedan variar de forma independiente.	✓ Combina las abstracciones e implementaciones en muchas clases distintas. ✓ Implementa las abstracciones e implementaciones como clases independientes que se pueden combinar dinámicamente.
	Facade	Simplifica o hace más amigable la complejidad asociada a una biblioteca de software.	✓ Simplifica la comprensión y el de una biblioteca de software. Encapsula una interfaz poco amigable en una más consistente o mejor estructurado.
Comportamiento	Command	Permite parametrizar operaciones, de tal forma que se pueda controlar su selección y secuencia, ponerlas en la cola, deshacerlas y manipularlas	✓ Permite parametrizar operaciones, encapsula una petición en un objeto.
	Iterator	Dado un lenguaje, define una representación de su gramática junto con un intérprete que usa dicha representación para interpretar las sentencias del lenguaje.	✓ Se usa para definir un lenguaje para representar expresiones regulares. ✓ Permite representar las distintas instancias de una familia de problemas.

	Observer	Existe una dependencia de uno a varios objetos y cuando cambia un objeto de estado el resto son notificados y cambia de estado automáticamente.	✓ Mantienen la consistencia entre los objetos relacionados sin aumentar el acoplamiento entre las clases.

Fuente: (EcuRed, 2014)
Elaboración: Autor.

En la sección anterior se definió los conceptos sobre los estilos arquitectónicos, patrones arquitectónicos y patrones de diseño, en la Figura 2 se presenta la relación que existe entre ellos y su categorización en sus distintos niveles.

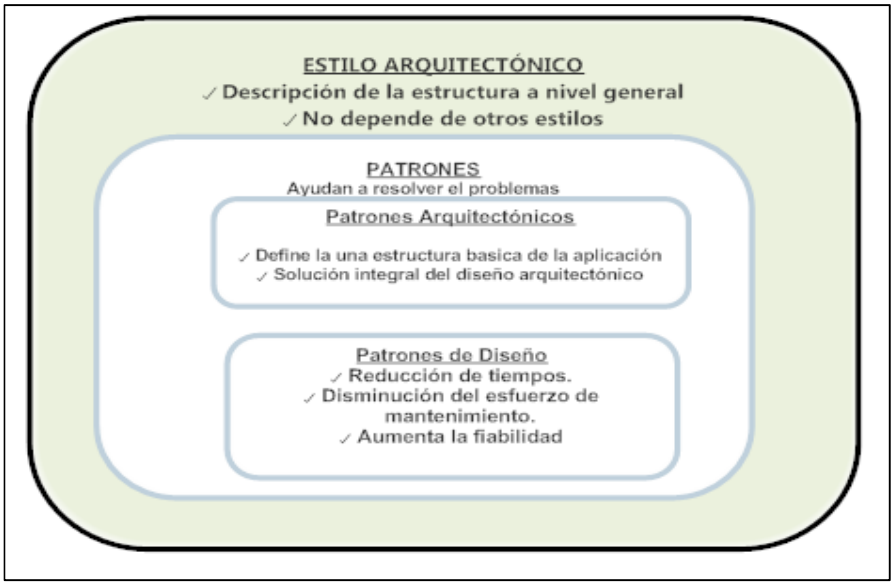

Figura 2. Relación entre estilos y patrones de software
Fuente: Autor.
Elaboración: Autor

La arquitectura en capas aumenta la flexibilidad y facilidad de mantenimiento de un software, haciendo que este estilo arquitectónico se encuentre estructurado y organizado en capas: Capa de presentación (Es la interfaz de usuario que interactúa directamente con el cliente) , capa lógica de negocio (donde se encuentra la lógica de programación), capa de acceso a datos (la forma de almacenar y extraer datos), esto permite que un sistema sea escalable, en cuanto a la implementación de nuevos requerimientos o su vez permita realizar modificaciones en la capa que corresponde, de manera que no afecte a todo el sistema. A continuación se explica con mayor detalle el estilo arquitectónico basado en 3-layers.

1.3. Estilo Arquitectónico 3-Layers

1.3.1. Definición.

Según Sommerville,(2009) el estilo arquitectónico 3-layers organiza un sistema en capas, cada una de las cuales proporciona un conjunto de servicios que se enfoca en la distribución de roles y responsabilidades de forma jerárquica.

Peláez, (2009) nos indica que el estilo arquitectónico basado en capas se encuentra distribuida en roles y responsabilidades de una forma jerárquica, el rol indica el modo y tipo de interacción con otras capas, y la responsabilidad indica las operaciones que debe cumplir en el sistema para lo que fue desarrollada.

El estilo arquitectónico 3-layers está formado por tres elementos principales; capa de presentación, capa lógica de negocios y capa de enlace de datos; sin embrago el programador puede definir varias capas según lo crea conveniente. En las definiciones anteriores se mencionó que cada uno de estos elementos tiene responsabilidades dentro del funcionamiento de un sistema, es por esto que es importante definir una arquitectura principal antes de empezar a desarrolla un software.

Cuando se menciona el estilo arquitectónico 3-layers siempre existe la tendencia a confundir con Tiers, por aquello se da una breve explicación de las diferencias entre estos dos estilos. Layers se encarga de dividir la lógica de componentes y funcionalidades sin tomar en cuenta la ubicación física donde se encuentren los servidores, en cambio Tiers si toma en cuenta la ubicación física de los servidores y su distribución.

1.3.2. Características del estilo arquitectónico 3-Layers.

La arquitectura 3-layers es una evolución de las arquitecturas anteriores como 2-capas, ha incorporado conceptos nuevos, como la distribución lógica de las capas, asignación de roles y responsabilidades en cada capa, y adopta características propias las cuales se las describe a continuación (García Bautista, 2014):

- ✓ Separación de roles, responsabilidades y funcionalidades específicas en tres capas, haciendo fácil su modificación sin interferir entre ellas.
- ✓ Permite la reutilización del código.
- ✓ La capa intermedia puede ser reutilizada por varias aplicaciones.
- ✓ La migración de la base de datos se puede realizar sin grandes impactos en el resto de la aplicación.
- ✓ Permite la modificación del Front de la aplicación sin afectar la lógica de negocio ni la base de datos

- ✓ La distribución de las capas puede estar en la misma máquina física como en diferentes computadores.
- ✓ La comunicación entre los diferentes componentes se realiza por medio de interfaces bien definidas.
- ✓ Permite a los diseñadores crear la interfaz gráfica, mientras los programadores escriben el código de la aplicación.

1.3.3. Ventajas y desventajas de Arquitectura 3-Layers.

Al implementar el estilo arquitectónico 3-Layers en una aplicación web se logra obtener una estructura definida, ordenada y separada la codificación implementando el principio "divide y vencerás". La división de cada capa se lo realiza de acuerdo a la funcionalidad, lo cual facilita la modificación de cualquier capa sin alterar al resto (Vargas & Maltés, 2007). En la Tabla 3 se describen las ventajas y desventajas del estilo arquitectónico 3-Layers.

Tabla 3. Ventajas y desventajas 3-Layers.

Ventajas	Desventajas
Flexibilidad.- El software es susceptible a cambios, estos se los realiza de una manera organizada, y planificada en cada elemento o capa que se encuentra conformado el sistema.	Al realizar determinados cambios no se logra realizarlo de forma independiente en cada capa y se requiere una cascada de cambios en varias capas.
Mantenibilidad.- Si un software presente un error, este puede ser corregido oportunamente y de una forma efectiva sin que afecte a todo el sistema.	Pérdida de eficiencia del sistema al no lograr la independencia entre las capas.
Reutilización Código.- Se puede reutilizar código en los diferentes módulos del sistema.	Trabajo innecesario de las capas internas al redundar procesos.
Encapsulamiento.- Con el uso de patrones de diseño podemos encriptar código.	Dificultad de diseñar correctamente la granularidad de las capas con el fin de reutilizar las partes importantes de la misma.
Escalabilidad.- Un sistema desarrollado en 3-Layers permite que siga creciendo y desarrollándose con los nuevos requerimientos.	Se hace necesario incrementar recursos a nivel de hardware para lograr una escalabilidad y un funcionamiento adecuado del sistema.

Fuente:(EcuRed, 2014)
Elaboración: Autor.

1.3.4. Capas de la arquitectura 3-Layers.

El estilo arquitectónico 3-Layers se compone por tres elementos principales: capa de presentación, capa de negocios y enlace de datos; en la Figura 3 se representa los tres elementos que lo conforman.

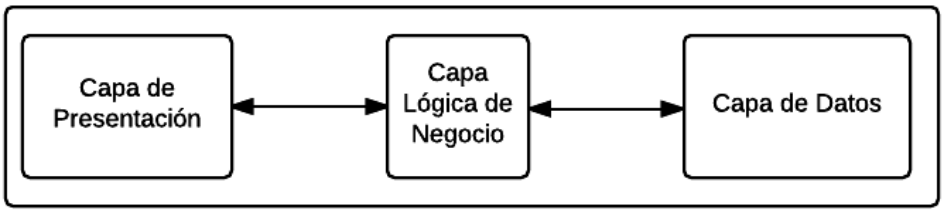

Figura 3. Elementos del estilo arquitectónico 3-Layers.
Fuente: (Echeverría, Ballari, Molina, Wainerman, & Olsina, n.d.).
Elaboración: Autor.

1.3.4.1. Capa de presentación.

La capa de presentación es el primer elemento del estilo arquitectónico 3-layers, la cual interactúa directamente con el usuario del sistema, permite a la capa de negocio mostrar y controlar el flujo de la información que se presentará a través de esta capa.

Es la capa donde se presenta los datos o información del programa ante el usuario, es por esto se debe manejar interfaces amigables para el usuario, facilitando la interacción con la aplicación mediante la utilización de patrones definidos. Las interfaces además de amigables deben ser fáciles de utilizar ya que el usuario es el encargado de manejar el sistema y dar retroalimentación al equipo de desarrollo para corregirlo y mejorarlo.

Las interfaces deben ser consistentes con la información que se maneje, es decir no requerir información innecesaria y solo la especificada de forma clara y concisa, las interfaces deben satisfacer los requerimientos del usuario, sin excluir la información solicitada ni incluir la información no solicitada por el usuario. En la capa de presentación existen elementos, requerimientos de usuario que se debe definir a la hora de implementarla; según Valencia & Gonzales,(2011) tenemos lo siguiente.

- ✓ **La interfaz de usuario:** Ofrece a los usuarios información, sugerencias, acciones y captura los datos de entrada que el usuario final proporciona a la aplicación web.
- ✓ **La lógica de presentación:** Se ejecuta en cliente y hace referencia a todo el procesamiento requerido para mostrar datos y transformar los datos de entrada en acciones que podemos ejecutar contra el servidor.

- ✓ **La lógica de control de peticiones:** Reside en el servidor y se encarga de interactuar con la lógica de presentación, encargándose principalmente de transformar el modelo de datos para la recepción/envío, gestionar las reglas de navegación y enlazar con la capa de servicios de negocio.
- ✓ **Entender los seguimientos:** Se debe identificar los usuarios, los procesos para cada tarea, formatos de presentación, tecnología de desarrollo para que la interfaz sea intuitiva y fácil de entender.
- ✓ **Elegir la tecnología:** Este paso depende mucho del anterior, existen varias tecnologías para los diferentes requerimientos; un factor importante que se debe utilizar para elegir el tipo de tecnología es su licencia, si es pagada o libre.
- ✓ **Diseño de los componentes:** Permite la organización de funcionalidades en los componentes; estos residen en la capa de presentación. Entre los principales tenemos.
    - Componentes de interfaz de usuario.
    - Componentes lógicos de la Presentación.
    - Componentes de los Modelos de la Presentación.

- ✓ **Determinar los requerimientos del enlace:** Brinda una manera de mantener un enlace entre los controles de interfaz de usuario y los datos o componentes de la capa lógica. Existen dos maneras de realizar esta tarea:
    - Los cambios en la fuente o destino de los datos se actualizan automáticamente en el otro, pero únicamente en una dirección.
    - Los cambios tanto en la fuente como en el destino son actualizados automáticamente.
- ✓ **Determinar una estrategia para el manejo de errores:** Se debe desarrollar una estrategia de manejo de errores en un componente por separado por cada capa. Por lo cual se debe tener en cuenta los siguientes aspectos:

    - Los log de excepciones, esto nos permitirá llevar un control de los errores a un detalle útil para su corrección.
    - Mostrar mensajes amigables al usuario, los cuales notifiquen las razones de los errores.
    - Permitir reintento, mostrar la razón del fallo y permitir reintento de la operación. Esto permitirá recuperar la operación de la aplicación ante errores temporales.

- ✓ **Determinar una estrategia de validación:** Se debe validar los datos de entrada, con el fin de filtrar datos no deseados o maliciosos. La validación de entradas de datos se la realiza en la capa de presentación y la validación de reglas de negocio se lo desarrolla en la capa de negocio.

La capa de presentación es un elemento fundamental dentro de la estructura de un sistema, la cual tiene funciones y responsabilidades tanto con el usuario y el sistema. según (Vignaga & Perovich, 2014) las principales responsabilidades son:

- ✓ La presentación de datos al usuario final.
- ✓ Controlar la interfaz de la aplicación.

1.3.4.2. Capa lógica de negocio o control.

La capa lógica de negocio denominada en ingles Business Logic Layer con sus siglas (BLL) según Microsoft (2009) en la Application Architecture Guide segunda edición, nos dice que se establecen todas las reglas, controles y el procesamiento de los datos; recibe las peticiones por parte del usuario a través de la capa de presentación. Esta capa contiene la lógica del programa, las estructura de datos y objetos encargados del manejo y procesamiento de los datos del usuario en la capa de presentación**.**

La capa de negocio es la capa central de la aplicación, por medio de ésta se comunican las capas de presentación y la capa de datos para cumplir con las funcionalidades. Esta capa recibe los datos que ingresa el usuario en la capa de presentación para procesarlos mediante el encapsulamiento y mostrarlos al usuario final en la capa de presentación.

Entre los principales componentes que constan en la capa lógica de negocio son:

- ✓ **Componentes de negocio.-** Encapsulan las reglas de negocio a la forma que los datos adquiridos desde la capa de presentación deben ser manipulados acorde al problema a resolver. Estos componentes pueden variar de acuerdo con los requerimientos de negocio; los componentes de negocio como parte fundamental dentro de la capa lógica se encargan de lo siguiente: (Arévalo, n.d.).
    - Pueden ser llamados desde las interfaces de servicio, flujos de negocio u otros componentes de negocio.
    - Verificar las entradas y las salidas.
    - Se comunican con la capa de acceso a datos, para obtener o actualizar los datos de la aplicación.
    - Pueden llamar componentes de negocio u otros flujos de negocios.

En la Figura 4 se muestra interacción de los componentes de negocios con los demás componentes de la arquitectura en donde la capa de presentación se comunica a través de la interfaz de servicio con los componentes de negocio y con los de accesos a datos también se puede realizar llamadas a otros servicios.

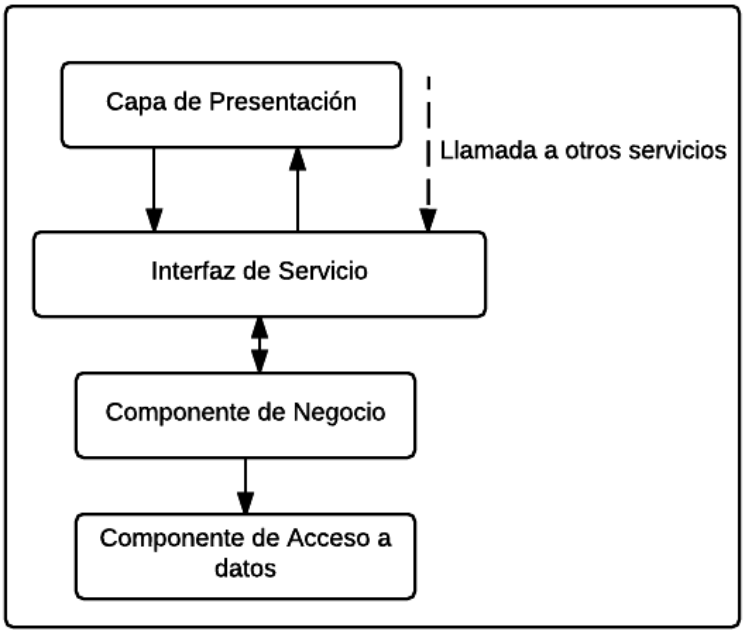

Figura 4. Interacción entre componentes de negocio
Fuente: (Arevalo, n.d.).
Elaboración: Autor.

- ✓ **Entidades de negocio.-** Permite intercambiar los datos entre los componentes de negocio. Estos están formados por las estructuras de datos o clases de lenguajes orientados a objetos serializables (Fernández Lanvin, n.d.).
- ✓ **Flujo de trabajo.-** Los flujos de trabajo son procesos de negocio de múltiples pasos que requieren intervención del usuario y son de larga duración (Ma, Tang, & Wang, 2009) .
- ✓ **Fachada de aplicación.-** Combina las operaciones de negocio dentro de un único mensaje. La fachada es un componente opcional que proporciona otra facilidad de proceso a nuestra capa de negocio.

Existen ciertos errores comunes que se cometen durante la programación de la capa lógica de negocio. Estos afectan directamente a la aplicación en desarrollo; en la tabla cuatro se a echo una recopilación de los principales errores.(Ayala, 2014).

Tabla 4. Errores de la capa lógica de negocio.

Categoría	Problemas comunes
Autenticación	✓ Aplicar la autenticación en la capa de negocios cuando no es necesario. ✓ Diseñar un mecanismo de autenticación personalizado, siempre se debe basar en estándares. ✓ No implementar mecanismos de single sign-on cuando se requiera.
Autorización	✓ Granularidad incorrecta para los roles (alguien hace más de lo permitido). ✓ Implementar las autorizaciones a nivel de la capa lógica de negocio.
Componentes de negocio	✓ Codificar la lógica de negocio en la capa de acceso a datos. ✓ No usar interfaces basadas en mensajes para exponer componentes de negocio
Entidades de negocio	✓ Uso de un dominio de negocio inadecuado para modelar las entidades. ✓ Mala selección de formatos de datos. ✓ No tomar en cuenta los elementos de serialización.
Acoplamiento y cohesión	✓ Alto acoplamiento entre las capas. ✓ Mala separación de responsabilidades en la capa de negocio. ✓ Fallo al utilizar una interfaz basada en mensajes entre capas de la aplicación.
Acceso a datos	✓ Acceder a la fuente de datos desde la capa de lógica de negocio. ✓ Mezclar lógica de acceso a datos con lógica de negocio.
Tratamiento de excepciones/errores	✓ Permitir visualizar datos sensibles en los mensajes de error. ✓ Mala utilización de los Logs en las excepciones.
Validación	✓ Suponer que los datos de usuario son siempre correctos. ✓ Fallo en el tratamiento de errores de validación ✓ Mala especificación de las reglas de negocio para las validaciones. ✓ No reutilizar la lógica de negocio para validación. ✓ No tomar en cuenta aspectos como rangos y tipos de datos en la entrada de información de los usuarios para la validación.

