Universidad de las Ciencias Informáticas

Facultad 5

Centro de Consultoría y Desarrollo de Arquitecturas Empresariales (CDAE)

Desarrollo de un mecanismo de autenticación centralizada para el entorno de aplicaciones de la UCI.

Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas.

Autor: Andy Torres Crego.

Tutores: Ing. Armando Masó Mosqueda.

Ing. Orson Omar Duque Cortina.

La Habana, Junio 2016

DECLARACIÓN DE AUTORÍA.

Declaro ser el autor de este trabajo y concedo a la Universidad de las Ciencias Informáticas con derechos patrimoniales del mismo, con carácter exclusivo.

Para así conste firmamos la presente a los __ días del mes de _____del año 2016.

_____

Autor

Andy Torres Crego

_____ _____

Tutor **Tutor**

Ing. Armando Masó Mosqueda Ing. Orson Omar Duque Cortina

DATOS DE CONTACTO.

Tutores:

Ing. Armando Masó Mosqueda

Universidad de las Ciencias Informáticas, Cuidad de La Habana, Cuba.

Correo: amaso@uci.cu

Ing. Orson Omar Duque Cortina

Universidad de las Ciencias Informáticas, Cuidad de La Habana, Cuba.

Correo: ooduque@uci.cu

DEDICATORÍA.

A la memoria de mi abuela Juanita que me enseñó muchas cosas de la vida y a la cual nunca olvidaré.

A mi mamá Ana, la mujer que más quiero en este mundo.

A mi tía Tere, por ser tan fuerte y decidida.

A mi hermano Landy, sin él no soy nadie en este mundo.

A mi primo Carlitos, el loco de la casa.

A mi papá Iván, por nuestras conversaciones.

A la revolución por darme la oportunidad de estudiar lo que me gusta y de convertirme en un ingeniero.

En general a toda mi familia por amarme tanto como lo han hecho y apoyarme en todas las decisiones que he tomado en mi vida.

AGRADECIMIENTOS.

A mi familia por todo el apoyo que me han brindado en estos cinco años de estudio y entrega.

A mi mamá que siempre me apoyó en mis sueños de convertirme en informático.

A mi tía, que es para mí mi segunda madre, que me brindo cariño en todo momento.

A mi hermano querido que siempre está ahí cuando lo necesito y representa mi modelo a seguir.

A mi primo con sus locuras que siempre me recuerda la familia feliz en la que vivo.

A mi papá por todos sus consejos.

Al tribunal por su labor.

A mis tutores por brindarme guía y consejo en la preparación de mi trabajo de diploma.

A mis compañeros de aula en especial con los que compartí apartamento y viví momentos inolvidables.

En general a todos los que conocí en el transcurso por la universidad.

A todos gracias.

RESUMEN.

El proceso de autenticación de usuarios en la Universidad de las Ciencias Informáticas (UCI) es de gran importancia para la seguridad de la intranet, la cual posee módulos de autenticación completamente independiente, provocando efectos negativos y debilitando la seguridad de los recursos. Por lo que se hace necesario la existencia de un sistema de autenticación única (SSO)[1] que controle centralizadamente los accesos garantizando la seguridad de la red y sus aplicaciones.

Tradicionalmente se ha pensado que implementar un *Single Sign-On* trae múltiples inconvenientes como afrontar problemas de seguridad y tener que asumir altos costos de implementación y mantenimiento. Sin embargo, las nuevas tecnologías desarrolladas y las infraestructuras de las aplicaciones han permitido que se reconsidere la posibilidad de implementar *Single Sign-On* como una estrategia para fortalecer la seguridad informática de las organizaciones.

En base a esto se comenzará por introducir el concepto de SSO indicando su utilidad, características distintivas, ventajas, requisitos, principales tipos de sistemas SSO existentes, concluyendo con algunas sugerencias y recomendaciones para la implantación de un SSO en una organización.

Palabras Clave.

Single Sign-On, Autenticación, Seguridad.

[1] SSO (*Single Sign-On*): Sistema de autenticación única.

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE IMÁGENES

INTRODUCCIÓN.

El desarrollo de las Tecnologías de la Información y las Comunicaciones (TIC) es incuestionable y forma parte de la cultura tecnológica con la que se interactúa en el mundo actualmente. Las TIC tienen, día a día, una mayor presencia en todos los aspectos de la vida laboral y personal, ofreciendo un nuevo espacio de innovación en ámbitos como la industria, los servicios, la salud, la administración, el comercio y la educación.

Actualmente el gran flujo de información que se maneja en el mundo mediante la red es abrumador y requiere que se proteja ante cualquier amenaza, ya sea la perdida de datos o la filtración de información confidencial e imprescindible para el negocio. Es por ello que para garantizar la seguridad de la información se hace necesario contar con un mecanismo de control de acceso, para el cual el proceso de autenticación es el primer paso sobre cualquier sistema de información.

La Universidad de Ciencias Informáticas cuenta con el Centro de Consultoría y Desarrollo de Arquitecturas Empresariales (CDAE) encargado de ejecutar proyectos de investigación, desarrollo u ofrecer servicios de consultoría tecnológica a partir del capital intelectual que se desarrolla en la universidad, como una actividad generadora de recursos financieros para el país.

A petición de la dirección de informatización de la universidad el CDAE pone en marcha el desarrollo de un mecanismo de autenticación única o *Single Sign-On* para el entorno de aplicaciones de la UCI, el cual tiene como objetivo proveer un mecanismo único de identificación para las personas en la universidad, adecuándose a las diferentes necesidades y condiciones de los usuarios. La universidad posee módulos de autenticación completamente independientes por lo que para acceder a la información alojada en varios sistemas los usuarios deben autenticarse usando el método de usuario/contraseña para cada uno de ellos, haciendo el trabajo más engorroso.

Por estas razones existe la necesidad de integrar varios subsistemas en uno mismo permitiendo al usuario una autenticación única a la entrada del sistema y que a la vez sus credenciales sean distribuidas a todas las aplicaciones de la organización, obteniendo mayor seguridad y control de la información. Por consiguiente, integrar algunos servicios como la autenticación del usuario en el sistema de manera centralizada constituye un problema a resolver.

A partir de la situación problemática antes expresada se define como **problema:**

¿Cómo centralizar el proceso de autenticación para facilitar a los usuarios el acceso a las aplicaciones web del entorno UCI?

Quedando definido como **objeto de estudio** El proceso de autenticación centralizada en sistemas web, especificándose como **campo de acción** El proceso de autenticación centralizada en sistemas web en la UCI.

De esta manera, la **idea a defender** con la presente investigación, es el despliegue de un modelo de integración, ajustada a las necesidades, estructura y estrategia de trabajo en la UCI, permitirá la gestión de autenticación centralizada.

Para la realización del trabajo se definieron sus objetivos de investigación, declarándose como **objetivo general**: Desplegar un sistema de integración que permita la autenticación centralizada para facilitar a los usuarios el acceso a las aplicaciones web del entorno UCI.

Se definen los siguientes **objetivos específicos:**

1. Realizar un estudio de los diferentes mecanismos de autenticación centralizada existentes.
2. Definir la arquitectura a utilizar.
3. Implementar y desplegar el sistema.
4. Validar el sistema desarrollado.

Para dar cumplimiento a las distintas tareas, se pusieron en práctica los siguientes métodos científicos:

Métodos Teóricos:

Histórico-Lógico: posibilitó conocer los antecedentes y tendencias actuales de la implementación de sistemas de SSO.
Analítico-Sintético: sirvió para realizar el procesamiento de toda la información, sintetizándola y diferenciándola para de esta forma enfocarla hacia el desarrollo.
Sistémico: facilitó la implementación de cada uno de los elementos desarrollados, en la conformación del sistema SSO.

Métodos Empíricos:

Medición: posibilitó establecer las comparaciones entre las arquitecturas básicas para implementar sistemas de SSO en cuanto a características, ventajas y desventajas que presentan.

El contenido del presente trabajo se encuentra comprendido en cuatro capítulos distribuidos de la siguiente manera:

Capítulo #1: Fundamentación teórica de la investigación.

En este capítulo se profundiza en los Sistemas de Autentificación Única (SSO), haciendo una explicación detallada en cuanto a características, ventajas y tipos que existen. Al mismo tiempo se realiza un análisis de las soluciones existentes, metodologías y herramientas a utilizar para su implementación.

Capítulo #2: Características del sistema.

En este capítulo se realiza un estudio y análisis de la arquitectura a usar para la implementación del SSO, los requerimientos de software, requerimientos funcionales y no funcionales, el modelo de dominio y de casos de uso, junto con la descripción detallada de cada caso de uso existente en la creación del sistema SSO.

Capítulo #3: Análisis y diseño del sistema.

En el capítulo se muestra el modelo de análisis y el modelo de diseño del sistema a desarrollar mostrando una visión más específica de los requisitos, describiendo cómo se va a implementar el software.

Capítulo #4: Implementación y prueba del sistema.

Se describen los estándares de codificación utilizados en la implementación. Se ilustran los principales resultados obtenidos y se aplican las pruebas a la solución propuesta.

CAPÍTULO 1: FUNDAMENTACIÓN TEAÓRICA DE LA INVESTIGACIÓN.

1.1 Introducción.

En el presente capítulo se pretende profundizar en los sistemas de autenticación única, haciendo una explicación detallada en cuanto a características, ventajas y los tipos existentes. También se refieren las definiciones o conceptos fundamentales: autenticación, centralizada, sistemas de autenticación única. Además, se describen las tecnologías y herramientas utilizadas para el análisis, diseño e implementación del sistema propuesto.

1.2 Conceptos Fundamentales.

A continuación, se presenta un conjunto de conceptos que ayudan a entender todo lo relacionado con el marco teórico de la investigación para lograr una mayor compresión y poder arribar a conclusiones.

El diccionario técnico de computación ofrece la siguiente definición:

Autenticación: Acto o proceso de validación de identidad, es la técnica mediante la cual un proceso comprueba que su compañero de comunicación es quien se supone que es y no un impostor. (1)

La real academia de la lengua española procura la siguiente definición:

Centralizada: Reunir varias cosas en un centro común o a hacer que varias cosas dependan de un poder central. (2)

Definiendo así autenticación centralizada como el proceso de validar la identidad de un sujeto en varios servicios mediante una única entrada al sistema.

Single Sign-On (SSO).

Existen muchas definiciones reflejadas en diferentes artículos de investigación refiriéndose a los sistemas de autenticación única las mismas se encuentran estrechamente relacionadas entre sí, pero no se debe confundir con la sincronización de contraseñas ya que en este no se requiere definir de igual forma todas las credenciales que utilizan los usuarios para cada aplicación o servicio, al contrario, permite poseer diferentes contraseñas para cada uno de los recursos, pero controlando el acceso a estos mediante una sola

identificación de usuario. Por lo que se bebe definir bien el concepto a tratar en el presente trabajo.

En el documento de *Aladdin Knowledge Systems* titulado Gestión de contraseñas se plantea:

¨Acceso a múltiples recursos por medio de un único proceso de ingreso. Gran cantidad de las arquitecturas implementadas en diferentes organizaciones han sido diseñadas con el objetivo de dar acceso a los usuarios a múltiples recursos, ya sean servicios Web y/o aplicaciones.¨ (3)

También en la documentación Implantación de un SSO se define el concepto:

¨Todas las aplicaciones dirigen la identificación de los usuarios al SSO, que, dependiendo del recurso al que desea el usuario acceder, exigirá un tipo de identificación más o menos fuerte, y permitirá o denegará el acceso una vez concluido el proceso de identificación del usuario.¨ (4)

El artículo de Jani Hursti hace referencia al SSO ideal:

¨En el mundo ideal, todo funciona sin problemas y los recursos informáticos se utilizan en la forma más beneficiosa. *Single Sign-On* no puede afectar a todo lo relacionado con la informática eficiente, pero hay cosas que puede hacer. Autentificarse en realidad significa la adquisición de una identidad electrónica. Por lo tanto, es un mapeo del mundo físico al mundo electrónico y lógico. En el mundo ideal, este mapeo es segura y conveniente. La identidad electrónica previsto se da sólo a ese individuo al que pertenece. Esto no sólo significa que el método de identificación no puede ser atacado contra sino también que no se puede utilizar de forma incorrecta.¨ (5)

Para este trabajo de diploma se tomará el concepto de SSO de *Aladdin Knowledge Systems* siendo el que más aterriza en el objetivo propuesto para poder realizar una aplicación que facilite la integración de los servicios Web y aplicaciones garantizando la seguridad de la información.

1.3 Estudio y desarrollo de un SSO en la universidad.

En la UCI hay un gran flujo de accesos a aplicaciones de la intranet, por lo que urge la existencia de un sistema que controle centralizadamente estos accesos y que garantice la seguridad de la red y sus aplicaciones. Debido a esta necesidad se han realizado

investigaciones y tesis sobre el tema para mayor conocimiento y posterior despliegue en la universidad.

Algunos trabajos sobre el tema son:

- Tesis Sistema de Gestión de Accesos (SGA), el cual brinda los servicios de autenticación y autorización de usuarios a las disímiles aplicaciones. (6)
- Tesis de Doctorado Modelo de Control de Acceso para Sistemas de Información en Entornos Multidominios (CAEM) en la cual se realiza una investigación del control de acceso a los sistemas de información y de la arquitectura de SSO ideal para unificar al proceso de autenticación en entornos multidominios. (7)
- Tesis Arquitectura, Análisis y Diseño del Sistema de Autenticación Única de RINDE (Red de Integración y Desarrollo Nacional de Software Libre de la República de Venezuela), con el objetivo de unificar el movimiento de Software libre en ese mediante la creación de un SSO que integraba tres subsistemas web. (8)
- Tesis Sistema de autenticación y autorización centralizado para el entorno de intranet de la empresa productora de petróleo PDVSA, el cual no contaba con un sistema de seguridad capaz de mantener el control interno, ayudando a los usuarios de la organización a tener acceso a múltiples aplicaciones de la infraestructura. (9)

1.4 Características de los SSO:

Los SSO, de manera general, presentan características distintivas, para su buen funcionamiento, como son:

- **Multiplataforma:**
  Facilita las tareas de inicio de sesión y de acceso a recursos de red desde distintas plataformas.
- **Transparencia:**
  El acceso a los recursos de sistemas se efectúa de forma transparente al usuario debido a la automatización del inicio de sesión.
- **Facilidad de uso:**
  El usuario se autentica una sola vez y el sistema le permite acceder a los recursos para los cuales está autorizado. Así se evita las interrupciones producidas por la solicitud de usuario y contraseña para el acceso a diferentes recursos.

- **Gestión sencilla:**
  El uso de SSO aconseja la sincronización de contraseñas e información de los usuarios. Esto implica la simplificación de la gestión de los recursos por parte de los administradores.
- **Control de acceso:**
  No se ve afectado por el uso de este sistema, SSO implica cambiar los mecanismos de autenticación del cliente y/o servidor, pero no modifica los permisos de los recursos.
- **Seguridad:**
  Depende de la arquitectura usada, pero en todos los casos la información viaja cifrada por la red. (10)

1.5 Ventajas de la implementación de un SSO.

La implementación de la estrategia de *Single Sign-On* sugiere algunos beneficios adicionales como son:

- Reducción de costos de administración de la seguridad.
- Disminución de la operatividad asociada con la administración de contraseñas.
- Incremento en los niveles de seguridad existentes.
- Control centralizado de autenticación para las aplicaciones corporativas.
- Mayor comodidad y facilidad de uso de las aplicaciones corporativas para los usuarios finales. (11)

1.6 Tipos de Sistemas Single Sign-On.

Existen cinco tipos principales de sistemas SSO, conocidos como *Reduced Sign-On Systems* (Sistemas de Autenticación Reducida). Los cuales consideran a continuación.

- ***Enterprise Single Sign-On (E-SSO):*** funciona luego de una autenticación primaria, interceptando los requerimientos de autenticación presentados por las aplicaciones secundarias para completarlos con el usuario y la contraseña.
- ***Web Single Sign-On (Web-SSO):*** conocido como *Web Access Management* (Web-AM), trabaja sólo con aplicaciones y recursos que se acceden vía Web.

- ***Kerberos*:** protocolo de autenticación mutua tanto cliente como servidor, y requiere un tercero de confianza.
- ***Federation:*** es una de las soluciones para abordar la gestión de identidad en los sistemas de información, permitiendo a los usuarios identificarse en redes de diferentes departamentos o incluso empresas.
- ***OpenID:*** es un proceso de SSO distribuido y descentralizado donde la identidad se compila en una URL de forma que cualquier aplicación o servidor puede verificar. (12)

Teniendo en cuenta que en la UCI el trabajo con aplicaciones y recursos se realiza vía Web, de los diferentes tipos de sistemas SSO existentes el más propicio a utilizar es *Web Single Sign-On.*

1.7 Soluciones existentes.

Existen en la actualidad muchas herramientas y soluciones que brindan un SSO, ya que permite una entrada sencilla a aplicaciones y plataformas a través de una única autenticación facilitando la vida de los usuarios e incrementando su productividad.

✓ CAS 4.0.1.

El servicio de autenticación central (CAS) es un único protocolo de inicio de sesión para la web. Su propósito es permitir a un usuario acceder a múltiples aplicaciones al tiempo que proporciona sus credenciales (tales como identificador de usuario y contraseña) sólo una vez. También permite que las aplicaciones web autentiquen a los usuarios, sin tener acceso a sus credenciales de seguridad, como una contraseña. El nombre CAS también se refiere a un paquete de software que implementa este protocolo.

El protocolo CAS implica al menos tres partes:

- Un navegador cliente web.
- La aplicación Web que solicita la autenticación.
- El servidor CAS. (13)

✓ OpenSSO Enterprise 8.0.

Es muy flexible en su configuración, cuenta con su propia *webapp* para brindar una interfaz de usuario sofisticada que le ayuda a configurar el *web-conteiner* con las diferentes aplicaciones a las que se le quiere dotar de SSO.

- Se necesita de un directorio LDAP[2] para proporcionar la SSO *authentication and Datastore.*
- Es una solución de nivel empresarial.
- Diseñado con tareas repetibles y escalables para el rápido despliegue de varias instancias.
- Proporciona un mayor conocimiento y control sobre cómo se comparten los datos de identidad con las redes de confianza. (14)

✓ JOSSO.

JOSSO, o Java Open inicio De sesión Único, es una fuente abierta de J2EE[3] y Spring-SSO basada en la infraestructura destinada a brindar una solución para centralizado, de plataforma neutral, autenticación y autorización de usuarios.
Las nuevas versiones de JOSSO son compatibles con la mayoría de los servidores web. Es una solución ampliamente difundida, y es más simple y fácil de configurar que OpenSSO. (15)

✓ adAS.

Servidor de Autenticación Avanzado que realiza funciones de Proveedor de Identidad con un entorno gráfico de configuración integrado que facilita su administración, puesta en marcha y mantenimiento.
A diferencia de otros productos de autenticación, adAS ofrece una gran flexibilidad adaptándose a cualquier tipo de entorno tecnológico y facilitando el despliegue de un servicio de *Single Sign-On*.

---

[2] LDAP (*Protocolo ligero de acceso a directorios*): Permite administrar directorios.

[3] J2EE (*Java Enterprise Edition*): Es una plataforma destinada a desarrollar aplicaciones empresariales distribuidas escrita en Java.

