Universidad de las Ciencias Informáticas

Facultad 5

Centro de Consultoría y Desarrollo de Arquitecturas Empresariales (CDAE)

Título: Gestión de la autenticación mediante Single Sign-On (SSO) para el Sistema de Reservación de Alimentos UCI.

Trabajo de Diploma para optar por el título de
Ingeniero Informático

Autor: David López Mallett

Tutor(es): Ing. Yuriseidis M. Reina Torres.

Ing. Armando Masó Mosqueda.

La Habana, junio de 2016

"Año 58 de la Revolución"

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de ____del año__.

David López Mallett

_____

Firma del Autor

Yuriseidis María Reina Torres

_____

Firma del Tutor

DATOS DE CONTACTO

Tutor(es):

Ing. Yuriseidis María Reina Torres

Universidad de las Ciencias Informáticas, Ciudad de la Habana, Cuba

Correo: ymreina@uci.cu

Ing. Armando Masó Mosqueda

Universidad de las Ciencias Informáticas, Ciudad de la Habana, Cuba

Correo: amaso@uci.cu

AGRADECIMIENTOS

Después de muchos años de estudio y esfuerzo, hoy prácticamente termina mi vida de estudiante universitario. Para hacer realidad este sueño muchas personas me ayudaron, apoyaron y depositaron su confianza en mí. Hoy tengo la oportunidad de hacerles saber cuan agradecido estoy.

A mis padres por todo su esfuerzo y sacrificio, por su cariño, su apoyo incondicional, gracias por estar siempre cuando los necesité. Quiero que sepan que los quiero y los admiro mucho y este logro es de ustedes también, nunca le podré estar lo suficientemente agradecido por todo lo que han hecho por mí, gracias por ser mi guía, mi apoyo y mi vida.

A mi abuela, mi hermano, que me ha dado el mejor regalo del mundo, mi sobrinita Vanesa, mi tía Virginia y mi prima Lupe, mis tíos que, aunque estén lejos siempre me han apoyado a lo largo de mi carrera. Solo quiero que sepan que los quiero con la vida y este logro es de ustedes también.

Por otra parte, quisiera agradecerles a mis tutores por todo su esfuerzo y paciencia, son una de las piezas más importantes en esta tesis. A la profesora Yirka, una de los mejores profesores que he conocido.

También a las personas que forman el día a día de mi vida, mi piquete, Ulises, Alejandro, Begoña y Amanda, gracias por estar ahí para mí siempre que los necesité, apoyarme y aconsejarme, quiero decirle que son más que mis amigos, son parte de mi familia. A mi piquete de la UCI, como olvidarme de ellos, Juan Carlos, Andy, el Tito, Henry, José Miguel, el Nino y a todos mis compañeros de la universidad con los cuales he vivido grandes momentos, los aprecio mucho y sepan que pueden contar conmigo para lo que sea. Gracias por estar ahí estos 5 años.

En fin, a todos aquellos que de una forma u otra han formado parte de mi vida y me han apoyado, gracias.

DEDICATORIA

A mi familia, en especial a mis padres, mi abuela y mi hermano y a todas las personas que me quieren y siempre confiaron en mí brindándome su apoyo para poder realizar este sueño, quiero regalarles este momento y honrarlos por tanto amor y dedicación. Los quiero mucho.

RESUMEN

Actualmente las empresas u organizaciones centran su atención en mejorar y mantener un seguimiento y control de la seguridad de sus sistemas informáticos para proteger la integridad, disponibilidad y confidencialidad de los datos. A causa de las constantes amenazas que exponen las debilidades de sus recursos se deben disminuir los riesgos o el impacto que estos puedan tener, velando por la protección de sus activos. La seguridad es primordial, y su correcta gestión obliga al diseño de entornos que faciliten su uso a innumerables usuarios, sin que provoquen una pérdida de seguridad, confidencialidad y disponibilidad.

Las organizaciones cubanas han implantado varios sistemas informáticos para la gestión de sus procesos de negocio que ayudan a tomar decisiones que garantizan dar cumplimiento a sus necesidades. La seguridad de estos sistemas es gestionada mayormente por el método tradicional de usuario/contraseña. Este método no garantiza en su totalidad la seguridad de la información, debe de estar acompañado de políticas de seguridad, infraestructura de autenticación de los sistemas operativos y otros mecanismos de autenticación.

La aplicación de un sistema *Single Sign-On (SSO)* es actualmente uno de los mecanismos más usados para garantizar la máxima protección de la información mediante autenticación múltiple. En la presente investigación se argumenta la implementación de un sistema SSO para el sistema de Reservación de Alimentos de la Universidad de las Ciencias Informáticas (UCI).

Palabras claves: autenticación, seguridad, información.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Garantizar la seguridad de la información es uno de los grandes retos de la informática corporativa y en cualquier organización se requiere que se proteja ante cualquier amenaza. Una de las medidas de seguridad más comunes que utilizan las organizaciones para acceder a los sistemas locales es la gestión de la identidad de los usuarios a través de un usuario y una contraseña. Es importante que la autentificación del usuario sea correcta para que la aplicación permita un acceso correspondiente al rol de cada usuario.

Hoy en día existen mecanismos que permiten reducir los ataques que ponen en evidencia la información al transitar por la red, por lo que ser interceptada sería un obstáculo si se tratase de información sensible. El fortalecimiento con el uso de contraseñas en la red bloquea el tránsito de los usuarios no autentificados. En diferentes entornos se ha demostrado que varios mecanismos de autenticación como el clásico usuario/contraseña son poco seguros o inadecuados, y más aún cuando los usuarios deben de acceder a cada aplicación por separado, ya que estas tienen su propio sistema de autenticación. Esto hace más engorroso el proceso de gestión de identidades y permisos a medida que aumenta la cantidad de usuarios en la red y debilitando la seguridad de acceso a las aplicaciones comprometiendo la confidencialidad de los datos.

La necesidad de que el cliente repita el ingreso de la contraseña cada vez que accede a un servicio trae consigo varias dificultades: tiempo prolongado al tener que registrarse en varios sistemas para consultar información, la ubicación del servidor en una máquina remota implica inseguridad. Cuando la red crece mucho porque se encuentran una gran cantidad de usuarios conectados, puede volverse muy complejo el proceso de autentificación, es decir que es afectada la escalabilidad.

Debido a esto se hace necesaria la gestión de la autenticación mediante la utilización de mecanismos que agilicen este proceso, sin que cause perdida de información, seguridad, confidencialidad o disponibilidad y que faciliten la gestión de grandes volúmenes de usuarios reduciendo el riesgo de caída de las aplicaciones y garantizando la gestión de identidades con el aumento de la seguridad, preferentemente con el uso de sistemas de identidad unificados.

La Universidad de las Ciencias Informáticas (UCI) cuenta con varios sistemas en la red donde se utiliza el método tradicional de usuario/contraseña como mecanismo de seguridad para acceder a diversas aplicaciones, este proceso de autentificación se hace muy

complejo al tener que acceder a cada uno de ellos de forma independiente. Para eliminar estas dificultades el Centro de Consultoría y Desarrollo de Arquitectura Empresariales(CDAE) perteneciente a la UCI se propuso el desarrollo de una integración que permita la gestión de autenticación centralizada mediante CAS (*Central Authentication Service*) para la implantación de un sistema de autenticación múltiple SSO. Esta propuesta tiene como objetivo un aumento en la productividad, tener mayor facilidad de acceso a los recursos, funciones de autenticación a través de una única plataforma, una administración sencilla de credenciales y sobre todo garantizar un aumento de la seguridad. Mediante este servicio el usuario podrá registrarse en el sistema una sola vez, con lo cual podrá acceder a todos los recursos sin tener que volver a autentificarse.

El centro CDAE en conjunto con la Dirección de Informatización de la UCI desarrolló un sistema para la reservación de alimentos. Este sistema ha sido seleccionado por la dirección de la UCI para la integración con el nuevo sistema de autenticación.

De la situación problemática planteada anteriormente se deriva la necesidad de solucionar el siguiente **problema de investigación**: ¿Cómo implementar un mecanismo de autentificación centralizada para el Sistema de Reservación de Alimentos UCI?

Quedando definido como **objeto de estudio**: El proceso de autenticación centralizada en sistemas web.

Para contribuir a la solución del problema se propone como **objetivo general:**

Implementar un mecanismo de autenticación centralizada mediante SSO-CAS para el Sistema de Reservación de Alimentos UCI.

Especificándose como **campo de acción:** El proceso de autenticación centralizada mediante SSO-CAS para el Sistema de Reservación de Alimentos UCI.

Con vista a dar cumplimiento al objetivo general se plantean las siguientes **tareas de investigación:**

1. Analizar de los diferentes mecanismos de autentificación centralizada existentes.
2. Analizar de las diferentes arquitecturas para sistemas de autenticación centralizada.
3. Seleccionar de una arquitectura que ayude a la implementación de un mecanismo de autenticación centralizada para el Sistema de Reservación de Alimentos UCI.
4. Analizar y diseño de un mecanismo de autenticación centralizada para el Sistema de Reservación de Alimentos UCI.

5. Implementar de un mecanismo de autenticación centralizada para el Sistema de Reservación de Alimentos UCI.
6. Validar un mecanismo de autenticación centralizada implementado para el Sistema de Reservación de Alimentos UCI.

Se espera obtener como **posible resultado:** El diseño e implementación de un mecanismo de autentificación centralizada mediante el CAS para la implantación de un sistema de autenticación múltiple SSO en el Sistema de Reservación de Alimentos UCI.

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 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.

La presente investigación estará desglosada en tres capítulos fundamentales:

En el **Capítulo 1:** se aborda el estado del arte de los Sistemas de Autentificación Única (SSO), su organización con respecto a los sistemas descentralizados y arquitecturas básicas existentes, realizando una explicación detallada en cuanto a características, ventajas y desventajas que presentan.