Fuente:(Ayala, 2014)
Elaboración: Autor

La capa lógica de negocio tiene responsabilidades dentro del software; entre las principales tenemos:

- ✓ La implementación de los objetos de negocios y reglas de negocio.
- ✓ Recepta los datos de la capa de presentación, los procesa y a su vez realiza peticiones a la base de datos y viceversa.
- ✓ No permite el acceso directo a los datos desde la capa de presentación.

1.3.4.3. Capa de datos.

Es la encargada de gestionar el acceso a la base de datos, esta capa permite almacenar y recuperar la información. Los datos se deben manejar de tal forma que exista consistencia y precisión de los mismos. En esta capa se definen las consultas simples y complejas para la generación de reportes específicos.

La capa de acceso a datos es la única que conoce la forma de almacenamiento y recuperación de los datos, es decir que la capa lógica de negocio no interactúa con la base de datos sino que lo realiza mediante esta capa; de esta manera se logra la independencia entre las capas.

La capa de datos envía la información directamente a la capa de lógica de negocio para ser procesada y convertida en objetos según se requiera, esta acción se conoce como encapsulamiento (Mendoza Quispe, 2014). En la Figura 5 se muestra una representación de la capa enlace de datos.

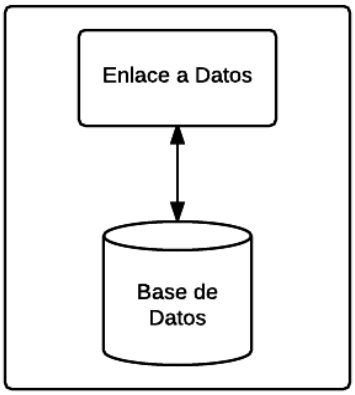

Figura 5. Capa de enlace de datos.
Fuente: autor.
Elaboración: Autor.

La capa de datos tiene las siguientes responsabilidades dentro de un sistema se software:

- ✓ Almacena los datos del sistema.

- ✓ Procesa y permite la recuperación de los datos.

1.3.5. Principios Fundamentales de 3-Layers.

El estilo arquitectónico dentro de un software debe cumplir ciertos parámetros para el buen funcionamiento de un sistema como: la cohesión, acoplamiento y funcionalidades claras bien definidas, a continuación se describe.

- ✓ **Cohesión.-** El sistema diseñado en un estilo arquitectónico 3-layers debe especificar el comportamiento que deben tener cada módulo del sistema o subsistema, con una alta cohesión entre ellos, lo cual permite alcanzar un solo propósito de manera más eficaz y rápida.
- ✓ **Bajo Acoplamiento.-** Un software desarrollado en un estilo 3-layers debe tener un bajo grado de dependencia entre las capas que lo conforman; esto se da por la división de módulos, con el fin de que cada uno tenga un funcionamiento específico y puede ser más factible la implementación por separado de cada capa. En caso de haber alto acoplamiento entre módulos no se estaría alcanzando el principal objetivo de este modelo, el cual es dividir una tarea grande en varias pequeñas, los módulos actuarían como uno solo al estar altamente acoplados entre sí, y se perdería el objetivo primordial de dividir el proyecto (Bermeo, 2012).
- ✓ **Funcionalidades claramente definidas**.- Los roles y responsabilidades de cada capa deben ser claras y específicamente definidas logrando de esta manera la independencia de cada capa; la comunicación entre las capas se da de manera bidireccional.

1.3.6. Comunicación entre las Capas.

Cada capa cuenta con varios elementos y componentes esenciales para el buen funcionamiento de un sistemas; entendiéndose por componente a: Una de las partes de la solución del software, sus componentes de software compilados y otros elementos del sistema como páginas web; estas cumplen con una función específica (Rojas, 2009).

En la Figura 6 se muestra los componentes y tecnologías que se utiliza para la construcción en una aplicación web y que permite la comunicación entre las capas.

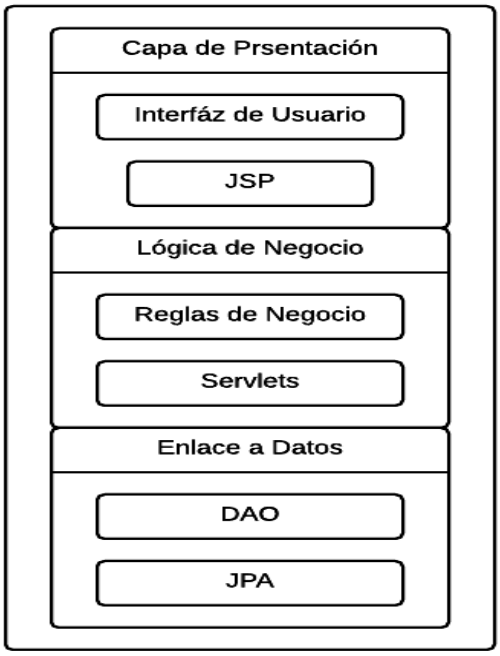

Figura 6. Elementos por capa en el estilo 3-layers
Fuente:(Rojas, 2009).
Elaboración: Autor.

Para la construcción de un software seguro, existen tecnologías que facilitan la implementación de mecanismos de seguridad como en la autenticación, autorización y criptografía de los datos que transmiten en las aplicaciones web. Una de estas herramientas es Apache Shiro, que es un framework de seguridad para aplicaciones web Java, el cual se describe a continuación.

1.4. Framework de Seguridad: Apache Shiro.

Un framework es una estructura de software compuesta de componentes personalizables e intercambiables para el desarrollo de una aplicación; uno de los principales aspectos en una aplicación web es la seguridad, así, es importante implementar un framework de seguridad en aplicaciones web, que permita implementar y gestionar la seguridad en sistemas de software como Apache Shiro el cual es un framework de seguridad que nos permite gestionar la seguridad a nivel de autenticación, autorización, criptografía y administración de sesiones. Apache Shiro es un framework Open Source permite implementar seguridad en aplicaciones web Java, sin necesidad de escribir todo el código desde el principio.

1.4.1. Características de Apache Shiro.

Según Garcés (2015) Apache Shiro como un framework de seguridad presenta las siguientes características:

- ✓ Implementación rápida y fácil.
- ✓ Simplifica la seguridad.
- ✓ Soporte Web a servicios como Rest.
- ✓ De fácil integración con aplicaciones web.
- ✓ El almacenamiento en caché la cual nos permite la eficiencia y rapidez en las aplicaciones web.
- ✓ Recuerda la identidad de los usuarios a través de sesiones (Remember), por lo cual no se necesita iniciar una sesión en cada momento siempre y cuando el usuario se encuentre activo dentro del sistema.

1.4.2. Arquitectura de Apache Shiro.

Apache Shiro simplifica las seguridades de las aplicaciones, es intuitivo y fácil de usar. Las aplicaciones de software se diseñan generalmente en base a los usuarios en la forma como trabajan o necesitan trabajar; este framework refleja estos conceptos en su propio diseño en cualquier aplicación.

En la Figura 7 se muestra la arquitectura de Apache Shiro de una forma detallada la cual nos indica los diferentes módulos y componentes que contiene (Foundation, 2015).

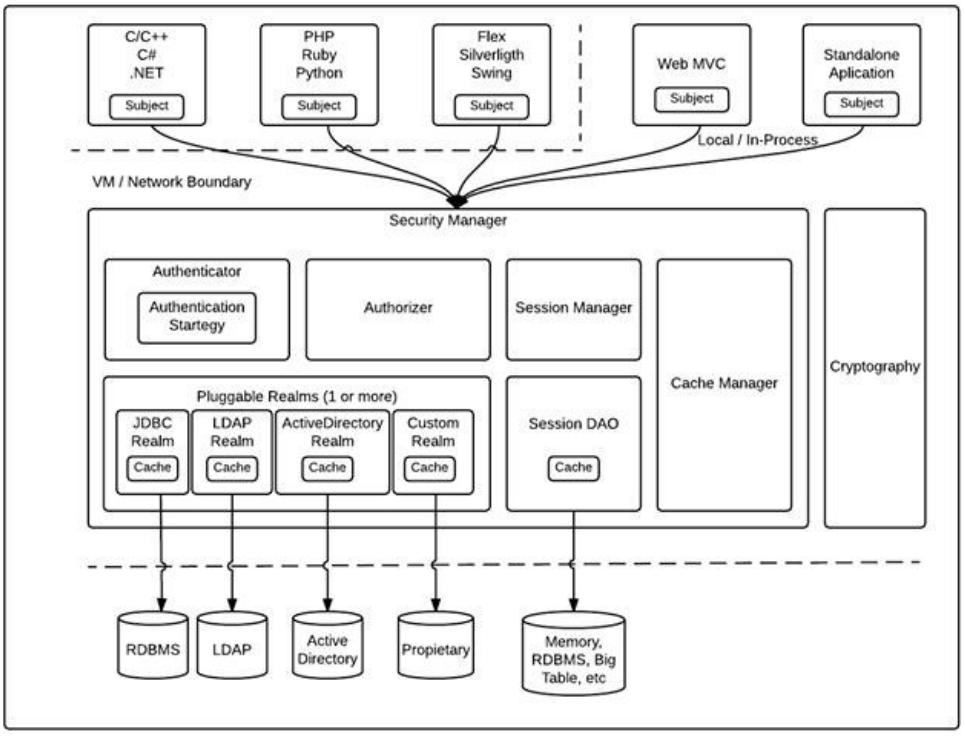

Figura 7. Arquitectura Apache Shiro.
Fuente: (Apache Software Fundación, 2015).
Elaboración: Autor.

Subject.- Es esencialmente una vista de seguridad, un Subject puede ser una persona, pero también puede representar a un servicio, básicamente es cualquier cosa que se está interactuando con el software.

SecurityManager.- es la parte medular del Framework de seguridad en donde coordina todos los componentes internos de seguridad y está conformado por:

- ✓ **Authenticator**.- La autenticación es el proceso de verificación de la identidad del usuario que está tratando de demostrar un usuario es quien dice ser; para ello, el usuario debe proporcionar algún tipo de prueba de identidad, mediante una **Authentication estrategy**, la cual determina las condiciones de éxito de una autenticación.
- ✓ **Authorizer.-** La autorización o control de acceso, es la función que nos permite decir a cada usuario que es lo que puede hacer dentro del sistema o a que recursos tiene acceso.
- ✓ **SessionManager.-** Este componente conoce como crear y administrar los ciclos de sesiones de las cuentas de usuario en cualquier ambiente. El **SessionDAO** es el encargado de las operaciones de persistencia (CRUD), en nombre de la sesión manager es decir permite enlazar base de datos con la administración de sesiones.
- ✓ **CacheManager.-** Crea y gestiona los ciclos de vida de la memoria cache utilizados por otros componentes de Apache Shiro, ayudando a mejorar el rendimiento durante en acceso a la base de datos.
- ✓ **Cryptography.-** Es un componente adicional de Apache Shiro. La criptografía es la protección de la información de acceso al sistema de un usuario que ingresa al mismo con el respectivo usuario y contraseña; para ello Apache Shiro utiliza el algoritmo de encriptación sha256Hash. El objetivo principal del Framework es utilizar algoritmos de encriptación con característica robusta.

Realms.- este componente interactúa como un puente o conector entre el framework y los datos de seguridad de la aplicación, los cuales se debe configurar en la aplicación para realizar la autenticación (Login) y autorización (control de acceso).

Almacenamiento Criptográfico.- Proteger los datos sensibles se ha convertido en una parte fundamental para la mayoría de las aplicaciones Web. Simplemente no cifrar datos sensibles es una falla de seguridad en aplicaciones que no lo realizan adecuadamente o la criptografía está mal diseñada, ya sea usando sistemas de cifrado no apropiados o cometiendo errores

serios al usar algoritmos de cifrado sólidos. Estos errores pueden conducir a la revelación de los datos sensibles y dando paso a violaciones de cumplimiento de estándares.

Los métodos del algoritmo de encriptación hash son muy utilizados para la protección de contraseñas cifrando los datos, haciendo que estos se transporten desde el navegador hasta el almacenamiento en la base de dados de forma segura y no se haga en texto plano, una función hash es como una firma para un texto o fichero. El algoritmo SHA-256 es un hash de 64 dígitos hexadecimales. Se considera seguro sin vulnerabilidades teóricas conocidas y tiene un tamaño razonable de 32 bytes. Un "HASH" es una función matemática y lógica que nos permite transformar un conjunto de datos, texto, imágenes, archivos, en un único valor numérico (Gutiérrez, 2013).

El uso de una clave compleja no permite que atacantes puedan adivinar fácilmente las claves y hackear una cuenta de usuario, OWASP recomienda las siguientes mejores prácticas para el uso de claves fuertes.

- ✓ Contraseñas que contengan al menos una letra mayúscula.
- ✓ Contraseñas que contengan al menos una letra minúscula.
- ✓ Contraseñas que contengan al menos un número o un caracter especial.
- ✓ Contraseñas cuya longitud sea como mínimo 8 caracteres.
- ✓ Contraseñas cuya longitud máxima no debe ser arbitrariamente limitada.

1.5. Vulnerabilidades

1.5.1. Definición.

Las vulnerabilidades en las aplicaciones por lo general se originan a partir de malas configuraciones y de errores en la programación, estas vulnerabilidades pueden ser complejas y pueden ocurrir bajo circunstancias distintas.

Una vulnerabilidad es un fallo o debilidad en el diseño, la implementación, el funcionamiento o la gestión de un sistema, que puede ser explotado con la finalidad de violar las políticas de seguridad del sistema (Candel, 2011).

La vulnerabilidad es la capacidad, las condiciones y características del sistema mismo que lo hace susceptible a amenazas, con el resultado de sufrir algún daño. En otras palabras, es la capacidad y posibilidad de un sistema de no responder o reaccionar a una amenaza o de recuperarse de un daño (Kimberly, Jiménez, & Valencia, 2013).

Una vulnerabilidad en un software es simplemente un error, un problema en su código o en su configuración; es muy probable que los programas contengan errores puesto que los desarrolladores de software no son perfectos y esto se puede dar en aplicaciones grandes y complejas.

Al construir una aplicación web en el estilo arquitectónico 3-layers, con frecuencia las vulnerabilidades se dan en las tres capas que lo conforman, por lo que es importante tener bien definida su arquitectura e implementar estándares de codificación, esto permite reducir la existencia de vulnerabilidades en un software. Dentro de esta arquitectura todos los elementos o capas son importantes, permiten el crecimiento y la extensibilidad de servicios para todas las aplicaciones existentes y futuras. (Herrera, 2007).

1.5.2. Clasificación de Vulnerabilidades

Como se mencionó una vulnerabilidad es una debilidad de cualquier tipo que compromete cualquier modulo del sistema. Se las agrupa de la siguiente manera según Mifsud (2012):

- ✓ **Diseño**
  - Debilidad en el diseño de protocolos utilizados.
  - Políticas de seguridad deficiente e inexistente.

- ✓ **Implementación**
  - Errores de programación.
  - Descuido de desarrolladores.

- ✓ **Uso**
  - Mala configuración de los sistemas informáticos.
  - Desconocimiento y falta de sensibilización de los usuarios y de los responsables de informática.
  - Disponibilidad de herramientas que facilitan los ataques.
  - Limitación gubernamental de tecnologías de seguridad.

- ✓ **Vulnerabilidad del día cero**

  La vulnerabilidad del día cero en ingles zero-day attack se produce en el momento que el atacante descubre una vulnerabilidad en un sistema en producción, esto se lo realiza a través de la ejecución de scripts y antes que el equipo de desarrollo cree o de una solución.

  El nombre 0-day (día cero) se debe a que aún no existe ninguna solución para mitigar el aprovechamiento de la vulnerabilidad. Éstas a veces se usan junto a troyanos,

rootkits, virus, gusanos y otros tipos de malware, con el fin de robar información confidencial, y propagarse e infectar el mayor número de equipos.

1.6. OWASP (The Open Web Application Security Project)

OWASP es un proyecto de código abierto sobre aplicaciones web, es una comunidad dedicada a asesorar a las organizaciones con el objetivo de fomentar el desarrollo de aplicaciones confiables (OWASP, 2014).

OWASP es una organización sin fines de lucro dedicada a asesorar a los desarrolladores de software para mejorar la seguridad y calidad en las aplicaciones web. Su principal objetivo es brindar información sobre las vulnerabilidades y fallos a las organizaciones y los desarrolladores, estos a su vez puedan aplicar técnicas y métodos para evitar riesgos reales de seguridad.

Toda la información que proporciona OWAPS se encuentra amparada bajo una licencia Open Source y está disponible para cualquier persona que lo requiera. Al tratarse de una organización desvinculada con empresas que brindan servicios informáticos o de software, proporcionan información a través de sus publicaciones de forma independientes sin ningún tipo de presiones. (Rodríguez, 2011). Dentro de OWASP se manejan diversos proyectos en los que se destaca el OWASP Top Ten 2013.

El Top Ten de OWASP 2013 presenta un listado de vulnerabilidades seleccionadas de acuerdo a la prevalencia que existe en las aplicaciones web y con el apoyo de estimaciones acordadas como la explotabilidad, detectabilidad e impacto. La versión 2013 de OWASP Top-Ten es la décima publicación de este proyecto, con el fin de concientizar sobre los riesgos de seguridad en las aplicaciones web. La primera publicación de OWASP Top Ten fue realizada en el año 2003, en el cual se identifican los riesgos que se enfrentan las organizaciones, proporcionando técnicas y recomendaciones de cómo proteger nuestra aplicación web en áreas que existe un alto riesgo que ocurra un ataque.

Dentro del OWASP Top Ten 2013 constan las siguientes vulnerabilidades:

- ✓ A1-Injection.
- ✓ A2- Broken Authentication and Session Management.
- ✓ A3- Cross-Site Scripting (XSS).
- ✓ A4- Insecure Direct Object References.
- ✓ A5-Security Misconfiguration.
- ✓ A6-Sensitive Data Exposure.
- ✓ A7-Missing Function Level Access.

- ✓ A8-Cross-Site Request Forgery (CSRF).
- ✓ A9-Using Known Vulnerable Components.
- ✓ A10-Unvalidated Redirects and Forwards.

Broken Authentication and Session Management

La vulnerabilidad Broken Authentication and Session Management, incluye todos los aspectos de manejo de autenticación de usuario y la gestión de las sesiones activas. La autenticación es un aspecto importante y crítica a la vez. Un mal diseño de los mecanismos de autenticación puede hacer a la aplicación vulnerable a un atacante, el cual puede apropiarse de información y del control de las aplicaciones; se expone a negocios o empresas en riesgo de perder o alterar datos confidenciales, como también se abre puertos a los atacantes internos como externos, pueden tomar ventaja de esta vulnerabilidad para robar cuentas de usuarios y hacerse pasar por los mismos, una vez que una cuenta es secuestrada, el atacante tiene la capacidad de hacer cualquier cosa que el titular de la cuenta tenga permiso y lo cual no solo afecta al usuario de la cuenta secuestrada si no también puede provocar consecuencias graves a toda la empresa (OWASP, 2013).

En la Figura 8 se identifica los riesgos e impactos que existe en Broken Authentication and Session Management.