Características:

- El usuario accede a las diferentes aplicaciones de su organización con una sola identificación.
- Reduce las incidencias relacionadas con el olvido de contraseñas y la existencia de múltiples nombres de usuarios para una misma persona.
- Dispone de una herramienta de administración sencilla e intuitiva que facilita la gestión del sistema. Incorpora un motor de *Business Intelligence* (BI) que utiliza el cuadro de mando para realizar análisis del estado del SSO.
- Pretende, no sólo ser una herramienta de SSO eficaz y segura, si no también hacer más fácil a los usuarios la migración a este entorno, llegando a ser transparente.
- Permite al usuario autenticarse presentando diversos tipos de credenciales, como por ejemplo nombre de usuario y contraseña, DNI[4] electrónico, certificados digitales, Kerberos, etc.
- Realiza el control de acceso a las aplicaciones basado en la información de los usuarios, pudiendo configurar las políticas de autorización desde la herramienta de administración de adAS.
- Reduce los costes de gestión de nuestras infraestructuras, puesto que evita a las aplicaciones tener que gestionar la identidad digital de los usuarios e implementar mecanismos de control extra.

Gracias a sus características específicas, se proporciona un entorno de seguridad y confianza en la gestión y uso de la identidad digital en la organización.

Es posible identificar al usuario en una organización externa, realizándose el control de acceso en adAS. Este modelo es conocido como Autenticación Delegada. (16)

Por todo lo antes expuesto, y teniendo en cuenta que el tipo de sistema SSO escogido, debido a las características del entorno de trabajo de la UCI, de las diferentes soluciones que implementan un sistema de autenticación única se trabajara con el servicio de autenticación central, único protocolo de inicio de sesión para la web.

[4](*DNI*) Documento Nacional de Identidad.

1.7.1 Metodología.

Dentro del desarrollo de software y a la alta necesidad de que los proyectos lleguen al éxito y obtener un producto de gran valor para nuestros clientes, generan grandes cambios en las metodologías adoptadas por los equipos para cumplir sus objetivos, puesto que, unas se adaptan mejor que otras, al contexto del proyecto brindando mejores ventajas.

Es por eso de la importancia de una metodología robusta que ajustada en un equipo cumpla con sus metas, y satisfaga más allá de las necesidades definidas al inicio del proyecto.

El éxito del producto depende en gran parte de la metodología escogida por el equipo, ya sea tradicional o ágil, donde los equipos maximicen su potencial, aumenten la calidad del producto con los recursos y tiempos establecidos. (17)

Metodología de Proceso Unificado Ágil (*Agile Unified Process*, AUP).

Es una versión simplificada del Proceso Unificado de Racional (RUP). Este describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de software de negocio usando técnicas ágiles conceptos que aún se mantienen válidos en RUP.

Características principales:

- Es iterativo incremental.
- Divide el proyecto en mini-proyectos.
- Detecta a tiempo los riesgos.
- Administra adecuadamente los cambios.
- Mayor grado de reutilización.
- Mayor experiencia de grupos.
- Los casos de uso son la base del desarrollo del proyecto.
- Realiza múltiples modelos y vistas que definan la arquitectura de software de un sistema. (18)

1.7.2 *Frameworks.*

Un *framework* es una estructura de software compuesta de componentes personalizables e intercambiables para el desarrollo de una aplicación. Puede considerarse como una

aplicación genérica incompleta y configurable a la que podemos añadirle las últimas piezas para construir una aplicación concreta. Entre los objetivos principales que persigue se encuentran: acelerar el proceso de desarrollo, reutilizar código ya existente y promover buenas prácticas de desarrollo como el uso de patrones. (19)

Spring 3.2.0.

Es un framework modular de código abierto para el desarrollo de aplicaciones Java, está dividido en siete módulos donde se destaca su flexibilidad y alto valor de configuración, se acopla a las aplicaciones sin tener que modificarlas para usar sus beneficios. Hace una clara división entre las capas de arquitecturas y está basada en interfaces para enmarcar las funcionalidades, disminuye el margen de errores, se obtiene una mayor limpieza y claridad en el código, reduce potencialmente el enlace entre los diferentes componentes de la aplicación. No necesita muchos recursos para su ejecución por lo que es descrito como liviano para la variedad de servicios que brinda. Propone un módulo que soporta los *frameworks* ORM[5] más populares del mercado dentro de los cuales se encuentra *Hibernate*. (20)

Hibernate 3.2.5.

Es una capa de persistencia objeto/relacional y un generador de sentencias SQL. Te permite diseñar objetos persistentes que podrán incluir polimorfismo, relaciones, colecciones, y un gran número de tipos de datos. De una manera muy rápida y optimizada podremos generar BBDD en cualquiera de los entornos soportados: Oracle, DB2, MySql, etc. Y lo más importante de todo, es *open source*, lo que supone, entre otras cosas, que no tenemos que pagar nada por adquirirlo. (21)

Bootstrap 3.0.

Es un framework de CSS[6], en otras palabras, es un conjunto de archivos que incluyes en tu página y puedes empezar a maquetar el sitio web en minutos, sin tocar una sola línea de CSS. Muy cómodo para las personas que desarrollan webs ya que ofrece las herramientas

[5] ORM (*Object-Relational Mapping*): Mapeo Objeto-Relacional.

[6] CSS (*cascading style sheets)* Hoja de estilo en cascada, lenguaje usado para estructurar un documento HTML o XML.

necesarias para crear cualquier tipo de sitio utilizando los estilos y elementos de sus librerías y es compatible con la mayoría de navegadores web del mercado. (22)

1.7.3 Servidor web.

Un servidor web es un programa que procesa una aplicación del lado del servidor realizando conexiones bidireccionales y/o unidireccionales con el cliente, generando o cediendo una respuesta en cualquier lenguaje. El código recibido por el cliente suele ser compilado y ejecutado por un navegador web. (23)

Apache Tomcat 7.0.41.

Apache Tomcat es un servidor web con soporte de servlets[7] y JSP[8] desarrollado bajo el proyecto Jakarta en la *Apache Software Foundation.* Su implementación es en el lenguaje de programación Java y es compatible con las API más recientes. Debido a su implementación en tecnologías libres puede ser usado en cualquiera de los sistemas operativos, solo necesitan tener instalado una máquina virtual de Java. Es uno de los servidores web más populares entre los desarrolladores. (24)

WampServer 2.2.

WampServer es un entorno de desarrollo web para Windows con el que se pueden crear aplicaciones web con Apache, PHP y bases de datos MySQL *database*.

Provee a los desarrolladores con los cuatro elementos necesarios para un servidor web: un Sistema Operativo (Windows), un manejador de base de datos (MySQL), un software para servidor web (Apache) y un software de programación script Web (PHP (generalmente), Python o PERL), debiendo su nombre a dichas herramientas. Incluye, además de las últimas versiones de Apache, PHP y MySQL, versiones anteriores de las mismas, para el caso de que se quiera testear en un entorno de desarrollo particular.

El uso de WAMP permite servir páginas HTML a Internet, además de poder gestionar datos en ellas, al mismo tiempo proporciona lenguajes de programación para desarrollar aplicaciones. (25)

---

[7] Servlets: Clase en el lenguaje de programación Java, utilizada para ampliar las capacidades de un servidor.

[8] JavaServer Pages (JSP): Tecnología que ayuda a los desarrolladores de software a crear páginas web dinámicas.

1.7.4 Herramienta CASE.

Esencialmente, un CASE (*Computer-Aided Software Engineering,* Ingeniería de Software Asistida por Computadora*)* es una herramienta que ayuda al ingeniero de software a desarrollar y mantener software.

En *The CASE Experience* se ofrece la siguiente definición:
"Una combinación de herramientas de software y metodologías de desarrollo". (26)

Visual Paradigm For UML 8.0.

Visual Paradigm es una herramienta visual de ingeniería de software para el modelado, que tiene un entorno de trabajo que muestra una colección de menús, barra de herramientas y ventanas que permiten realizar diagramas. Permite confeccionar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. Posibilita el uso de varios idiomas, soporta el modelado de sistemas web, es un producto de calidad, fácil de instalar, actualizar y posee gran compatibilidad entre ediciones. Posee como peculiaridad sobre el resto de las herramientas que cuenta con una potente funcionalidad para la creación de interfaces de usuario de las aplicaciones. Es una herramienta que es soportada por cualquier sistema operativo. (27)

1.7.5 Lenguajes de programación.

Los lenguajes de programación son herramientas que nos permiten crear programas y software. Entre ellos tenemos Delphi, Visual Basic, Pascal, PHP, Java, entre otros, en particular se conoce como código de máquinas o lenguaje de máquinas. (28)

Java.

Lenguaje de programación de propósito general orientado a objetos desarrollado por Sun Microsystems. Es una tecnología que no sólo se reduce al lenguaje, sino que además provee de una máquina virtual Java que permite ejecutar código compilado Java, sea cual sea la plataforma que exista por debajo; plataforma tanto hardware, como software (el sistema operativo que soporte ese hardware).

Razones para escoger Java por sobre otros lenguajes:

- Es orientado a objetos.
- Es muy flexible.

- ❖ Funciona en cualquier plataforma.
- ❖ Su uso no acarrea inversiones económicas.
- ❖ Es de fuente abierta.
- ❖ Es un lenguaje expandible. (29)

PHP 5.4.3.

PHP (acrónimo recursivo de *PHP: Hypertext Preprocessor*) es un lenguaje de código abierto muy popular especialmente adecuado para el desarrollo web.

Lo que distingue a PHP de algo del lado del cliente como Javascript es que el código es ejecutado en el servidor, generando HTML[9] y enviándolo al cliente. El cliente recibirá el resultado de ejecutar el script, aunque no se sabrá el código subyacente que era. El servidor web puede ser configurado incluso para que procese todos los ficheros HTML con PHP, por lo que no hay manera de que los usuarios puedan saber qué se tiene debajo de la manga. Lo mejor de utilizarlo es su extrema simplicidad para el principiante, pero a su vez ofrece muchas características avanzadas para los programadores profesionales. (30)

1.7.6 Entorno de desarrollo integrado.

Eclipse Mars.

Según la web oficial de Eclipse se define como "*An IDE for everything and nothing in particular*" (un IDE para todo y para nada en particular).

Eclipse es, en el fondo, únicamente un armazón (*workbench*) sobre el que se pueden montar herramientas de desarrollo para cualquier lenguaje, mediante la implementación de los *plugins*[10] adecuados. El entorno de desarrollo Eclipse, incluyendo sus *plugins*, está desarrollado por completo en el lenguaje Java. Un problema habitual en herramientas Java (como NetBeans) es que son demasiado "pesadas". Es decir, necesitan una máquina muy potente para poder ejecutarse de forma satisfactoria. (31)

---

[9] HTML(*HyperText Markup Language*): Hace referencia al lenguaje de marcado para la elaboración de páginas web.

[10] *Plugin*: Un complemento es una aplicación que se relaciona con otra para aportarle una función nueva y generalmente muy específica.

1.7.7 Sistema gestor de bases de datos.

Un Sistema Gestor de Bases de Datos o SGBD, también llamado DBMS (*Data Base Management System*) como una colección de datos relacionados entre sí, estructurados y organizados, y un conjunto de programas que acceden y gestionan esos datos. La colección de esos datos se denomina Base de Datos o BD, (DB *Data Base*). Algunos ejemplos de SGBD son Oracle, DB2, PostgreSQL, MySQL, MS SQL Server, entre otros.

Un SGBD debe permitir:
• Definir una base de datos: especificar tipos, estructuras y restricciones de datos.
• Construir la base de datos: guardar los datos en algún medio controlado por el mismo SGBD.
• Manipular la base de datos: realizar consultas, actualizarla, generar informes. (32)

PostgreSQL 9.2.4.

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD y con su código fuente disponible libremente. Es el sistema de gestión de bases de datos de código abierto más potente del mercado y en sus últimas versiones no tiene nada que envidiarles a otras bases de datos comerciales.

PostgreSQL utiliza un modelo cliente/servidor y usa *multiprocesos* en vez de *multihilos* para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando.

Sus características técnicas la hacen una de las bases de datos más potentes y robustas del mercado:

- Es una base de datos 100% ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad).[11]
- Integridad referencial.
- Copias de seguridad.
- *Unicode.*
- Juegos de caracteres internacionales.
- Regionalización por columna.

[11] ACID: Características de los parámetros que permiten clasificar las transacciones de los SGBD.

- Múltiples métodos de autentificación.
- Acceso encriptado vía *Secure Sockets Layer* (SSL).[12]
- Completa documentación. (33)

MySQL 5.5.24.

Es un sistema de administración de bases de datos (*Database Management System, DBMS*) para bases de datos relacionales. Así, MySQL no es más que una aplicación que permite gestionar archivos llamados de bases de datos.

Existen muchos tipos de bases de datos, desde un simple archivo hasta sistemas relacionales orientados a objetos. MySQL, como base de datos relacional, utiliza múltiples tablas para almacenar y organizar la información. Fue escrito en C y C++ y destaca por su gran adaptación a diferentes entornos de desarrollo, permitiendo su interactuación con los lenguajes de programación más utilizados como PHP, Perl y Java y su integración en distintos sistemas operativos.

También es muy destacable, la condición de open source de MySQL, que hace que su utilización sea gratuita e incluso se pueda modificar con total libertad, pudiendo descargar su código fuente. Esto ha favorecido muy positivamente en su desarrollo y continuas actualizaciones. (34)

1.7.8 Pruebas al sistema.

Las pruebas son básicamente un conjunto de actividades dentro del desarrollo de *software*. Dependiendo del tipo de pruebas, estas actividades podrán ser implementadas en cualquier momento de dicho proceso de desarrollo. Existen distintos modelos de desarrollo de software, así como modelos de pruebas. A cada uno corresponde un nivel distinto de involucramiento en las actividades de desarrollo. (35)

Apache-jmeter-2.10.

JMeter es una herramienta Java desarrollada para realizar pruebas de rendimiento y pruebas funcionales sobre aplicaciones web. Permite la ejecución de pruebas distribuidas entre distintos ordenadores, para realizar pruebas de rendimiento. Además, activar o

[12] (SSL): Seguridad de la capa de transporte.

desactivar una parte del test, lo que es muy útil cuando se está desarrollando un test largo, y se desea deshabilitar ciertas partes iniciales que sean muy pesadas o largas. Tiene la forma de generar un caso de prueba a través de una navegación de usuario.

Para realizar pruebas de Carga con JMeter ante todo se debe de tener en cuenta los siguientes requerimientos:

- **Hardware:**

PC de 1 GB o superior de RAM.

- **Software:**

Instalación de una máquina virtual de Java.

- **Para interpretar los resultados:**

- Cantidad de usuarios concurrentes que necesita soportar la aplicación.
- Tiempo de respuesta en situaciones de carga de una transacción.
- Tiempo de recuperación del sistema.
- Caso de prueba.

El proyecto debe realizar el caso de prueba donde describa los escenarios significativos que serán sometidos a la prueba. Dígase camino básico a realizar para llegar hasta esos escenarios. (36)

Después de haberse realizado un análisis de los conceptos, los diferentes tipos de SSO existentes y el estudio de los *frameworks* definidos, se llegó a la conclusión.

1.8 Conclusiones parciales.

- Después de haberse realizado un análisis de los conceptos y los diferentes tipos de SSO existentes, se llegó a la conclusión que para aplicar en la Universidad de las Ciencias Informáticas un sistema de acceso a múltiples recursos por medio de un único proceso de ingreso, el más propicio a utilizar es *Web Single Sign-On*.

- Teniendo en cuenta el tipo de sistema SSO escogido que se utilizara el servicio de autenticación central, único protocolo de inicio de sesión para la web.

- El sistema se desarrollará utilizando gestor de base de datos PostgreSQL, la programación se hará con Java por las ventajas que esta brinda y, para el análisis y desarrollo se utilizara la Metodología AUP que a su vez hará uso del lenguaje de modelado (UML) utilizando Visual Paradigm como herramienta CASE.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA.

2.1 Introducción.

En el presente capítulo se realiza un estudio y análisis de las causas por las cuales se decide adoptar la realización del SSO utilizando la herramienta CAS, mostrando así los beneficios que aportaría a la Universidad de las Ciencias Informáticas. Se muestran los requerimientos de software, requerimientos funcionales y no funcionales, el modelo de dominio y de casos de uso, junto con la descripción detallada de cada caso de uso existente en la implantación del sistema SSO.

2.2 Propuesta de solución.

Mediante la investigación realizada en el capítulo anterior se sentaron las bases para dar solución al problema de cómo poder centralizar el proceso de autenticación a las aplicaciones web del entorno UCI, el sistema a desarrollar tiene como objetivo acceder a diferentes aplicaciones web por medio de un único ingreso de credenciales.

Se realizará el despliegue de un sistema de autenticación única de tipo web (SSO-Web) utilizando el Servicio de Autenticación Centralizada vinculándolo con dos aplicaciones una en el lenguaje Java y el otro en PHP, que les facilitará el trabajo a los usuarios a la hora de interactuar con la intranet de la universidad, Permitiendo que con solo un ingreso de credenciales las mismas sean distribuidas por las aplicaciones proporcionando una mayor seguridad al solo tener que ingresar una sola vez sus datos al sistema.

2.3 Arquitectura del sistema.

Existen diversos tipos de arquitecturas que permiten implementar un SSO en una determinada organización dependiendo de los recursos tecnológicos y económicos disponibles, la complejidad deseada en la infraestructura y la decisión de diseño establecida para el desarrollo del sistema.

Las diferentes arquitecturas SSO están compuestas por tres componentes básicos:

- *Interface:* El modo en que el SSO interactúa con una determinada aplicación. Usualmente reside en el cliente, y es conocido como Agente SSO.
- Administración: El mecanismo que permite configurar, mantener y monitorear el proceso de SSO.

- ❖ Credenciales: Cada aplicación a la que se accede requiere información confidencial (nombre de usuario y contraseña), que agrupada recibe el nombre de credenciales. Las credenciales deben almacenarse de manera protegida para que sea únicamente el agente SSO quien pueda acceder a ellas.

2.3.1 Diseño de la arquitectura a utilizar.

Arquitectura de administración y almacenamiento de credenciales centralizados:

La arquitectura SSO con administración y almacenamiento centralizado pretende solucionar los principales inconvenientes encontrados en la arquitectura que almacena las credenciales localmente, teniendo un mayor grado de seguridad en cuanto al componente de las credenciales, que se almacenan en un lugar centralizado, facilitando así su protección, administración y manejo.