En el **Capítulo 2:** se realiza un estudio y análisis de las causas por las cuales se decide adoptar un mecanismo de autenticación SSO, mostrando así los beneficios que esto reportaría para el Sistema de Reservación de Alimentos de la UCI. Se muestran las especificaciones de los requerimientos de software, requerimientos funcionales y no funcionales. Se realiza un análisis de la arquitectura a usar para la implementación del SSO, el modelo de dominio y de casos de uso.

En el **Capítulo 3:** en correspondencia con los aspectos tratados en el capítulo anterior (requerimientos, casos de uso, etc.) se lleva a cabo la descripción de las tareas de implementación generadas obteniendo en cada una de ellas una versión del producto, así como las pruebas de aceptación y de rendimiento efectuadas sobre el sistema.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En este capítulo se explican los aspectos más significativos en el proceso de investigación para la implementación del proceso de autenticación centralizada para el Sistema de Reservación de Alimentos UCI, sus ventajas y desventajas, sus principales características, una descripción detallada de los sistemas de autenticación centralizada existentes, como se aplican estos procesos de autenticación, se hace mención y una breve explicación de las diferentes arquitecturas que implementan servicios de autenticación única en el dominio, la selección de la metodología de desarrollo, plataformas, herramientas, lenguajes de programación y *frameworks* de desarrollo a utilizar para darle solución a la problemática planteada.

1.2 Autenticación.

Consiste en un sistema para certificar que el usuario es quien dice ser; lo más común es utilizar una combinación de identificador de usuario único y contraseña, aunque existen otros. Un sistema de *single sign-on* consiste, por tanto, en un protocolo de autenticación que funcione en más de un sistema; es federado si el sistema responsable de la autenticación puede ser cualquiera que cumpla el estándar definido por el sistema; por otro lado, será un sistema delegado si el sistema que autentica es uno predeterminado. (1)

1.3 *Single Sign-On*.

Single Sign-On se refiere al acceso a múltiples recursos por medio de un único proceso de ingreso o autenticación. Muchas de las arquitecturas implementadas en diferentes organizaciones han sido diseñadas con el objetivo de dar acceso a los usuarios a múltiples servicios Web y/o aplicaciones.

El principal objetivo de una arquitectura que implemente *Single Sign-On* es transferir la funcionalidad y complejidad de todos los componentes de seguridad a un solo servicio de *Single Sign-On (SSO)*. En una arquitectura SSO solo existe un único punto de autenticación y registro en el sistema.

Con la implementación de un mecanismo *Single Sign-On* el sistema se encarga de almacenar en una base de datos o directorios protegidos, las credenciales que le permiten al usuario acceder a las aplicaciones en el momento que desee. Dicha implementación simplifica y centraliza el control de acceso a todos los servicios de las organizaciones,

reduciendo el costo en la administración de seguridad, lo que logra un mejor rendimiento y velocidad de los procesos de autenticación. Facilitando a los usuarios la interacción con los sistemas de la organización, simplificando el manejo de claves y lo fundamental es que aumenta los niveles de seguridad.

Las arquitecturas diseñadas en diferentes organizaciones tienen el objetivo de permitir a los usuarios el acceso a varios servicios web y aplicaciones, en momentos se encuentran con problemas de seguridad y tener que afrontar altos costos de implementación y mantenimiento. Con las nuevas tecnologías, las infraestructuras de las aplicaciones que son cada vez más firmes y la madurez de las diferentes técnicas de autentificación y seguridad, han permitido que se reconsidere la posibilidad de implementar el SSO como una estrategia para fortalecer la seguridad informática de las organizaciones. (2)

1.3.1 Tipos de sistemas *Single Sign-On*.

Existen 5 tipos principales de sistemas SSO, también conocidos como Sistemas de Autenticación Reducidas (*Reduced Sign-On System*). Los cuales se consideran a continuación.

- ***Enterprise Single Sign-On (E-SSO):*** conocido como *Legacy Single Sign-On*, el cual funciona luego de una autenticación primaria, interceptando los requisitos de autenticación presentados por las aplicaciones secundarias para completarlos con el usuario y contraseña. Los sistemas E-SSO permiten interactuar con sistemas que pueden deshabilitar la presentación de la pantalla de login.
- ***Web Single Sign-On (Web-SSO):*** conocido como *Web Access Management* (Web-AM), trabaja solo con aplicaciones y recursos que se acceden vía Web. Los accesos son interceptados con la ayuda de un servidor Proxy o de un componente instalado en el servidor Web destino. Los usuarios no autenticados que tratan de acceder son redirigidos a un servidor de autenticación y regresan solo después de haber logrado un acceso exitoso. Se utilizan *cookies*, para reconocer aquellos usuarios que acceden y su estado de autenticación.
- ***Kerberos:*** es un método popular de externalizar la autenticación de los usuarios. Los usuarios se registran en el servidor *Kerberos* y reciben un *ticket*, que luego utilizan para obtener acceso.
- ***Federation:*** es una nueva manera de concebir este tema, también para aplicaciones Web. Utiliza protocolos basados en estándares para habilitar que las

aplicaciones puedan identificar los clientes sin necesidad de autenticación redundante.

- ***OpenId:*** es un protocolo de SSO distribuido y descentralizado donde la identidad se compila en una URL de forma que cualquier aplicación o servidor puede verificar. (3)

1.3.2 Características de un sistema *Single Sign-On*.

- **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 vez y el sistema le permite acceder a los recursos para los cuales está autorizado. Así se evita 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 (SSL, certificados digitales). (4)

1.3.3 Ventajas de un sistema Single Sign-On.

- **Aumento en la productividad:** la productividad del usuario en beneficio de la organización se ve mejorada, ya que el tiempo que se tarda en brindar los permisos de acceso a todas las aplicaciones que requerirá el usuario, disminuiría notablemente por el hecho de solo tener que facilitar una identificación a este. Este tiempo antes perdido puede ser mejor aprovechado por el usuario.
- **Facilidad de acceso a los recursos:** permite el acceso de los usuarios desde diferentes plataformas a las cuales tiene permiso en un menor tiempo.
- **Administración sencilla de credenciales:** el aprovisionamiento, eliminación, y mantenimiento de las credenciales del usuario llevado a cabo por el administrador,

es mucho más sencillo debido a que SSO administra las credenciales y demás información de manera centralizada con la utilización de un servicio directorio.

- **Aumento de seguridad:** la seguridad aumenta con el SSO, ya que el usuario al utilizar solamente una identificación para el ingreso, no se ve obligado a escribir en un documento las contraseñas para poder recordarlas como sucede a veces, con lo cual podría exponerlas a individuos no permitidos. (5)

1.3.4 Desventajas de un sistema Single Sign-On.

- **Punto de falla centralizado:** debido a que la autenticación inicial llevada a cabo por medio del *Single Sign-On* para la validación del usuario, se realiza mediante un solo proceso, se crea un punto de falla único, ya que, si este proceso falla, sea por fallo en el servidor de aplicaciones, fallo en el servidor de credenciales o fallo en el agente SSO podría afectar todo el proceso y no permitir al usuario realizar sus labores.
- **Acoplamiento con Sistemas Operativos:** a pesar de que *Single Sign-On* contempla gran variedad de ambientes de sistemas operativos desde los cuales se puede acceder a las aplicaciones, otros ambientes quedan por fuera de este proceso, y algunos requieren de herramientas adicionales para poder ser parte del proceso, lo cual para algunas organizaciones dificulta la implementación ya que utilizan diferentes sistemas operativos.
- **Herramientas para fortalecimiento de seguridad:** como se mencionó en las ventajas de SSO, este permite adicionar procesos de autenticación fuerte con el fin de aumentar la seguridad, pero cabe resaltar que estos procesos no son parte de una parte de una solución *Single Sign-On*, por lo que toda organización que requiera robustecer el proceso de autenticación requerirá de recursos adicionales para implementarlo. (5)

1.4 Arquitecturas SSO.

La implementación de un sistema SSO debe estar en consideración a las diferentes arquitecturas. Pueden brindar solución para cada uno de los aspectos tales como los recursos computacionales y económicos. Cada una de ellas posee las características necesarias para determinar el más apropiado tipo de organización.

Las arquitecturas constan de tres componentes importantes:

- **La interface**: se le encomienda el almacenamiento de los documentos de los usuarios en una base de datos. Conocido como el agente SSO, el cual se localiza en el cliente y permite el acceso a las aplicaciones.
- **La administración**: es el mecanismo por medio del cual se realiza configuración, mantenimiento y monitorio del proceso SSO. La localización desde donde se realice depende de la arquitectura implementada.
- **Las credenciales**: Esta es la información confidencial del usuario para poder acceder a las aplicaciones, la cual es el nombre de usuario y la contraseña, pero además de estos datos, se almacena otra información personal del usuario. Toda esta información solo debe ser accedida por el Agente SSO y se encuentra ubicada dependiendo de la arquitectura empleada, esta puede ser en el cliente mismo, o en una base de datos central. (3)

1.4.1 Arquitectura *Password vault*.

Esta arquitectura demanda de una configuración muy básica para la implementación de SSO. En este caso los tres elementos mencionados anteriormente se encuentran ubicados en el cliente, por lo que deben acceder a las aplicaciones justamente desde allí. El primer paso es la registración de las credenciales que el usuario solicitará para que puedan proporcionarse a las aplicaciones y poder accederlas cuando él desea, como se ilustra en la figura.