Agentes de Amenaza	Vectores de Ataque	Debilidades de Seguridad		Impactos Técnicos	Impactos al negocio
Específico de la Aplicación	**Explotabilidad PROMEDIO**	**Prevalencia DIFUNDIDO**	**Detección PROMEDIO**	**Impacto SEVERO**	**Específico de la aplicación/negocio**
Considere atacantes anónimos externos, así como a usuarios con sus propias cuentas, que podrían intentar robar cuentas de otros. Considere también a trabajadores que quieran enmascarar sus acciones.	El atacante utiliza filtraciones o vulnerabilidades en las funciones de autenticación o gestión de las sesiones (ej. cuentas expuestas, contraseñas, identificadores de sesión) para suplantar otros usuarios.	Los desarrolladores a menudo crean esquemas propios de autenticación o gestión de las sesiones, pero construirlos en forma correcta es difícil. Por ello, a menudo estos esquemas propios contienen vulnerabilidades en el cierre de sesión, gestión de contraseñas, tiempo de desconexión (expiración), función de recordar contraseña, pregunta secreta, actualización de cuenta, etc. Encontrar estas vulnerabilidades puede ser difícil ya que cada implementación es única.		Estas vulnerabilidades pueden permitir que algunas o todas las cuentas sean atacadas. Una vez que el ataque resulte exitoso, el atacante podría realizar cualquier acción que la víctima pudiese. Las cuentas privilegiadas son objetivos prioritarios.	Considere el valor de negocio de los datos afectados o las funciones de la aplicación expuestas. También considere el impacto en el negocio de la exposición pública de la vulnerabilidad.

Figura 8.Broken Authentication and session management
Fuente: ("OWASP," 2014)
Elaboración: Autor.

En la Figura 9 se ejemplariza un usuario que tiene el rol administrador con su respectiva contraseña "admin" no cumple con los estándares de contraseñas y se encuentra almacenados en texto plano lo cual hace que sean vulnerables.

PER_USUARIO	PER_CLAVE
administrador	admin

Figura 9. Usuario y contraseña en texto plano.
Fuente: Autor.
Elaboración: Autor.

Por tal razón, se debe seguir las recomendaciones de OWASP en donde claramente dice que:

- ✓ Se debe usar algoritmos fuertes en cuanto al cifrado.
- ✓ No se debe almacenar datos innecesarios que sean sensibles.
- ✓ Se debe inhabilitar la función de autocompletar de los formularios.
- ✓ Hacer de un canal seguro de transporte como HTTPS.

Unvalidated Redirects and Forwards

La vulnerabilidad Unvalidated Redirects and Forwards, se da cuando las aplicaciones web redirigen a los usuarios hacia otras páginas y sitios web no deseables que no son de confianza para el usuario, sin la validación adecuada, los atacantes pueden redirigir las víctimas de phishing o malware, para acceder a las páginas no autorizadas (Sheehan, 2015).

En la Figura 10 muestra los riesgos e impactos que existen en una aplicación web.

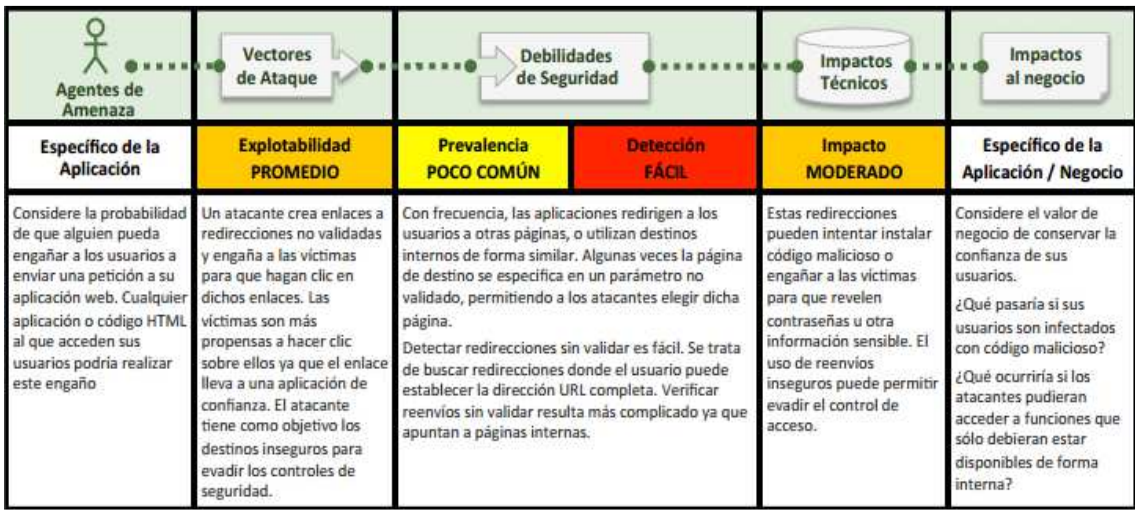

Figura 10. Unvalidated redirects and forwards
Fuente: ("OWASP," 2014)
Elaboración: Autor.

Entre las recomendaciones de OWASP para mitigar este tipo de ataques tenemos:

- ✓ No permitir el uso de redirecciones y reenvíos.
- ✓ Si se utiliza, no permitir que se involucre parámetros que sean manipulables por el usuario para definir su destino.

- ✓ Si los parámetros de destino no pueden ser evitados, se debe asegurar que el valor suministrado sea válido y autorizado para el usuario.

CAPITULO II

PROPUESTA DE SOLUCIÓN

En el presente capítulo se expone una propuesta de solución para la construcción del prototipo de la aplicación web tomando como referencia el capítulo anterior, en el cual se analizó los estándares de codificación, framework de seguridad y siguiendo las recomendaciones de OWASP. Se realiza la construcción de un prototipo web seguro con el fin de evitar las vulnerabilidades de tipo Broken Authentication and Session Management y Unvalidated Redirects and Forwards, con lo cual se garantiza el buen funcionamiento de la aplicación asegurando la integridad de los datos.

2.1. Objetos de implementación del prototipo

2.1.1. Selección de Vulnerabilidades

Una aplicación web basada en un estilo arquitectónico 3-layers, al igual que cualquier aplicación, deben ser seguras en: autenticación, autorización, integridad y confidencialidad de los datos que se manejen. Estos aspectos se ven reflejadas en el OWASP Top Ten 2013 con las vulnerabilidades de Broken Authentication and Session Management (Pérdida de Autenticación y Gestión de sesiones, en español) y Unvalidated Redirects and Forwards Redirecciones (Redirecciones y reenvíos no validados, en español) las cuales se han sido seleccionadas de acuerdo a un estudio realizado en los diferentes software por parte de OWASP y de WASC (The Web Application Security Consortium) como las amenazas y riesgos más frecuentes en las aplicaciones web basadas en este estilo arquitectónico. Además, estas vulnerabilidades ocurren debido a una mala validación de las credenciales de autenticación de los usuarios, ya que éste control muchas de las veces se lo hace en la capa de lógica de negocio y no en la capa de presentación, es decir antes que las peticiones ingresen a la aplicación, se controlen en el navegador del cliente.

Para evitar estos dos tipos de vulnerabilidades, se debe iniciar con el diseño de una arquitectura donde se contemplen los aspectos relacionados tanto en la funcionalidad y la seguridad del sistema que se implementará, tomando en cuenta la utilización de patrones de diseño, estándares y recomendaciones para la implementación de un software seguro.

2.1.2. Patrones de Diseño

2.1.2.1. DAO.

El patrón de diseño Data Access Object (DAO) permite la abstracción y encapsulamiento del acceso a la base de datos, permite realizar la conexión con la fuente de datos con el fin de extraer y almacenar datos independientemente de que tipo de base de datos se esté utilizando, así mismo oculta los detalles de configuración del acceso a los datos. DAO

mediante su Api JPA permite automatizar la persistencia y el mapeo de la base de datos y la implementación de los métodos CRUD (Create, Read, Update, Delete).

Para el desarrollo del prototipo se implementará el patrón de diseño DAO, que permite tener un bajo acoplamiento entre clases, esto hace que cada capa tenga sus roles y responsabilidades, característica en una arquitectura 3-layers.

2.1.2.2. Facade

El patrón Facade en español Fachada se caracteriza por ser una puerta de entrada hacia otro subsistema. Provee una interfaz unificada y sencilla que haga de intermediaria entre el cliente y una interfaz o grupo de interfaces más complejas.

En la aplicación web nos va a permitir dividir en subsistemas y proporcionar una interfaz para poder acceder a ellos, con esto se reduce la comunicación y la dependencia de los mismos haciendo que la aplicación web sea sencilla, por lo tanto existen razones para el uso en el desarrollo del prototipo:

- ✓ Proporcionar una interfaz sencilla cuando se tiene sistemas complejos
- ✓ Permite desacoplar un subsistema de sus clientes u otros.
- ✓ subsistemas, haciéndolo más independiente y portable.
- ✓ Hacer que la aplicación web esté bien estructurada.

2.1.3. Ambiente de desarrollo del prototipo

2.1.3.1. Framework de seguridad Apache Shiro.

Es un framework de seguridad el cual permite gestionar e implementar seguridad en diferentes aspectos de una aplicación desarrollada en el lenguaje de programación Java. Shiro permite incorporar seguridad en los siguientes aspectos: autenticación, autorización, encriptación y funciones de administración del periodo de sesiones ya que contiene paquetes de seguridad e implementa un algoritmo criptográfico seguro y recomendado por OWASP, además es flexible en su implementación e integración con otros framework de desarrollo.

2.1.3.2. Java.

Es un lenguaje de programación orientado a objetos, es uno de los más populares y utilizados por los desarrolladores de software, permite diseñar y desarrollar diferentes tipos de aplicaciones como: dispositivos móviles, aplicaciones web y de escritorio; entre sus principales características tenemos que: lenguaje Open Source y es multiplataforma es decir no depende de ningún sistema operativo, cualquier aplicación desarrollada en el lenguaje de programación

Java funciona correctamente en los diferentes plataformas de sistemas operativos existentes; establece un puente de comunicación entre Java y el sistema operativo, mismo que es conocido como la máquina virtual. Entre las ventajas de desarrollar con Java tenemos: flexibilidad, seguridad, multiplataforma, open source, etc.

En el desarrollo del prototipo de la aplicación web se utilizara Java EE en la versión 7 ya que esta plataforma de programación permite utilizar la arquitectura en 3-layers, mediante la utilización de componentes, el desarrollo modular y la construcción de una aplicación estructurada.

2.1.3.3. Glassfish.

Glassfish es un servidor de aplicaciones open source que permite implementar tecnologías que son definidas por Java EE, soporta JSF, framework de seguridad, patrones de diseño, EJB, herramientas utilizadas en el desarrollo de la aplicación web. Glassfish siendo de código libre posee las siguientes ventajas: escalabilidad, velocidad, integralidad y extensible.

Glassfish es una implementación de JAVA EE la cual permite la implantación de un framework de seguridad como Apache Shiro y algoritmos de autenticación para la implementación de seguridad en la aplicación web.

2.1.3.4. MYSQL.

My Structured Query Language más conocido por sus siglas “MySQL”, es un gestor de base de datos relacional open source, en la actualidad tiene una gran demanda por parte de los desarrolladores de software en los pequeños, medianos y grandes proyectos ya que cumple con ciertas características que los desarrolladores requieren como: robustez, velocidad y gran almacenamiento de datos, portabilidad, multiusuario y multiplataforma es decir se puede hacer uso de MySQL en los diferentes sistemas operativos que existen; estas características son indispensables en las aplicaciones modernas, además de brindar la seguridad necesaria ya que soporta procedimientos almacenados. Al utilizar MySQL como un gestor de la aplicación web tenemos las siguientes ventajas.

- ✓ Velocidad en la realización de operaciones.
- ✓ No se limita al almacenamiento de datos.
- ✓ Permite implementar y administrar permisos y privilegios a los usuarios de la aplicación.
- ✓ Permite la integración con los diferentes lenguajes de programación.

2.1.3.5. Java Server Faces.

Java Server Faces conocido por sus siglas JFS es una tecnología para el desarrollo de aplicaciones web Java; la cual permite que en el presente prototipo se pueda simplificar el desarrollo de interfaces de usuario en aplicaciones Java EE. La aplicación web se define como paginas XHTML las cuales se las denomina como paginas JSF. JSF promociona algunas ventajas como.

- ✓ Muestra los datos o información al usuario en cajas de texto y tablas.
- ✓ Recolecta los datos proporcionados por parte del usuario en los formularios de la aplicación.
- ✓ Controla los eventos que se dan al pulsar botones, teclas y hacer algún movimiento con el mouse.
- ✓ Realiza validaciones y conversiones de datos introducidos por parte del usuario

En la Figura 11 se muestra la generación de una página JSF; el navegador hace una llamada a una URL en donde se encuentra la página JSF, el motor JSF también conocida como Servlet, recoge la llamada.

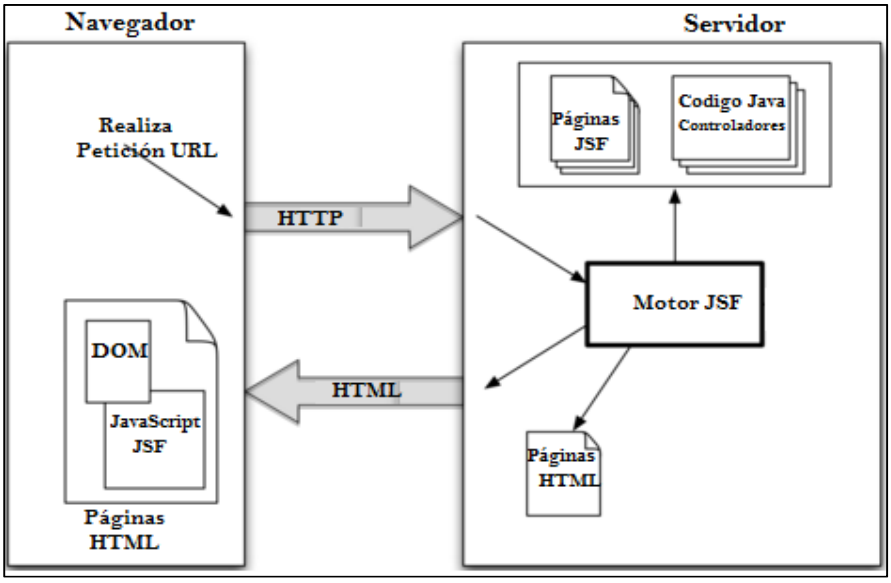

Figura 11. Arquitectura JSF.
Fuente: (Such, 2013).
Elaboración: Autor.

2.2. Resumen de objetos de implementación

En la Tabla cinco se realiza un resumen de las tecnologías que se utilizarán para la implementación del prototipo con las seguridades adecuadas para evitar las vulnerabilidades antes seleccionadas.

Tabla 5. Tecnologías utilizadas

Nombre	Tecnología
Patrones de diseño	DAO
	FAÇADE
Framework de seguridad	Apache Shiro
Framework de presentación	Java server faces (JSF)
Lenguaje de programación	JAVA
Servidor de aplicaciones	GlassFish Server Version 4.1
Base de datos	MySQL
Transporte.	HTTPS/SSL

Fuente: Autor
Elaboración: Autor.

CAPITULO III

DISEÑO DE LA SOLUCIÓN

En este capítulo se presenta el diseño de prototipo utilizando un estilo arquitectónico 3-layers, que permitirá la creación y actualización de propuestas de proyectos de fin de titulación y a su vez dar el debido seguimiento a los mismos.

Con el objetivo de brindar una seguridad adecuada a la aplicación web en cuanto a la autenticación, autorización, encriptación de los datos sensibles como el nombre de usuario y contraseña, se implementará estándares de seguridad como framework, mencionados en los capítulos anteriores y se seguirá las recomendaciones de OWASP Top Ten 2013 para evitar que existan vulnerabilidades y de esta manera proporcionar un grado de seguridad a la aplicación web y a los usuarios finales.

3.1. Diseño Arquitectónico de la aplicación.

El diseño de la aplicación web (prototipo) se basará en el estilo arquitectónico 3-layers en base a la asignación de roles y responsabilidades descrito en el capítulo anterior. La aplicación web está estructurada en 3 capas: Presentación, Lógica de negocio y Acceso a datos. Incluyendo la implementación de seguridad mediante procesos almacenados y el framework Apache Shiro para la autenticación, autorización y gestión de sesiones, como se puede ver en la Figura 12.

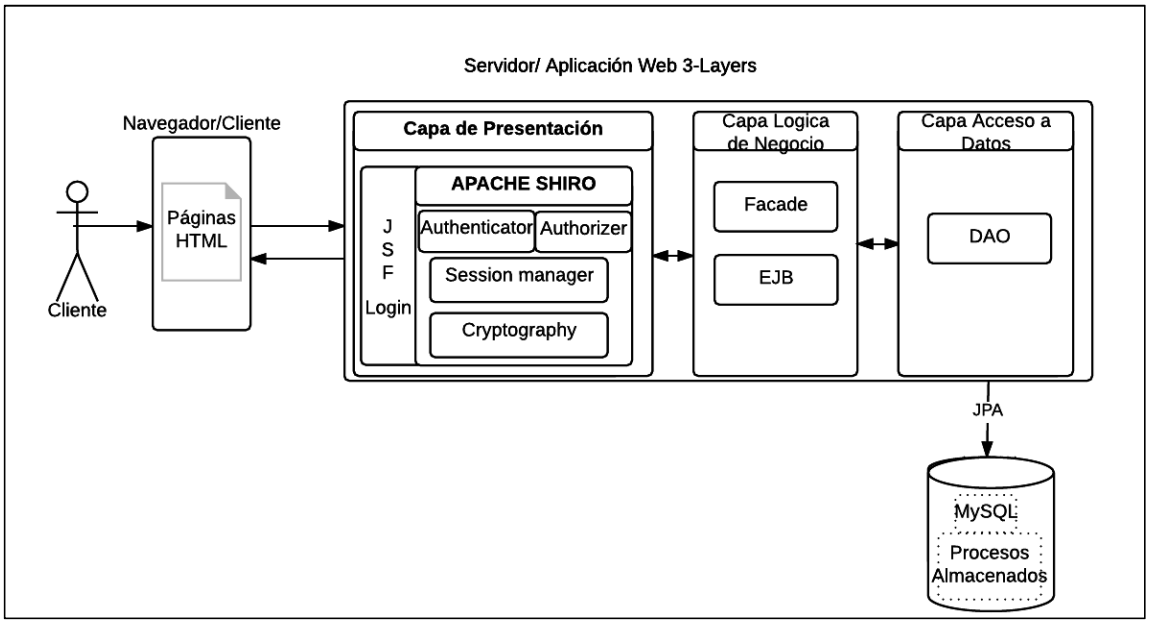

Figura 12. Diseño de la solución.
Fuente: Autor
Elaboración: Autor.

Donde la aplicación está conformada por:

- **Capa de Presentación.-** La capa presentación que contendrá las páginas web mediante el framework JSF que visualizará el cliente en el navegador, y además contendrá la seguridad en lo que respecta para la autenticación, autorización, criptografía y gestión de sesiones, mediante la implementación del framework Apache Shiro.
- **Capa de Lógica de Negocio.-** La capa de lógica de negocio contendrá el procesamiento de los datos extraídos de la capa inferior de acceso a datos, para ello hace uso del patrón de diseño Facade para ocultar la lógica mediante la implementación de interfaces.
- **Capa de Acceso a Datos.-** La capa de acceso a datos contendrá la lógica para el acceso a la base de datos y las operaciones CRUD con el patrón de diseño DAO y de JPA para el mapeo relacional.