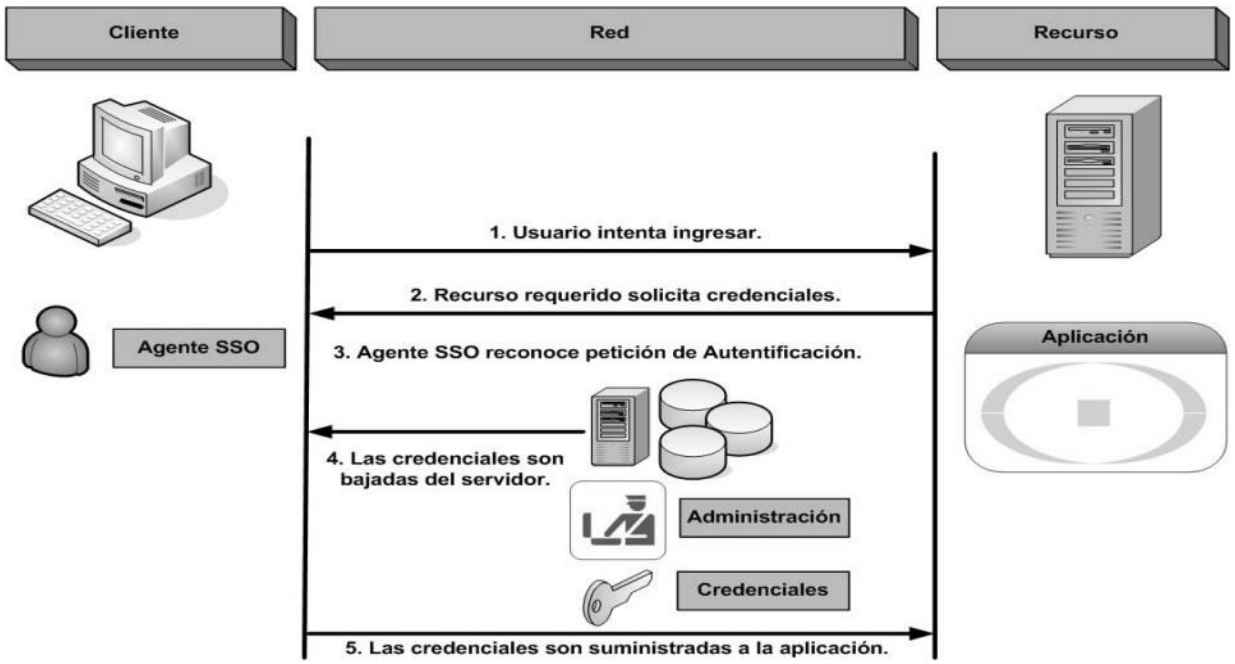

Ilustración 1. Arquitectura de administración y almacenamiento de credenciales centralizados.

Características:

- ➢ Las credenciales son migradas a un servidor central, quien entrega las credenciales al cliente correspondiente en el momento de hacer el ingreso.
- ➢ El administrador determina la frecuencia con que se descargan las credenciales del servidor.

Ventajas:

- Permite a los usuarios el acceso a las aplicaciones desde cualquier estación, previa autenticación del mismo.
- Ofrece administración centralizada de credenciales disminuyendo posible manipulación de la misma en el cliente. (37)

2.4 Modelo de Dominio.

El Modelo de Dominio o Modelo Conceptual como se muestra en la Ilustración 1 es una representación visual de los conceptos u objetos del mundo real significativos para el problema o área de interés. Permite de manera visual mostrar al usuario los principales conceptos que se manejan en el dominio del sistema en desarrollo. Esto ayuda a los usuarios, clientes, desarrolladores e interesados a utilizar un vocabulario común para poder entender el contexto en que se enmarca el sistema. (38)

2.4.1 Diagrama conceptual del Modelo de Dominio.

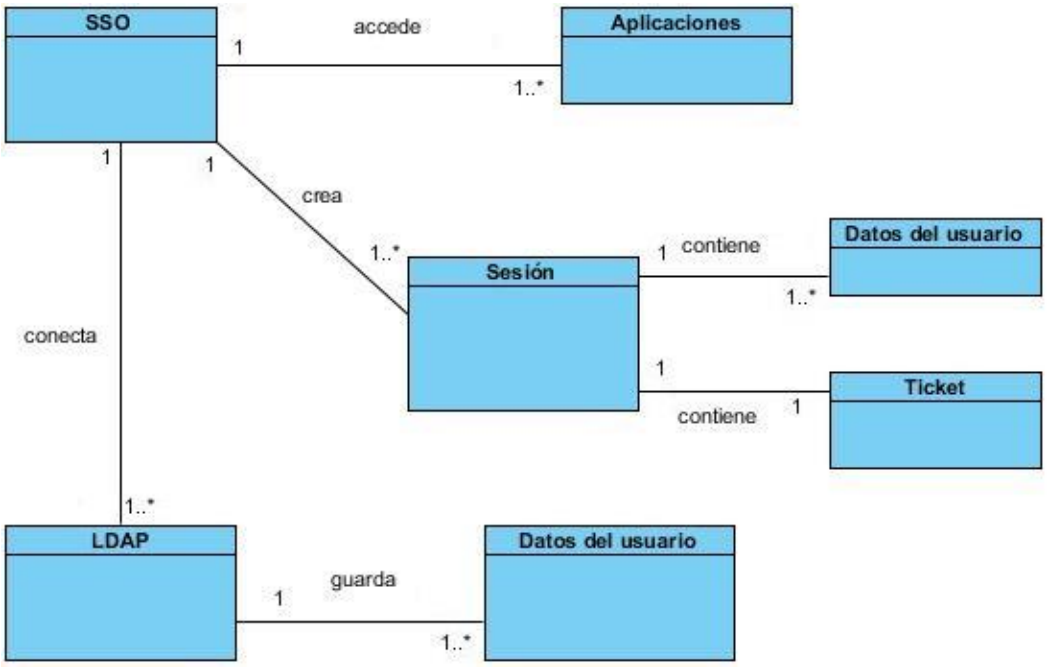

Ilustración 2. Modelo de Dominio.

2.4.2 Conceptos de las entidades utilizados en el diagrama:

- **SSO:** Sistema de acceso a múltiples recursos ya sea servicios Web y/o aplicaciones por medio de un único proceso de ingreso.
- **Aplicaciones:** Programas diseñados como herramientas que permiten a un usuario realizar uno o diversos tipos de trabajos mediante la web.
- **LDAP:** Es un servicio de directorio funcionando como una base de datos especializada para hacer búsquedas.
- **Sesión:** Relación del usuario con el sistema en el inicio de sesión.
- **Datos de los usuarios:** Información referente al usuario.
- **Ticket:** Identificador que se les proporciona a las aplicaciones para interactuar con el sistema.

2.5 Especificación de los requisitos de software.

Las especificaciones de los requerimientos del software especifican qué es lo que el sistema debe hacer (sus funciones) y sus propiedades esenciales y deseables. (39)

El sistema SSO que se pretende desplegar, según la dirección de informatización de la universidad, debe permitir a los usuarios autenticarse sin problemas usando las credenciales del dominio UCI accediendo a diferentes aplicaciones con sólo un método de ingreso permitiéndoles navegar por la intranet.

2.5.1 Requisitos funcionales:

Los requisitos funcionales de un sistema describen lo que el sistema debe hacer. Estos requerimientos dependen del tipo de software que se desarrolle, de los posibles usuarios y del enfoque general de los requerimientos. (40)

Las condiciones que el sistema debe cumplir son:

RF1. El sistema debe permitir la autenticación de un usuario.

RF2. El sistema debe validar el ticket de seguridad.

RF3. El sistema debe verificar si una aplicación tiene permisos para utilizarlo.

RF4. El sistema debe permitir a una aplicación cerrar sesión.

2.5.2 Requisitos no funcionales:

Los requisitos no funcionales son aquellos requerimientos que no se refieren directamente a las funciones específicas que proporciona el sistema, sino a las propiedades emergentes de éste. (41)

- **Fiabilidad:** Capacidad de un sistema o componente para desempeñar las funciones especificadas, cuando se usa bajo unas condiciones y periodo de tiempo determinados.

  - **Rendimiento:** Capacidad del sistema de ajustarse a las expectativas del cliente en cuanto a los tiempos de respuesta y capacidad de prestar servicio adecuadamente. (42)

RNF1. La aplicación debe responder en un tiempo razonable.

- **Compatibilidad:** Capacidad de dos o más sistemas o componentes para intercambiar información y/o llevar a cabo sus funciones requeridas cuando comparten el mismo entorno hardware o software.
- **Interoperabilidad:** Capacidad para intercambiar información entre dos o más componentes y utilizarla. (42)

RNF2. La aplicación debe poder vincularse con los sistemas web implementados en la universidad.

- **Usabilidad:** Capacidad del producto software para ser entendido, aprendido, usado y resultar atractivo para el usuario, cuando se usa bajo determinadas condiciones.
- **Estética de la interfaz de usuario:** Interfaz agradable para satisfacer la interacción con el cliente. (42)

RNF3. La aplicación debe poseer una interfaz agradable para el usuario.

- **Seguridad:** Capacidad de protección de la información y los datos de manera que personas o sistemas no autorizados no puedan leerlos o modificarlos.
- **Autenticidad**: Capacidad de demostrar la identidad de un sujeto o recurso. (42)

RNF4. La aplicación debe poder identificar al usuario que ingresa al sistema.

2.6 Modelos de Casos de Uso del Sistema.

A continuación, se reconocen los posibles actores que van a interactuar con el sistema a desarrollar, así como los casos de uso del sistema.

2.6.1 Definición de los actores.

El primer paso al escribir un caso de uso consiste en definir el conjunto de ¨actores¨ que estarán involucrados con la historia. Los actores son las diferentes personas (o dispositivos) que utilizarán el sistema o producto dentro del contexto de la función y el comportamiento que se describirá. (43)

Actores	**Descripción**
Aplicación	Permite a los usuarios realizar uno o diversos tipos de operaciones en el sistema.
Usuario	Encargado de identificarse para navegar en el sistema.

Tabla 1. Definición de los actores.

2.6.3 Casos de Uso del Sistema.

Un caso de uso es una descripción de los pasos o las actividades que realiza un actor al interactuar con el sistema en un conjunto específico de circunstancias. (43)

CU 1. Autenticar usuario.

CU_1	Autenticar usuario
Actores	Aplicación
Descripción	El caso de uso comienza cuando la aplicación quiere autentificar un usuario en el sistema devolviendo un ticket de seguridad que identifica la sesión.
Referencia	RF1

Tabla 2. CU 1. Autenticar usuario.

CU 2. Validar ticket.

CU_2	Validar ticket
Actores	Aplicación
Descripción	El caso de uso comienza cuando la aplicación le pide al sistema validar el ticket, si existe devuelve la respuesta de que la autenticación del usuario es válida.
Referencia	RF2

Tabla 3. CU 2. Validar ticket.

CU 3. Verificar permisos.

CU_3	Verificar permisos
Actores	Aplicación
Descripción	El caso de uso comienza cuando accede una aplicación al sistema y el mismo verifica los permisos de la aplicación.
Referencia	RF3

Tabla 4. CU 3. Verificar permisos.

CU 4. Cerrar sesión.

CU_4	Cerrar sesión
Actores	Aplicación
Descripción	El caso de uso comienza cuando la aplicación le pide al sistema cerrar la sesión.
Referencia	RF4

Tabla 5. CU 4. Cerrar sesión.

2.6.4. Diagramas de Casos de Uso.

Los diagramas de casos de uso se utilizan para ilustrar los requerimientos del sistema representando gráficamente la interacción entre los actores del sistema con los procesos del negocio.

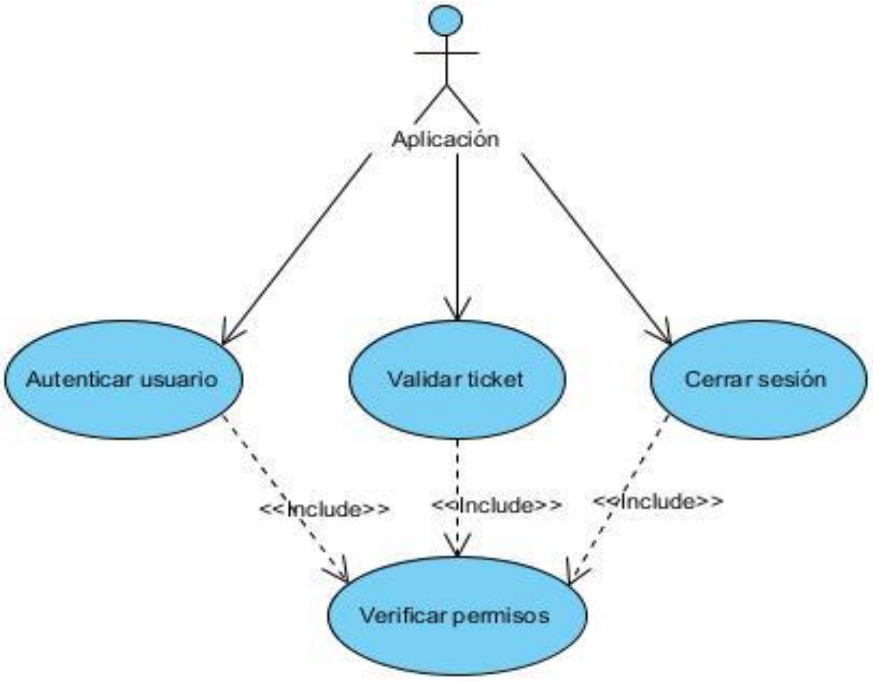

Ilustración 3. Diagrama de Casos de Uso.

2.6.5 Descripción de los Casos de Uso del Sistema.

CU 1	**Autenticar usuario**
Propósito:	Validar los datos ingresados por el usuario para autenticar al sistema
Actor:	Aplicación
Resumen:	El caso de uso comienza cuando la aplicación quiere autentificar un usuario en el sistema devolviendo un ticket de seguridad que identifica la sesión.
Precondiciones:	La aplicación debe tener permisos.
Referencias:	RF1
CU asociados:	CU_3
Prioridad:	Crítico
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**

1. Una aplicación desea autenticar a un usuario entrando los datos para su autenticación.	1.1 Se ejecuta el CU asociado. 1.2 El sistema verifica que los datos sean correctos. 1.3 El sistema autentifica al usuario. 1.4 El sistema crea el ticket de seguridad. 1.5 El sistema devuelve el ticket de seguridad.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	1.1 El sistema muestra mensaje de error ¨La aplicación no posee los permisos necesarios para acceder al sistema¨. 1.2 El sistema muestra mensaje ¨Datos incorrectos¨.
Poscondiciones	El usuario se encuentra autenticado.

Tabla 6. Autenticar usuario.

CU 2	**Validar ticket**
Propósito:	Validar que una aplicación tenga los permisos necesarios para interactuar con el sistema.
Actor:	Aplicación
Resumen:	El caso de uso comienza cuando la aplicación le pide al sistema validar el ticket, si existe devuelve la respuesta de que la autenticación del usuario es válida.
Precondiciones:	La aplicación debe tener permisos.
Referencias:	RF2
CU asociados:	CU_3
Prioridad:	Crítico
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. Una aplicación le pide al sistema que valide un ticket de seguridad.	1.1 Se ejecuta el CU asociado. 1.2 El sistema verifica que la sesión de la aplicación se encuentre abierta. 1.3 El sistema valida que el ticket exista. 1.4 El sistema devuelve el la respuesta de que la autenticación es válida.

Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	1.1 El sistema muestra mensaje de error ¨La aplicación no posee los permisos necesarios para acceder al sistema¨. 1.2 El sistema muestra mensaje de error si la sesión no se encuentra abierta. 1.3 El sistema muestra mensaje de error si el ticket no existe.
Poscondiciones	El sistema valida el ticket de seguridad y concede los permisos necesarios.

Tabla 7. Validar ticket.

CU 3	**Verificar permisos**
Propósito:	Que la aplicación posea los permisos necesarios para interactuar con el sistema.
Actor:	Aplicación
Resumen:	El caso de uso comienza cuando accede una aplicación al sistema y el mismo verifica los permisos de la aplicación.
Precondiciones:	La aplicación debe tener permisos.
Referencias:	RF3
CU asociados:	
Prioridad:	Crítico
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. La aplicación desea acceder al sistema.	1.1 El sistema verifica la lista de aplicaciones registradas. 1.2 El sistema concede los permisos para su interacción con la aplicación.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**

	1.1 El sistema muestra mensaje de error ¨La aplicación no posee los permisos necesarios para acceder al sistema¨.
Poscondiciones	El sistema concede los permisos.

Tabla 8. Verificar permisos.

CU 4	**Cerrar sesión**
Propósito:	Una aplicación le solicite al sistema cerrar la sesión.
Actor:	Aplicación
Resumen:	El caso de uso comienza cuando la aplicación le pide al sistema cerrar la sesión.
Precondiciones:	La aplicación debe tener permisos.
Referencias:	RF4
CU asociados:	CU_3
Prioridad:	Secundario
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. Una aplicación desea cerrar la sesión enviando el ticket de seguridad al sistema.	1.1 Se ejecuta el CU asociado. 1.2 El sistema verifica que el ticket de seguridad sea válido. 1.5 El sistema cierra la sesión.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	1.1 El sistema muestra mensaje de error ¨La aplicación no posee los permisos necesarios para acceder al sistema¨. 1.2 El sistema muestra mensaje de error ¨ Ticket invalido¨.
Poscondiciones	El sistema cierra la sesión.

Tabla 9. Cerrar sesión.

2.7 Conclusiones parciales.

En el presente capítulo se presentaron los conceptos del sistema a desarrollar mediante el modelo de dominio, se obtuvo un listado con los requisitos funcionales y no funcionales del software representados mediante un diagrama de casos de uso. Se describieron las acciones de los actores del sistema mediante los casos de uso reflejando el comportamiento del sistema a desarrollar y también se definió la arquitectura a utilizar.

CAPÍTULO 3: ANÁLISIS Y DISEÑO DEL SISTEMA.

3.1 Introducción.

En el capítulo se muestra el modelo de análisis y diseño del sistema a desarrollar mostrando una visión más específica de los requisitos describiendo cómo se va a implementar el software.

3.2 Modelo de análisis.

El modelo de análisis proporciona una representación de información, función y comportamiento del sistema, funcionando como un puente entre la descripción del sistema y el modelo de diseño y ofrece al desarrollador y al cliente los medios para evaluar la calidad una vez construido el software. (44)

Posee además tres clases de análisis centradas en los requisitos funcionales representando conceptos y relaciones de dominio:

Clase entidad: donde ocurren las acciones del propietario.

Clase interfaz: modela las interacciones entre el sistema y sus autores.

Clase control: coordinan el trabajo delos casos de uso.

Debe cumplir los siguientes tres objetivos primarios:

1. Describir los que requiere el cliente.
2. Establecer una base para la creación de un diseño de software.
3. Definir un conjunto de requisitos que puedan validarse una vez construido el software. (45)

3.2.1 Diagrama de clases de análisis.

- **Aplicación:** actor responsable de iniciar los casos de uso que contienen las acciones (autenticar usuario, validar ticket, verificar permisos y cerrar sesión).
- **CI_SSO:** es la clase interfaz que actúa como controladora la cual verifica la existencia de los usuarios en el directorio de usuarios del sistema (LDAP).
- **CE_Aplicación:** clase entidad donde se guarda la información de las aplicaciones alas que accede el usuario.
- **CE_Usuario:** clase entidad donde se guarda la información del usuario autenticado en el sistema.

- **CE_Sesión:** clase entidad donde se guarda la información de las sesiones creadas para cada usuario.