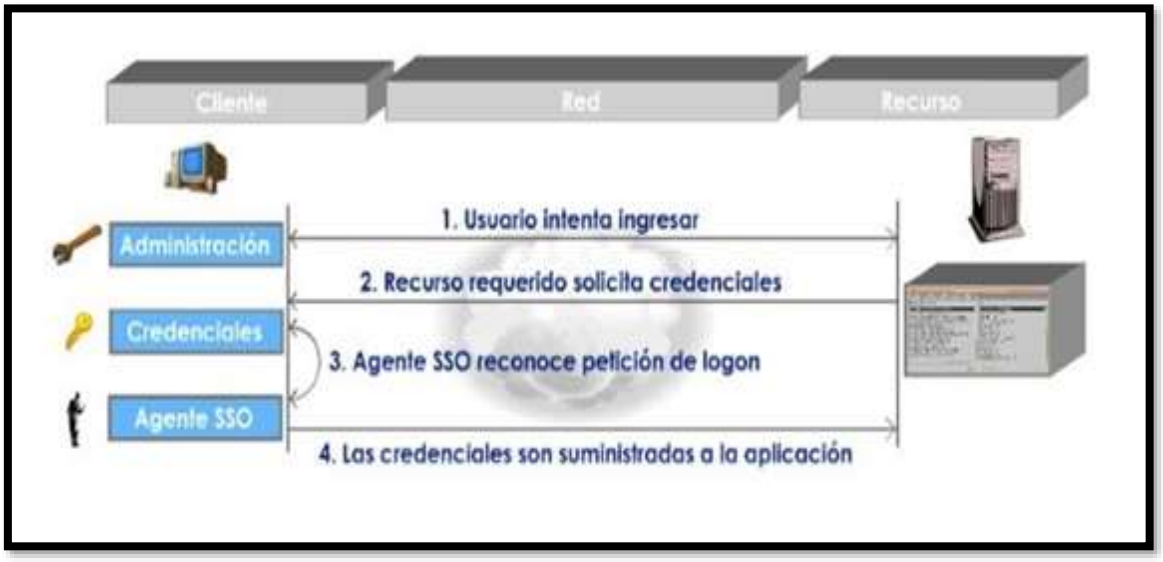

Figura 1.Arquitectura Password vault.

Otras de las características que posee la arquitectura *Password vault* se encuentran las funciones administrativas limitadas donde la administración de cada uno de los clientes debe realizarse desde la estación correspondiente y por lo tanto generalmente termina quedando a cargo del usuario. No es posible actualizar los clientes de manera masiva en toda la organización, requiere que se realice máquina por máquina.

Dentro de las ventajas de este tipo de arquitectura se encuentran:

- La implementación de este proyecto no es muy complicada ya que figura de tantos recursos por configurar.
- Requiere de pocos recursos computacionales, solamente un servidor central donde ocupen las aplicaciones y los clientes necesarios.

Dentro de sus desventajas se pueden mencionar:

- La administración es muy limitada, por lo que las empresas deben tomar medidas de seguridad informática y control de acceso sobre los recursos.
- La administración en cada una de las máquinas incita la configuración del sistema SSO en cada máquina, por lo que hace la labor más lenta.
- El paso de los usuarios a las aplicaciones desde múltiples estaciones se ve limitado debido a que las credenciales se almacenan en una sola máquina.

1.4.2 Administración centralizada con almacenamiento local de credenciales.

El diseño de arquitectura con administración centralizada con almacenamiento local de credenciales soluciona los inconvenientes que presenta el *Password vault*. Brinda un mecanismo para controlar y monitorear el proceso de ingreso. Elimina la necesidad de configurar el SSO en cada uno de los clientes.

Figura 2.Administración centralizada con almacenamiento local de credenciales.

Características de la arquitectura:

- Contiene un componente de administración de manera centralizada.
- Solamente el agente SSO y las credenciales permanecen en el cliente.
- El *software* cliente es autónomo durante el proceso de autenticación, debido a que durante este proceso la labor de administración se restringe a realizar el monitoreo de los clientes.

Ventajas:

- Por medio de un servidor central, el diseño realiza la administración de forma centralizada.
- Las labores de administración, control de la configuración y monitoreo del *software* del cliente posee un bajo nivel de complejidad.

Desventajas:

- Las credenciales se almacenan en el cliente por lo que deben vigilar por precisar las medidas de control de acceso y confidencialidad de la información.
- La información entre el cliente que utiliza SSO y el servidor no transita cifrada.
- El cliente al realizar la conexión, el administrador SSO solo puede monitorear la conexión. No consigue efectuar acciones de desconexión o cambios de configuración.

1.4.3 Administración y almacenamiento de credenciales centralizadas.

Con el modelo de arquitectura administración y almacenamiento de credenciales centralizadas, se intenta solucionar los problemas encontrados en la arquitectura que almacena las credenciales localmente. Se estima un mayor grado de seguridad en cuanto al elemento de las credenciales, que permanecen en un lugar centralizado. Esta arquitectura se representa en la figura 3.

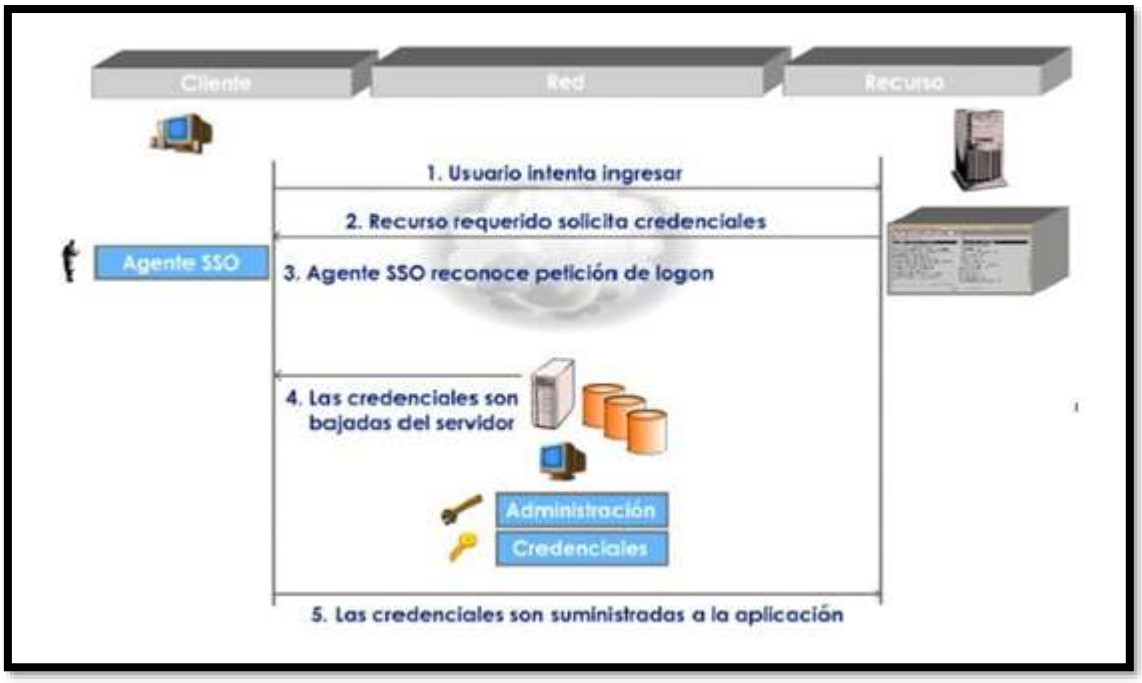

Figura 3.Administración y almacenamiento de credenciales centralizadas.

Características de la arquitectura:

- Las credenciales permanecen en un servidor central, que entrega las credenciales al cliente en el momento en que hace el ingreso.
- El administrador determina la frecuencia con que se descargan las credenciales del servidor (por sesión, por autentificación).

Ventajas:

- Los usuarios acceden a las aplicaciones de una manera centralizada.
- La posibilidad de acceder a los recursos desde cualquier estación en la que se encuentren, aumentando la disponibilidad de los recursos.

- La administración apropiada de las credenciales reduce la manipulación de la misma desde los clientes.

Desventajas:

- Se genera un único punto de falla, cambiando al SSO en una puerta de enlace para todos los recursos de la empresa, ya que el servidor debe ser contactado cada vez que se realice un ingreso.
- El acceso a todas las aplicaciones de la organización depende del servidor central.
- La configuración escasea de redundancia y respaldos de la información.

1.4.4 Arquitectura SSO totalmente distribuida.

Esta arquitectura es caracterizada por el alejamiento del servidor y la base de datos, lo cual lo hace totalmente modular. Supera los inconvenientes que pueden presentar las arquitecturas anteriores ofreciendo múltiples ventajas. Posee la mayor complejidad en su implementación y requiere de mayor inversión en los equipos de cómputo, el funcionamiento de esta arquitectura se puede ver en la siguiente figura.

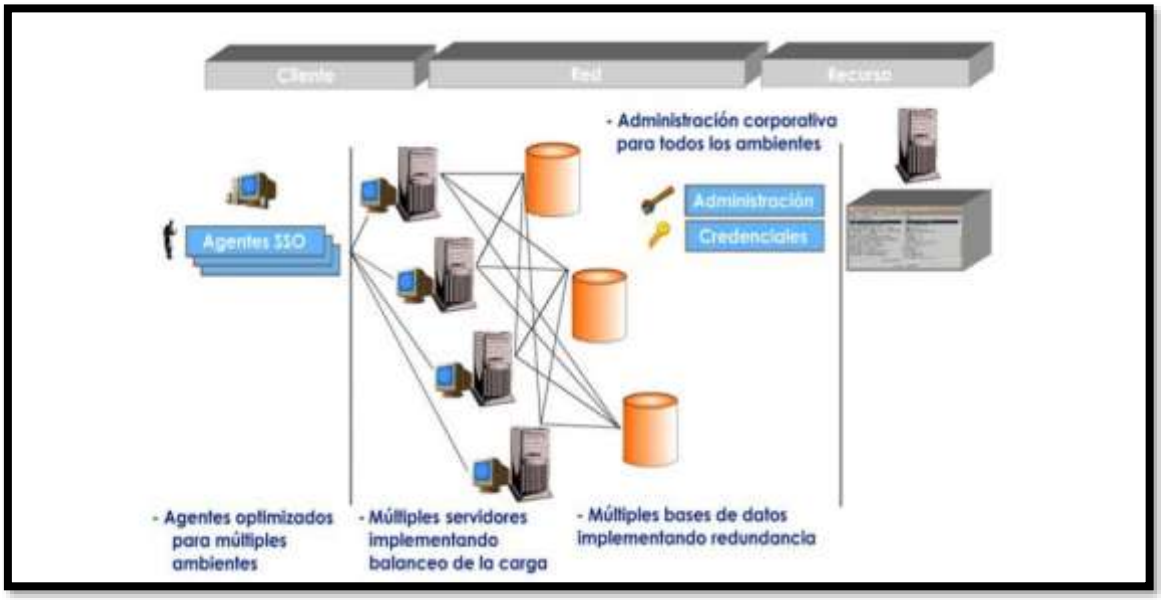

Figura 4.Arquitectura SSO totalmente distribuida.