3.2. Diseño de base de Datos

El modelo entidad relación de base de datos a utilizar está basada en información obtenida de los proyectos de fin de titulación de la Universidad Técnica Particular de Loja. En el presente trabajo se hace uso de ciertas tablas del modelo (ptf_configuración, ptf_persona, ptf_proyecto ptf_catalogo), que permitan construir el prototipo de la aplicación web y poder validar su seguridad, el modelo de entidad relación de la base de datos se lo puede visualizar en el Anexo A.

3.3. Diseño de Seguridad

Para el diseño de la seguridad del presente prototipo se realizó un análisis de acuerdo al estilo arquitectónico 3-layers y las vulnerabilidades seleccionadas y estudiadas Broken Authentication and Session Management y Unvalidated Redirects and Forwards, las cuales serán implementadas a nivel de: codificación y configuración, descritas a continuación:

3.3.1. Codificación

Para la seguridad y evitar la vulnerabilidad de Authentication and Session Management se implementó el framework Apache Shiro, el cual nos permite gestionar la autenticación, autorización, criptografía y control de gestiones de los usuarios que hagan uso del prototipo. La autenticación es uno de los aspectos más importantes y críticos en el desarrollo de software, ya que es la primera pantalla que se presenta al cliente para que proporciones sus credenciales de autenticación (usuario y contraseña), por tal motivo se requiere de la implementación de métodos de autenticación probados. En cuanto a la gestión de sesiones, Apache Shiro permite el control en cuanto a roles y privilegios de usuarios del sistema.

Apache Shiro realiza la autenticación y autorización de la siguiente manera:

- **Autenticación.-** La autenticación a la aplicación web es el método de seguridad para verificar que la persona que intenta acceder a la aplicación, es en realidad quien dice ser, todo esto se lo realiza en base a los datos proporcionados como usuario y clave. En la Figura 13 se indica el procedimiento detallado que permite la autenticación mediante el framework Apache Shiro:

    1. El usuario ingresa sus credenciales con las cuales realiza una petición para ingresar al sistema,
    2. Crea una instancia de Subject de Apache Shiro, la contraseña en texto plano que ingresa por parte del usuario es cifrada haciendo uso de algoritmo HAS-256 bits,
    3. Se procede a realizar la autenticación mediante el método de Apache Shiro,
    4. posteriormente se realiza una consulta a la base de datos en donde se permite o se niega el inicio de sesión.

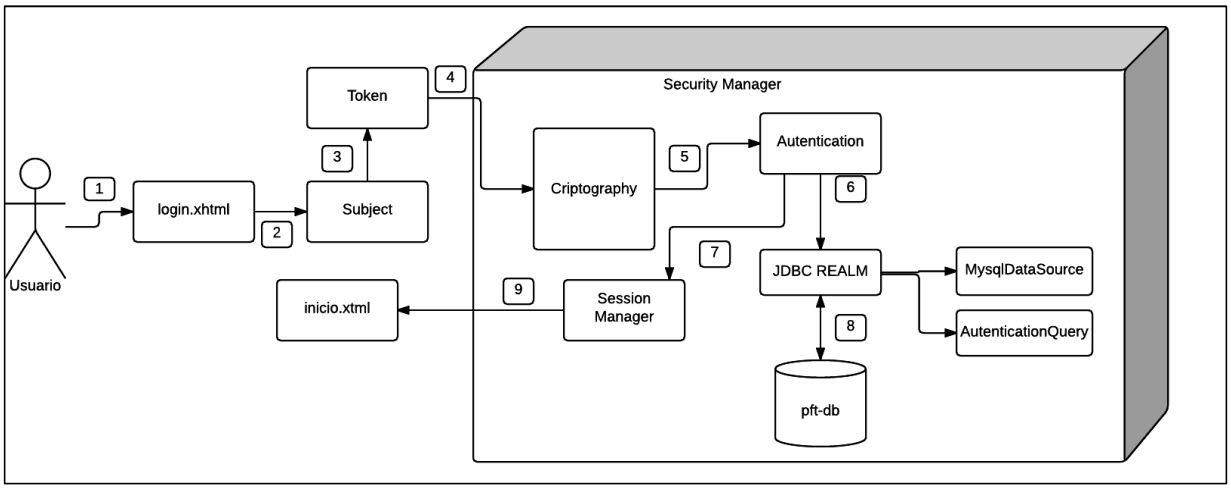

Figura 13. Proceso de autenticación.
Fuente: autor.
Elaboración: Autor.

- **Autorización.-** La autorización conocida como el control de acceso permite o niega el acceso a los diferentes módulos de la aplicación, a través de los roles y privilegios asignados a cada usuario lo cual determina a que funcionalidad del sistema se encuentra autorizado a acceder. En el caso del prototipo se va a tener dos tipos de roles y a cada rol se asigna el acceso a la aplicación:

    - ✓ **Administrador**: Un administrador está autorizado para: crear, editar, eliminar tanto a un usuario y un proyecto de fin de titulación.

- ✓ **Usuario/Estudiante:** Un usuario se encuentra autorizado para crear o editar un proyecto de fin de titulación. Esto hace que el administrador sea el único que pueda modificar los datos de un usuario de la aplicación, restringiendo el acceso a los usuarios.

  En la Figura 14 se muestra el proceso de autorización que se encuentra implementado en el prototipo.

  1. Un usuario realiza una petición para ingresar a una funcionalidad del sistema.
  2. Se realiza la consulta a la base de datos mediante el modulo Shiro.ini por el cual se hace la petición y se realiza la verificación de sus roles dentro del sistema.
  3. Si el usuario tiene un rol que corresponde a la URL que está intentando ingresar el sistema navega a la funcionalidad, caso contrario le aparece una página la cual se informa al usuario que no tienes los permisos correspondientes para ingresar.

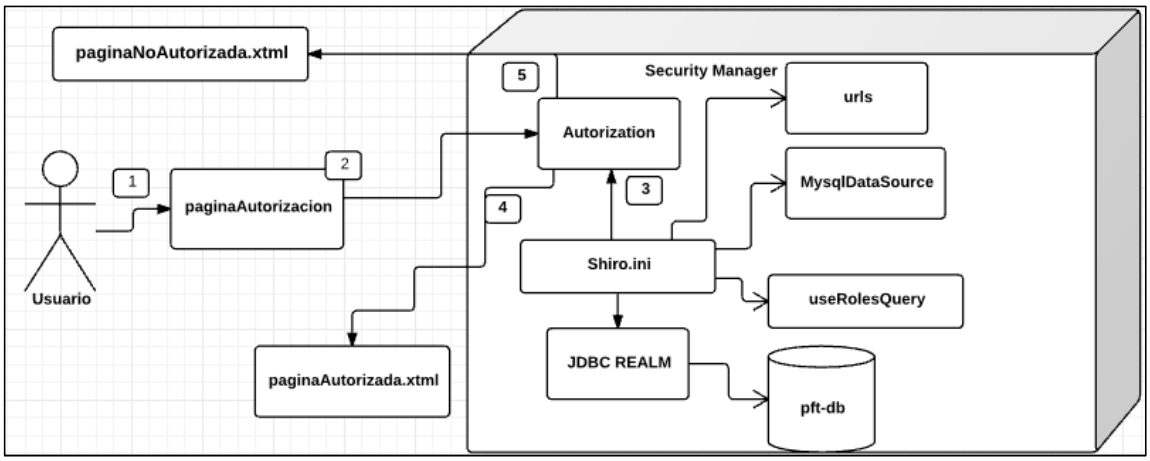

Figura 14. . Proceso de autorización
Fuente: Autor.
Elaboración: Autor.

- ✓ **Unvalidated Redirects and Forwards:** Las aplicaciones web frecuentemente redirigen a los usuarios hacia otras páginas y sitios web, como por ejemplo http://localhost:8080/pft-web-secure/proyectos/. En el enlace anterior, en lugar de la página **/proyectos/** se puede suplantar otro parámetro y llevar al cliente hacia donde se indique que ingrese usuario y contraseña y a través de un monitoreo de red o fishing captar las credenciales.

3.3.2. Configuración

Las aplicaciones web utilizan generalmente protocolos de trasporte HTTP, que es un protocolo no seguro, en el cual se puede capturar las credenciales del usuario mediante mecanismos de capturas de tráfico de red o sniffer; por tal razón se debe implementar un protocolo de transporte de hipertexto seguro como HTTPS, que permite el transporte de los datos de manera segura desde el inicio hasta el punto o nodo final.

HTTPS utiliza la tecnología SSL denominada capa de conexión segura, la cual permite la encriptación de los datos que se trasmite entre el navegador y el servidor web; OWASP recomienda utilizar este tipo de protocolo de trasferencia de texto con el fin de garantizar, la confidencialidad e integridad de los datos.

En el Anexo B se puede visualizar la configuración del protocolo HTTPS en el servidor Glassfish.

Cabe recalcar que se hará uso de procedimientos almacenados, los cuales trabajarán directamente con el servidor de base de datos alivianando la sobrecarga en la aplicación ya que las consultas SQL no estarán alojadas en el código, evitando así la exposición del esquema de la base de datos para los atacantes. Además los procedimientos almacenados como recomendación de OWASP ayudan a evitar ataques de Inyecciones SQL.

CAPITULO IV

IMPLEMENTACIÓN DE LA SOLUCIÓN

En este capítulo se detalla el proceso de implementación del prototipo diseñado en el capítulo anterior, tomando en cuenta las consideraciones de seguridad para proteger la información mediante el uso del lenguaje de programación J2E, el framework Apache Shiro, JSF y el uso de procedimientos almacenados en MySQL. A continuación se detallan los niveles de implementación del prototipo.

4.1. Implementación a nivel de base de datos

A nivel de base de datos como recomendación de OWASP se implementó procedimientos almacenados, los cuales permiten manejar todas las consultas SQL directamente con el motor de base de datos y no con la lógica de negocio del prototipo, ayudando a evitar la inserción de consultas SQL maliciosas, previniendo así ataques como SQL Injection. En la Tabla 6 se listan los procedimientos almacenados implementados:

Tabla 6. Procedimientos Almacenados

Nombre de Procedimiento	Parámetros	Resultado
create_persona	Datos de Usuario.	Permite crear un usuario, Estudiante, Docente.
edit_persona	Datos de Usuario.	Permite editar un usuario (Estudiante, Docente).
sp_login	Usuario. Contraseña.	Permite la autenticación de un usuario
get_modalidades	No aplica.	Permite devolver todas las modalidades
get_Programas	No aplica.	Permite obtener todos los programas educativos.
sp_ultimo_intento_usuario	Username.	Permite devolver el último registro de intento de autenticación de un usuario.

Fuente: Autor
Elaboración: Autor

Después de realizar el diseño y la implementación de la base de datos se realizó la conexión con el software, en la Figura 15 se muestra la conexión lógica entre el software y la base de datos.

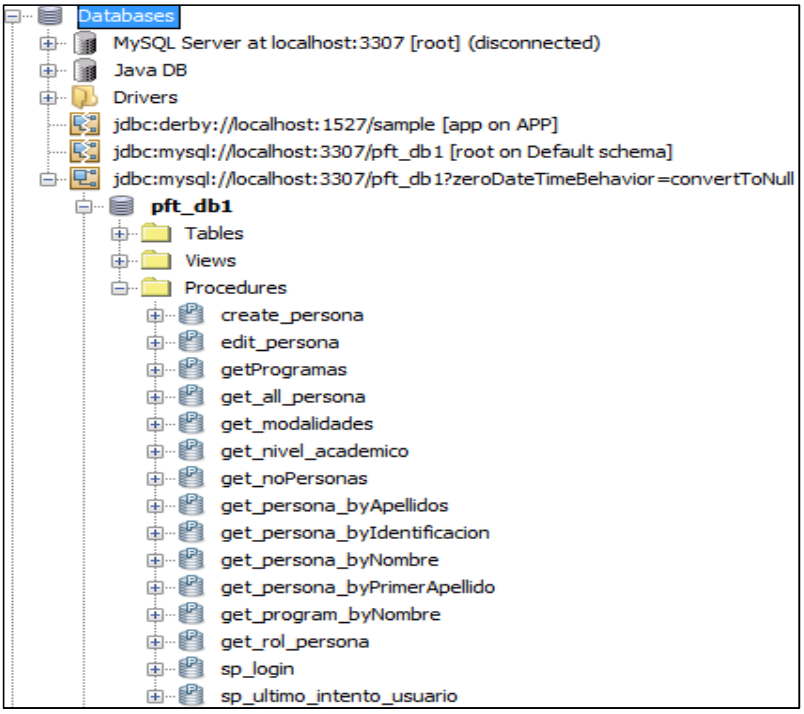

Figura 15. Conexión de base de datos.
Fuente: Autor.
Elaboración: Autor.

4.2. Implementación a nivel de codificación

Se desarrolló el prototipo con el nombre **pft-web**, en el cual se hizo uso de los estándares y tecnologías abordadas en los capítulos anteriores. En la figura 16 se muestra la estructura general del prototipo con sus respectivos paquetes.

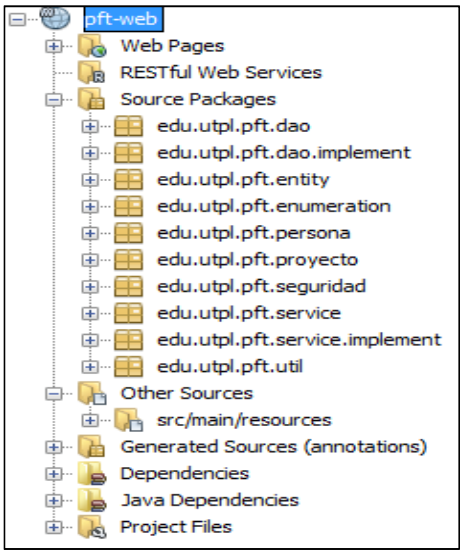

Figura 16. Estructura general del prototipo.
Fuente: Autor.
Elaboración: Autor.

En la construcción del prototipo se hace uso de dependencias, por ello se tiene un modelo de objetos de proyecto denominado POM, donde las dependencias son colocadas en un archivo pom.xml.

En la tabla 7, se lista las dependencias utilizadas en la construcción del prototipo.

Tabla 7. Dependencias utilizadas en el prototipo.

Dependencia	Descripción
org.primefaces	Permite el uso de componentes visuales, en JSF.
org.apache.commons	Permite obtener y establecer valores de las propiedades en clase java que utilizan los patrones de diseño.
org.apache.shiro	Permite hacer uso del api de Apache Shiro, que permite implementar los diferentes módulos de seguridad.
javax.mail	Permite hacer uso de interfaces y clases que se utilizan para enviar, leer y borrar mensajes de correo electrónico.

Fuente: Autor.
Elaboración: Autor.

En el archivo pom.xml, se implementan las dependencias descritas en la tabla anterior tal como se muestra en la Figura 17.

```
<dependencies>
    <dependency>
        <groupId>org.primefaces</groupId>
        <artifactId>primefaces</artifactId>
        <version>5.0</version>
    </dependency>
    <dependency>
        <groupId>commons-logging</groupId>
        <artifactId>commons-logging</artifactId>
        <version>1.1.1</version>
    </dependency>
    <dependency>
        <groupId>commons-codec</groupId>
        <artifactId>commons-codec</artifactId>
        <version>1.6</version>
        <type>jar</type>
    </dependency>
```

Figura 17. Configuración de dependencias.
Fuente: Autor.
Elaboración: Autor.

Paquetes

La estructura del prototipo se conforma de paquetes tal como se muestra en la figura 18.

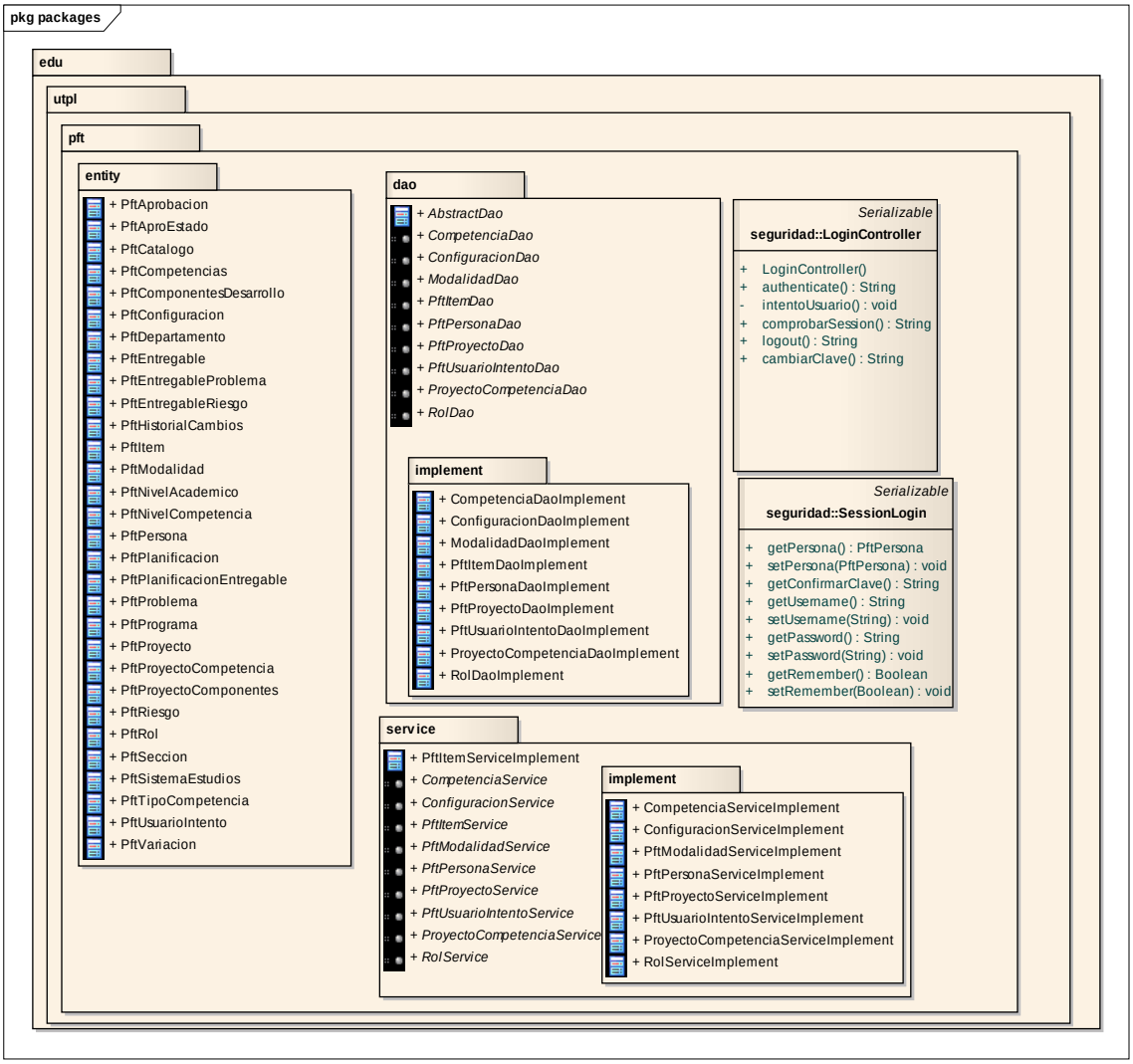

Figura 18. Paquetes de la aplicación web
Fuente: Autor.
Elaboración: Autor.