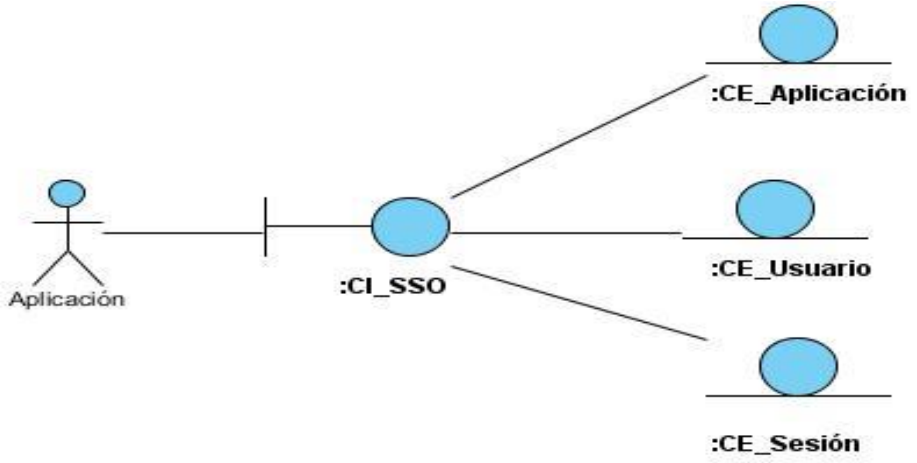

Ilustración 4. Diagrama de clases de análisis: Autenticar usuario.

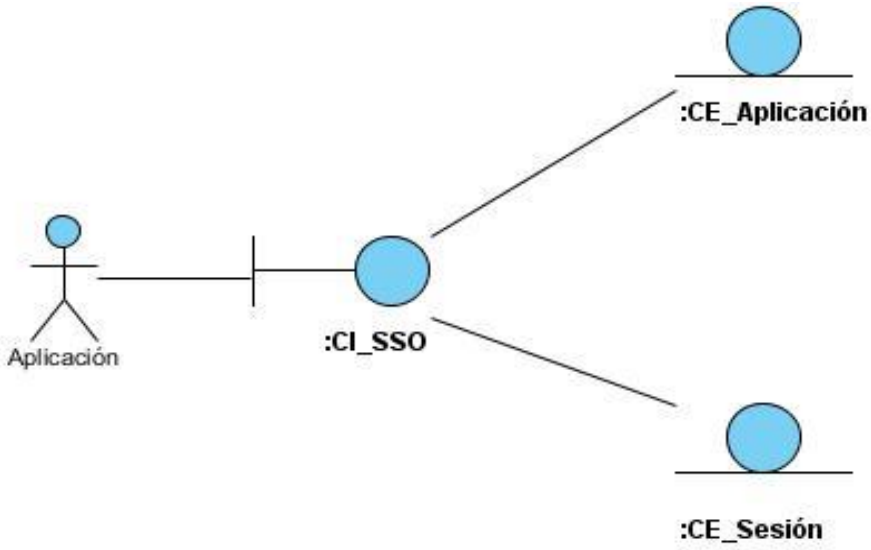

Ilustración 5. Diagrama de clases de análisis: Validar ticket.

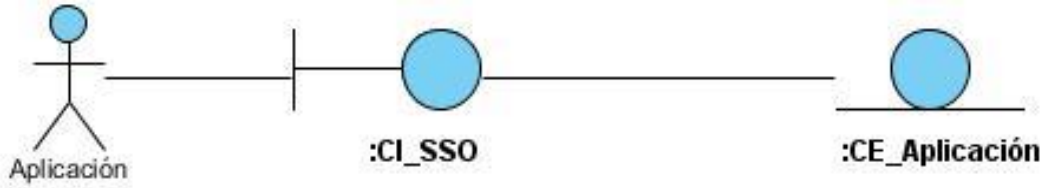

Ilustración 6. Diagrama de clases de análisis: Verificar permisos.

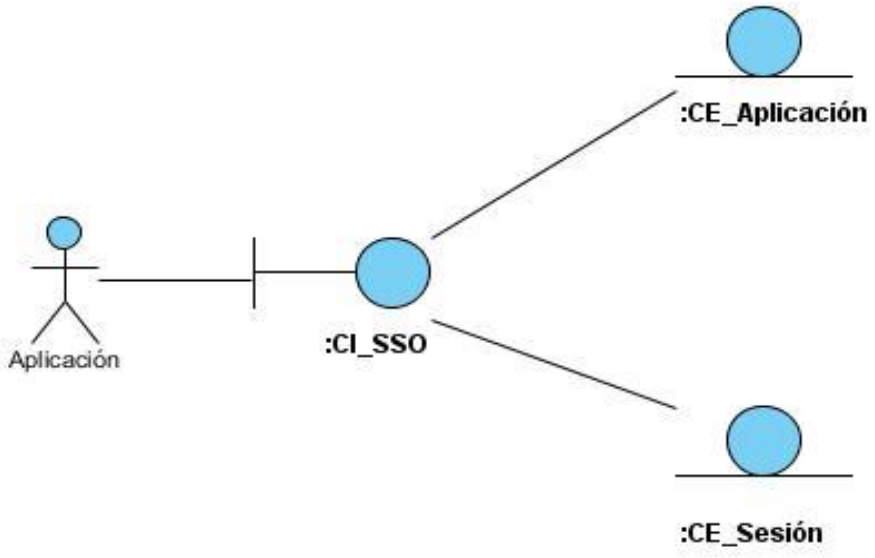

Ilustración 7. Diagrama de clases de análisis: Cerrar sesión.

3.2.2 Diagrama de colaboración.

El diagrama de colaboración es un tipo de diagrama de interacción cuyo objetivo es describir el comportamiento dinámico del sistema de información mostrando cómo interactúan los objetos entre sí, es decir, con qué otros objetos tiene vínculos o intercambia mensajes un determinado objeto. En estos diagramas la comunicación entre objetos se denomina vínculo o enlace (*link*) y estará particularizada mediante los mensajes que intercambian. (46)

Los diagramas de colaboración no son de fácil interpretación por lo que luego de cada diagrama se realizará una descripción del mismo para mayor entendimiento.

- **Flujo de sucesos-análisis: Autenticar usuario.**

Para autenticar un usuario la aplicación envía los datos del mismo (1), la clase controladora (CI_SSO) verifica los permisos de la aplicación (2), es decir, si la misma se encuentra vinculada con la aplicación CAS, después verifica los datos (3) entrados al sistema por el usuario mediante la comprobación de los mismos con el LDAP. El sistema autentica al usuario (4) si son correctos sus datos, luego genera un ticket (5) para que la aplicación pueda identificarse en el servidor del sistema y devuelve este a la aplicación (6).

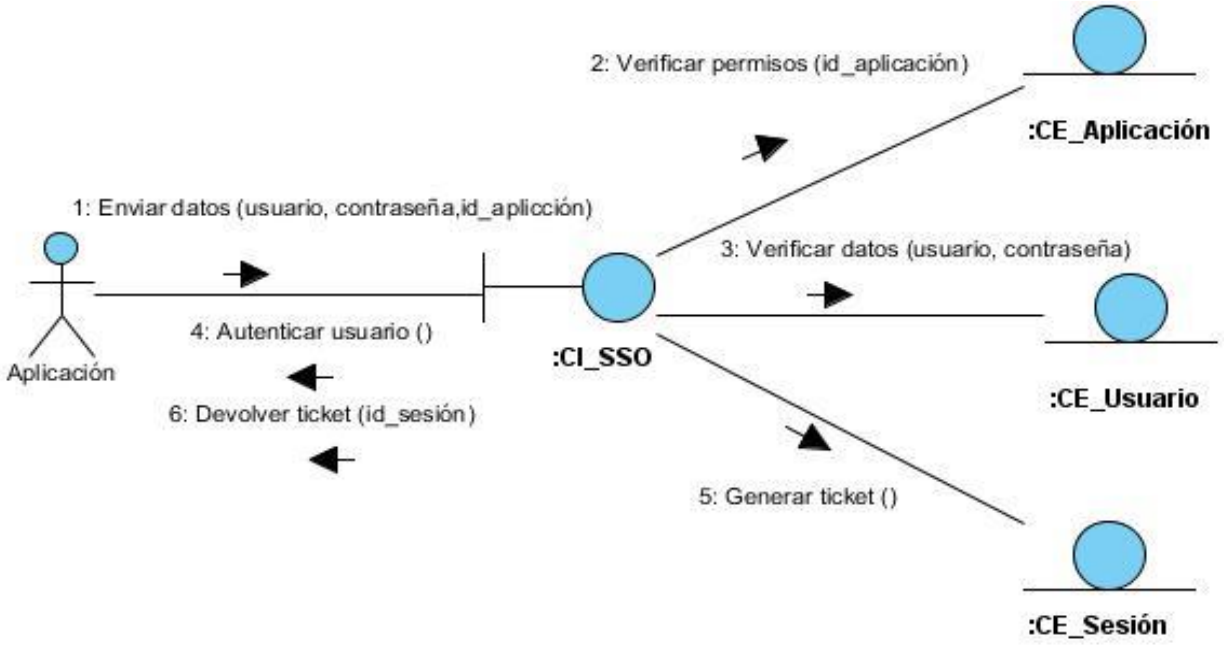

Ilustración 8. Diagrama de clases de colaboración: Autenticar usuario.

- **Flujo de sucesos-análisis: Validar ticket.**

La aplicación le solicita validar el ticket (1) a la clase controladora (CI_SSO) la cual verifica los permisos de la aplicación (2), es decir, si la misma se encuentra vinculada con la aplicación CAS, después valida el ticket (3) para que la aplicación pueda identificarse en el servidor del sistema y le envía una respuesta a la aplicación (4).

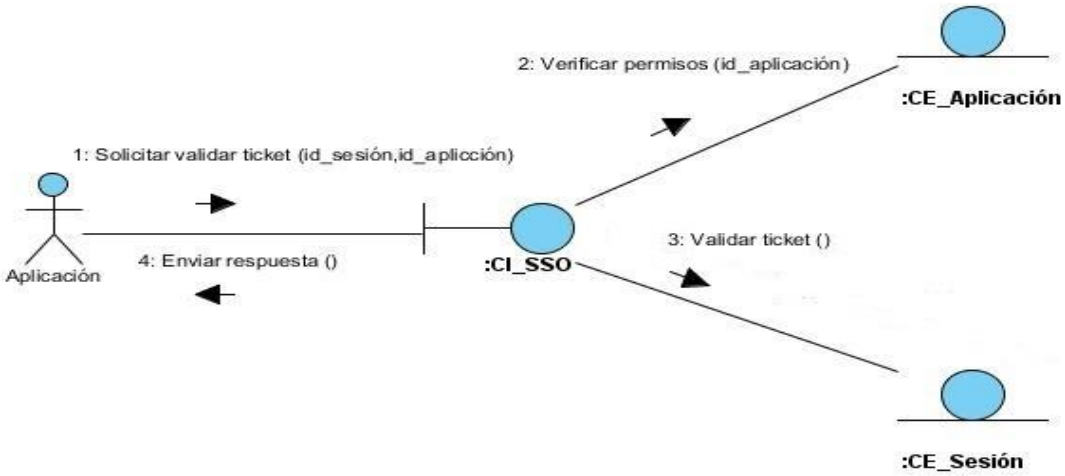

Ilustración 9. Diagrama de clases de colaboración: Validar ticket.

❖ Flujo de sucesos-análisis: Verificar permisos.

La aplicación le solicita acceder al sistema (1) a la clase controladora (CI_SSO) la cual verifica el identificador de la aplicación (2), es decir, si el identificador de la misma se encuentra registrado en el sistema y luego le brinda los permisos a la aplicación (3).

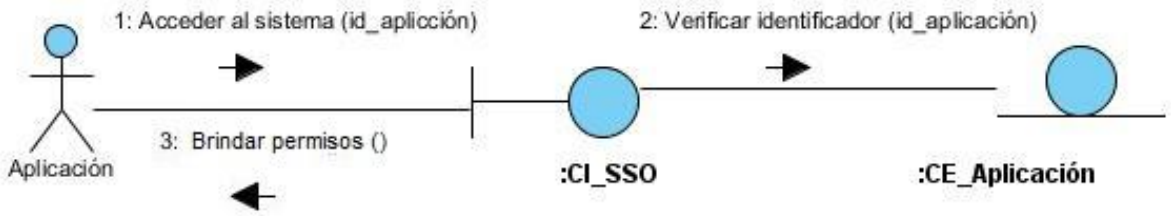

Ilustración 10. Diagrama de clases de análisis: Verificar permisos.

❖ Flujo de sucesos-análisis: Cerrar sesión.

La aplicación solicita cerrar sesión (1) a la clase controladora (CI_SSO) la cual verifica los permisos de la aplicación (2), es decir, si la misma se encuentra vinculada con la aplicación CAS, después valida el ticket (3) para que la aplicación pueda identificarse en el servidor del sistema y luego elimina el ticket (4) para que se elimina la sesión del servidor.

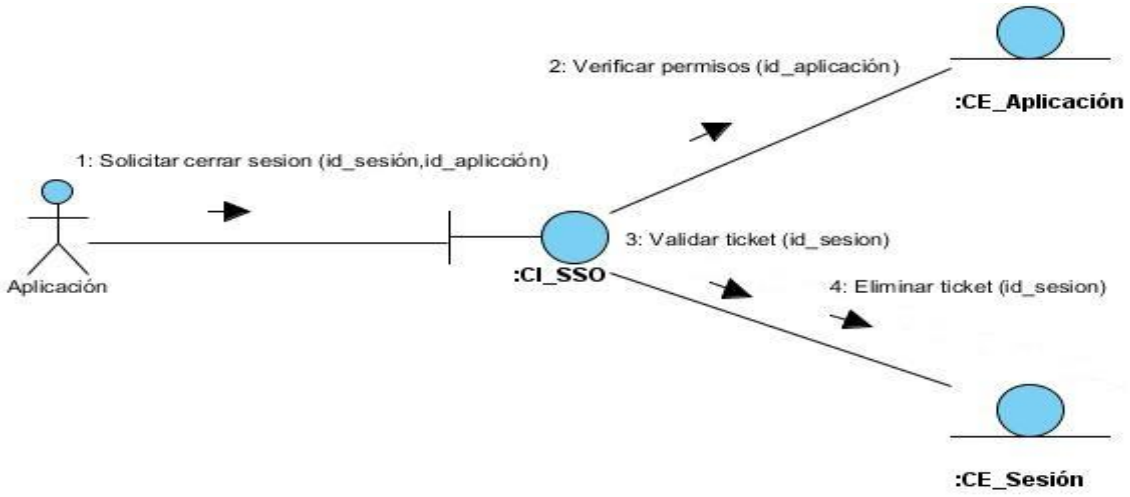

Ilustración 11. Diagrama de clases de análisis: Cerrar sesión.

3.3 Diseño.

El diseño del software es un proceso iterativo mediante el cual los requisitos se traducen en un ¨plano¨ abstracto el cual puede rastrearse de manera directa hasta conseguir el objetivo específico del sistema y requisitos más detallados del comportamiento, funcionales y de datos. (47)

3.4 Modelo de diseño.

El modelo de diseño crea una representación o modelo de software, que a diferencia del modelo de análisis (que se enfoca en la descripción de los datos, las funciones y el comportamiento requerido), proporciona detalles de las estructuras de los datos, las arquitecturas, las interfaces y los componentes del software que son necesarios para implementar el sistema. (48)

3.4.1 Diagrama de clases de diseño.

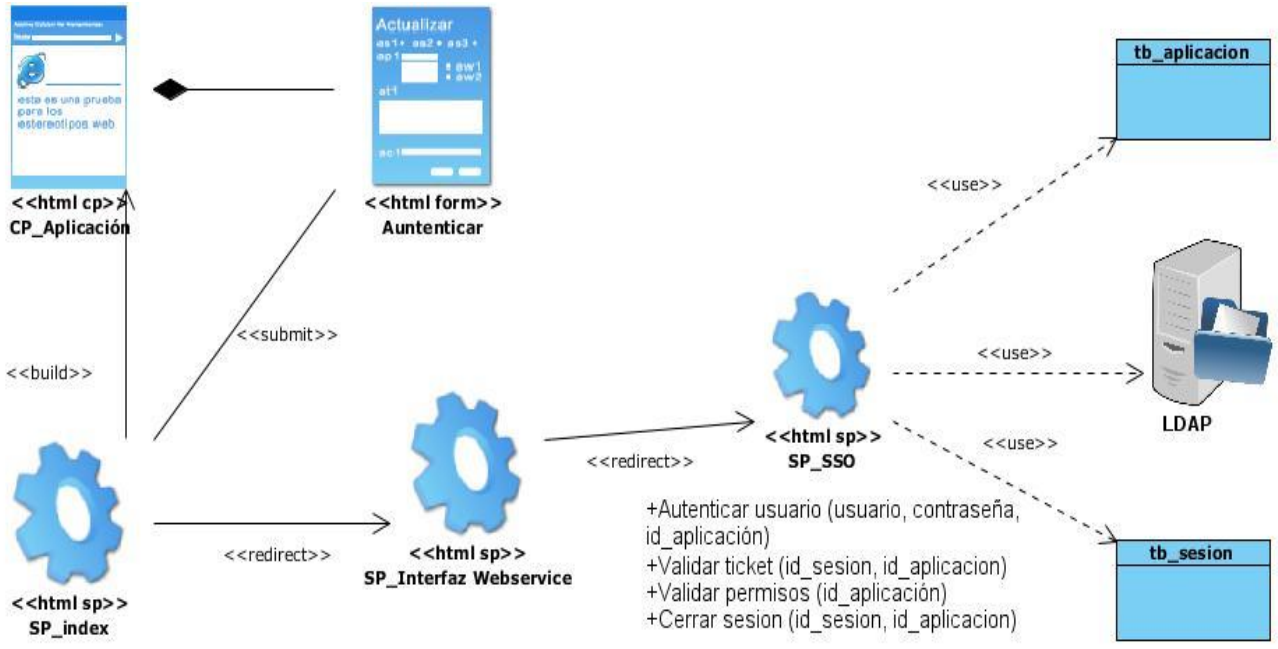

Ilustración 12. Diagrama de clases de diseño: Autenticar usuario.

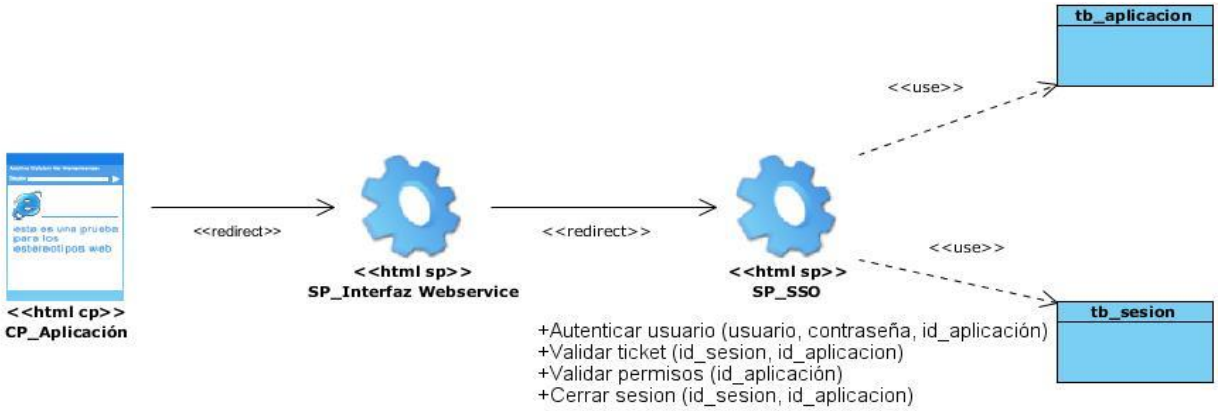

Ilustración 13. Diagrama de clases de diseño: Validar ticket.