La arquitectura SSO totalmente distribuida presenta las siguientes características:

- El servidor resulta ser una aplicación independiente que cuenta con un administrador diferente.
- El proceso de autentificación se remite a un recurso de red, cuando el agente SSO implanta conexión con un servidor SSO, las credenciales podrán solicitarse (y almacenarse en memoria caché para realizar *offline login*) y el proceso de autentificación podrá realizarse.
- La información se accede en el momento de ingreso.
- Las bases de datos se hallan combinadas con el fin de lograr redundancia y respaldo.
- La información permanece en bases de datos o en directorios de forma encriptada, aunque la información entre el cliente y el servidor transita cifrada.
- Cuenta con un diseño SSO avanzado que utiliza bases de datos escalables que aguantan redundancia (MySQL, PostgreSQL, SQL Server, Oracle).

Ventajas:

- Posee múltiples bases de datos sincronizadas, con lo que se consigue la redundancia y el respaldo de la información.
- Los varios servidores administrativos reducen los peligros, aumentan la disponibilidad de la información y el tiempo de respuesta a los clientes.
- El disfrute de múltiples servidores adicionales hace que se disminuya la latencia en la red.
- Los agentes SSO se encuentran optimizados para múltiples ambientes (Terminal Server, Win32, GNU/Linux, Mac).

Desventajas:

- La solución es costosa debido a que se requiere recursos de alto costo.
- Retrasa la implementación técnica por interacción entre múltiples sistemas operacionales.
- El soporte y la administración se vuelve más complejo debido a la suma de recursos.

1.4.5 Administración y almacenamiento de credenciales centralizados garantizando alta disponibilidad y redundancia.

La arquitectura de administración y almacenamiento de credenciales centralizadas garantizando alta disponibilidad y redundancia, no pertenece a las arquitecturas anteriores, pues es una adaptación de la arquitectura de administración y almacenamiento de credenciales centralizadas y reúne algunas de las ventajas de la arquitectura totalmente distribuida.

Es provechoso la descripción y exhibición de la arquitectura debido a sus características, con la cual se adquieren ventajas de mayor disponibilidad, donde el *hardware* y el *software* están capacitados para situaciones de contingencia.

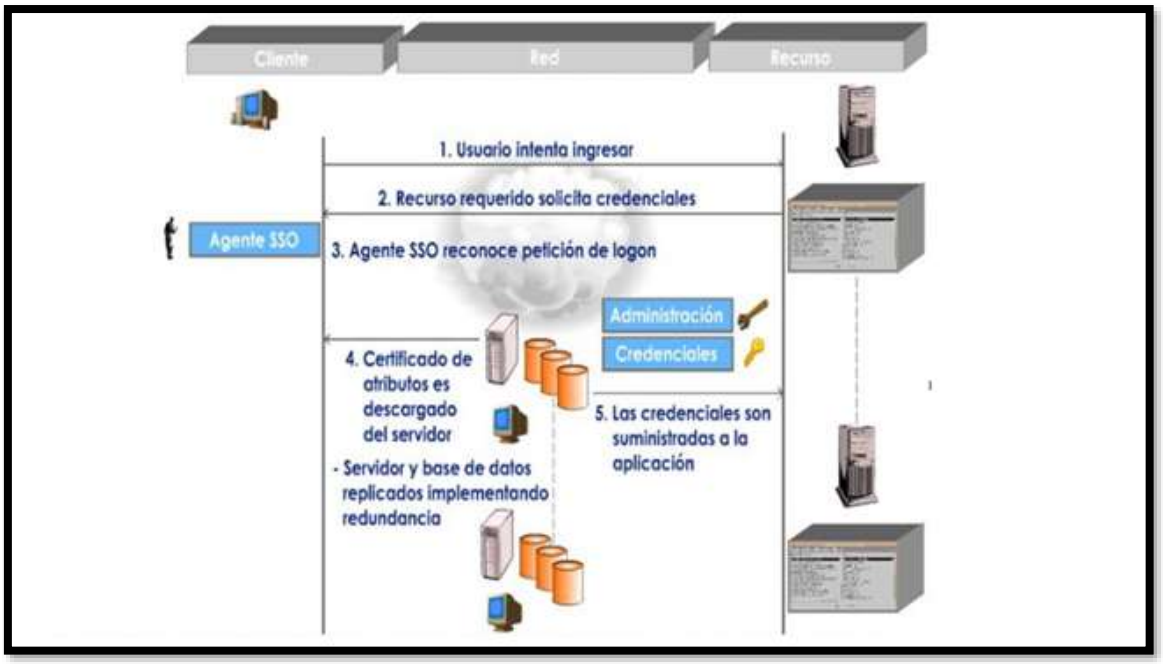

Figura 5.Administración y almacenamiento de credenciales centralizados garantizando alta disponibilidad y redundancia.

Características:

- El diseño se basa en administrar y almacenar las credenciales de forma centralizada el cual entrega un certificado al cliente y las credenciales son verificadas para la pertinente aplicación en el momento del ingreso.
- Brinda mayor disponibilidad mediante *software*.

- Concentra infraestructura replicada con el fin de manipular la contingencia y la redundancia en tiempo real.

Ventajas:

- Su infraestructura duplicada permite implementar alta disponibilidad y redundancia.
- Permite a los usuarios conectarse desde varias estaciones.
- El *hardware* como el *software* se encuentran en forma para afrontar una situación de contingencia.

Desventajas:

- El elevado costo de *software* y *hardware*, como de administración y control de sistema es debido a la alta disponibilidad y redundancia que brinda asentada en su infraestructura replicada.
- La información entre el cliente SSO y el servidor no viaja cifrada.

1.5 Aplicaciones exitosas que hacen uso de *Single Sign-On*.

Se puede decir que hoy en día existen muchas aplicaciones exitosas que utilizan la implementación de un SSO, ya que permite un acceso fácil a plataformas y aplicaciones a través de una interfaz de *login* única facilitando el trabajo de los usuarios e incrementando su productividad. A demás incrementa su nivel de seguridad usando un buen nivel de autenticación y el manejo de las contraseñas.

1.5.1 Google.

Google, es uno de los ejemplos de soluciones de seguridad debido a que se integra con sus sistemas de autenticación y autorización que brinda *Google Search Appliance* para proporcionar un modo seguro. En la actualidad, las compañías tienen un gran número de información muy importante y el servicio prestado por Google para indexar información restringida o de uso público habilita las políticas de seguridad para la información de la organización en el momento de la búsqueda. Además, proporciona a los usuarios un mecanismo por medio del cual pueden buscar información restringida de forma segura. La tecnología de búsqueda empresarial de Google se integra con sus sistemas de autenticación y autorización para proporcionar una manera de utilización segura a los usuarios.

Se puede decir que *Google Search Appliance* es un potente motor de búsqueda que rastrea el contenido y una vez encontrado la muestra de forma segura. Por lo general el motor de búsqueda puede configurarse para que guarde tanto el contenido público como protegido. Cuando el usuario accede a un contenido protegido, aparece un cuadro de dialogo en la sesión del navegador que solicita del usuario las credenciales necesarias, evento que tiene lugar una vez por sesión. El motor de búsqueda ha estructurado la autenticación que brinda el servicio en dos fases principales: rastreo/indexación y publicación. La fase de rastreo crea un índice de información que adquiere a través de los mecanismos de acceso a contenidos integrados.

Una vez que el servicio rastrea y adquiere la información, la aplicación utiliza credenciales de acceso facilitadas por el administrador del sistema. Estas pueden ser de inicio de sesión único (SSO) para sistemas basados en formularios donde la autenticación de implanta en entornos Web protegidos por el inicio de sesión única y está basado en formularios mediante solicitudes HEAD, cuyo contenido está basado en aplicaciones Web para un servidor Web. El motor de búsqueda puede utilizar tanto el reenvío de *cookies* como la suplantación absoluta del usuario. En ambos casos el motor captura la información de acceso en una *cookie* y la reenvía a los sistemas que se están rastreando. Para sistemas de credenciales de autenticación básica como las credenciales NTLM (nombre de usuario, contraseña y dominio) y certificados del cliente, por ejemplo, el X.509, funciona casi en todas implementaciones del servidor Web que son compatibles al menos con HTTP/1.0. También es admitido por servidores basados en el sistema operativo (SO) Microsoft. Generalmente si las credenciales de un usuario residen en un SO Microsoft y se utiliza NTLM, el motor de búsqueda podrá aprovecharlas.

El proceso de publicación tiene lugar cuando los usuarios realizan una consulta y en ese momento puede especificar, a través de la interfaz de búsqueda, si quiere buscar información de carácter público o privilegiado. Si el usuario se decide por la privilegiada, el sistema le solicita al usuario sus credenciales de acceso basándose en los métodos de autenticación y autorización configurados, es decir, como ya la aplicación está integrada por un inicio de sesión única (SSO) el usuario es dirigido directamente al servidor de autenticación en el cual se registrará y luego el sistema se encargará de darle las credenciales correspondientes para poder realizar la búsqueda y si el usuario ya está autenticado podrá continuarla sin necesidad de autenticarse.

El sistema realiza la búsqueda en el índice para obtener una lista de los posibles resultados que coincidan con lo que se está buscando. Sin embargo, antes de mostrar la lista de los resultados obtenidos, la aplicación utiliza la *cookie* de SSO en nombre del usuario para autorizar los resultados ante el sistema de origen. Los que no superan la autorización se excluyen de la lista, y únicamente se muestran los usuarios aquellos que hayan sido validados y el sistema asegura que solo puedan verlos los usuarios que están correctamente registrados.