- **Paquete Entity.-** En este paquete se encuentra las entidades mapeadas con la base de datos, donde cada clase entidad representa a cada tabla de la base de datos llamada pft_db. Para realizar el mapeo se utilizó el ORM denominado JPA, el cual contiene una serie de anotaciones que permiten realizar el mapeo de las entidades de persistencia. Entre las anotaciones que se utilizó en el presente prototipo tenemos las listadas en la Tabla 8:

Tabla 8. Anotaciones utilizadas en el prototipo

Nombre	Descripción
@Entity	Declara la clase como una entidad.
@Table	Permite definir la tabla, el catálogo y los nombres del esquema para el mapeo de entidades, de lo contrario las tablas tomarán el valor defecto.
@Lob	Indica que la propiedad debe persistir en un Blob o Clob dependiendo del tipo de propiedad.

Fuente: (Bernard, 2014).
Elaboración: Autor

Cuando se trabaja con JPA en un proyecto se genera un archivo de configuración denominado persistence.xml, en este archivo se realizan la configuración de mapeo de las entidades de persistencia con la base de datos, como se puede ver en la Figura 19.

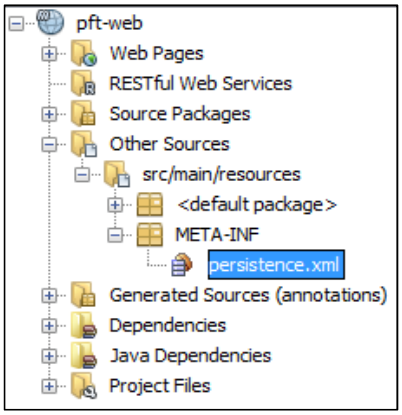

Figura 19. Archivo de persistence.xml.
Fuente: autor
Elaboración: Autor

En la Figura 20, se muestra como está configurado el archivo persitence.xml en el prototipo:

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.1" xmlns="http://xmlns.jcp.org/xml/ns/persistence"
  <persistence-unit name="gestionTesisPU" transaction-type="JTA">
    <provider>org.eclipse.persistence.jpa.PersistenceProvider</provider>
    <jta-data-source>jdbc/pft</jta-data-source>
    <class>edu.utpl.pft.entity.PftProyecto</class>
    <class>edu.utpl.pft.entity.PftAprobacion</class>
    <class>edu.utpl.pft.entity.PftEntregable</class>
    <class>edu.utpl.pft.entity.PftPlanificacionEntregable</class>
    <class>edu.utpl.pft.entity.PftPersona</class>
    <class>edu.utpl.pft.entity.PftProyectoComponentes</class>
    <class>edu.utpl.pft.entity.PftHistorialCambios</class>
    <class>edu.utpl.pft.entity.PftPrograma</class>
    <class>edu.utpl.pft.entity.PftEntregableProblema</class>
    <class>edu.utpl.pft.entity.PftAproEstado</class>
    <class>edu.utpl.pft.entity.PftModalidad</class>
    <class>edu.utpl.pft.entity.PftSeccion</class>
    <class>edu.utpl.pft.entity.PftSistemaEstudios</class>
    <class>edu.utpl.pft.entity.PftConfiguracion</class>
    <class>edu.utpl.pft.entity.PftNivelCompetencia</class>
    <class>edu.utpl.pft.entity.PftVariacion</class>
    <class>edu.utpl.pft.entity.PftComponentesDesarrollo</class>
```

Figura 20. Configuración del archivo percistence.xml.
Fuente: Autor.
Elaboración: Autor.

- ✓ **Paquete DAO.-** Este paquete contiene la clase con la lógica de acceso a datos, mediante estas clases la aplicación se comunica con los procedimientos almacenados construidos en la base de datos. El patrón de diseño utilizado para el acceso a datos es el patrón Data Access Object (DAO), este patrón se representa por la clase denominada AbstractDao en la cual se define el EntityManager que es una instancia de la clase EntityManagerFactory en JPA (Figura 21), lo que hace el EntityManager es representar la configuración para acceder a la base de datos que utiliza el prototipo.

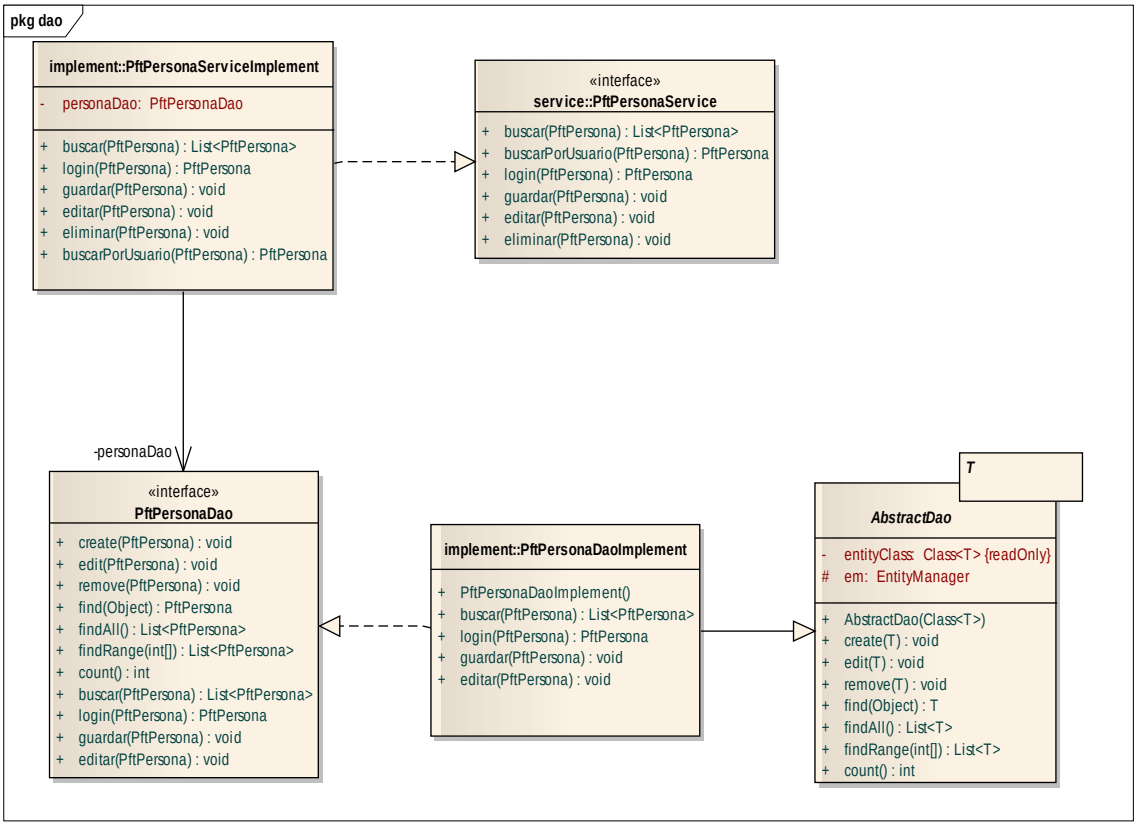

Figura 21. Paquete DAO
Fuente: Autor.
Elaboración: Autor.

Del mismo modo se implementó el patrón de diseño Facade en el prototipo, el mismo que permite el encapsulamiento de los métodos para el acceso a datos tal como se lo muestra en la Figura 22.

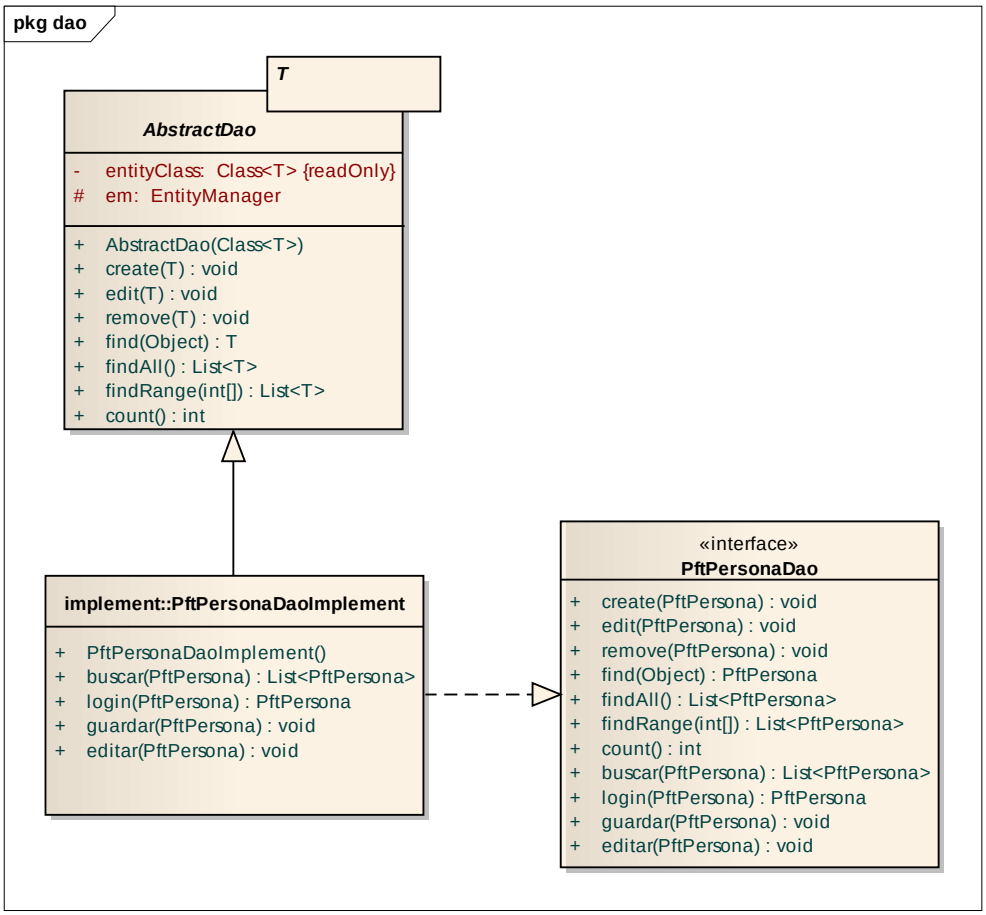

Figura 22. Patrón de diseño Facade.
Fuente: Autor.
Elaboración: Autor.

- ✓ **Paquete Service.-** En este paquete se maneja la lógica de negocio para el procesamiento de los datos del prototipo, este paquete contiene las clases que permiten comunicarse con las clases del paquete DAO para el acceso a los datos.
- ✓ **Paquete Seguridad.-** Este paquete contiene las clases que utilizan JSF, que es un framework para el desarrollo de aplicaciones web. Las clases que conforman este paquete son:
    - **FacesAjaxAwareUserFilter.-** Esta clase permite controlar el redireccionamiento de las páginas de usuarios no autenticados.
    - **LoginController.-** Esta clase permite la autenticación y la encriptación de una forma segura utilizando el framework Apache Shiro.
    - **SessionLogin.-** Esta clase controla lo referente a las claves de acceso al prototipo, como el cambio y recuperación de claves.

Estas clases se detallaran más a profundidad en el apartado de implementación de seguridad de este capítulo.

4.3. Implementación a nivel de seguridad

4.3.1. Configuración de Apache Shiro

Para la seguridad del prototipo se utilizó el framework de seguridad Apache Shiro el cual nos permite implementar seguridad a nivel de autenticación, autorización y gestión de sesiones mediante archivos de configuración. Para añadir Apache Shiro como dependencia en el prototipo se debe configurar y colocar en el archivo pom.xml, como se puede ver en la Figura 23.

```
<dependency>
    <groupId>org.apache.Shiro</groupId>
    <artifactId>Shiro-core</artifactId>
    <version>1.2.3</version>
</dependency>
<dependency>
    <groupId>org.apache.Shiro</groupId>
    <artifactId>Shiro-ehcache</artifactId>
    <version>1.2.3</version>
</dependency>
<dependency>
    <groupId>org.apache.Shiro</groupId>
    <artifactId>Shiro-faces</artifactId>
    <version>2.0-SNAPSHOT</version>
</dependency>
```

Figura 23. Configuración de pom.xml con Apache Shiro.
Fuente: Autor.
Elaboración: Autor.

Archivo de configuración.- El framework de seguridad Apache Shiro contiene un archivo de configuración llamado Shiro.ini, en el cual se encuentra toda la configuración para obtener la seguridad necesaria en el prototipo; en este archivo definiremos los usuarios, los roles con sus permisos y el control del acceso a cada una de las URL que referencien a las páginas de la aplicación, en la Figura 24 se muestra las respectivas configuraciones como los roles y la conexión con la base de datos.

```
[main]
# Create and setup user filter.
user = edu.utpl.pft.seguridad.FacesAjaxAwareUserFilter
user.loginUrl = /faces/login.xhtml
#JDBC
jdbcRealm=org.apache.Shiro.realm.jdbc.JdbcRealm
jdbcRealm.authenticationQuery = select PER_CLAVE from pft_persona where PER_USUARIO=?
jdbcRealm.userRolesQuery = SELECT r.rol_tipo FROM pft_rol r JOIN pft_persona p ON p.per_id_rol=r.rol_id WHERE p.PER_USUARIO= ?
ds=com.mysql.jdbc.jdbc2.optional.MysqlDataSource
ds.serverName=localhost
ds.user=root
ds.password=root
ds.port=3307
ds.databaseName=pft_db1
jdbcRealm.dataSource=$ds
authc.loginUrl =/login.xhtml
roles.unauthorizedUrl =/faces/denegado.xhtml
[urls]
/faces/login.xhtml=user
/personas/**=user,roles[ADMINISTRADOR]
/editarPersona/**=user,roles[ADMINISTRADOR]
/crearPersona/**=user,roles[ADMINISTRADOR]
/editarClave/**=user,roles[ADMINISTRADOR]
/proyectos/**=user
/crearProyecto/**=user
/editarProyecto/**=user
/cambiarClave/**=user
```

Figura 24. Archivo de configuración shiro.ini.
Fuente: Autor.
Elaboración: Autor.

En la configuración se indica que en la página /faces/login.xhtml se encuentra el formulario para la autenticación de un usuario. En este archivo además se configura la conexión JDBC que permite la conexión con la base de datos pft_db, así mismo se configura la consulta SQL para la autenticación de un usuario.

Autenticación.- La autenticación se maneja mediante la clase **LoginController,** la cual se encuentra dentro del paquete de seguridad y permite la autenticación segura de un usuario y la encriptación de los datos proporcionados por el usuario, e inicia una instancia de la persona tal como se lo muestra en la siguiente Figura 25.

```
public String authenticate() {
    Calendar fechaActual = Calendar.getInstance();
    org.apache.shiro.subject.Subject currentUser = SecurityUtils.getSubject();//Obtiene la configuración del sh
    try {
        Sha256Hash sha256Hash = new Sha256Hash(sessionLogin.getPassword(),
                (new SimpleByteSource("random_salt_value_" + sessionLogin.getPassword())).getBytes()); // insta
        String result = sha256Hash.toHex(); //Encripto el pasword en 256 bits
        UsernamePasswordToken token = new UsernamePasswordToken(sessionLogin.getUsername(), result);
        token.setRememberMe(sessionLogin.getRemember());
        PftPersona personaBuscar = new PftPersona(); // inicia una instancia de persona
        personaBuscar.setPerUsuario(sessionLogin.getUsername());
        sessionLogin.setPersona(personaService.buscarPorUsuario(personaBuscar));
        currentUser.login(token);//ejecuto compara
        if (currentUser.isAuthenticated()) {
            sessionLogin.setEsAdmin(Boolean.FALSE);
            if (!sessionLogin.getPersona().getEntActivo()) {
                currentUser.logout();
                cabeceraController.getMessageView().message(FacesMessage.SEVERITY_ERROR, "Usuario Desactivado",
                return "";
```

Figura 25. Clase LoginController.
Fuente: Autor.
Fuente: Autor.

Autorización.- En toda aplicación web se necesita tener un nivel de seguridad en el acceso a una funcionalidad a través de la URL, por ello se debe implementar un manejo adecuado de las seguridades de las mismas.

Mediante Apache Shiro se puede implementar seguridades para que un determinado usuario pueda acceder a ciertas funcionalidades de acuerdo a los roles que posee siempre y cuando se haya autentificado en el formulario login.

En el archivo Shiro.ini se configura una consulta para obtener los roles de un usuario que se autentica, luego en la URL se debe definir que roles de usuario tienen autorización:

user = edu.utpl.pft.seguridad.FacesAjaxAwareUserFilter

Esta línea de comando permite realizar una llamada a una clase denominada **FacesAjaxAwareUserFilter**, esta clase permite hacer un filtro que redirecciona a la página login si el usuario no está autenticado, en la Figura 26 se puede observar lo que contiene la clase FacesAjaxAwareUserFilter.

```
public class FacesAjaxAwareUserFilter extends UserFilter {
    private static final String FACES_REDIRECT_XML = "<?xml version=\"1.0\" encoding=\"UTF-8\"?>"
            + "<partial-response><redirect url=\"%s\"></redirect></partial-response>";
    @Override
    protected void redirectToLogin(ServletRequest req, ServletResponse res) throws IOException {
        HttpServletRequest request = (HttpServletRequest) req;
        if ("partial/ajax".equals(request.getHeader("Faces-Request"))) {
            res.setContentType("text/xml");
            res.setCharacterEncoding("UTF-8");
            res.getWriter().printf(FACES_REDIRECT_XML, request.getContextPath() + getLoginUrl());
        } else {
            super.redirectToLogin(req, res);
        }
    }
}
```

Figura 26. Clase FacesAjaxAwareUserFilter.
Fuente: Autor.
Elaboración: Autor.

Contraseñas.- El uso de contraseñas fuertes minimiza los riesgos de un ataque interno o externo, se implementó estándares de uso de contraseña descritos en el capítulo anterior mediante expresiones regulares, esto hace permitirá que al ingreso o cambio de contraseñas estas no sean débiles y cumplan los estándares descritos en la sección anterior. En la Figura 27 se muestra la implementación de las expresiones regulares en prototipo.

```
nelGrid columns="3">
n:outputLabel value="Clave:"/>
o:password id="psw" value="#{sessionLogin.persona.perClave}" validatorMessage="La Contraseña debe c
   <f:validateRegex pattern="(?=^.{7,}$)((?=.*\d)|(?=.*\W+))(?![.\n])(?=.*[A-Z])(?=.*[a-z]).*$"/>
   <p:ajax update="mspsw"/>
/p:password>
o:message for="psw" id="mspsw" />
n:outputLabel value="Confirmar Clave:"/>
o:password id="cfpsw" value="#{sessionLogin.confirmarClave}" validatorMessage="La Contraseña debe c
   <f:validateRegex pattern="(?=^.{7,}$)((?=.*\d)|(?=.*\W+))(?![.\n])(?=.*[A-Z])(?=.*[a-z]).*$"/>
   <p:ajax update="mscfpsw"/>
/p:password>
o:message for="cfpsw" id="mscfpsw" />
anelGrid>
et name="footer">
ui:include src="botonesCambiarClave.xhtml"/>
```

Figura 27. Implementación de expresiones regulares.
Fuente: Autor.
Fuente: Autor.

4.3.2. Configuración de Filtros de Seguridad

Esta clase contiene la seguridad para evitar las vulnerabilidades de ClickJacking, X-Frame-Options Header Not Set, XSS, a continuación en la Figura 28 se muestra el código implementado.