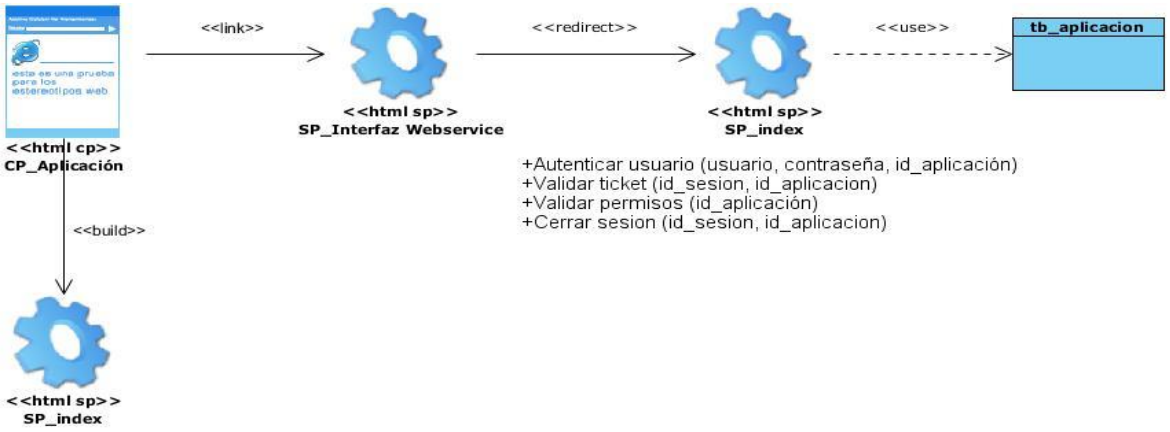

Ilustración 14. Diagrama de clases de diseño: Verificar permisos.

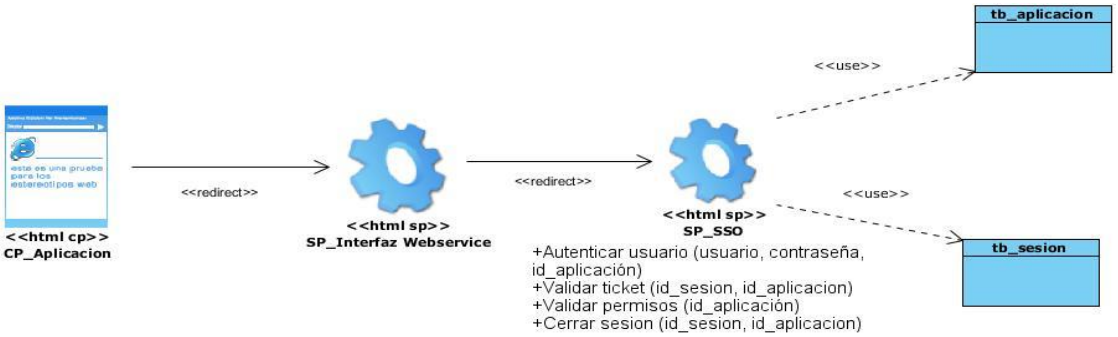

Ilustración 15. Diagrama de clases de diseño: Cerrar sesión.

3.4.2 Diagrama de secuencia de diseño.

Los diagramas de secuencia muestran la forma en que los objetos se comunican entre sí al transcurrir el tiempo.

El diagrama muestra:

- Los objetos participando de la interacción
- La secuencia de mensajes intercambiados
- Objetos con su línea de vida.
- Mensajes intercambiados entre objetos de una secuencia ordenada.
- Línea de vida activa. (49)

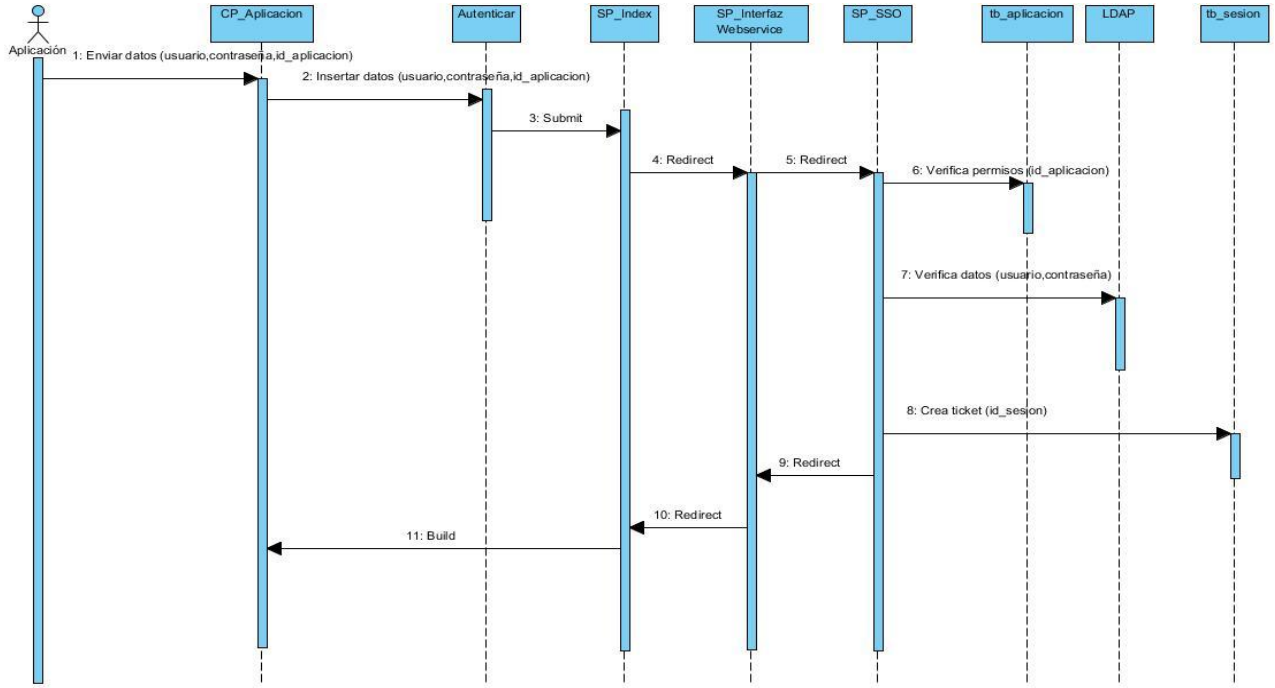

Ilustración 16. Diagrama de secuencia: Autenticar usuario.

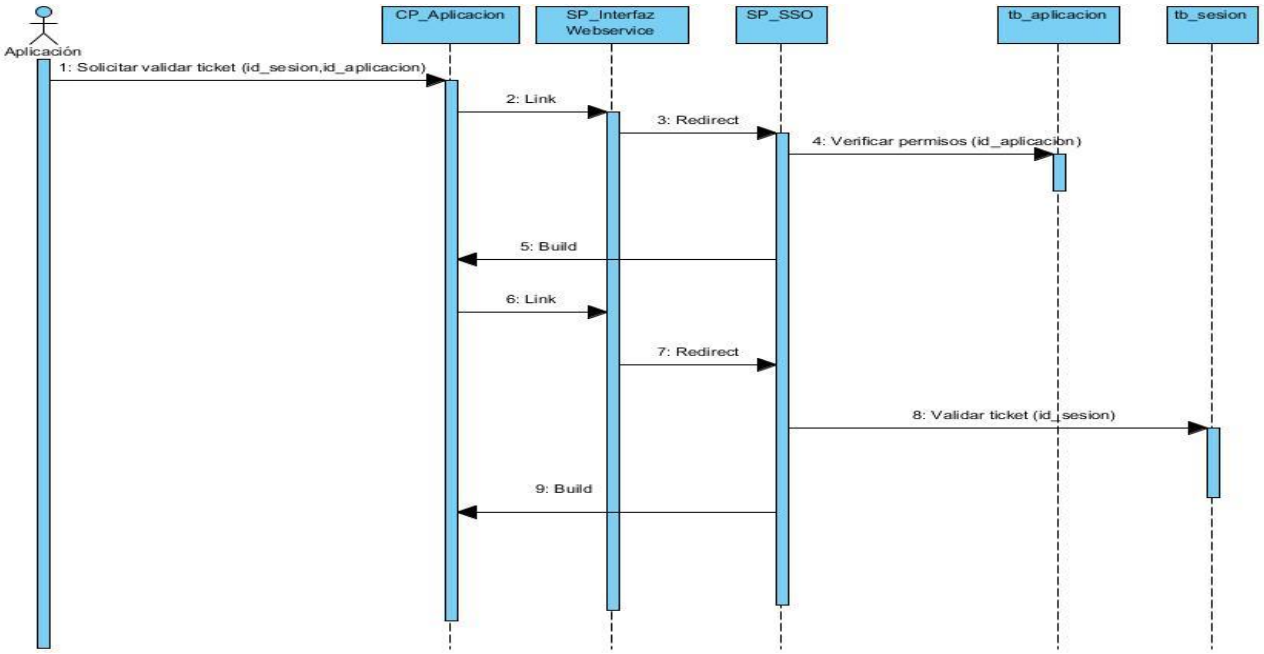

Ilustración 17. Diagrama de secuencia: Validar ticket.

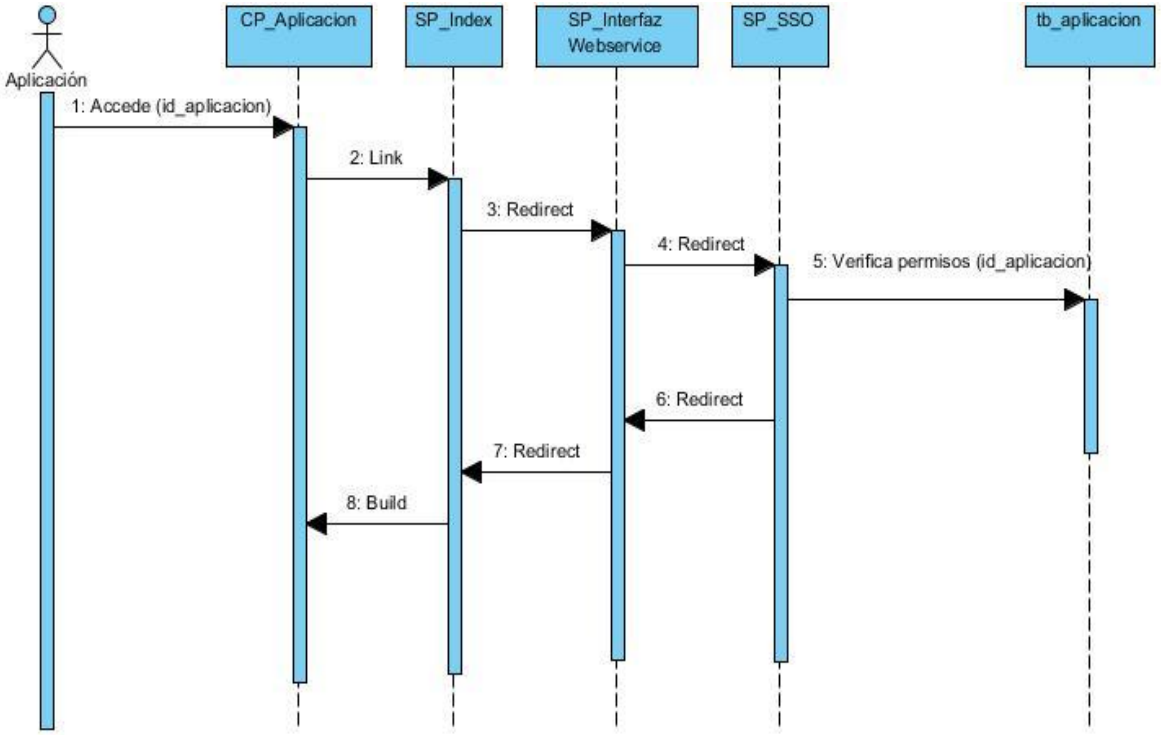

Ilustración 18. Diagrama de secuencia: Verificar permisos.

3.5 Patrones de diseño.

Los patrones de diseño se utilizan una vez se ha desarrollado el modelo de análisis, aplicándose a un elemento específico del diseño como un agregado de componentes para resolver algún problema de diseño, relaciones entre los componentes o los mecanismos para efectuar la comunicación de componente a componente. (50)

3.5.1 Patrón de diseño GRASP.

Los patrones de diseño GRASP (*General Responsibility Assignment Software Patterns*) son patrones generales de software para asignación de responsabilidades. Aunque se considera que más que patrones propiamente dichos, son una serie de "buenas prácticas" de aplicación recomendable en el diseño de software. (51)

- **Patrón experto:** indica que la responsabilidad de la creación de un objeto o la implementación de un método, debe recaer sobre la clase que conoce toda la información necesaria para crearlo. (52)

Este patrón se pone de manifiesto en la clase cas-servlet.xml.

- **Patrón controlador:** la mayor parte de los sistemas reciben eventos de entrada externa, los cuales generalmente incluyen una interfaz gráfica para el usuario operado por una persona. Este patrón ofrece una guía para tomar decisiones apropiadas que generalmente se aceptan. (52)

Este patrón se pone de manifiesto en la clase web.xml.

- **Alta cohesión:** caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme. (52)

Un ejemplo de este patrón se evidencia en la clase loginServlet.java y login.php.

- **Bajo acoplamiento**: significa que una clase no depende de muchas clases. (52)

Un ejemplo de este patrón se evidencia en la clase mbeans.xml y clearpass-configuration.xml.

3.6 Diseño de la base de datos.

Un diseño correcto es esencial para lograr los objetivos fijados para la base de datos, siendo necesario decidir qué información se necesita, dividirla en tablas y columnas y relacionarlas entre sí. Una base de datos correctamente diseñada permite obtener acceso a información exacta y actualizada. (53)

3.6.1 Diagrama de clases persistentes.

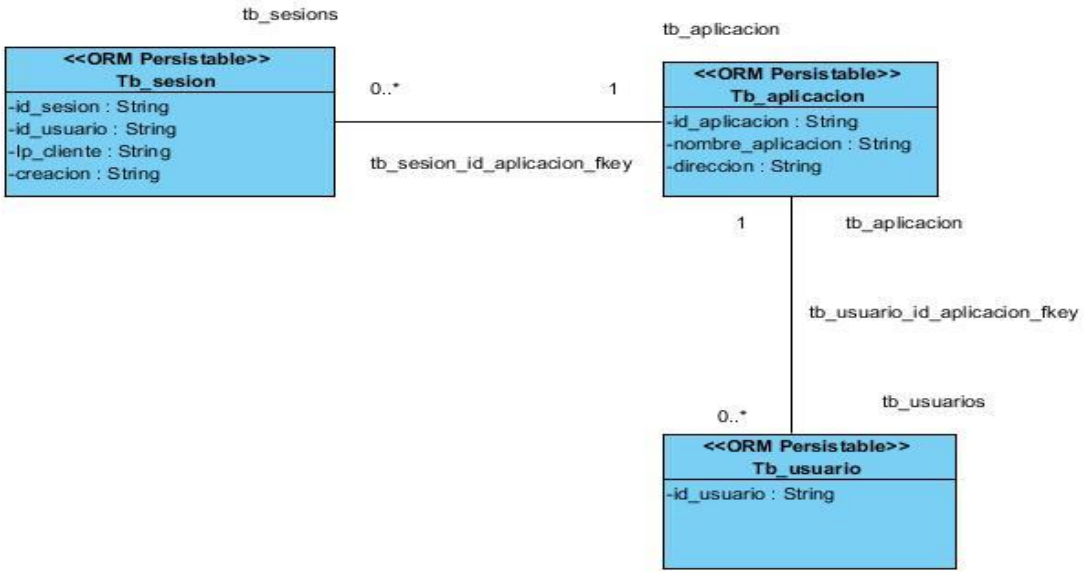

Ilustración 19. Diagrama de clases persistentes.

3.6.2 Modelo de datos.

Un modelo de datos es un conjunto de herramientas conceptuales para describir los datos, las relaciones entre ellos, su semántica y sus limitantes. (53)

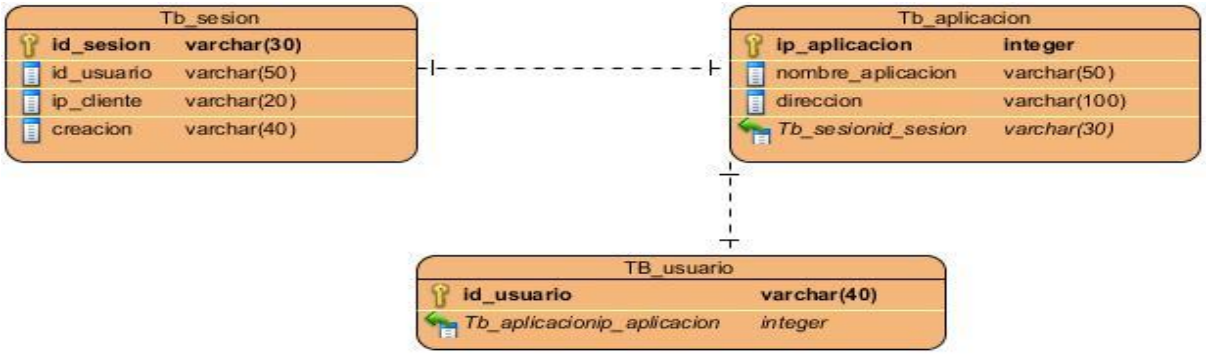

Ilustración 20. Diagrama de modelo de datos.

3.7 Conclusiones parciales.

Al concluir este capítulo se tiene concebido detalladamente el diseño completo del sistema mediante el modelado de los diagramas de clases de análisis, diagramas de colaboración y diagramas de clases de diseño correspondiente a cada caso de uso. Se mostró además el modelo de datos sobre el que se va a implementar la aplicación que se ha diseñado, con lo que se puede pasar a la etapa de implementación del proyecto.

CAPÍTULO 4: IMPLEMENTACIÓN Y PRUEBA DEL SISTEMA.

4.1 Introducción.

El presente capítulo contiene diagramas que reflejan la distribución de los diferentes componentes del Sistema de Autenticación Centralizada representando de forma visual las partes físicas que dispondrá el sistema, también se realizarán un conjunto de pruebas que validen las funcionalidades del mismo.