El sistema de búsqueda *Google Search Appliance* cumple con estándares abiertos establecidos. La aplicación puede utilizar un gran número de mecanismos en el sector de la autenticación de usuarios, incluyendo también la autenticación basada en LDAP y *Active Directory*, sistema de acceso único (SSO) basados en formularios, como *Netegrity* y *Oblix,* y certificados de un cliente, digamos el X.509. La autorización proporciona a la aplicación el nivel del sistema-origen del documento, utilizando de nuevo el sistema SSO empresarial, autenticación HTTP básica o autorización basada en NTLM. También es compatible con interfaces que tiene su base en SAML para la autenticación y la autorización. La interfaz de proveedor basada en SAML 2.0 utiliza los estándares XML emergentes para permitir la autenticación de terceros, así como la autorización de resultados externos. Con las interfaces de proveedor de servicios de autorización y autenticación, *Google Search Appliance* se puede integrar de forma fácil y segura a todos los entornos de control de acceso empresarial. Además, proporciona seguridad a las empresas con la facilidad de integrarlas de inmediato después de haber implantado el sistema. (6)

Al sistema de búsqueda *Google Search Appliance* no le hace falta la creación de identidades de usuarios nuevos o listas de control de acceso (ACL) para implementar una búsqueda segura en la empresa. En su lugar Google utiliza el sistema de gestión de identidades existentes y las políticas de control de acceso de que ya dispone en sus sistemas de contenido.

1.5.2 Windows Live ID.

La plataforma .NET contiene los cimientos para la nueva generación del software, ella utiliza los servicios Web como medio para poder interactuar con distintas tecnologías que permiten conectar diversos sistemas operativos, dispositivos físicos, información y usuarios. La idea central de la plataforma .NET es proveer servicios, a los cuales se puede acceder desde Internet y su capacidad de distribución para los usuarios accedan desde cualquier

dispositivo, en cualquier sistema operativo y lugar por la funcionalidad que los servicios Web proveen. Unos de los componentes más importantes que lo integran, provistos por Microsoft es .NET Passport o Windows Live ID, como se denomina actualmente, que es un conjunto de servicios Web. (7)

Desde 1999 *Windows Live ID* permite que un usuario se autentique una sola vez y luego accede a los sitios que hacen uso del servicio sin necesidad de volver a identificarse. Estos sitios pueden tener en cuenta datos del usuario como su perfil de usuario para adaptar la interfaz o proveer los servicios específicos. El sitio debe conectarse al servicio Web .NET Passport y hacer llamadas a componentes del servicio .NET Passport para hacer uso de un servicio. El usuario, por su parte, debe tener una cuenta en Hotmail o MSN y haber sido autenticado por el servicio .NET Passport. Una vez hecho esto, él puede navegar por distintos sitios y si éstos utilizan .NET Passport entonces el usuario no tiene necesidad de identificarse o de ingresar sus datos personales.

1.5.3 OpenID.

Hoy en día se sabe que en Internet se encuentran aplicaciones Web que utilizan OpenID que permanecen accesibles al servicio prestado por Internet, por esto elegir un método de autenticación y autorización de los usuarios a dichas aplicaciones y servicios es siempre un factor determinante en la seguridad de los sistemas informáticos.

El proyecto OpenID no es más que un estándar que posee un sistema descentralizado de información, él permite a los sitios Web que le brindan soporte el acceso a los usuarios sin tener necesidad de crear su propia cuenta. Simplemente se necesita un proveedor de identidad (IdP), un ejemplo de esto es Yahoo.

Este estándar OpenID es un sistema de autenticación seguro, libre y muy fácil de usar, ya que facilita la identificación en muchas páginas Web con solo introducir la dirección URL de OpenID, de forma que solo mantendrá una ID global para los accesos a Web con soporte OpenID, que es un registro rápido y único, pues posee la facilidad de autenticarse una sola vez y tener acceso a muchas aplicaciones tipo Web.

Se puede decir que el usuario en el sistema de autenticación crea un identificador en un servidor que verifique OpenID y el proveedor de identidad confirma la identificación de los usuarios a los sitios que soporten este sistema. Seguidamente el usuario, después de haberse registrado, obtiene la URL que él puede usar para identificarse, además si posee

una página Web puede incluir unas etiquetas en ella y usar su propio dominio como código de identificación.

La utilización de OpenID implica una arquitectura *Single Sign-On*, que no especifica el mecanismo de autenticación por lo que la seguridad de la conexión dependerá de la confianza del cliente en el proveedor de identidad. Se puede destacar que según sea la implementación de este ofrece una autenticación segura y mayor seguridad en el sistema. (8)

1.6 Servidor Central de Autenticación (*Central Authentication Server*):

Central Authentication Service (CAS) es una aplicación web que nos permite implementar el conocido *Single Sign-On (SSO)* que es un procedimiento de autenticación que habilita a un usuario para acceder a distintas aplicaciones web en diferentes dominios y en diversos servidores con autenticarse una única vez. En general, cuando un usuario se conecta a una de estas aplicaciones el sistema comprueba si esta autenticado y si no lo está, lo redirecciona a la pantalla del servidor de autenticación. Si es correcto el sistema, en este caso CAS, vuelve a redirigir al usuario a la página a la que quería acceder en un primer momento. El CAS se encarga única y exclusivamente de la autenticación, es decir, de comprobar contra una fuente de datos específica si el usuario y contraseña facilitados existen, no se encarga de la autorización, que sería la gestión de lo que puede o no puede hacer ese usuario en función de sus roles.

1.6.1 Funcionamiento.

Cuando el usuario esta autenticado, el usuario puede moverse de una aplicación a otra sin necesidad de autenticarse de nuevo, simplemente se envía la *cookie* que tiene guardado el usuario en su navegador hacia el servicio CAS, el CAS valida la *cookie* y lo redirecciona a la aplicación con un *ticket* único, la aplicación solicita la validación del *ticket* y obtiene el nombre del usuario.

1.6.2 Ventajas.

- El servidor CAS hace un *logout* en todas las aplicaciones. Simplemente invalida al usuario, realiza una petición de *logout* en todas las demás aplicaciones donde ha entrado el usuario (utilizando *tickets*) y lo redirecciona al *login* del CAS.

- Acepta conexiones de servidores en modo Proxy. El CAS permite que un servidor pueda realizar una acción en representación del CAS. Con ello un servidor proxy puede solicitar un *ticket* para un usuario autenticado.

Para proporcionar seguridad al Sistema Reservación de Alimentos UCI, se estudió la manera de utilizar el servicio de autenticación CAS. Este servicio es utilizado en varias aplicaciones de todo el mundo y la idea es poder integrarlo a nuestro proyecto. (9)

1.7 Tecnologías y herramientas utilizadas.

Se decide trabajar sobre herramientas libres debido al principal propósito de fomentar el desarrollo del Software Libre en la Universidad de las Ciencias Informáticas.

1.7.1 Lenguaje de programación. Java

Java es un lenguaje compilado de programación orientado a objetos. También se puede decir que Java es una tecnología que no solo se reduce al lenguaje, sino que además provee de una máquina virtual Java que permite ejecutar código compilado en Java, sea cual sea la plataforma que exista por debajo; plataforma tanto *hardware* como *software* y siendo esta la que mantiene el control sobre las clases que se estén ejecutando. Además de ser un lenguaje multiplataforma, simple, robusto, seguro y de arquitectura neutral. (10)

1.7.2 Servidor Web. Apache Tomcat.

El servidor HTTP Apache Tomcat es un contenedor de servlets que se utiliza en la Referencia oficial de la implementación para Java Servlets y Java Server Page(JSP). Las especificaciones Java Servlets y Java Server Page son desarrolladas por Sun Microsystems cuyas especificaciones vienen dadas por la JCP (*Java Community Process*). Apache Tomcat es desarrollado en un entorno abierto y participatorio, bajo la licencia de Apache Software License. (11)

Ventajas del uso de Tomcat:

- Servidor de aplicaciones *open source*.
- Fácil integración con Apache HTTP Server.
- No requiere mucha memoria para arrancar.

1.7.3 Entorno de desarrollo integrado. Eclipse

Es un entorno de desarrollo integrado capaz de proveer un conjunto de herramientas para administrar espacios de trabajo, construir, correr y depurar aplicaciones, además posee una arquitectura de módulos llamados *plug-ins* que le permite incorporarle múltiples funcionalidades, como lo es la instalación de un *plug-ins* para la creación de *portlets.* Es un entorno de desarrollo diseñado para ser extendido con sofisticadas herramientas. (12)

1.8 Frameworks de desarrollo:

Es una estructura de soporte definida en la cual otro proyecto de *software* puede ser organizado y desarrollado. Puede ser considerado como el conjunto de procesos y tecnologías usados para resolver un problema complejo. Es el esqueleto sobre el cual varios objetos son integrados para una solución dada.

1.8.1 Spring

Spring es *framework* para desarrollar aplicaciones empresariales Java. La ventaja de utilizar Spring por encima de otros *framework* es que es de código abierto. Lo que figura que los desarrolladores pueden montar código reutilizable sin dependencia de un distribuidor. (13)

Características:

- Spring gestiona y contiene la configuración y ciclo de vida de aplicaciones de los objetos.
- Cuando se trata de claridad y tamaño, Spring es un *framework* de aplicaciones ligero.
- Utiliza el estilo arquitectónico modelo-vista-controlador (MVC).

Ventajas:

- Proporciona la habilidad de integrarse con otros *framework* para que el desarrollador obtenga los beneficios que desea.
- Brinda distintos módulos de acuerdo a la herramienta o *framework* que se utilicen.
- La complicación de la aplicación es proporcional a la complejidad del problema que se está solucionando.

Desventajas:

- Es preciso escribir numeroso código dentro de los archivos JSP, independientemente de los *tags* que brinda Spring.
- En muchos casos es necesario utilizar diversos archivos XML, lo que es inevitable para configurar ciertos aspectos del *framework*.