```
*/
public class SeguridadFilter implements Filter {



    @Override
    public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain) throws IOException, ServletException {
        HttpServletResponse res = (HttpServletResponse) response;
        res.addHeader("X-FRAME-OPTIONS", "DENY");
        res.addHeader("Cache-Control", "no-cache");
        res.addHeader("Pragma", "no-cache");
        res.addHeader("Expires", "0");
        res.addHeader("X-XSS-Protection", "1; mode=block");
        res.addHeader("X-XSS-Protection", "0");
        res.addHeader("X-Content-Type-Options", "nosniff");
        chain.doFilter(request, response);
    }
```

Figura 28. Codificaión SeguridadFilter.
Fuente: Autor.
Elaboración: Autor.

Para complementar la seguridad se debe ingresar en el archivo web.xml el siguiente filtro haciendo referencia a la clase SeguridadFilter como se muestra en la Figura 29.

```
<filter>
    <filter-name>ShiroFiltro</filter-name>
    <filter-class>org.apache.shiro.web.servlet.IniShiroFilter</filter-class>
</filter>
<filter>
    <filter-name>SeguridadFiltro</filter-name>
    <filter-class>edu.utpl.pft.util.SeguridadFilter</filter-class>
</filter>
<filter-mapping>
    <filter-name>ShiroFiltro</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
<filter-mapping>
    <filter-name>SeguridadFiltro</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
<servlet>
    <servlet-name>Faces Servlet</servlet-name>
    <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
</servlet>
```

Figura 29. Filtro de SeguridadFilter.
Fuente: Autor.
Elaboración: Autor.

De la misma forma OWASP en una de sus recomendaciones explica que no se debe mostrar los errores de la aplicación a los usuarios y realiza recomendaciones para manejar los mismos. En la Figura 30 se indica el código que se debe colocar en el archivo web.xml de la aplicación web para personalizar los errores y no indicar a los usuarios los errores propios del sistema.

```
<error-page>
    <error-code>500</error-code>
    <location>/faces/error.xhtml</location>
</error-page>
<error-page>
    <exception-type>java.lang.Throwable</exception-type>
    <location>/faces/error.xhtml</location>
</error-page>
```

Figura 30. Encapsulamiento de errores
Fuente: Autor.
Elaboración: Autor.

En el web.xml del proyecto web se coloca un filtro que hace referencia a Apache Shiro. El archivo web.xml contiene toda la información necesaria para desplegar de forma correcta la aplicación web en el servlet/jsp container o servidor de aplicaciones en este caso Glassfish tal como lo muestra en la Figura 31.