4.2 Implementación.

El modelo de implementación es comprendido por un conjunto de componentes y subsistemas que constituyen la composición física de la implementación del sistema. Entre los componentes podemos encontrar datos, archivos, ejecutables, código fuente y los directorios. Fundamentalmente, se describe la relación que existe desde los paquetes y clases del modelo de diseño a subsistemas y componentes físicos. (54)

4.2.1 Diagrama de despliegue.

Los diagramas de despliegue muestran las relaciones físicas de los distintos nodos[13] que componen un sistema y el reparto de los componentes sobre dichos nodos. La vista de despliegue representa la disposición de las instancias de componentes de ejecución en instancias de nodos conectados por enlaces de comunicación. (55)

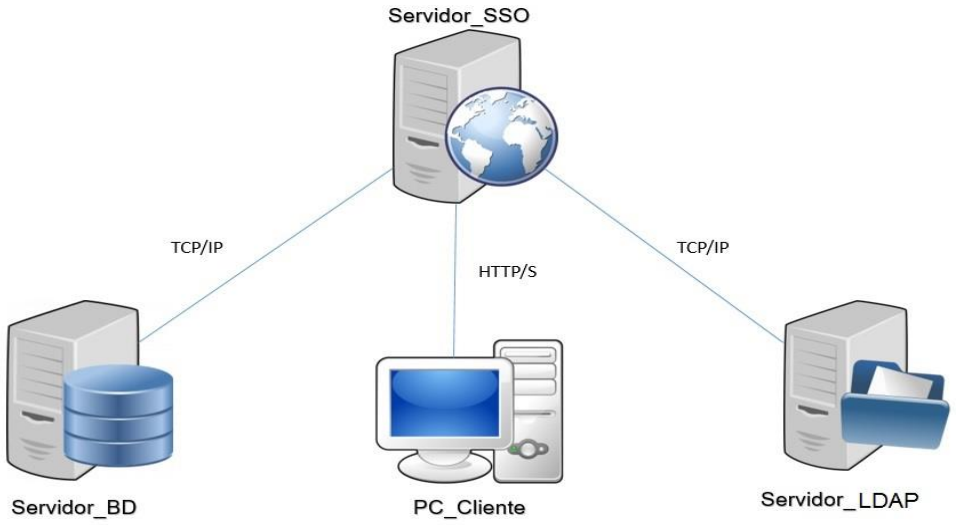

Ilustración 21. Diagrama de despliegue.

[13] nodos: recurso de ejecución tal como un computador, un dispositivo o memoria.

- **Servidor_SSO:** contiene la aplicación a la cual se accederá desde las estaciones de trabajo.
- **Servidor_BD:** contiene la base de datos donde se encuentran registrados los datos pertinentes a los usuarios que ingresan al sistema.
- **PC_Cliente:** se conecta al servidor web a través del protocolo HTTPS para visualizar la aplicación.
- **Servidor_LDAP:** servidor que utiliza la universidad para gestionar a los usuarios de manera centralizada utilizando el protocolo de acceso a directorios ligeros.

4.2.2 Diagrama de componentes.

Los diagramas de componente describen los elementos físicos del sistema y sus relaciones. Muestran las opciones de realización incluyendo código de fuente, binario y ejecutable, representando todos los tipos de elementos de software que entran en la fabricación de una aplicación informática. (56)

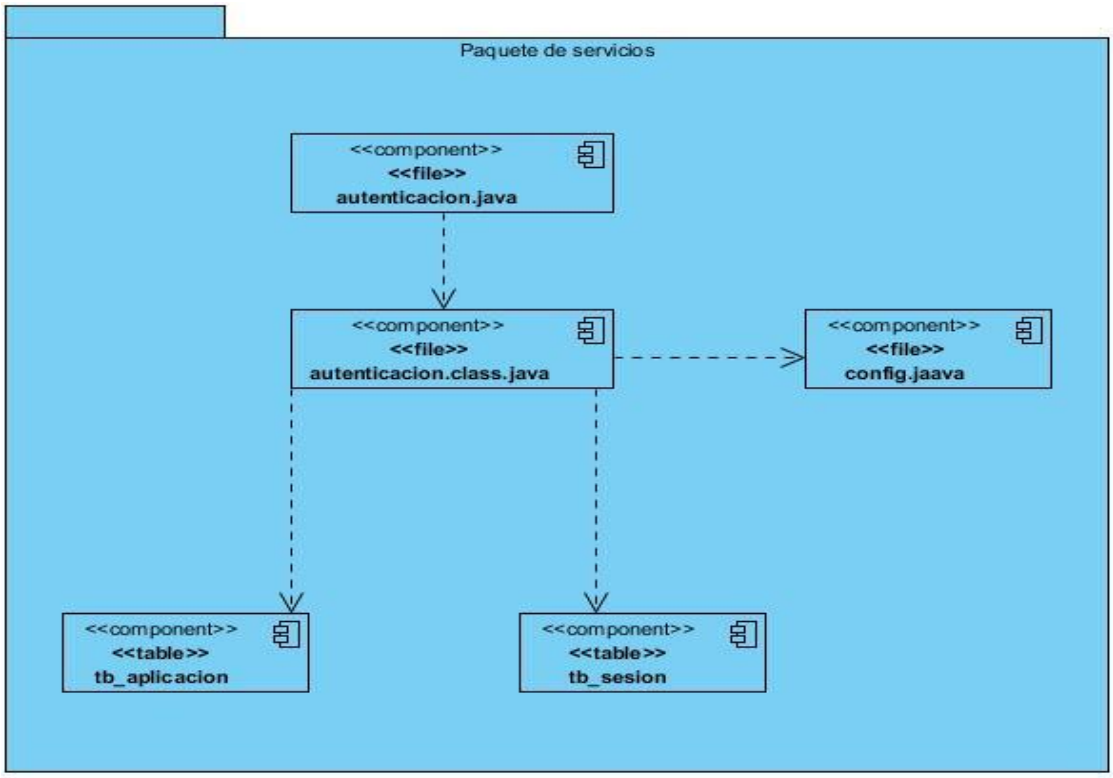

Ilustración 22. Diagrama de componentes: Paquete de servicios Java.

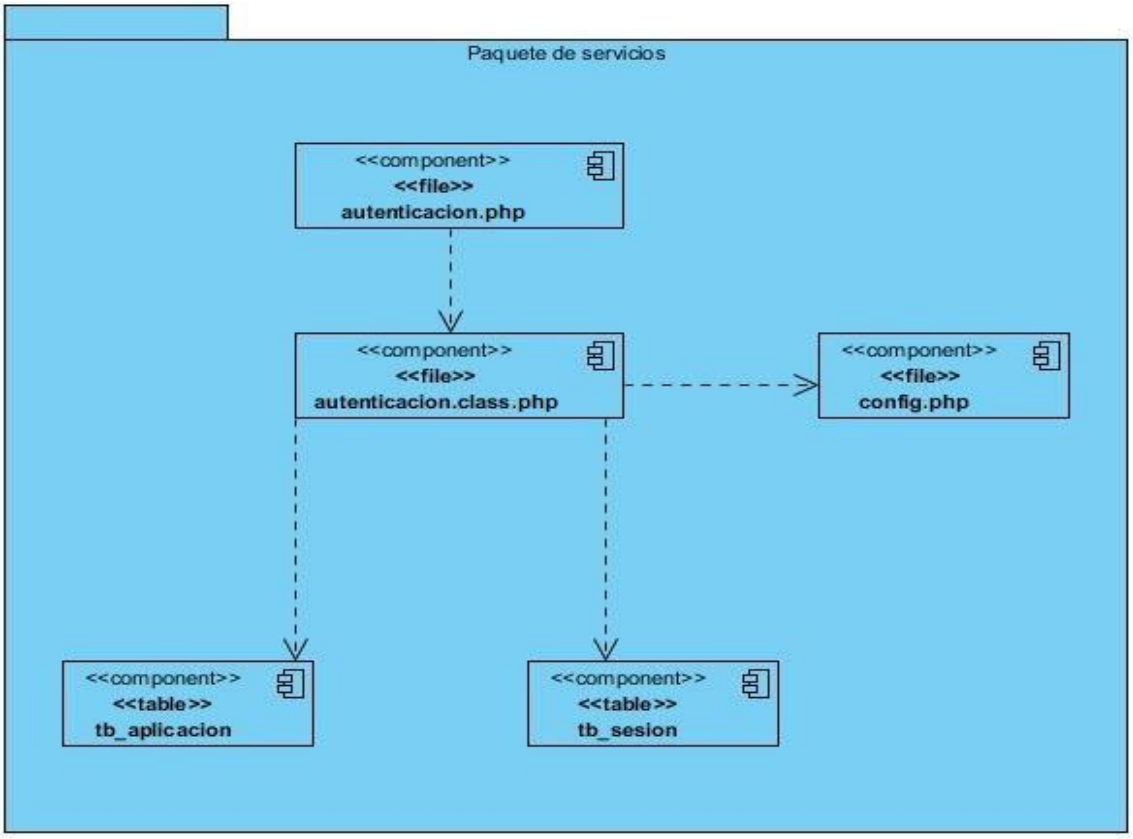

Ilustración 23. Diagrama de componentes: Paquete de servicios PHP.

4.3 Pruebas.

Las pruebas son un conjunto de actividades que se plantean con anticipación y se realizan de manera sistemática. Por tanto, se deben definir un conjunto de pasos en que se puedan incluir técnicas y métodos específicos del diseño de casos de prueba. (57)

4.3.1 Estrategia de prueba.

- **Prueba de unidad:**

  Se concentra en el esfuerzo de verificación de la unidad más pequeña del diseño del software (el componente o módulo del software).

- **Prueba de integración:**

  Técnica sistemática para construir la arquitectura del software mientras, al mismo tiempo, se aplican las pruebas para descubrir errores asociados a la interfaz.

- **Prueba de regresión:**

  Ejecuta nuevamente el mismo subconjunto de pruebas que ya se ha aplicado para asegurarse de que los cambios no han propagado efectos colaterales indeseables.

- **Prueba del sistema:**

  Verifica que se hayan integrado adecuadamente todos los elementos del sistema y que realicen las funciones apropiadas.

- **Prueba de aceptación:**

  Valida el cumplimiento de los requisitos funcionales del software. (58)

4.3.2 Técnicas de prueba.

Existen dos maneras de probar cualquier producto construido.

- ✓ Si se conoce la función específica para la que se diseñó, se aplican pruebas que demuestren que cada función es operacional, mientras se buscan los errores de cada función.
- ✓ Si se conoce el funcionamiento interno del producto, se aplican pruebas para asegurarse de que las operaciones internas se realizan de acuerdo con las especificaciones, y que se han probado todos los componentes internos de manera adecuada.

Al primer enfoque de prueba se le denomina prueba de caja negra que son las que se le aplican a la interfaz del software, examinando algún tipo de aspecto funcional del sistema. Mientras que el segundo es una prueba de caja blanca basándose en un examen cercano al detalle procedimental, probando las rutas lógicas del sistema y la colaboración entre componentes. (59)

4.3.3 Pruebas de aceptación.

El uso de cualquier producto de software tiene que estar justificado por las ventajas que ofrece. Sin embargo, antes de empezar a usarlo es muy difícil determinar si sus ventajas realmente justifican su uso. El mejor instrumento para esta determinación es la llamada “prueba de aceptación”. En esta prueba se evalúa el grado de calidad del software con relación a todos los aspectos relevantes para que el uso del producto se justifique. (60)

Estas pruebas las realiza el cliente. Son básicamente pruebas funcionales, sobre el sistema completo, y buscan una cobertura de la especificación de requisitos y del manual del usuario. (61)

Para las pruebas del software se decidió aplicar las pruebas de aceptación bajo la siguiente estructura:

Caso de prueba de Aceptación	
Número: N	**Historia de Usuario:** N
Nombre: Nombre	
Descripción: Descripción	
Condiciones de ejecución: Condiciones	
Entradas/Pasos de ejecución: Pasos	
Resultado esperado: Resultado esperado.	
Evaluación de la prueba: Evaluación.	

Número: Representa el número del caso de prueba. Este debe ser consecutivo.

Historia de usuario: Número de la historia de usuario a la que responde el caso de prueba.

Nombre: Nombre de la historia de usuario a la que responde el caso de prueba.

Descripción: Descripción de la historia de usuario a la que responde el caso de prueba.

Condiciones de ejecución: Condiciones previas que deben cumplirse para la realización del caso de prueba.

Entradas/Pasos de ejecución: Secuencia de pasos o entradas que se realizan para validar la funcionalidad que se prueba.

Resultados esperados: Describe los objetivos concretos de la funcionalidad.

Evaluación de la prueba: Evaluación obtenida luego de realizada la prueba. Puede ser satisfactoria o no satisfactoria.

A continuación, se muestran algunos casos de pruebas relacionados con las historias de usuarios.

Caso de prueba de Aceptación	
Número: HU1-P1	**Historia de Usuario:** #1
Nombre: Autenticar usuario.	
Descripción: Se ingresan los datos de acceso a la aplicación.	
Condiciones de ejecución: Los datos ingresados del usuario deben ser correctos.	
Entradas/Pasos de ejecución: 1. Ingresar usuario. 2. Ingresar contraseña. 3. Presionar en el botón “Aceptar”.	
Resultado esperado: La aplicación muestra el mensaje: ¨Inicio de sesión exitoso¨.	
Evaluación de la prueba: Satisfactoria.	

Tabla 10. Caso de prueba de aceptación: HU1-P1.

Caso de prueba de Aceptación	
Número: HU2-P1	**Historia de Usuario:** #2
Nombre: Autenticar usuario.	
Descripción: Se ingresan los datos de acceso a la aplicación. (Datos erróneos)	
Condiciones de ejecución: Los datos ingresados del usuario deben ser correctos.	
Entradas/Pasos de ejecución: 1. Ingresar usuario. 2. Ingresar contraseña. 3. Presionar en el botón “Aceptar”.	
Resultado esperado: La aplicación muestra el mensaje: ¨Credenciales inválidos¨.	
Evaluación de la prueba: Satisfactoria.	

Tabla 11. Caso de prueba de aceptación: HU2-P2.

Caso de prueba de Aceptación	
Número: HU3-P1	**Historia de Usuario:** #3
Nombre: Cerrar sesión.	
Descripción: Se desea salir de la aplicación.	
Condiciones de ejecución: El usuario debe de estar autenticado.	
Entradas/Pasos de ejecución: 1. Presionar en el botón “Cerrar sesión”.	
Resultado esperado: La aplicación muestra el mensaje: ¨Cierre de sesión exitoso¨.	
Evaluación de la prueba: Satisfactoria.	

Tabla 12. Caso de prueba de aceptación: HU3-P1.

4.3.4 Pruebas de rendimiento.

Las pruebas de rendimiento son aquellas que son realizadas para determinar qué tan rápido un sistema realiza una tarea bajo ciertas condiciones pre-planificadas de trabajo. Estas también son utilizadas para validar y verificar diferentes aspectos de la calidad de software, como, por ejemplo, escalabilidad, fiabilidad y el buen uso de los recursos. (62)

Resultados de las pruebas de rendimiento al sistema:

Etiqueta	# Muestras	Media	Mediana	Linea de 90%	Mín	Máx	% Error	Rendimiento	Kb/sec
/AP1/jsps/welco...	50	2450	2423	2540	2322	2596	0,00%	18,6/sec	38,7
/sso/js/jquery-1....	50	2023	2014	2044	2005	2159	0,00%	18,9/sec	39,3
/sso/js/bootstra...	50	2012	2009	2028	2005	2030	0,00%	16,6/sec	34,5
/sso/css/login.c...	50	2012	2011	2019	2004	2042	0,00%	15,4/sec	32,0
/sso/css/bootstr...	50	2011	2008	2022	2005	2037	0,00%	14,5/sec	30,1
/sso/js/classie.j...	50	2010	2005	2013	2003	2164	0,00%	13,8/sec	28,7
/favicon.ico	100	2010	2007	2020	2004	2043	0,00%	16,7/sec	34,8
/sso/login;jsess...	50	2009	2007	2015	2004	2042	0,00%	12,3/sec	25,5
/sso/login?servi...	50	2007	2007	2012	2004	2019	0,00%	12,1/sec	25,2
/AP1/jquery-2.0....	200	2007	2006	2011	2003	2029	0,00%	10,6/sec	22,0
/AP1/usuario.jsp	50	2007	2006	2011	2003	2031	0,00%	12,1/sec	25,1
/AP1/js/main.js	50	2006	2006	2009	2003	2010	0,00%	11,9/sec	24,6
/AP1/js/main2.js	50	2005	2005	2008	2003	2021	0,00%	11,6/sec	24,1
/AP1/UsuarioSe...	100	2007	2006	2013	2003	2025	0,00%	11,5/sec	24,0
Total	950	2032	2007	2024	2003	2596	0,00%	22,4/sec	46,5

Ilustración 24. Informe agregado. Aplicación.

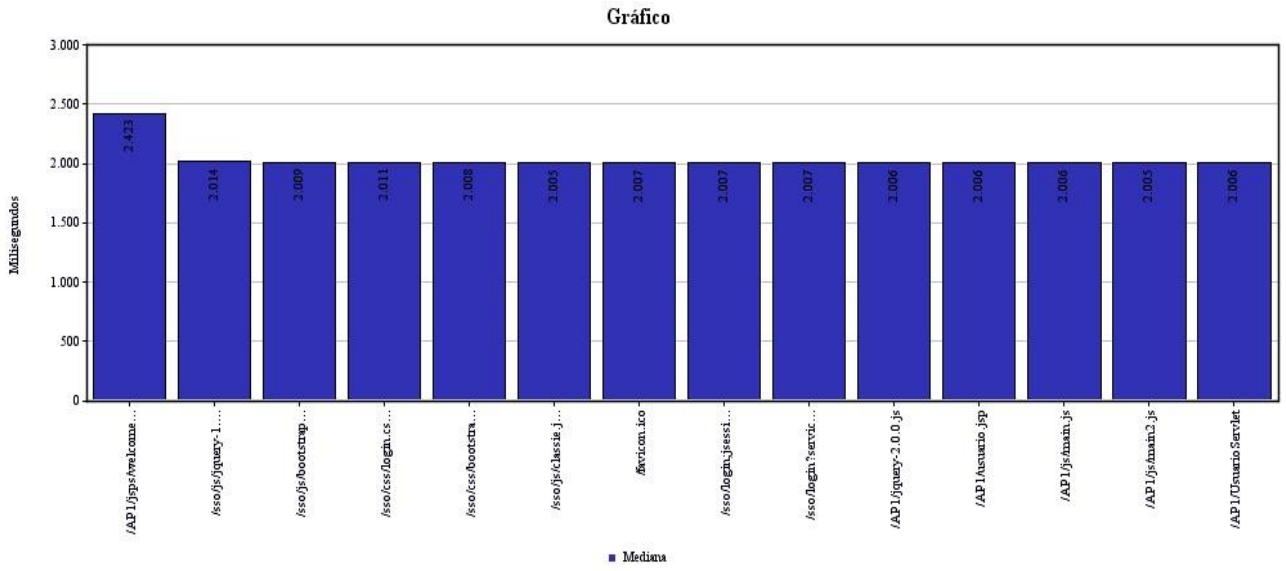

Ilustración 25. Aplicación.

Teniendo 50 usuarios interactuando en cada página llegado a la suma de 950 en total, el tiempo de respuesta máximo fue de 2s y medio, el tiempo mínimo fue de 2s con un rendimiento de 22.4 peticiones por segundo.

4.5 Conclusiones parciales.

En este capítulo quedó reflejado como se ha realizado la implementación de las principales funcionalidades que debe tener el sistema, mediante el modelado del diagrama de despliegue y el diagrama de componentes quedando representada la distribución del sistema en un plano físico y la estructura lógica del mismo. Además, se realizaron pruebas para comprobar el correcto funcionamiento del sistema validando el cumplimiento de los requisitos establecidos, garantizando así la calidad del software desarrollado.

CONCLUSIONES.