1.9 Framework Mapeo Objeto Relacional (ORM).

Es una técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y el utilizado en una base de datos relacional. Esto posibilita el uso de las características propias de la orientación de objetos (básicamente herencia y polimorfismo). (14)

1.9.1 Hibernate.

Hibernate tiene como objetivo facilitar la persistencia de objetos Java en base de datos relacionales y al mismo tiempo la consulta de estas bases de datos para obtener objetos, a través de archivos (XML) o con anotaciones en las entidades. Además, resuelve el conflicto de la diferencia entre los dos modelos de datos simultáneos en una aplicación: el usado en la memoria (orienta a objeto) y el usado en la base de datos (modelo relacional), es flexible en cuanto al esquema de tablas usadas. Realiza la función de introducir la base de datos a partir de la información utilizable. Permite diseñar objetos firmes que podrán incluir un gran número de tipos de datos. Presenta gran escalabilidad e implementa la gestión de la API de la persistencia Java y el mapeado objeto-relacional. (15)

1.10 Sistema Gestor de Bases de Datos (SGBD).

Un Sistema Gestor de Base de Datos es un sistema de software que permite la definición de bases de datos; así como la elección de las estructuras de datos necesarios para el almacenamiento y búsqueda de los datos, ya sea de forma interactiva o a través de un lenguaje de programación. Un SGBD relacional es un modelo de datos que facilita a los usuarios describir los datos que serán almacenados en la base de datos junto con un grupo de operaciones para manejar los datos. (16)

1.10.1 PostgreSQL

Es un sistema gestor de base de datos objeto-relacional, bajo licencia BSD (*Berkeley Software Distribution*). Esta licencia tiene menos restricciones en comparación con otras

como la GPL (Licencia Pública General) estando muy cercana al dominio público. La licencia BSD al contrario que la GPL permite el uso del código fuente en software no libre. Es el sistema de gestión de bases de datos de código abierto más avanzado del mundo y en sus últimas versiones posee muchas características que solo se podían ver en productos comerciales de alto calibre. (17)

Características de PostgreSQL:

Atomicidad (Indivisible): es la propiedad que asegura que la operación se ha realizado o no, y por lo tanto ante un fallo del sistema no puede quedar a medias.

Consistencia: es la propiedad que asegura que sólo se empieza aquello que se puede acabar. Por lo tanto, se ejecutan aquellas operaciones que no van a romper la reglas y directrices de integridad de la base de datos.

Aislamiento: es la propiedad que asegura que una operación no puede afectar a otras. Esto asegura que dos transacciones sobre la misma información nunca generarán ningún tipo de error.

Durabilidad: es la propiedad que asegura que, una vez realizada la operación, esta persistirá y no se podrá deshacer, aunque falle el sistema.

1.11 Herramienta CASE.

Son diversas Aplicaciones informáticas destinadas a aumentar la productividad en el Desarrollo de software reduciendo el coste de las mismas en términos de tiempo y dinero. Estas herramientas nos pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del software en tareas como el diseño de proyectos, cálculos de costes, implementación de parte del código automáticamente con el diseño dado, Compilación automática, documentación o detección de errores entre otras. (18)

1.11.1 Visual Paradigm for UML:

Es una herramienta multiplataforma que utiliza UML como lenguaje de modelado. Soporta el ciclo de vida completo del *software* (análisis y diseño orientados a objetos, construcción, pruebas y despliegue). Permite realizar todos los tipos de diagramas de clases, ingeniería inversa, generar código desde diagramas, entre muchas otras funcionalidades. Presenta una excelente documentación la herramienta UML CASE basada en abundantes tutoriales de UML, demostraciones interactivas y proyectos UML.

Visual Paradigm ofrece un diseño enfocado al negocio, permitiendo así generar un software de mayor calidad. Presenta una licencia *Open Sour*ce a diferencia de su homólogo *Rational Rose*.

Se caracteriza por:

- Disponibilidad en múltiples plataformas.
- Diseño centrado en casos de uso y encaminado al negocio.
- Uso de un lenguaje estándar común a todo el equipo de desarrollo. (19)

1.12 Metodología a usar:

El Proceso Unificado Ágil (*Agile Unified Process*, más conocido como AUP) es una versión simplificada del Proceso Unificado Racional (RUP). Este describe una manera simple y fácil de entender la forma de desarrollar aplicaciones de software de negocio utilizando técnicas ágiles y conceptos que aún se mantienen validos en RUP.

En AUP se establecen cuatro fases que transcurren de manera consecutiva y que acaban con hitos claros alcanzados. El objetivo es obtener una comprensión común cliente-equipo de desarrollo del alcance del nuevo sistema.

- Elaboración: el objetivo es que el equipo de desarrollo profundice en la comprensión de los requisitos del sistema y validar la arquitectura.
- Construcción: durante la fase de construcción el sistema es desarrollado y probado al completo en el ambiente de desarrollo.
- Transición: el sistema se lleva a los entornos de preproducción donde se somete a pruebas de aceptación y validación y finalmente se despliega en los sistemas de producción. (20)

En general, el Proceso Unificado Ágil supone un enfoque intermedio entre XP (*Extreme Programming*) y el Proceso Unificado Racional, y tiene la ventaja de ser un proceso ágil que incluye explícitamente actividades y artefactos a los que la mayoría de desarrolladores ya están, de alguna manera, acostumbrados.

1.13 Conclusiones parciales.

En el capítulo se expusieron características y conceptos importantes que se deben tener presentes durante todo el ciclo de vida del producto, así como las herramientas y tecnologías que serán utilizadas, seleccionando la Metodología Proceso Unificado Ágil

(AUP) para el desarrollo, empleando sistemas de autenticación centralizada CAS y como mecanismos de autenticación SSO, utilizando Java como lenguaje de programación y como entorno de desarrollo Eclipse por las características que presenta. Como sistema gestor de base de datos PostgreSQL es el seleccionado; siendo Visual Paradigm elegido para el modelado.

CAPÍTULO 2: DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA

2.1 Introducción

Durante el desarrollo de este capítulo se definen los requisitos funcionales y no funcionales, se dará a conocer la propuesta del sistema y los diagramas para apoyar la comprensión del funcionamiento del mismo, se define la arquitectura que se usará en la implementación del sistema de autenticación centralizada SSO y se describen las necesidades del mismo mediante casos de uso.

2.2 Objeto de Automatización

El sistema CAS garantiza el uso de herramientas para un proceso fácil y fluido, de forma que el usuario no tenga que hacer ningún tipo de trámite para acceder de uno a otro sitio Web, haciendo la navegación más agradable. Todo incurre en un mayor grado de seguridad, confidencialidad, coherencia y portabilidad en el sistema creado.

2.3 Propuesta de solución

La Universidad de las Ciencias Informáticas UCI actualmente no cuenta con un mecanismo de autenticación centralizada que les permita a los usuarios tener acceso a todos los sistemas web disponibles.

Para dar solución a esta problemática se propone la implementación de un mecanismo de autenticación centralizada SSO haciendo uso del CAS, lo cual permitirá a los usuarios autenticarse en un sistema y tener acceso al resto de las aplicaciones disponibles en la UCI. La solución se verá en funcionamiento mediante la integración del Sistema de Reservación de Alimentos UCI con el CAS.

2.4 Modelo de Dominio del SSO.

El sistema de autenticación y control centralizado se desarrolla en el ambiente de las aplicaciones y servicios con que cuenta la infraestructura del sistema Reservación de Alimentos UCI. El sistema de autenticación y control centralizado es un sistema que permite al usuario autenticarse una sola vez, además proporciona un mecanismo de autenticación que permite el acceso a los usuarios a múltiples aplicaciones, así como también establecer un componente de seguridad y un alto nivel de confiabilidad a los usuarios, a las aplicaciones y a los servicios que se ejecutan en el sistema.

El sistema presenta la aplicación Reservación de Alimentos UCI que puede hacer uso de él, para que cuando haga uso del sistema el automáticamente dejarla usar los servicios que posee el sistema. Ya una vez que la aplicación hace uso del sistema, esta la redirecciona al usuario a un formulario de autenticación para poder hacer uso de la aplicación.

Este acceso se hace por medio de la inserción de un *login* especificado y un *password* que a su vez debe validarse. Una vez registrado el usuario y después de haberse validado el registro y contraseña del usuario, el sistema le da un *token* de seguridad y es almacenado junto con las credenciales del usuario. El sistema devuelve a la aplicación el *token* de seguridad junto con el rol o roles que le fueron asignados al usuario para así él saber hasta donde el usuario tiene los permisos necesarios para hacer uso de los servicios que brinda la aplicación. (21)

En caso de que el usuario ya esté autenticado, el sistema verifica que el usuario esté registrado, de ser así entonces el sistema comprueba que exista una sesión abierta y si es cierto, el sistema valida el *token* de seguridad que le fue enviado a la aplicación comprobando que el *token* de la aplicación coincida con los datos almacenados. Después de esto el sistema autoriza a la aplicación hacer uso de él como se muestra en la figura.