```
    <context-param>
        <param-name>com.ocpsoft.pretty.SCAN_LIB_DIRECTORY</param-name>
        <param-value>true</param-value>
    </context-param>
    <filter>
        <filter-name>ShiroFiltro</filter-name>
        <filter-class>org.apache.shiro.web.servlet.IniShiroFilter</filter-class>
    </filter>
    <filter>
        <filter-name>SeguridadFiltro</filter-name>
        <filter-class>edu.utpl.pft.util.SeguridadFilter</filter-class>
    </filter>
    <filter-mapping>
        <filter-name>ShiroFiltro</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>
    <filter-mapping>
```

Figura 31. Configuración Filter.
Fuente: autor.
Elaboración autor.

CAPITULO V

PRUEBAS

Para comprobar la validez del prototipo implementado en el capítulo anterior, en necesario realizar las respectivas pruebas, establecidas en tres niveles: diseño arquitectónico, codificación y seguridad, dichas pruebas forman parte de un proceso de control de calidad del software mediante la utilización de herramientas open Source, descritas a continuación.

5.1. Pruebas de diseño

Para la validación el diseño se realizó con una herramienta Open Source, la cual permite validar el diseño planteado.

5.1.1. Structural Analisys for Java.

Structural Analisys for Java es una herramienta que permite evaluar el diseño de nuestra aplicación, el análisis se trata de descomponer la complejidad del sistema y dejar que el usuario revise sus artefactos en cualquier nivel y desde diferentes perspectivas.

El análisis que realiza la herramienta Structural Analisys for Java, permite validar el modelo planteado frente al software codificado. Esta herramienta realiza un análisis detallado y a su vez una diagramación del código escrito permitiendo evaluar de manera profunda la aplicación.

En la Figura 32, se muestra la arquitectura de la aplicación la cual coincide con la arquitectura propuesta.

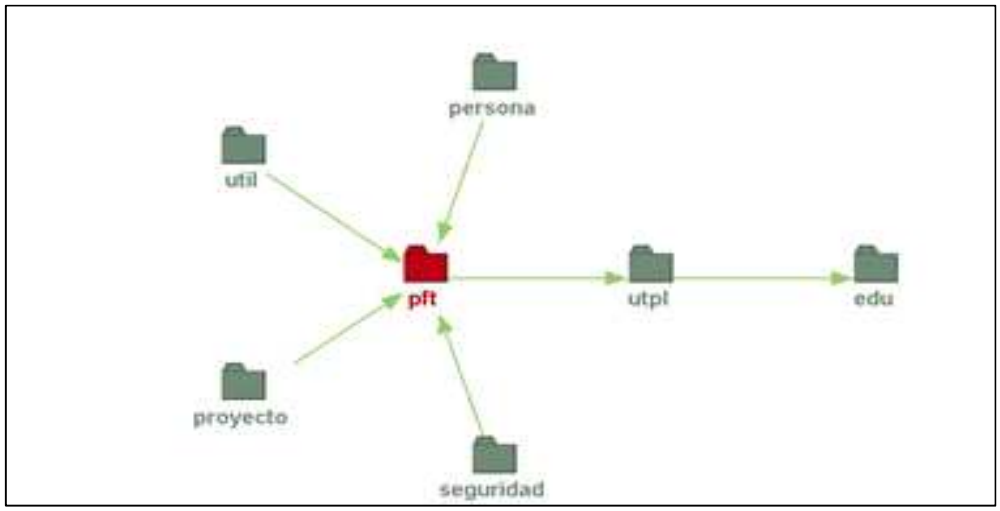

Figura 32. Validación de la Arquitectura-prototipo
Fuente Autor.
Elaboración: Autor

En la figura 33, nos muestra la implantación de patrón DAO el cual nos permite visualizar su implementación y sus entidades.

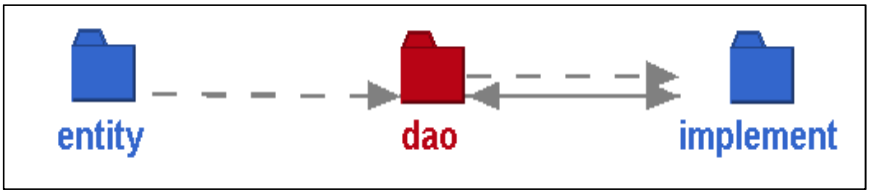

Figura 33. Validación del patrón DAO.
Fuente: Autor.
Elaboración: Autor.

5.2. Pruebas de Codificación.

Con el objetivo de asegurar la calidad del software, es necesario realizar evaluaciones en las distintas clases y métodos para verificar la correcta codificación del prototipo, para ello se hará uso de SonarQube.

5.2.1. SonarQube

Es una herramienta que nos permite evaluar el código fuente de las aplicación web que se encuentran construidas en distintas plataformas como Java, Php, C#; SonarQube es una plataforma muy completa que permite analizar una aplicación web de las distintas fases de codificación, permite detectar problemas de código en la aplicación y hacer análisis utilizando métricas de calidad específicas mediante configuraciones en la misma.

Una de las características importantes de SonarQube es que es Open Source. Categoriza los errores de una manera que podemos verificar los de incidencia negativa en nuestra aplicación y a su vez poderlos corregir.

Las pruebas se las realizo en dos fases o interacciones, los resultados obtenidos durante cada interacción fueron analizados y posteriormente resueltos.

5.2.1.1. Problemas encontrados.

Iteración 1

En la Figura 34 se muestra los resultados de la primera iteración en donde la escala de SQALE Rating A (deuda técnica) indica que el prototipo consta de una estabilidad óptima a nivel de codificación del prototipo. La Tabla 9 muestra un resumen de los resultados dados por la herramienta.

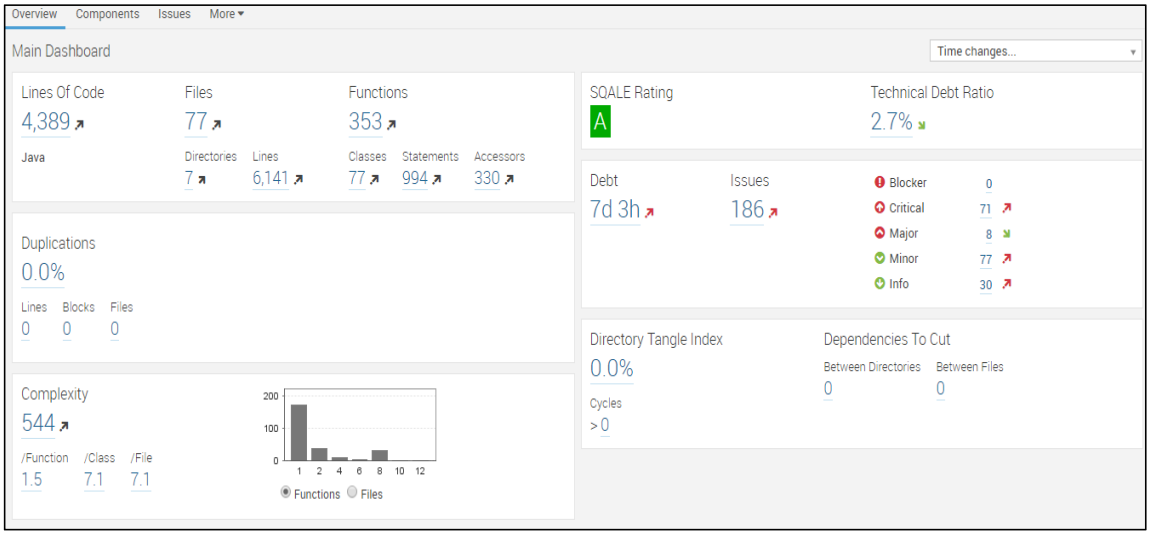

Figura 34. Primera Iteración con SonarQube.
Fuente: Autor.
Elaboración: Autor.

Tabla 9. Detalle de pruebas de SonarQube-Iteración 1.

Nombre	**Cantidad**
Líneas de código	4389
Clases	77
Directorio	7
Funciones	553
Deuda Técnica	2.7%
ISSUES	186

Fuente: Autor.
Elaboración: Autor.

Después de realizar la primera interacción, se presentan los siguientes errores detallados en la Tabla 10.

Tabla 10. Resumen de los diferentes errores encontrados

Problema	**Cantidad**	**Categoría**
Transient or serializable	91	Critico
Los manejadores de excepciones deben preservar la excepción original	1	Critico
Importaciones Inútiles deben retirarse	2	Mayor
Convención de Nombres de Paquetes	5	Menor

Convención de Nombres de Campo	3	Menor
Secciones de código no debe estar comentadas.	30	
Los literales de cadena no deben ser duplicados.	2	Menor
Las declaraciones deben utilizar interfaces de la colección de Java tales como "Lista" en lugar de clases específicas de implementación, tales como "Lista enlazada".	6	mayor
Los métodos o bloques código no deben dejarse vacíos.	1	mayor

Fuente: Autor.
Elaboración: Autor.

5.2.1.2. Solución de Errores Encontrados

Los errores encontrados fueron solucionados de acuerdo a la complejidad de los mismos, en la Tabla 11 se muestra el error y su solución dada.

Tabla 11. Solución de problemas

Problema	Solución
Manejo de excepciones deben preservar la excepción original.	Al manejar una excepción capturada, el mensaje de excepción original y de seguimiento de la pila deben ser registrados.
Los campos en una clase "Serializable" o bien debe ser transitorio o serializable.	Serializar todas las clases de la aplicación.
Las declaraciones deben utilizar interfaces de la colección de Java en lugar de clases específicas de implementación, tales como "Lista enlazada".	Proporcionar una jerarquía bien definida de las interfaces con el fin de ocultar los detalles de implementación. Las clases de ejecución se deben utilizar para crear instancias nuevas.
Bloques de código anidados no deben dejarse vacíos.	Un bloque vacío es defectuoso y se lo debe quitar.
Los métodos no deberían tener demasiados parámetros.	Una larga lista de parámetros puede indicar que una nueva estructura debería ser creada para agrupar

	los numerosos parámetros o que la función está haciendo demasiadas cosas.
Los métodos no deben estar vacíos.	No tener métodos vacíos o en su defecto comentar el por qué están vacíos.
El código o secciones de código no deben estar comentados.	Eliminar comentarios.
Los literales de cadena no deben ser duplicados.	Implementar métodos con el fin de no duplicar ciertos bloques.
Nombre de Paquetes mal llamados.	Se los debe llamar de acuerdo a la convención de nombres de java como package edu.utpl.pft.persona.
Nombres de campos incorrectos.	Se debe utilizar la sintaxis y gramática al momento de llamar un paquete o clase como: private PersonaSession personaSession.

Fuente: Autor.
Elaboración: Autor.

Iteración 2

Una vez corregido los errores obtenidos de la primera iteración, se realizó un nuevo análisis al prototipo con la misma herramienta SonarQube dando los siguientes resultados resumidos en la Tabla 12.

Tabla 12. Detalle de pruebas de SonarQube-Iteración 1.

Nombre	Cantidad
Líneas de código	4373
Clases	77
Directorio	7
Funciones	553
Deuda Técnica	1.6%
ISSUES	91

Fuente: Autor.
Elaboración: Autor.

Después de haber terminado de dar solución de los errores se realizó la segunda interacción, se encontró con problemas o issues, como: transient or serializable la cual no se puede dar

una solución, debido a que todas las 91 clases del prototipo están correctamente serializadas, la herramienta SonarQube no reconoce los objetos declarados dentro de una clase y los toma como un error.

5.3. Pruebas de seguridad

La implementación de estándares y framework implementados en el prototipo hace que éste sea seguro, sin embargo es necesario realizar pruebas con el objetivo de encontrar algún fallo de seguridad en el prototipo mediante herramientas como OWASP ZAP, Vega, Wireshark y pruebas manuales.

5.3.1. Pruebas con Herramientas

OWASP ZAP

Es una poderosa herramienta Open Source y multiplataforma para la verificación de vulnerabilidades en aplicaciones web mediante ataques de penetración, con esta herramienta se puede monitorear las entradas y salidas desde el usuario hacia el servidor y permite escanear principalmente las vulnerabilidades de OWASP Top ten 2013 ya que está basado en ello.

Se realizó comprobaciones a las dos aplicaciones web: pft-web que no cuenta con ninguna seguridad y a **pft-web** en la cual se implementaron las seguridades que se ha mencionado en el capítulo anterior obteniendo los siguientes resultados (Tabla 13):

Tabla 13. Resultados por niveles de Aplicación pft-web

Nivel de Riesgo	Número de Alertas
Alto	0
Medio	4
Bajo	4
Informativos	0

Fuente: Autor.
Elaboración: Autor.

Entre las principales vulnerabilidades encontradas en la aplicación pft-web agrupados en los niveles medio, bajo tenemos los siguientes.

Tabla 14. Número de Vulnerabilidades de pft-web.

Nivel Medio	Número de errores
X-Frame-Options Header Not Set	31
Session ID in URL Rewrite	21
Application Error Disclosure	4

Buffer Overflow	12
Nivel Bajo	
Web Browser XSS Protection Not Enabled	31
-Password Autocomplete in browser	4
X-Content-Type-Options Header Missing	31
Referer Exposes Session ID	3

Fuente: Autor.
Elaboración: Autor.

Wireshark

Wireshark es una herramienta que permite capturar el tráfico en ciertos protocolos como HTTP el mismo que permite visualizar los datos en texto plano de aquellas aplicaciones que no cuentan con la seguridad necesaria.

Para el presente caso de estudio necesitamos capturar el tráfico de red filtrando el protocolo HTTP para la aplicaciones web y así obtener la cookie del usuario, en la Figura 35 se muestra los datos del cliente como usuario y contraseña que accede al prototipo, estos datos son capturados con la herramienta y se los puede visualizar ya que viajan en un protocolo de transporte inseguro como es HTTP.

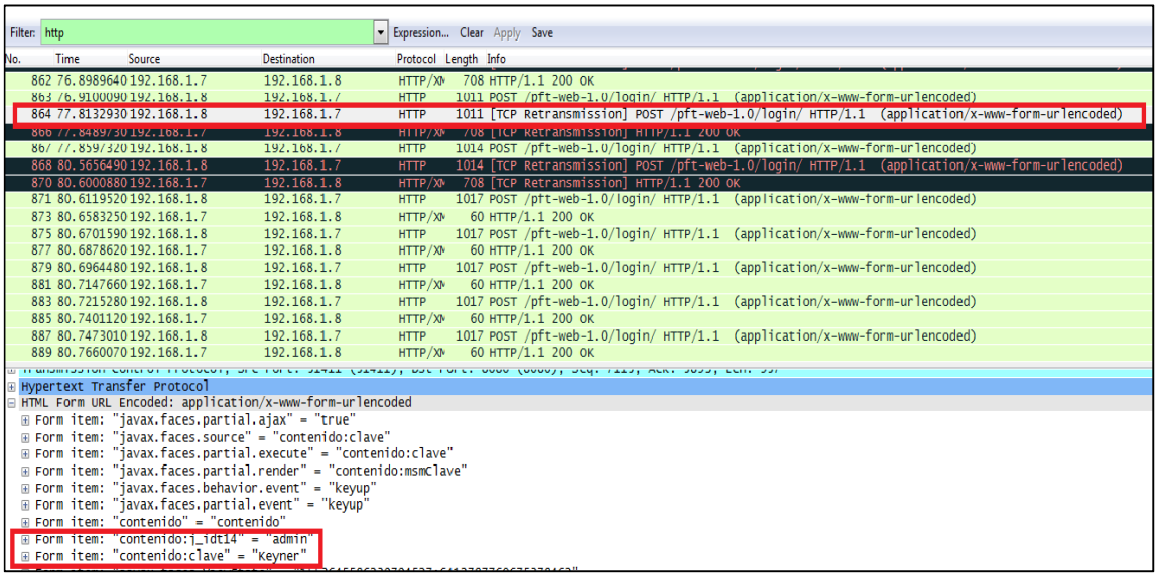

Figura 35.Monitoreo y captura del Tráfico de red con el protocolo HTTP.
Fuente: Autor.
Elaboración: Autor.

Después de haber terminado con las pruebas de la aplicación **pft-web** y haber obtenido los resultados, se encontraron una gran cantidad de errores de seguridad, se analizaron cada uno de ellos y siguiendo las recomendaciones de OWASP se implementó las seguridades

utilizando el framework y algoritmos descritos en el capítulo anterior; tenemos un nuevo prototipo al cual se le ha realizado el mismo proceso obteniendo los siguientes resultados (Tabla 15).

Tabla 15. Resultados por niveles de la Pft-Web con seguridad

Nivel de Riesgo	Número de Alertas
Alto	0
Medio	1
Bajo	2
Informativos	0

Fuente: Autor.
Elaboración: Autor.

Como nos podemos dar cuenta el número de vulnerabilidades detectadas por la herramienta han disminuido, a continuación en la Tabla 15 se detallan las vulnerabilidades encontradas.

Tabla 16. Número de Vulnerabilidades de pft-web con seguridad

Nivel Medio	Número de errores
X-Frame-Options Header Not Set	2
Nivel Bajo	
X-Content-Type-Options Header Missing	2
Web Browser XSS Protection Not Enabled	2

Fuente: Autor.
Elaboración: Autor.

Existe un número reducido de alertas aun detectadas por la herramienta, entre las cuales tenemos:

- ✓ **X-Frame-Options Header Not Se.- Esta alerta es más conocida como** ClickJacking, en donde se pretende engañar a un usuario que ingrese a enlaces falsos y posteriormente obtener información como sus credenciales; en estos casos la cabecera **X-Frame-Options indica al navegador si hace una página <frame> o un objeto <Object>**
- ✓ **X-Content-Type-Options Header Missing. - Es un tipo de cabecera, hace que un navegador no pueda leer otro tipo de contenido que no sea el correcto.**
- ✓ **Web Browser XSS Protection Not Enabled. -** Es una cabecera HTTP que permite la entrada de código malicioso a la aplicación, para aquello se implementa un filtro de seguridad.

Para las vulnerabilidades mencionadas anteriormente se implementó la seguridad necesaria dentro del framework de seguridad Apache Shiro descrito en la sección anterior.

De la misma manera se implementó un protocolo seguro como HTTPS, ya que este protocolo utiliza un cifrado SSL; de esta manera se hace que los datos viajen de manera segura y no permite que se muestren datos que son sensibles. Utilizando la herramienta Wireshark se procedió a capturar el tráfico de la red, en la figura 36 se muestra el contenido en donde no se permite visualizar los datos sensibles del cliente como son usuario y contraseña.

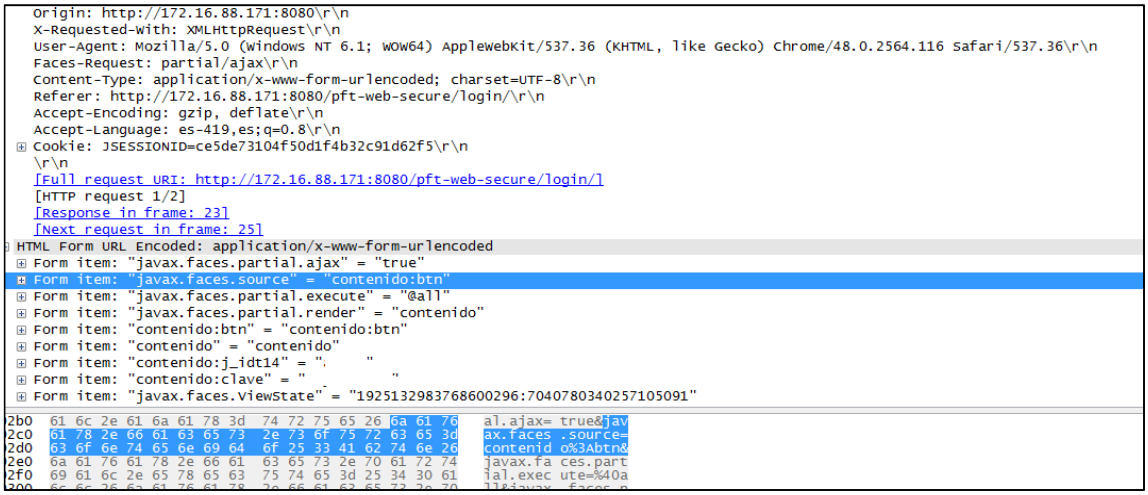

Figura 36. Monitoreo y captura del Tráfico de red con el protocolo HTTPS.
Fuente: Autor
Elaboración: Autor.

5.3.2. Validación Manual.

Para Verificar el almacenamiento Criptográfico, autenticación y el uso de contraseñas fuertes, se lo va realizar manualmente en el prototipo que no posee ningún tipo de seguridades y en el prototipo que se encuentra implementado estándares y el framework de seguridad.

Almacenamiento Criptográfico.

Las contraseñas de los usuarios almacenadas en texto plano en la base de datos, son fáciles de hackear y apropiarse de cuentas de usuario legítimas, muchos de los ataques son internos, en la Figura 37 se muestra el almacenamiento criptográfico, en este caso es el prototipo sin seguridad en donde las contraseñas son visibles sin ninguna protección.

PER_PRIMER_APELLIDO	PER_SEGUNDO_APELLIDO	PER_FECHA_NACIMIENTO	PER_USUARIO	PER_CLAVE	per_id_rol	per_id_departame
JIMENEZ	CUEVA	2015-10-27 12:02:48	1104780836	1104780836	3	
SÁNCHEZ	SARAGURO	2015-10-27 12:02:48	1104885114	1104885114	3	
PALACIOS	CASTILLO	2015-10-27 12:02:48	1104922024	1104922024	3	
YAGUANA	CASTILLO	2015-10-27 12:02:48	1104475825	1104475825	3	
YAGUANA	SARANGO	2015-10-27 12:02:48	1104856701	1104856701	3	
CUEVA	JIMENEZ	2015-10-27 12:02:48	1104787963	1104787963	3	
CUEVA	QUEZADA	2015-10-27 12:02:48	1104318520	1104318520	3	
TAPIA	BURI	2015-10-27 12:02:48	1104294234	1104294234	3	
INGA	AGUIRRE	2015-10-27 12:02:48	1104594229	1104594229	3	
GONZÁLEZ	GUACHISACA	2015-10-27 12:02:48	1104735772	1104735772	3	
TOCTO	MALDONADO	2015-10-27 12:02:48	1104977630	1104977630	3	
POMA	PINEDA	2015-10-27 12:02:48	1104899164	1104899164	3	
CABRERA	MALLA	2015-10-27 12:02:48	1104191752	1104191752	3	
JARAMILLO	CUEVA	2015-10-27 12:02:48	1104042914	1104042914	3	
SIGCHO	GONZALEZ	2015-10-27 12:02:48	1104680101	1104680101	3	
LOYOLA	ROMERO	2015-10-27 12:02:48	1104592595	1104592595	3	
CHUQUIHUANCA	SOLORZANO	2015-10-27 12:02:48	1104085723	1104085723	3	
JARAMILLO	JAPON	2015-10-27 12:02:48	1104536972	1104536972	3	

Figura 37. Almacenamiento de contraseñas sin seguridad.
Fuente: Autor.
Elaboración: Autor.

De acuerdo a la guía de pruebas de OWASP las claves de los usuarios deben estar cifradas con algoritmos para ello diseñados. En la Figura 38 se muestra el almacenamiento de las contraseñas cifradas y se hace uso del algoritmo sha256Hash descrito en la sección anterior.

PER_FECHA_NACIMIENTO	PER_USUARIO	PER_CLAVE	per_id_rol
2015-10-26 21:34:52	admin	24bccaf7cfb23ef25b5a6c1f4f779c17ed17e273790b13a5168940f3f192de45	4
2015-10-25 00:01:15	1104136138	762d05bf6756918cee53afa5c77a2469250714faca8334af9a5e2e6997452912	3
2015-10-25 00:01:15	1104102890	0e37733be0cc229e993746e4da16fcace35b4696f38f7d21e1d76f55c7adb269	3
2015-10-22 01:57:16	1104510753	89b13595bc54ecab54b186ad6791dc66c5ee47e2cbe5aede8e31baa04a46661e	3
2015-10-25 00:01:15	1104780836	6d96109cc8b839776bc888f61778a117a0393c269c098dd89ccf910b53a77af0	3
2015-10-25 00:01:15	1104885114	544499a96b459742b96746c167c825d1f89fadb01cb53a1da324b2175096aaed	3
2015-10-25 00:01:15	1104922024	7598e2f43a67b16100995da638fb97cfe2a7ec5363bc6d25144dcc05368189f8	3
2015-10-25 00:01:15	1104475825	82d44c538c195c92eee14ea3ea733c4a95bb8655406087656925249439a3fc23	3
2015-10-25 00:01:15	1104856701	ae7022d682a3847491cbd45358da11b594d9565e3dc6f15148d486fd1b390549	3
2015-10-25 00:01:15	1104787963	a132990017d851fe5eae38df8bc6aee6610bb4c2e61d5d496bfe6ddae11aab3f	3
2015-10-25 00:01:15	1104318520	f910a86b81d76223be31c13197b7d3354704fec82e8ee1d86c45e522ea571a48	3
2015-10-25 00:01:15	1104294234	e63335c18cd75d16b5b124d9bb007da64046bd02869fcaa6fa2875b68e79ac0b	3
2015-10-25 00:01:15	1104594229	ee4ba3966a48ac748a57d4271c71aedc8cb115f378c7a3dbd5268b1bf3a2425e	3
2015-10-25 00:01:15	1104735772	cbea4ffb34a1dbea8f478c2a6685791fe5e443e7d4979f2d83cb4c382ac2bbe5	3
2015-10-25 00:01:15	1104977630	4ccaed2fe16933730cdcade8c876ddc742989cd679aa9d4931f4852e80666432	3
2015-10-25 00:01:15	1104899164	4b1751b034714b0b9fe4678b28212b09cf2ca33ebc68e60c64fae4b8950e7718	3
2015-10-25 00:01:15	1104191752	a5bb57b71d173d23c338841a1ea6868afac1d5b9ee0b4f0323ca38fb44b5de17	3
2015-10-25 00:01:15	1104042914	dc515b4a99e6ceb15cca74a76d08c8171418e5e38b70c253aad899c076f194be	3

Figura 38. Almacenamiento de contraseñas cifradas.
Fuente: Autor.
Elaboración: autor.

De igual forma al momento de intentar ingresar un usuario en el prototipo sin seguridad se permite que se pueda ingresar claves débiles, fácil de poder adivinar, debido a que no se utiliza ninguna validación ni estándares de uso de contraseñas; en la figura 43 se muestra el ingreso de una nueva contraseña que no cumple con los estándares que debe tener descritos en la sección anterior.