Con la realización de este trabajo y luego de los resultados obtenidos se puede llegar a las siguientes conclusiones:

- ✓ El estudio de los diferentes sistemas de autenticación única existentes, permitió obtener los conocimientos suficientes para lograr una mejor comprensión en la investigación.

Escogiendo el sistema de autenticación única de tipo web utilizando la herramienta CAS.

- ✓ Se diseñó un mecanismo de autenticación centralizada para el entorno de aplicaciones de la UCI, utilizando para ello las herramientas y lenguajes definidos.

Vinculando dos aplicaciones una en el lenguaje java y otra en php con el servicio de autenticación centralizada.

- ✓ Se validó la aplicación mediante pruebas de aceptación y rendimiento verificando así la calidad de la solución propuesta.

Arrojando como resultado que el sistema funciona rápidamente para satisfacción del usuario.

RECOMENDACIONES.

Durante la realización de la aplicación se cumplieron los objetivos propuestos, dándole solución a las dificultades existentes, pero es necesario dejar plasmadas algunas recomendaciones para el posterior despliegue del sistema:

- Continuar investigando sobre las diferentes librerías que posee un sistema de autenticación centralizada vinculándolo esta vez con una aplicación en el lenguaje .NET.
- Perfeccionar las funcionalidades del sistema, a partir de las variaciones que puedan surgir durante el tiempo de explotación en la universidad.

De forma general, dar continuidad y profundizar en los sistemas de autenticación centralizada.

Referencias bibliográficas:

1. Sotelo, Martin R. Mondragón. *Seguridad Informática. Capítulo 4: Autenticación.* http://www.ecured.cu/.

2. *Diccionario de la real academia de la lengua española.* 2016. http://dle.rae.es.

3. Systems, Aladdin Knowledge. *Gestión de contraseñas. Single Sign On.* 2006. http://www.aladdin.es/news/2006/etoken/gestion_contrasenyas.asp.

4. (Nacho), Juan Ignacio Martín. *Implantación de un SSO (Single Sign On).* openaccess.uoc.edu/webapps/o2/bistream/10609/28021/6/nacho_martinTFM0114memoria.pdf.

5. Hursti, Jani. *Ideal SSO.* Jani.Hursti@hut.fi.

6. Pérez, Sandra Menéndez Alonso y Ingrid Tobío. *Tesis Sistema de Gestión de Accesos.* 2008.

7. Baryolo, Oiner Gómez. *Modelo de Control de Acceso para Sistemas de Información en Entornos Multidominios .* 2012.

8. Grillo, Alexander Reyes. *Arquitectura, Análisis y Diseño del Sistema de Autenticación Única de RINDE mediante la creación de un SSO.* 2008.

9. Alvarado, Lázaro Antonio Marín Mártires y Luis Ramón Capriles. *Sistema de autenticación y autorización centralizado PDVSA.* 2008.

10. Trilla, Miquel. *Single Sign-On.* Facultad Informática de Barcelona : s.n., 2006. http://www.documents.mx/documents/facultat-dinformatica-de-barcelona-single-sign--on-miguel-trilla.html.

11. Jeimy J. Cano, Ivan M. Caballero. *CONSIDERACIONES PARA IMPLEMENTAR UNA ARQUITECTURA SINGLE SIGN-ON.* http://www.criptored.upm.es/guiateoria/gt_m142j.html.

12. Terrero, Eliurkis Díaz. *Single Sign On: Sistema de Autentificación Único.* 2006. http://www.deepinphp.com/2007/09/01/single-sign-on-sistema-de-autenticacion-unico/.

13. Valls, Juan C. Sánchez – Jordi. *Autenticación centralizada.* Córdoba : s.n., 18 de noviembre de 2010 . https://www.rediris.es/jt/jt2010/ponencias/jt2010-jt-serv_feder_2-2.pdf.

14. Corporation, Oracle. *OpenSSO.* http://www.oracle.com/technetwork/testcontent/opensso-091890.html.

15. Moreno, Manuel. *JOSSO ó CAS para certificados digitales.* http://www.javamexico.org/foros/java_standard_edition/josso_o_cas_para_certificados_digitales.

16. Vinuesa, García de. *adAS Single Sign-On.* http://www.adas-sso.com/es/sso/sso.php.

17. Roberth G. Figueroa, Camilo J. Solís y Armando A. Cabrera. *METODOLOGÍAS TRADICIONALES VS. METODOLOGÍAS ÁGILES.* s.l. : Universidad Técnica Particular de Loja, Escuela de Ciencias en Computación. http://ort-proyecto.googlecode.com.

18. Yuneikys Recio Miranda, Osbel Montero Pérez, Deiler Sevila Fernández. *Desarrollo de la Ampliación del Portal Web del CICPC.* Universidad de las Ciencias Informáticas. : s.n.

19. Sánchez, Jordi. *¿Qué es un 'framework'?* Mallorca : s.n., 2006. http://jordisan.net/blog/2006/que-es-un-framework/.

20. Rod, Johnson y. *Spring Framework Reference Documentation.* 2004. http://docs.spring.io/spring/docs/current/spring-framework-reference/html/single/.

21. González, Héctor Suárez. *Manual Hibernate.* 21.03.2003. static1.1.sqspcdn.com/static/f/ManualHibernate.pdf.

22. Chavez, Alan. *Si eres desarrollador web, debes utilizar Bootstrap y punto.* https://alanchavez.com/si-eres-desarrollador-web-debes-utilizar-bootstrap-y-punto/.

23. Santo Tomás : s.n., 12 de Enero de 2015. http://news.netcraft.com/archives/2015/02/24/february-2015-web-server-survey.html..

24. Terbush, Randy. *Apache Fundation.* 20 de marzo de 2015. Netcraft.com.

25. Rodriguez, Aime Rodriguez. *Que es wamp server.* http://es.slideshare.net/.

26. McClure, Carma. *The CASE Experience.* http://www.users.dsic.upv.es.

27. Esteban, Pozos Méndz y Pablo. *Herramienta Case Visual Paradigm | Herramientas Automatizadas.* 24 de Enero de 2013. http://dianbeel.blogspot.com/2012/06/segundo-trabajo-herramienta-case-visual.html.

28. *Lenguaje de Programación.* España : s.n. http://www.lenguajes-de-programacion.com/lenguajes-de-programacion.shtml.

29. Alvarez, Miguel Angel. *Qué es Java.* 18 de julio de 2001. http://www.desarrolloweb.com/articulos/497.php.

30. —. *Qué es PHP.* 9 de mayo de 2001. http://www.desarrolloweb.com/articulos/392.php.

31. Jose, Juan. *Eclipse IDE.* 10 Enero 2014. http://www.genbetadev.com/herramientas/eclipse-ide.

32. Avila, Katty. *¿Qué es un Sistema Gestor de Bases de Datos o SGBD?* http://www.cavsi.com/preguntasrespuestas/que-es-un-sistema-gestor-de-bases-de-datos-o-sgbd/.

33. Martinez, Rafael. *Sobre postgresql.* 2013. http://www.postgresql.org.es/sobre_postgresql.

34. Peña, Alvaro. *MYSQL ¿Qué es y para qué sirve?* España : s.n., 26 agosto, 2014. http://www.isocialweb.es/mysql-que-es-y-para-que-sirve/.

35. *Javier Zapata Sánchez.* 13 enero, 2013 . https://pruebasdelsoftware.wordpress.com/.

36. *Pruebas de rendimiento.* http://www.ecured.cu/Jmeter.

37. Ivan M. Caballero, Jeimy J. Cano. *CONSIDERACIONES PARA IMPLEMENTAR UNA ARQUITECTURA SINGLE SIGN-ON.*

38. *Tecnología y Synergix.* 18 de febrero de 2013. http://synergix.wordpress.com/2008/07/10/modelo-de-dominio/.

39. Parte_I_Vision_General, Sommerville 7ma Edición. *Definición de los requerimientos de sistema.* Capitulo 2 epigrafe 2.2.1.

40. II, Sommerville 7ma Edición Parte. *Requerimientos funcionales.* Capitulo 6 Requerimientos epigrafe 6.1.1.

41. —. *Requerimientos no funcionales.* Capitulo 6 Requerimientos epigrafe 6.1.2.

42. *Calidad del producto de software ISO-IEC 25010.* 2015. http://www. iso25000.com.

43. II, Pressman 6ta Edición Parte. *Desarrollo de Casos de Uso.* Capitulo 7 Ingenieria_de_Requisitos epigrafe 7.5.

44. 1, Pressman parte. *Modelo de análisis Capitulo 8 epígrafe 8.1 Análisis de requisitos.*

45. —. *Capitulo 8 modelo de análisis epígrafe 8.1.1 Filosofía y objetivos generales.*

46. Cillero, Manuel. *Diagrama de colaboración.* https://manuel.cillero.es/doc/metrica-3/tecnicas/diagrama-de-colaboracion/.

47. edición, Pressman 6ta. *Capitulo 9 Ingenieria del diseno epígrafe 9.2 Proceso y calidad del diseño.*

48. —. *Capítulo 9 Ingenieria del diseno.*

49. Diaz, Yuli Bargas y Mabel. *Diagramas de secuencia.* http://exposicinds.blogspot.com/2009/05/diagrama-de-secuencia.html.

50. edición, Pressman 6ta. *Capítulo 9 Ingenieria del diseno epígrafe 9.5.2 Utilización de patrones de diseño.*

51. Grosso, Andrés. *Patrones GRASP.* 21 de marzo de 2011. http://www.practicasdesoftware.com.ar/2011/03/patrones-grasp/.

52. Mora, Roberto Canales. https://www.adictosaltrabajo.com/tutoriales/grasp/.

53. *Diseño de bases de datos Departamento de informática Universidad FASTA.* http://www.ufasta.edu.ar/MediatecaWeb/catalogo/detalles.asp?M=3278.

54. Rumbaugh, James y Grady Jacobson, Booch. *El Proceso Unificado de Desarrollo de Software.* s.l. : [En línea] [Citado el: 16 de Abril de 2015.], 2000. http://www.fiury.net/ebooks-gratis/4102674/el-proceso-unificado-de-desarrollo-de-software.html.

55. Marca Huallpara Hugo Michael, Quisbert Limanchi Nancy Susana. *Análisis y diseño de sistemas II: Diagrama de despliegue.* http://es.slideshare.net/.

56. Terreros, Julio Casal. *Desarrollo de Software basado en Componentes.* https://msdn.microsoft.com/es-es/library/bb972268.aspx.

57. edición, Pressman 6ta. *Capítulo 13 Estrategia Prueba epígrafe 13.1 Un enfoque estratégico para la prueba del software.*

58. —. *Capítulo 13 Estrategia Prueba epígrafe 13.1.3 Estrategia de prueba del software.*

59. —. *Capítulo 14 Tecnicas de Prueba Parte 1 epígrafe 14.2 Pruebas de caja negra y caja blanca. .*

60. Villareal, Jean Carlos. *Gestión de calidad y pruebas del software.* España : s.n., 2005. La prueba de aceptación es la prueba más importante para los productos software.htm.

61. Jazmin, Zareth. *Pruebas de aceptación al cliente.* https://prezi.com/pghxcs44c0m-/pruebas-de-aceptacion-del-cliente/.

62. Mingo, Pedro Sebastian. *Para qué sirven las pruebas de rendimiento (I). Introducción.* 04/02/2013 . http://pedrosebastianmingo.com/para-que-sirven-las-pruebas-de-rendimiento-i-introduccion/.

ANEXOS.

Anexo 1. Imágenes del sistema.

Ilustración 26. Inicio de sesión único.

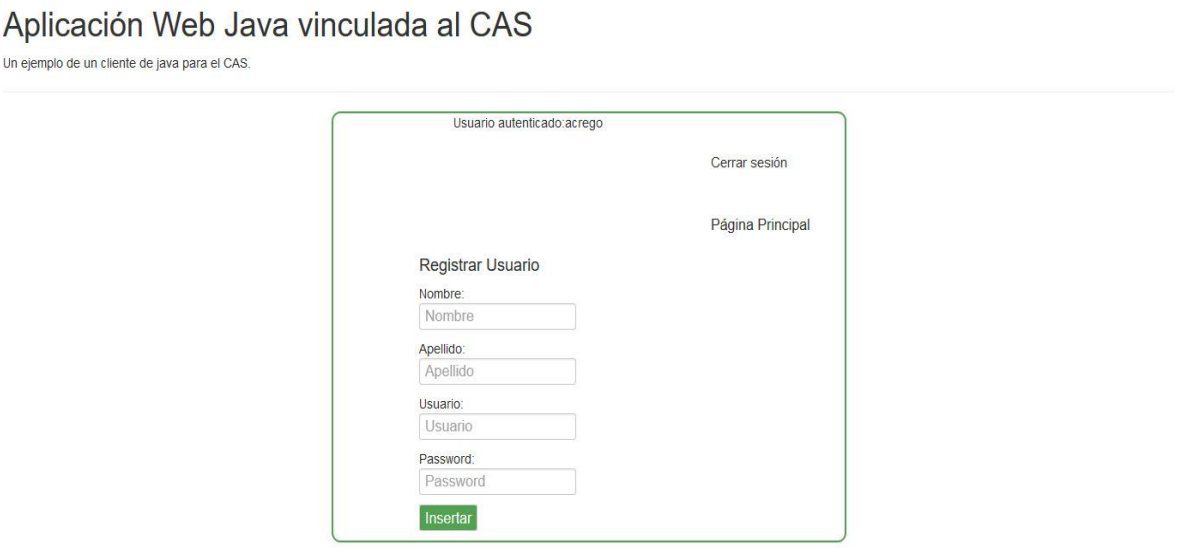

Ilustración 27. Página Registrar usuario aplicación java.

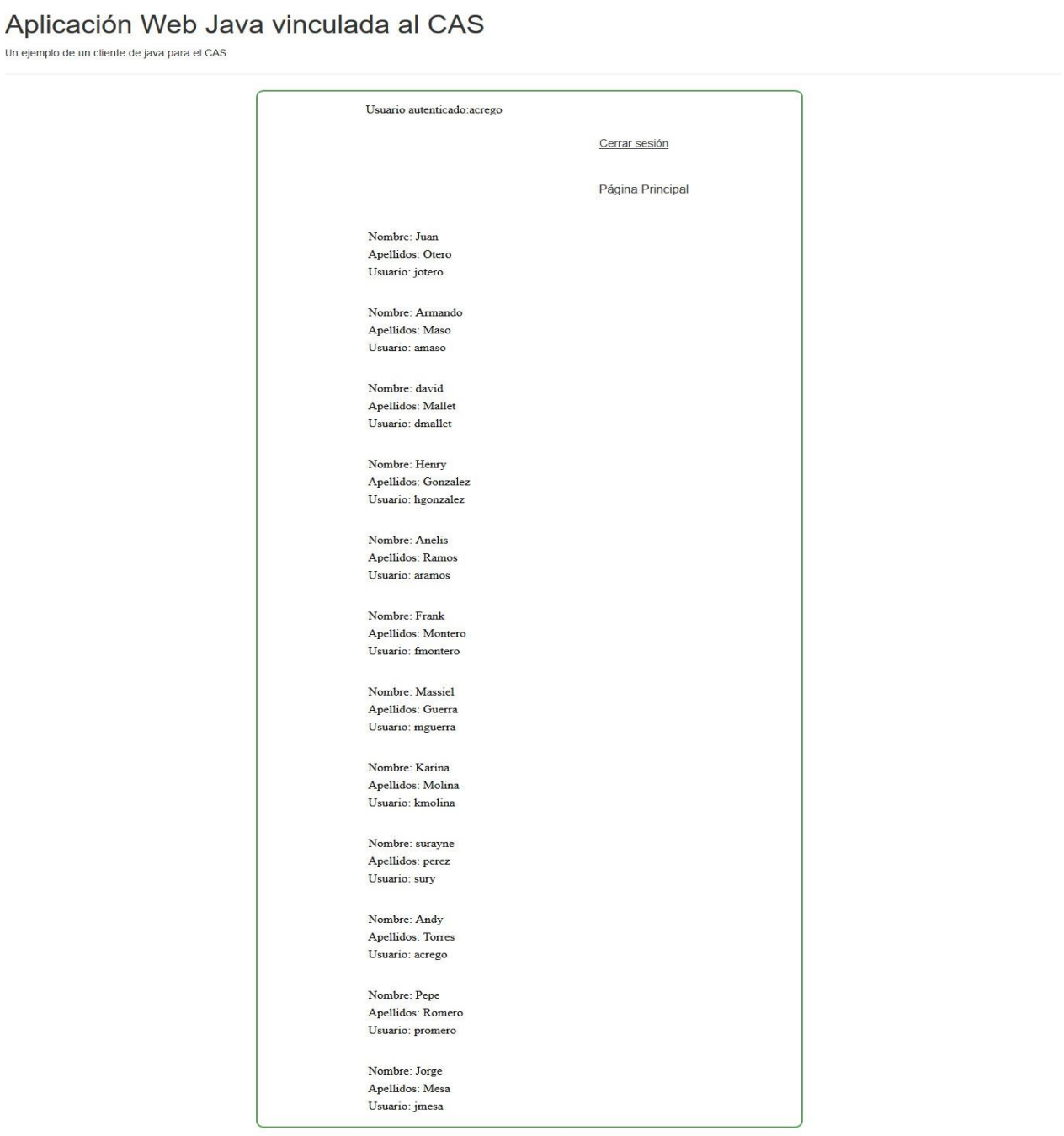

Ilustración 28. Página Listar usuario.

Anexo 2. Pruebas de rendimiento del sistema.

Proyecto JMeter.

Ilustración 29. Proyecto jMeter. Aplicación.

1 Vista por páginas.

1.1 *SSO*-login.

Etiqueta	# Muestras	Media	Mediana	Linea de 90%	Mín	Máx	% Error	Rendimiento	Kb/sec
/sso/js/jquery-1....	50	2023	2014	2044	2005	2159	0,00%	18,9/sec	39,3
/sso/js/bootstra...	50	2012	2009	2028	2005	2030	0,00%	16,6/sec	34,5
/sso/css/login.c...	50	2012	2011	2019	2004	2042	0,00%	15,4/sec	32,0
/sso/css/bootstr...	50	2011	2008	2022	2005	2037	0,00%	14,5/sec	30,1
/sso/js/classie.j...	50	2010	2005	2013	2003	2164	0,00%	13,8/sec	28,7
Total	250	2014	2009	2028	2003	2164	0,00%	21,2/sec	44,1

Ilustración 30. SSO-login. Informe agregado.

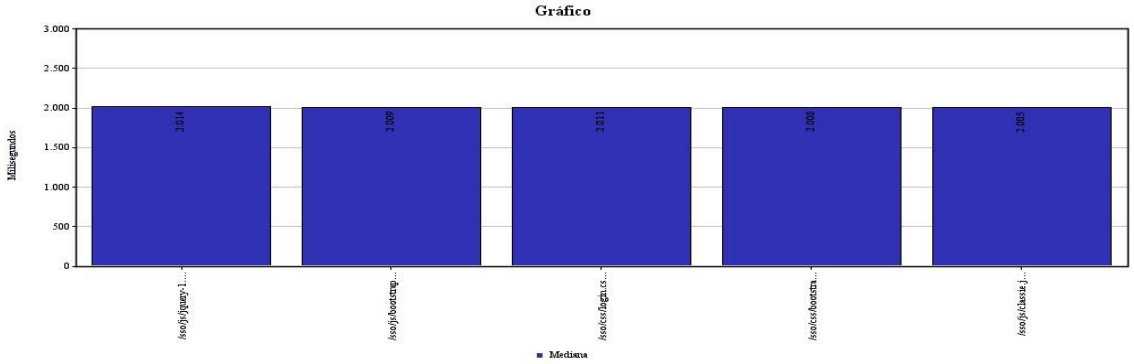

Ilustración 31. SSO-login. Gráfico.