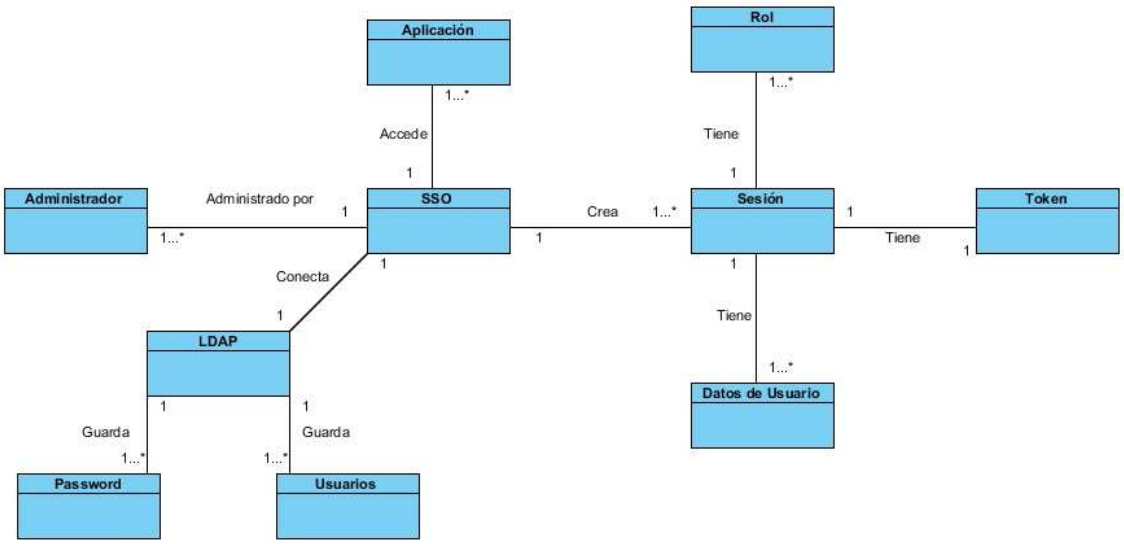

Figura 6.Modelo de Dominio del SSO.

2.5 Descripción del Modelo de Dominio.

El usuario accede al Sistema de Reservación de Alimentos UCI, y este lo redirecciona hacia el CAS, el CAS muestra la interfaz de autenticación para que el usuario se autentique en el

sistema, luego de ingresar sus credenciales, el CAS envía los datos al LDAP y una vez verificados los datos del usuario, el CAS envía un tique al sistema, el cual será almacenado con los datos y las cookies del navegador, para luego poder crear secciones de autenticación para ese mismo usuario en diferentes aplicaciones. Luego de que el sistema recibe el tique, consulta los datos del usuario que son necesarios para el funcionamiento del sistema que se encuentran almacenados en el LDAP.

2.6 Especificación de requisitos de software.

El flujo de especificación de los requisitos de software radica su mayor esfuerzo en el reconocimiento del problema como lo ve el usuario. Se define una evaluación del problema y se dan respuestas a las funcionalidades que tendrá el sistema. Con él se pretende entender el comportamiento del software y se establecen las características de la interfaz y el descubrimiento de restricciones adicionales de diseño. (22)

2.6.1 Requerimientos funcionales:

Los requerimientos funcionales son capacidades o condiciones que el sistema debe cumplir, no alteran la funcionalidad del producto y se mantienen invariables sin importar con qué propiedades o cualidades se relacionen. Son el punto de partida para identificar qué debe hacer el sistema.

A continuación, se describen los requisitos funcionales del sistema.

No.	Nombre	Descripción	Complejidad
[RF1.]	Acceder al CAS mediante el Sistema de Reservación de Alimentos.	El sistema debe permitir una vez que el usuario accede a la aplicación direccionarlo para el CAS y mostrar una interfaz de autenticación.	Alta
[RF2.]	Autenticar usuario en el sistema.	El sistema debe permitir que el usuario se autentique mediante la interfaz del CAS.	Alta
[RF3.]	Validar datos del usuario en el LDAP.	El sistema debe permitir que el CAS envié los datos del usuario autenticado para ser verificados en el LDAP.	Alta

[RF4.]	Registrar usuario en el Sistema de Reservación de Alimentos UCI.	El sistema debe permitir que luego de ser validadas las credenciales del usuario, este pueda acceder al Sistema de Reservación de Alimentos UCI.	Alta

Tabla 1.Requisitos funcionales.

2.6.2 Requisitos no funcionales.

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener. Representan las características que hacen el producto atractivo, rápido o confiable, son fundamentales en el éxito del producto.

A continuación, se describen los requisitos no funcionales del sistema.

- **Apariencia o interfaz externa.**
  - ✓ Diseño sencillo, permitiendo la utilización del sistema sin mucho entrenamiento.
  - ✓ Diseño adaptable
- **Soporte.**
  - ✓ Garantía de instalación y prueba del sistema, además de un breve entrenamiento a los futuros usuarios.
- **Implementación.**
  - ✓ Usar Java como lenguaje de programación.
- **Confiabilidad.**
  - ✓ Garantía de un tratamiento adecuado de las excepciones y validación de las entradas del usuario.

2.7 Definición de los casos de uso:

Los casos de uso son artefactos narrativos que describen, bajo la forma de acciones y reacciones el comportamiento del sistema desde el punto de vista del usuario. Por lo tanto, establece un acuerdo entre clientes y desarrolladores sobre las condiciones y posibilidades (requisitos) que debe cumplir el sistema. (23)

2.8 Definición de los actores:

Los actores del sistema son los trabajadores del negocio que tiene actividades a automatizar es un candidato a actor del sistema. Si algún actor del negocio va a interactuar con el sistema, entonces también será un actor del sistema.

Los actores del sistema:

- No son parte de él.
- Pueden intercambiar información con él.
- Pueden ser un recipiente pasivo de información.
- Pueden representar el rol que juegan una o varias personas, un equipo o un sistema automatizado.

Actores del sistema	Descripción
Usuario	Generaliza a todos los usuarios del sistema. Realiza las operaciones comunes a ellos.

Tabla 2.Definición de los actores.

2.9 Diagramas de casos de uso del sistema.

Un diagrama de casos de uso del sistema contiene los actores y los casos de uso del sistema, mostrando las diferentes relaciones que existen entre ellos. Son fragmentos de la funcionalidad del software, en ellos se describe la secuencia determinada que sigue un actor en su interacción con el sistema.

2.10 Diagrama de casos de uso. Acceder CAS.

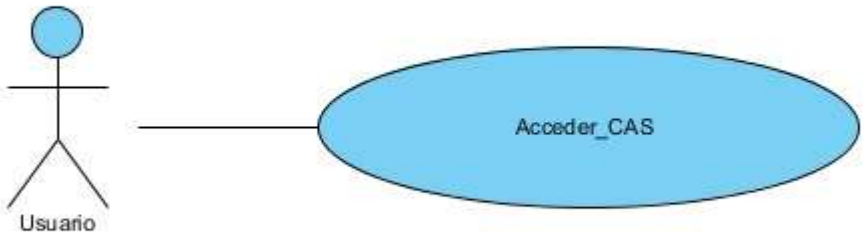

Figura 7.Diagrama de casos de uso. Acceder CAS.

CU-1	**Acceder al CAS mediante el Sistema de Reservación de Alimentos.**
Actores	Usuario
Propósito	Mostrar interfaz de autenticación CAS.
Resumen: el caso de uso inicia cuando el usuario ingrese al sistema Reservación de Alimentos UCI y se muestre la interfaz de login del servidor CAS.	
Referencia	RF1
Pre-condiciones	
Post-condiciones	
Flujo de trabajo	
Acción del Actor	**Respuesta del sistema**
1-El usuario accede al sistema Reservación de Alimentos UCI.	2-El sistema muestra una interfaz de login del servidor CAS.
Curso alternativo de los eventos.	

Tabla 3. Acceder CAS.

2.10.1 Diagrama de casos de uso. Autenticación.

Figura 8.Diagrama de casos de uso. Autenticación

CU-2	**Autentificar Reservación de Alimentos**
Actores	Usuario
Propósito	Autentificar usuario
Resumen: el caso de uso inicia cuando el usuario ingresa su identificador y contraseña en el sistema de Reservación de Alimentos, concluye con los datos introducidos correctamente o un mensaje de error en caso contrario.	
Referencia	R2
Pre-condiciones	El usuario debe haber sido creado.
Post-condiciones	El usuario es autentificado.
Flujo de trabajo	
Acción del Actor	**Respuesta del sistema**
1-El usuario Reservación de Alimentos ingresa a la aplicación. 3-El usuario Reservación de Alimentos introduce sus datos e envía la información. El usuario Reservación de Alimentos accede al sistema. 5-El usuario Reservación de Alimentos accede al sistema.	2-El sistema muestra una ventana dándole al usuario la posibilidad de ingresar sus datos. 4-El sistema procesa los datos ingresados por el usuario y le asigna un token de sesión después de verificar que esté registrado en la base de datos de Reservación de Alimentos y en el LDAP, asignándole el permiso.

Curso alternativo de los eventos.	
	➢ Si el usuario no está registrado en la base de datos Reservación de Alimentos y si en la central (LDAP), lo registra en Reservación de Alimentos y le permite el acceso. ➢ Si el usuario no está registrado en la base de datos Reservación de Alimentos ni en LDAP, le muestra un mensaje de error informando que los datos son incorrectos, dando la oportunidad de volver a ingresar los datos.

Tabla 4.Autenticar usuario Reservación Alimentos UCI

2.10.2 Diagrama de casos de uso. Validar.

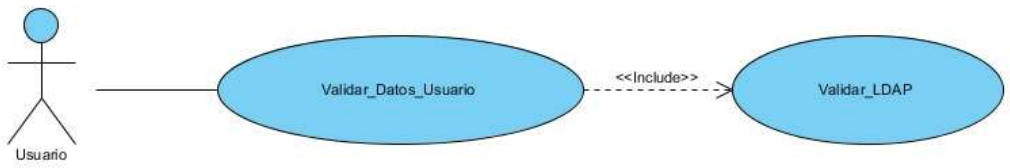

Figura 9.Diagrama de casos de uso. Validar

CU-3	Validar datos de usuario en el LDAP
Actor	Aplicación
Propósito	Comprobar que la aplicación tenga los permisos correspondientes para hacer uso del sistema.
Resumen: el caso de uso inicia cuando la aplicación le solicita al sistema de autenticación validar un token de seguridad, el sistema chequea que la aplicación	

tenga permisos para usar el sistema de autenticación, el sistema elimina todas las sesiones que hayan expirado o no sean válidas, el sistema chequea que el token exista y devuelve una respuesta de si (login de usuario correspondiente a la sesión) o no.	
Referencia	RF-3
Pre-condiciones	La aplicación tenga permisos para usar el sistema.
Post-condiciones	El sistema valida el token de seguridad y brinda los permisos.
Flujo de trabajo	
Acción del Actor	**Respuesta del sistema**
1- La aplicación le solicita al sistema validar un token de seguridad.	2-Se ejecuta el caso de uso incluido (verificar permisos) 3-El sistema chequea que la aplicación tenga una sesión abierta 4-El sistema elimina todas las sesiones que hayan expirado o que no sean válidas. 5-El sistema chequea que el token de seguridad exista. 6-El sistema devuelve una respuesta afirmativa.
Curso alternativo de los eventos	
	➢ El sistema muestra un mensaje de error. ➢ El sistema muestra un mensaje de error en caso de que la sesión no esté abierta. ➢ El sistema muestra un mensaje de error en caso de que el token de seguridad no exista.