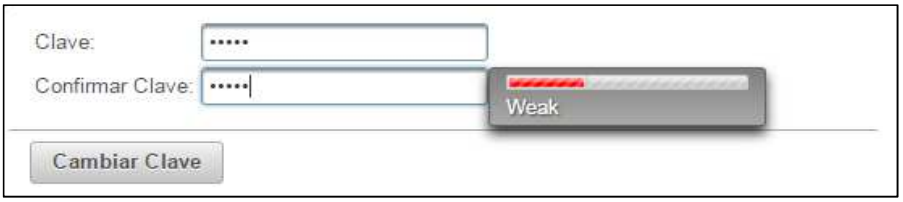

FIGURA 39. Ingreso de claves débiles.
Fuente: Autor.
Elaboración: Autor.

En la aplicación pft-web con seguridad, se ha implementado expresiones regulares como se lo describió en la sección anterior, lo cual hace que solo permita ingresar claves que como lo recomienda OWASP contengan una longitud mínima, caracteres especiales, número y letras; en la Figura 40 se muestra que no se puede hacer uso de contraseñas débiles que no cumplan las condiciones antes descritas.

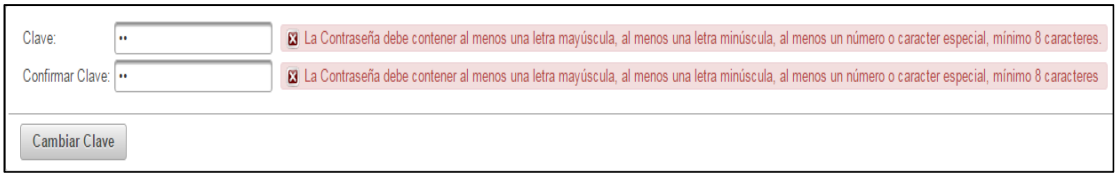

Figura 40. Validación de contraseñas.
Fuente: Autor
Elaboración autor.

Control de Acceso a nivel de funcionalidades

En la presente aplicación no existe ninguna seguridad en la URLS para accesos no autorizados, así que los atacantes pueden navegar a una URL de la aplicación sin necesidad de tener permisos ni estar autenticados.

En la Figura 45 se puede observar que un usuario puede acceder a un URL del sistema sin necesidad que se haya autenticado.

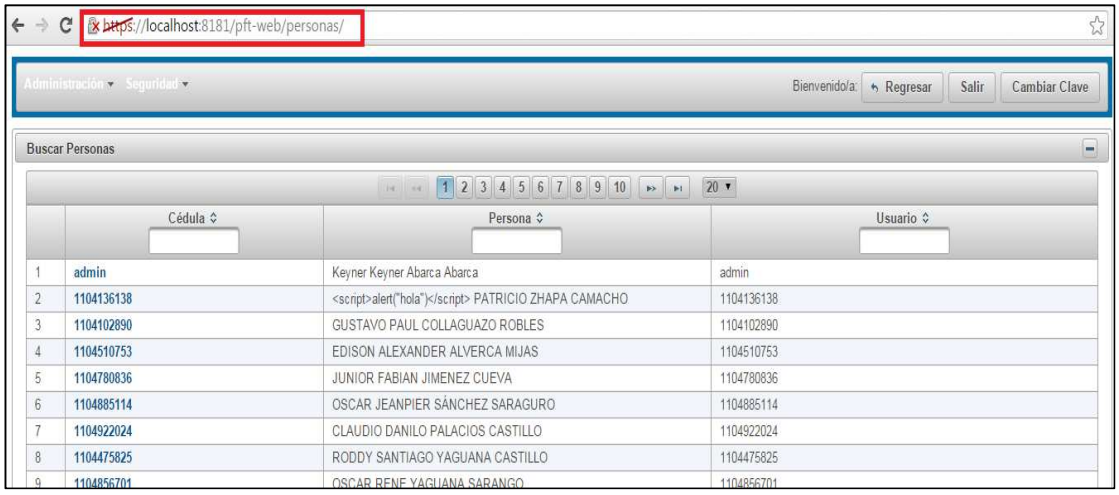

Figura 41. Seguridad a nivel de URL.
Fuente: Autor.

Elaboración: Autor.

Para la implementación de seguridad en el acceso a seguridades en el prototipo, se implementó el framework de seguridad Apache Shiro, el cual permite la configuración de la autenticación, restringiendo el acceso a usuarios no autenticados y negando que posean los correspondientes permisos tal como se observa en la figura 42.

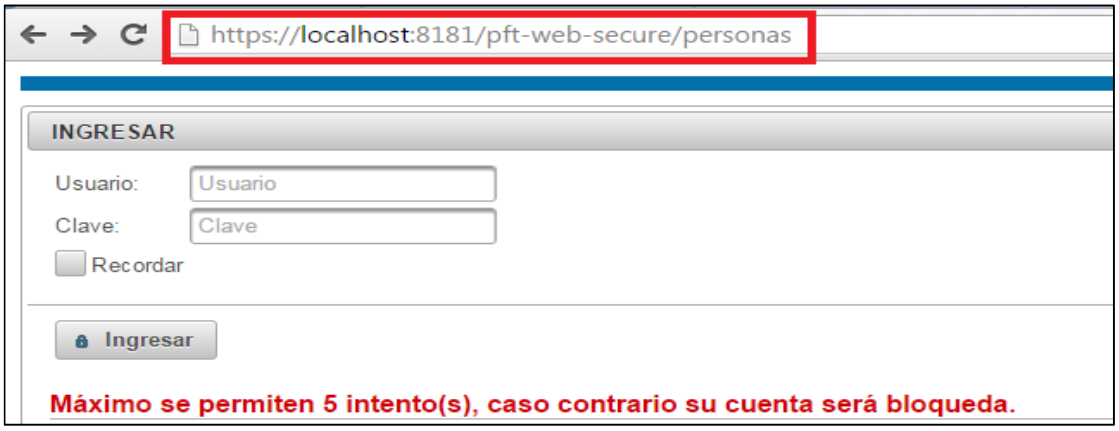

Figura 42. Restricción de acceso a usuarios no autenticados.
Fuente: Autor.
Elaboración: Autor.

Así también, la presente aplicación con las seguridades implementadas, permite denegar el acceso a una URL a usuarios que no poseen los permisos necesarios. En la Figura 43 se puede observar que una vez autenticado un usuario, éste desea ingresar a una URL que no le corresponde, el sistema automáticamente le arrojará un mensaje que no posee permisos de acceso.

Figura 43. Seguridad de Accesos.
Fuente: Autor.
Elaboración: Autor

ya se mencionó anteriormente se realizó las pruebas a las aplicaciones PFT- WEB sin ninguna seguridad y PFT-WEB con las seguridades propuestas en función a las vulnerabilidades, en la Tabla 17 se detalla un resumen los resultados de las pruebas.

Tabla 17. Resumen de pruebas

Prueba	Prototipo con seguridad	Prototipo Sin seguridad	Resultado 1 (Con seguridad)	Resultado 2 (Sin seguridad)
Criptografía	Algoritmo de encriptación sha256Hash	Texto Plano	Contraseñas encriptados en un array de 256 bits.	Contraseñas visibles.
Autenticación	Autenticación utilizando el Framework Apache Shiro	Ninguna	Contiene métodos ya implementados para la autenticación con un algoritmo hasher e inicio de sesión, la configuración se la realiza en un archivo llamado Shiro.ini y luego agregar los filtros en el web.xml para su funcionamiento.	Permite ingresar un usuario al sistema utilizando JPA y un procedimiento almacenado llamado login e inicia una sesión.
Autorización	Autorización con Framework Apache Shiro	Ninguna	Permite restringir el acceso a las URL de acuerdo al rol de cada usuario, para ello se debe configurar en el archivo apache.Shiro (ver anexo)	No tiene ninguna configuración ni restricción en el acceso a las URL (ver anexo).
Exposición de datos sensibles	Aplicación con Protocolo HTTPS	Aplicación con el protocolo HTTP	No permite capturar el usuario y contraseña por un Snnifer	Permite capturar y descifrar el usuario y contraseña por un ataque con Snnifer

ClickJacking	Configuración de la cabecera X-Frame-Options	Ninguna	Permite proteger contra la técnica maliciosa de engañar al usuario en hacer clic en funciones diferentes a las que se percibe.	Vulnerable a un ataque ClickJacking de acuerdo a la herramienta OWASP ZAP
Cache-Control	Configurar la cabecera Cache-Control mediante un filtro.	Ninguna	Permite que las respuestas del servidor no puedan se almacenadas en cache en los navegadores.	Las respuestas del servidor son almacenadas en las cache de los navegadores de acuerdo a la herramienta OWAS ZAP
Password Autocomplete in browser	Colocar autocomplete off en el componente password en la página de login.	Ninguna	No permite que en los navegadores guarden las contraseñas.	Permite que las contraseñas se almacenen en los navegadores.
Application Error Disclosure	Mostrar páginas de error personalizadas.	Ninguna	Permite redireccionar a una página los errores de la aplicación que corresponden al error HTTP 500	Permite que se pueda revelar errores detallados como la versión del servidor y en algunos casos fragmentos de código.
XSS Protection	Configurar la cabera X-XSS-Protection mediante un filtro.	Ninguna	Permite evitar las inyecciones XSS	Es vulnerable a ataques XSS

Fuente Autor.
Elaboración: Autor.

En la tabla 18, se muestra la relación entre los tipos de pruebas que se realizó en el prototipo asociadas con la vulnerabilidad seleccionada para el presente estudio.

Tabla 18. Relación Prueba-Vulnerabilidad

Prueba	Vulnerabilidad
Criptografía	Broken Authentication and Session Management
Autenticación	
Autorización	
Exposición de datos sensibles	
Cache-Control	
Password Autocomplete in browser	
Application Error Disclosure	
XSS Protection	
ClickJacking	Unvalidated Redirects and Forwards

Fuente: Autor.
Elaboración: Autor:

CONCLUSIONES

Como conclusiones del presente trabajo de fin de titulación y dando cumpliendo con los objetivos planteados inicialmente se menciona.

- ✓ Los patrones de diseño Facade y DAO, permiten un adecuado manejo de la estructuración a nivel de código del prototipo, permitiendo la encapsulación y manipulación de paquetes especializados para la recuperación de datos desde la base de datos.

- ✓ El framework de seguridad Apache Shiro permite implementar seguridades a nivel de la capa de presentación, utilizando métodos seguros como LoginController y SesionLogin, para la autenticación y autorización, permitiendo verificar la identidad del usuario con lo cual se proporciona seguridad al prototipo.

- ✓ La incorporación del algoritmo de encriptación SHA-256 en el framework Apache Shiro ayuda a la confidencialidad de los datos, convirtiendo las credenciales del usuario (username y password) en un arreglo de 256 bits, haciendo que estos viajen encriptados a través de la red y se almacenen de la misma forma en la base de datos.

- ✓ La implementación de procedimientos almacenados a nivel de base de datos permite reducir los ataques de inyección de código SQL, de esta manera contribuye a la seguridad de los datos que se encuentran almacenados y a su vez mejoran los tiempos de respuesta del prototipo hacia el usuario final.

- ✓ Apache Shiro permite controlar el acceso de los usuarios y que éstos no puedan acceder a las URL restringidas si no tienen la autorización y privilegios requeridos, esto se logra a través de la configuración de archivo Shiro.ini.

- ✓ La implementación de controles de seguridad a nivel de código en la capa de presentación, tales como número de intentos permitido en el logeo, tiempo de inactividad de sesión, longitud y complejidad de las contraseñas, permiten reducir el riesgos de ataques y de esta manera proporcionan seguridad a los sistemas.

RECOMENDACIONES

- ✓ Para la seguridad de las aplicaciones web es recomendable la utilización de frameworks de seguridad que estén sujetos a una comunidad activa en continuo mejoramiento y actualización proporcionando mayor información para la protección a los sistemas.
- ✓ Se recomienda la utilización de Apache Shiro como framework de seguridad en aplicaciones web, ya que el mismo posee un API de fácil implementación y configuración en el lenguaje de programación JAVA.
- ✓ Los certificados para la implementación de HTTPS no pueden ser generados por el servidor local de la aplicación, debido a que las páginas se mostrarían a los usuarios como no seguras, por ello se sugiere que los certificados se los pueda adquirir por unidades certificadoras.
- ✓ Para la generación de claves a los usuarios se debe utilizar un algoritmo hasher que no permita desencriptar una clave a través de una llave.
- ✓ Se recomienda la estructuración de la codificación a nivel de capas en la cual estén separados el acceso a datos, la presentación y la lógica de negocio.
- ✓ Tener en cuenta que las herramientas de validación de seguridad de las aplicaciones web poseen muchas otras funciones, como por ejemplo puede servir como proxys con la debida configuración.

BIBLIOGRAFÍA

Apache Software Fundación. (2015). Java Guía de Autorización con Apache Shiro. Retrieved from http://shiro.apache.org/java-authorization-guide.html

Arevalo, M. (n.d.). PATRÓN DE ARQUITECTURA POR CAPAS. COMPONENTES PRESENTES EN LA CAPA DE LÓGICA DE NEGOCIOS. Retrieved from https://arevalomaria.wordpress.com/2011/03/20/patron-de-arquitectura-por-capas-componentes-presentes-en-la-capa-de-logica-de-negocios/

Ayala, W. (2014). Arquitectura de Aplicaciones Web - Capa de Negocio. Retrieved from http://jmaw.blogspot.com/2013/01/arquitectura-de-aplicaciones-web-capa_5.html

Bass, B. L., Clements, P., & Kazman, R. (2003). *software architecture in practice* (Second Edi). Addison Wesley. Retrieved from http://disi.unal.edu.co/dacursci/sistemasycomputacion/docs/SWEBOK/Addison Wesley - Software Architecture In Practice 2nd Edition.pdf

Bermeo, E. (2012). *Análisis de la Arquitectura de desarrollo de sistemas N-Capas*. Retrieved from http://186.42.96.211:8080/jspui/bitstream/123456789/243/1/UNIVERSIDAD TECNOLOGICA ISRAEL.pdf

Bernard, E. (2014). Mapping Entities. Retrieved from https://docs.jboss.org/hibernate/stable/annotations/reference/en/html/entity.html

Buschmann, F., Meunie, R., Rohnert, H., Sommerlad, P., & Stal, M. (2001). *SOFT WARE ARCHITECTURE*. Retrieved from https://wiki.sch.bme.hu/images/9/98/Sznikak_jegyzet_Pattern-Oriented-SA_vol1.pdf

Candel, F. (2011). *II Congreso sobre las Nuevas Tecnologías y sus repercuciones en el seguro: Internet, Biotecnología y Nanotecnologia*. Barcelona. Retrieved from http://fundacionmapfre.com/ccm/content/documentos/fundacion/cs-seguro/libros/II-Congreso-Nuevas-Tecnologias-y-su-repercusion-en-el-seguro.pdf#page=97

Charlascylon, R. (2014). Patrones de diseño: qué son y por qué debes usarlos. Retrieved from http://www.genbetadev.com/metodologias-de-programacion/patrones-de-diseno-que-son-y-por-que-debes-usarlos#comments

Dávila, A., & Melendez, K. (2006). *Calidad del Producto Software Basados en Normas Internacionales* (Vol. 4, pp. 100–106). Retrieved from http://www.ewh.ieee.org/reg/9/etrans/ieee/issues/vol04/vol4issue2April2006/4TLA2_06Davila.pdf

Echeverría, E., Ballari, T., Molina, H., Wainerman, E., & Olsina, L. (n.d.). *Arquitectura Centrada en la Web para el Control y Monitoreo de Funcionalidad Domótica*. Retrieved from http://sedici.unlp.edu.ar/bitstream/handle/10915/23567/Documento_completo.pdf?sequence=1

EcuRed. (2014). Arquitectura de capas en sistemas de información. Retrieved from http://www.ecured.cu/Arquitectura_de_capas_en_sistemas_de_información#Arquitectura_de_Capas

Fernández Lanvin, D. (n.d.). *Definición de una arquitectura software para el diseño de aplicaciones web basadas en tecnología Java-J2EE*. Retrieved from http://di002.edv.uniovi.es/~dflanvin/doctorado/ArquitecturaJ2EE.PDF

Fielding, R. T. (2000). Software Architecture. Retrieved from https://www.ics.uci.edu/~fielding/pubs/dissertation/software_arch.htm#sec_1_5

Foundation, A. S. (2015). Java Authentication Guide with Apache Shiro. Retrieved from http://shiro.apache.org/java-authentication-guide.html

Garcés, K. (2015). ARQUITECTURA Y DISEÑO DE SOFTWARE. Retrieved from https://sistemasacademico.uniandes.edu.co/~isis2503/dokuwiki/doku.php?id=laboratorios:shiro

García Bautista, J. L. (2014). Programación en 3 capas. Retrieved from http://joseluisgarciab.blogspot.com/2014/09/programacion-en-3-capas.html

Gardazi, S. U., & Shahid, A. A. (2009). *Survey of software architecture description and usage in software industry of Pakistan*. Retrieved from http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=5353137&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D5353137

Guerrero, C. (2013). Reutilización del Software, *24*(3), 103–114. http://doi.org/10.4067/S0718-07642013000300012

Gutiérrez, P. (2013). Qué son y para qué sirven los hash?: funciones de resumen y firmas digitales. Retrieved from http://www.genbetadev.com/seguridad-informatica/que-son-y-para-que-sirven-los-hash-funciones-de-resumen-y-firmas-digitales

Herrera, M. (2007). Capa Lógica de Negocios. Retrieved from http://jmhogua.blogspot.com/2007/02/capa-lgica-de-negocios.html

Kimberly, A., Jiménez, J., & Valencia, O. (2013). *ANALISIS PARA LA DETECCION DE VULNERABILIDADES EN LA APLICACIÓN WEB COLIBRI II* (pp. 1–6). Retrieved from http://revista.upta.edu.ve/index.php/TI/article/view/1/1

Ma, S., Tang, J., & Wang, D. (2009). Process Based Application Level Architecture for RFID System. *2009 5th International Conference on Wireless Communications, Networking and Mobile Computing*, 1–5. http://doi.org/10.1109/WICOM.2009.5301380

Microsoft. (2009). Microsoft Application Architecture Guide. Retrieved from https://msdn.microsoft.com/en-us/library/ee658109.aspx

Mifsud, E. (2012). Introducción a la seguridad informática - Vulnerabilidades de un sistema informático. Retrieved from http://recursostic.educacion.es/observatorio/web/ca/software/software-general/1040-introduccion-a-la-seguridad-informatica?start=3

OWASP. (2013). *OWASP TOP-10 2013*. Retrieved from https://www.owasp.org/images/5/5f/OWASP_Top_10_-_2013_Final_-_Español.pdf

OWASP. (2014). Retrieved from https://www.owasp.org/index.php/Sobre_OWASP

Pavlova, M. (2010). Design Patterns. Retrieved from http://sourcemaking.com/design_patterns

Peláez Juan. (2009). Arquitectura basada en Componentes. Retrieved from http://geeks.ms/blogs/jkpelaez/archive/2009/04/18/arquitectura-basada-en-componentes.aspx

Pressman, R. (n.d.). *Ingeniería del Software*.

Rodríguez, N. (2011). OWASP: Creando aplicaciones seguras. Retrieved from http://www.genbetadev.com/seguridad-informatica/owasp-creando-aplicaciones-seguras#comments

Rojas, H. (2009). *Capítulo III. Análisis y diseño. 3.1*. Retrieved from http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/rojas_a_hi/capitulo3.pdf

Rosenblum, D. (2007). *Architectural Styles*. Retrieved from http://www.ccs.neu.edu/home/lieber/courses/csg110/sp08/lectures/april7/rosenblum-implicit-invoc.pdf

Scientist, C. (2008). A Comparison of Service-oriented, Resource-oriented, and Object-oriented Architecture Styles. Retrieved from http://research.microsoft.com/pubs/117710/3-arch-styles.pdf

Sheehan, J. (2015). Unvalidated Redirects and Forwards. Retrieved from http://www.montana.edu/itcenter/security/web/unvalidated-redirects-and-forwards.html

Sommerville, L. (2009). *Ingeniería del software Ingeniería del software*. *Benet Campderrich Falgueras II Editorial UOC*. Retrieved from http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Ingeniería+del+software+Ingeniería+del+software#0

Such, A. (2013). Componentes de Presentación. Retrieved from http://www.jtech.ua.es/j2ee/publico/jsf-2012-13/wholesite.pdf

Terreros, C. J. (2010). Desarrollo de Software basado en Componentes.

Tortosa, S. O. (2006). *Propuesta de una arquitectura de software*. Retrieved from http://dspace.uah.es/dspace/bitstream/handle/10017/472/Tesis.pdf?sequence=3&isAllowed=y

Vaencia, A., & Gonzales, M. (2011). Documentación y analisis crítico de algunas arquitecturas de software en aplicaciones empresariales. *Chemistry & …*. Retrieved from http://repositorio.utp.edu.co/dspace/bitstream/11059/2460/1/00422V152.pdf

Vargas, R., & Maltés, J. (2007). *Programación en Capas.* Retrieved from http://www.di-mare.com/adolfo/cursos/2007-2/pp-3capas.pdf

Venete, A. (2011). *Introducción a los Patrones de Diseño* (pp. 1–4). Retrieved from http://mahara.uji.es/artefact/file/download.php?file=54536&view=4648

Vignaga, A., & Perovich, D. (2014). Arquitecturas y tecnólogias para el desarrollo. Retrieved from

http://moodle2.unid.edu.mx/dts_cursos_mdl/pos/TI/LP/AM/01/Arquitecturas_y_tecnologias_para_el_desarrollo_de_aplicaciones_web.pdf

ANEXOS

A. Diseño de Base de Datos

Para la presente aplicación se utilizó la base de datos MySQL, la cual ofrece una serie de ventajas, tiene un motor de base de datos robusto, permite almacenar una gran cantidad de datos, y es open source; en la Figura 44 se puede observar un modelo conceptual de la base de datos.

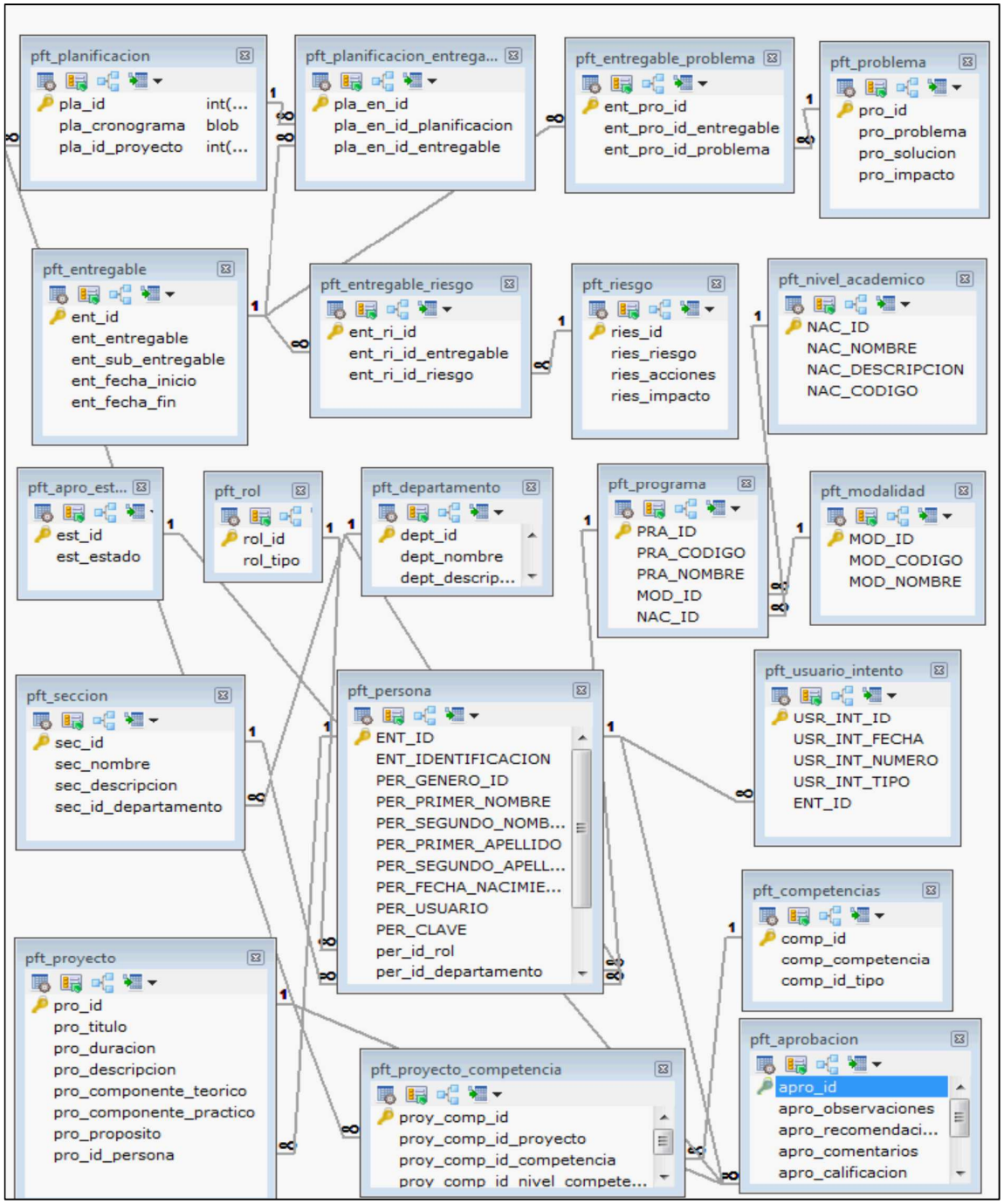

Figura 44. Modelo Conceptual.
Fuente: Autor.
Elaboración: Autor.

B. Configuración de servidor web.

Se realizó una configuración en dicho servidor para que todas las aplicaciones corran en un puerto seguro tal como se muestra en la figura 45.

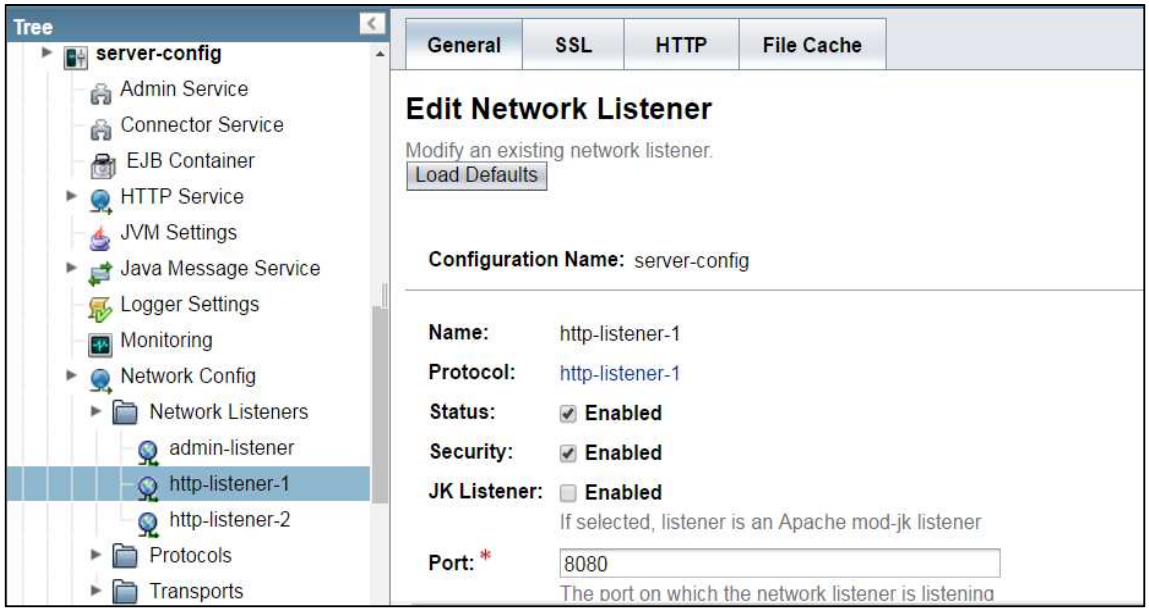

Figura 45. Configuración del protocolo HTTPS.
Fuente: Autor.
Elaboración: Autor.