1.2 Favicon.

Etiqueta	# Muestras	Media	Mediana	Linea de 90%	Mín	Máx	% Error	Rendimiento	Kb/sec
/favicon.ico	100	2010	2007	2020	2004	2043	0,00%	16,7/sec	34,8
Total	100	2010	2007	2020	2004	2043	0,00%	16,7/sec	34,8

Ilustración 32. Favicon. Informe agregado.

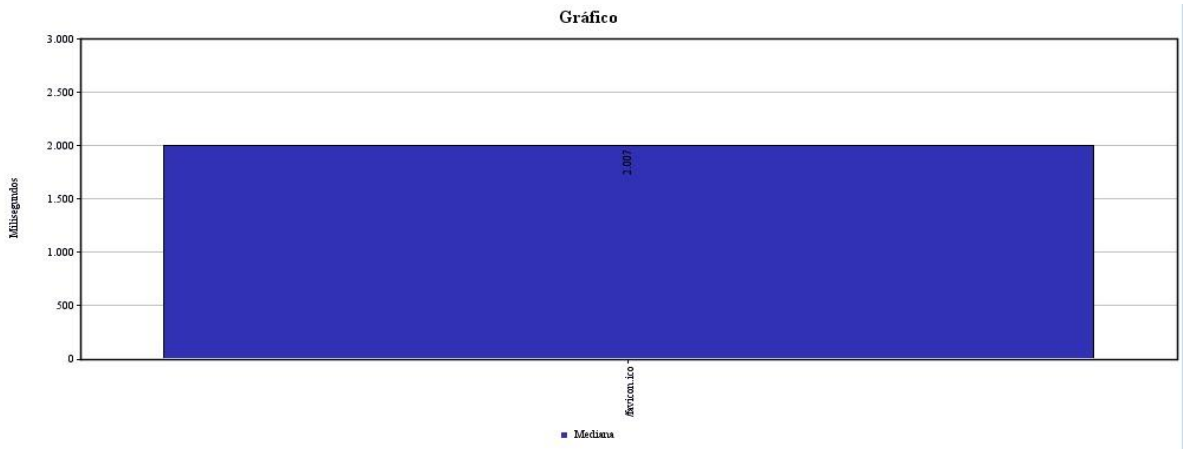

Ilustración 33. Favicon. Gráfico.

1.3 SSOservice.

Etiqueta	# Muestras	Media	Mediana	Linea de 90%	Mín	Máx	% Error	Rendimiento	Kb/sec
/sso/login?servi...	50	2007	2007	2012	2004	2019	0,00%	12,1/sec	25,2
Total	50	2007	2007	2012	2004	2019	0,00%	12,1/sec	25,2

Ilustración 34. SSOservice. Informe agregado.

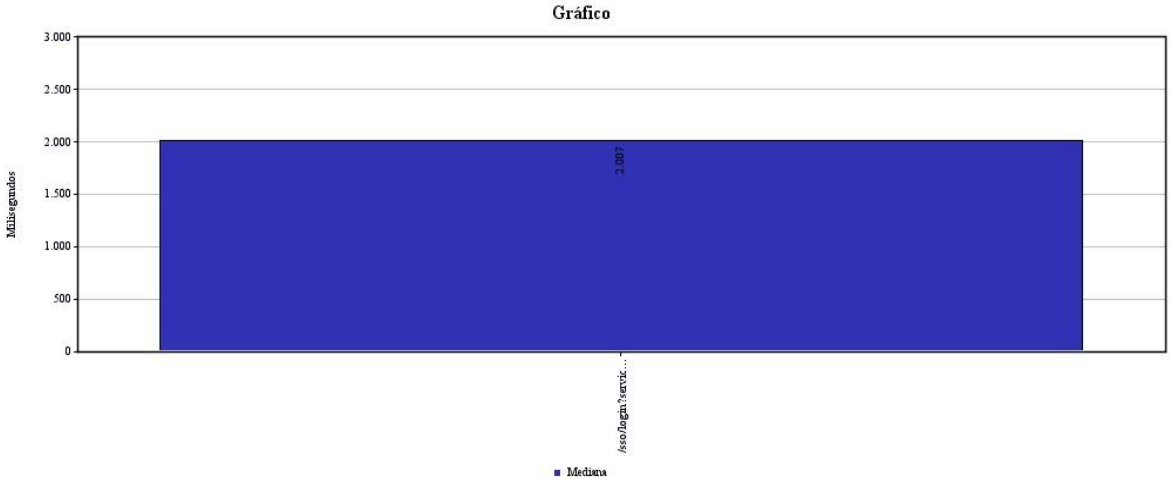

Ilustración 35. SSOservice. Gráfico.

1.4 SSOsesion.

Etiqueta	# Muestras	Media	Mediana	Linea de 90%	Mín	Máx	% Error	Rendimiento	Kb/sec
/sso/login;jsess...	50	2009	2007	2015	2004	2042	0,00%	12,3/sec	25,5
Total	50	2009	2007	2015	2004	2042	0,00%	12,3/sec	25,5

Ilustración 36. SSOsesion. informe agregado.

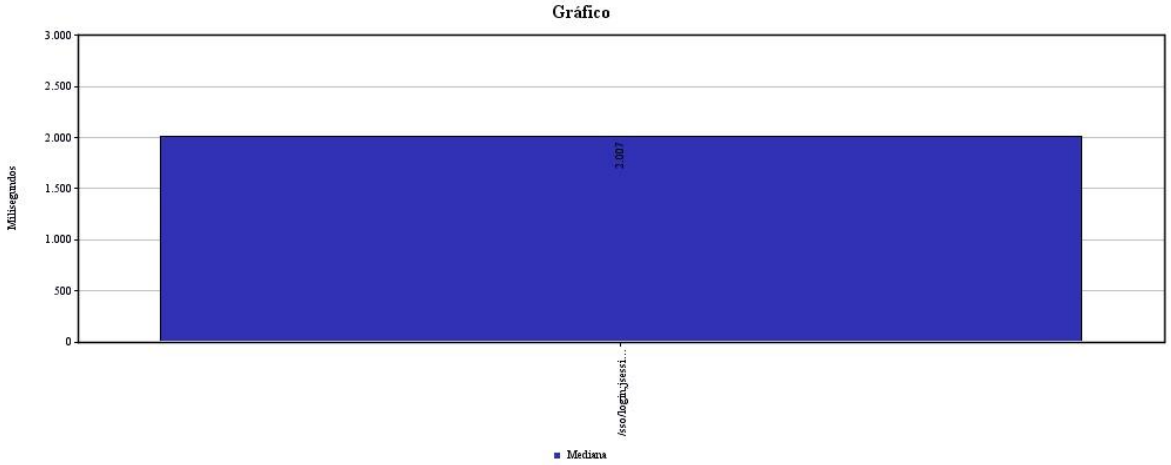

Ilustración 37. SSOsesion. Gráfico.

1.5 Usuario.

Etiqueta	# Muestras	Media	Mediana	Linea de 90%	Mín	Máx	% Error	Rendimiento	Kb/sec
/AP1/usuario.jsp	50	2007	2006	2011	2003	2031	0,00%	12,1/sec	25,1
/AP1/jquery-2.0....	100	2006	2006	2010	2003	2029	0,00%	9,5/sec	19,8
/AP1/js/main.js	50	2006	2006	2009	2003	2010	0,00%	11,9/sec	24,6
/AP1/js/main2.js	50	2005	2005	2008	2003	2021	0,00%	11,6/sec	24,1
Total	250	2006	2006	2009	2003	2031	0,00%	20,0/sec	41,5

Ilustración 38. Usuario. Informe agregado.

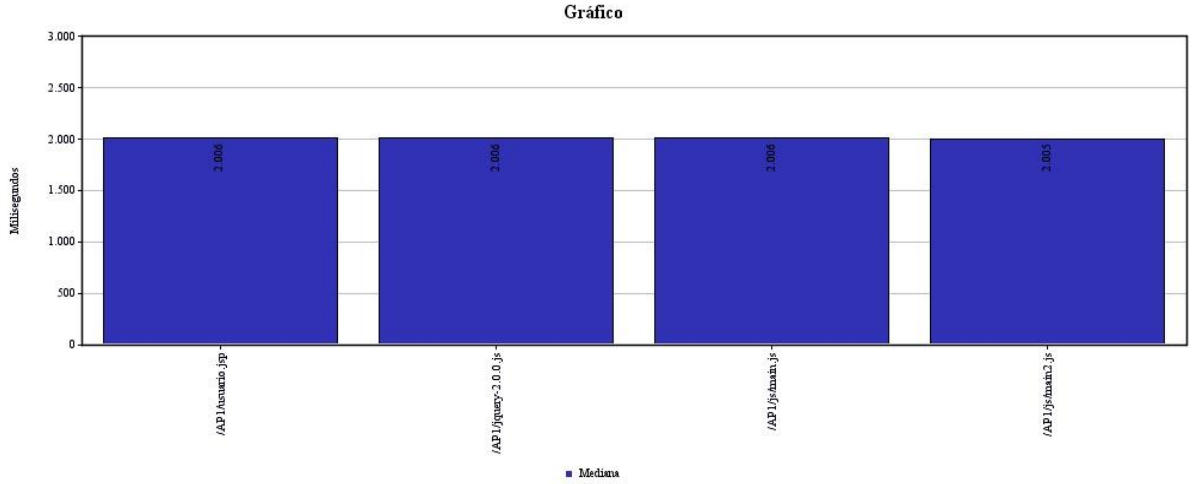

Ilustración 39. Usuario. Gráfico.

1.6 UsuarioServlet.

Etiqueta	# Muestras	Media	Mediana	Linea de 90%	Mín	Máx	% Error	Rendimiento	Kb/sec
/AP1/UsuarioSe...	50	2009	2007	2019	2004	2025	0,00%	11,1/sec	23,0
Total	50	2009	2007	2019	2004	2025	0,00%	11,1/sec	23,0

Ilustración 40. UsuarioServlet. Informe agregado.

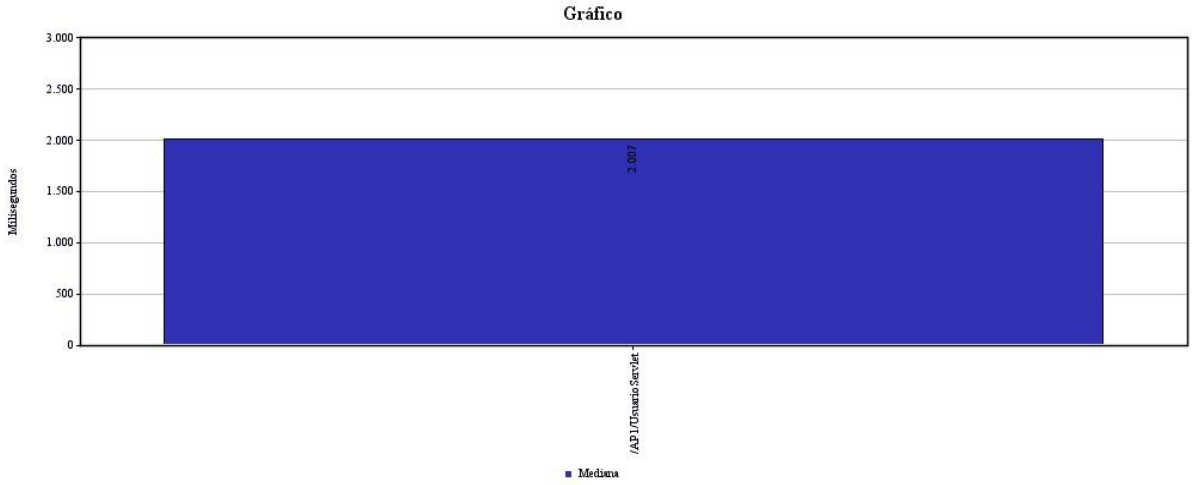

Ilustración 41. UsuarioServlet. Gráfico.

1.7 UsuarioServletJQuery.

Etiqueta	# Muestras	Media	Mediana	Linea de 90%	Mín	Máx	% Error	Rendimiento	Kb/sec
/AP1/UsuarioSe...	50	2006	2006	2010	2003	2022	0,00%	11,6/sec	24,0
/AP1/jquery-2.0....	50	2006	2006	2008	2003	2027	0,00%	11,3/sec	23,4
Total	100	2006	2006	2009	2003	2027	0,00%	15,4/sec	32,0

Ilustración 42. UsuarioServletJQuery. Informe agregado.

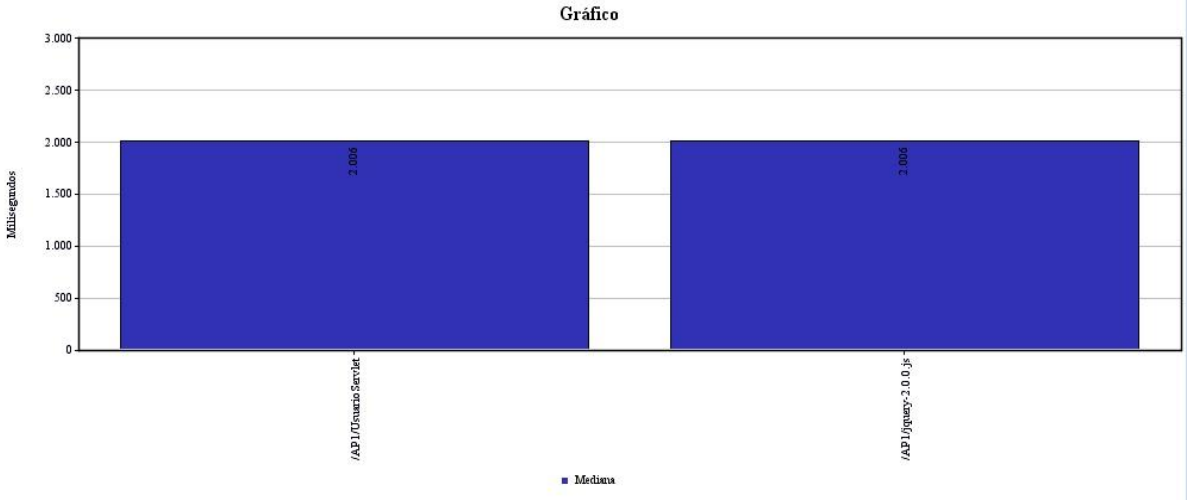

Ilustración 43. UsuarioServletJQuery. Gráfico.

1.8 Welcome.

Etiqueta	# Muestras	Media	Mediana	Linea de 90%	Mín	Máx	% Error	Rendimiento	Kb/sec
/AP1/jsps/welco...	50	2450	2423	2540	2322	2596	0,00%	18,6/sec	38,7
Total	50	2450	2423	2540	2322	2596	0,00%	18,6/sec	38,7

Ilustración 44. Welcome. Informe agregado.

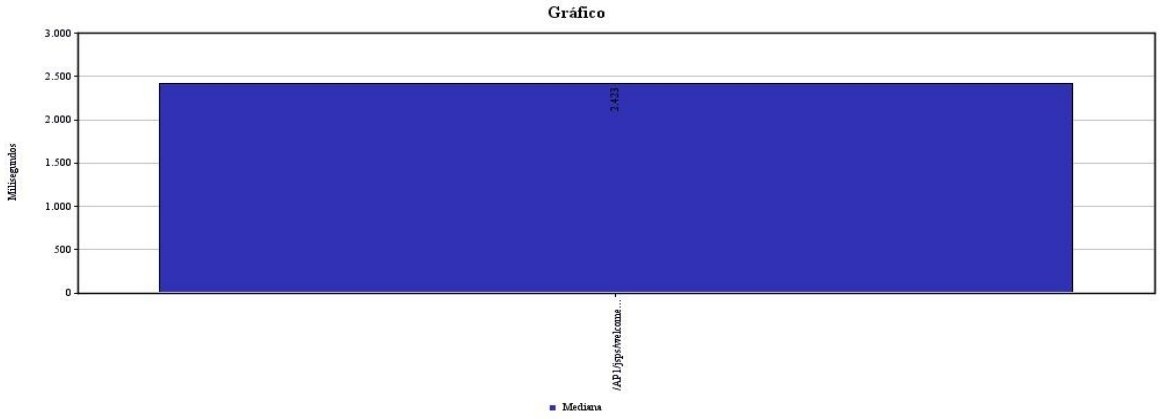

Ilustración 45. Welcome. Gráfico.

1.9 WelcomeSesion.

Etiqueta	# Muestras	Media	Mediana	Linea de 90%	Mín	Máx	% Error	Rendimiento	Kb/sec
/AP1/jquery-2.0....	50	2008	2006	2015	2003	2028	0,00%	11,9/sec	24,8
Total	50	2008	2006	2015	2003	2028	0,00%	11,9/sec	24,8

Ilustración 46. WelcomeSesion. Informe agregado.

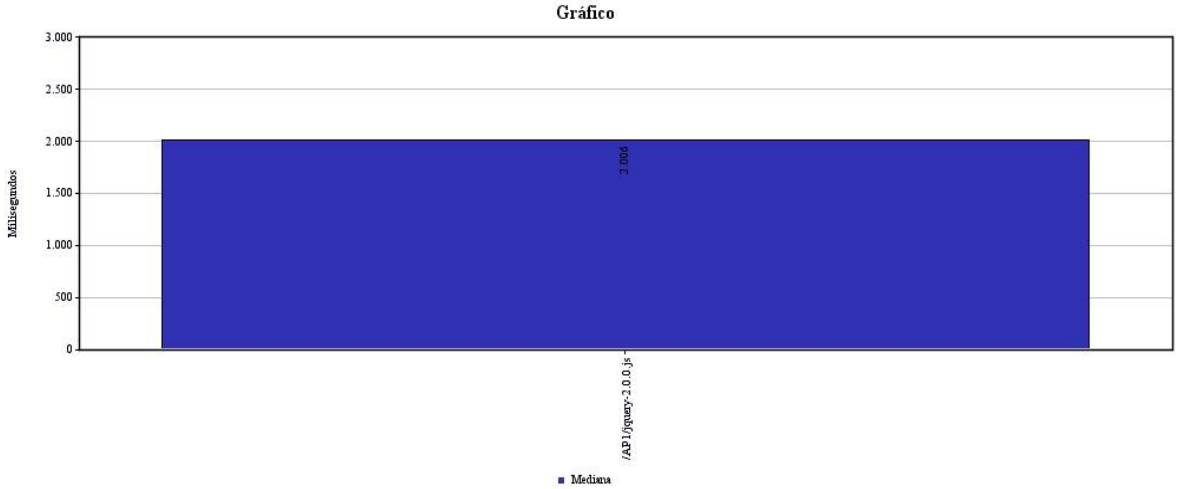

Ilustración 47. WelcomeSesion. Gráfico.