Tabla 5.Validar datos de usuario

2.10.3 Diagrama de casos de uso. Registrar.

Figura 10.Diagrama de casos de uso. Registrar

CU-4	**Registrar usuario Reservación de Alimentos UCI.**
Actor	Usuario
Propósito	Registrar usuario en la base de datos.
Resumen: El caso de uso inicia cuando el usuario ingresa a la aplicación y registra sus datos. El sistema se asegura que los datos estén correctos y que no estén registrados anteriormente en la base de datos de LDAP. De cumplirse esto el usuario adiciona un nuevo registro en la base de datos.	
Referencia	R4
Pre-condiciones	El usuario no debe estar registrado en la base de datos LDAP.
Post-condiciones	
Flujo de trabajo	
Acción del Actor	**Respuesta del sistema**
1-El usuario ingresa en la aplicación. 3-Ingresa los datos.	2-El sistema muestra una elección para registrar un usuario. 4-Verifica los datos y se asegura que no esté registrado en la base de datos LDAP. 5-El sistema registra al usuario en la base de datos de Reservación de Alimentos y le envía un aviso de la activación de la cuenta del usuario.

6-Se le otorga al usuario un aviso y realiza la activación de su cuenta.	
Curso alternativo de los eventos	
	➢ La aplicación recoge los datos y si están presentes en la base de datos LDAP muestra un mensaje de error.

Tabla 6. Registrar usuario Reservación Alimentos UCI.

2.11 Arquitectura del sistema.

Para el desarrollo de la solución se decidió utilizar la arquitectura SSO de administración y almacenamiento de credenciales centralizadas garantizando alta disponibilidad y redundancia, ya que se ajusta más a las necesidades del mecanismo de autenticación centralizada a implementar.

2.12 Patrones arquitectónicos utilizados.

El Modelo Vista Controlador (MVC) es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos.

El patrón MVC se ve frecuentemente en aplicaciones Web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página, el modelo es el Sistema de Gestión de Base de Datos y el controlador representa la lógica de negocio. (24)

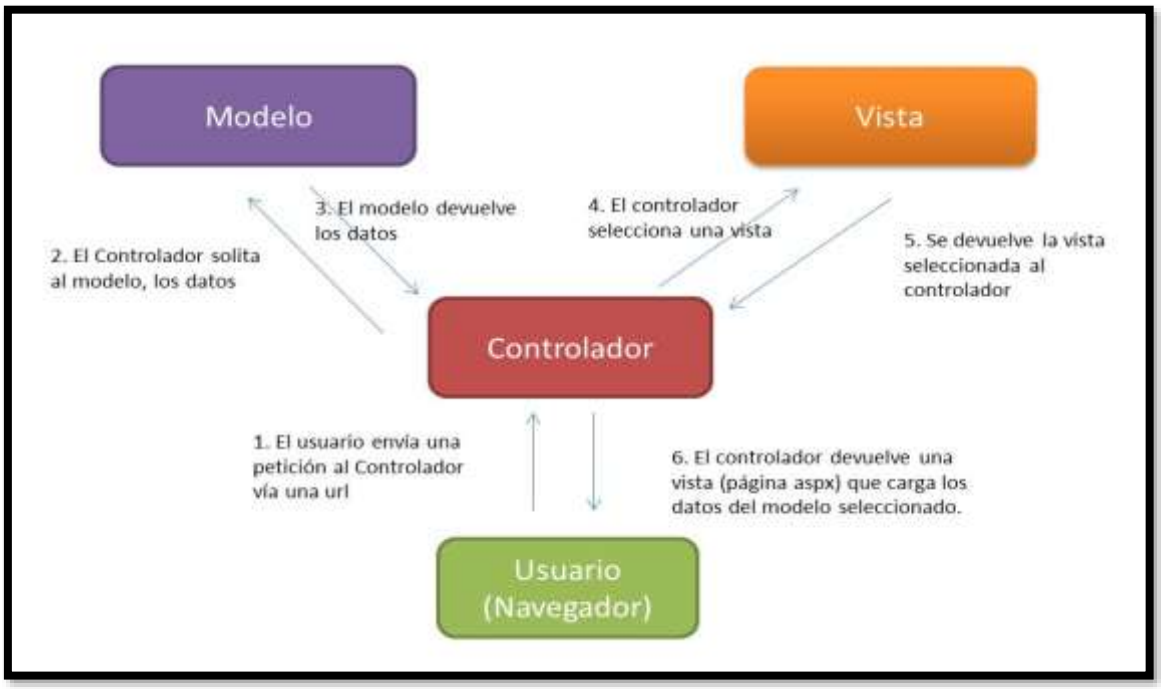

Figura 11.Modelo Vista Controlador (MVC).

2.13 Componentes utilizados en la elaboración de la propuesta.

2.13.1 Usuario:

Los usuarios pueden acceder al sistema desde dispositivos tradicionales (pc).

2.13.2 Servidor de Aplicaciones (Tomcat):

El servidor de aplicaciones tendrá alojado el sistema, el cual consumirá información de servidores universitarios, como es el caso del servidor LDAP.

2.13.3 Protocolo Ligero de Acceso a Directorios (LDAP):

Proporciona un repositorio único y centralizado para todas las aplicaciones (*single login*) posibilitando el acceso a la información de usuarios, grupos, perfiles, directorio de servicios y entidades. Ofrece el acceso más eficiente a los datos del directorio en comparación con estructuras basadas en Sistemas de Gestión de Bases de Datos Relacionales (RDBMS). Garantía de transportabilidad futura de la información almacenada en LDAP al tratarse de un estándar con amplia implantación. (25)

2.13.4 Servidor Central de Autenticación (*Central Authentication Server*):

La función del CAS implementado en el sistema Reservación de Alimentos UCI es única y exclusivamente de autenticación.

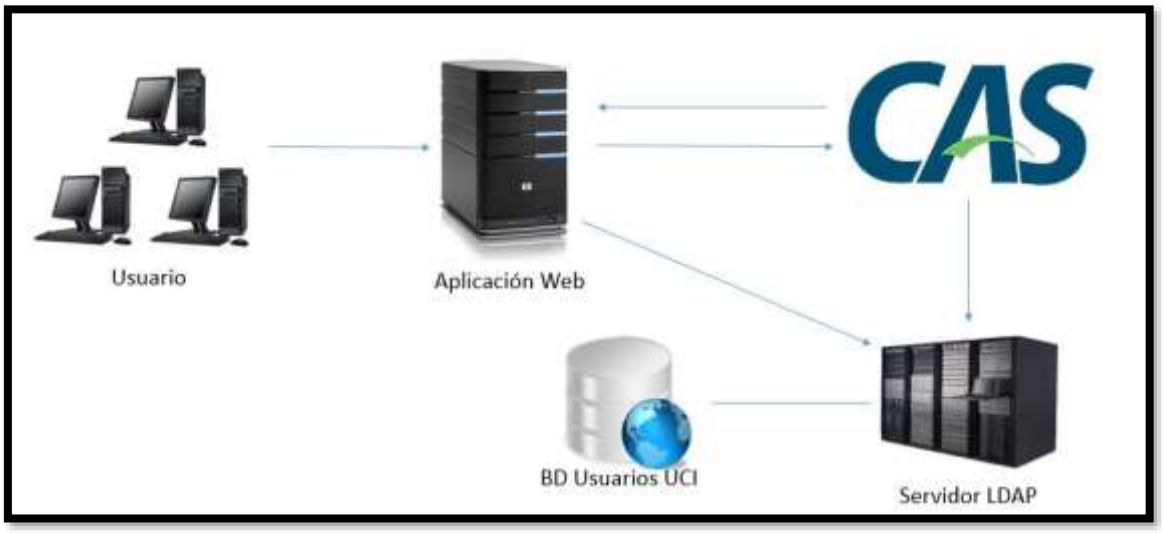

Figura 12. Servidor Central de Autenticación.

2.14 Características de la solución propuesta.

La implementación del mecanismo de autenticación centralizada tendrá las características de un sistema de autenticación única que le permitirá a los usuarios de la UCI fácil acceso a las aplicaciones y a la información de forma segura, para lograr este objetivo se ofrece este mecanismo como valor añadido que inicialmente se integrará el Sistema de Reservación de Alimentos UCI con el CAS y en un futuro cercano el resto de las aplicaciones web de la universidad.

2.15 Conclusiones parciales.

El desarrollo de este capítulo facilita el análisis y comprensión del mecanismo propuesto. Se describen las tecnologías y las herramientas empleadas en el proceso de desarrollo del sistema, la metodología que guía la construcción de la solución y la arquitectura a emplear en el sistema, la cual posibilita que el sistema sea fácilmente corregible e integrable con otros sistemas.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBAS

3.1 Introducción.

Para la realización del CAS personificado del Sistema de Reservación de Alimentos, fue necesario crear la arquitectura del mismo estableciendo un modelo de análisis y diseño que responda a las necesidades del cliente registradas en la previa fase de captura de requisitos.

3.2 Modelo de análisis del SSO.

El modelo de análisis contiene clases de análisis y sus objetos organizados en paquetes que colaboran.

Las clases de análisis se centran en los requisitos funcionales y son evidentes en el dominio del problema porque representan conceptos y relaciones del dominio. Tienen atributos y entre ellas se establecen relaciones de asociación, agregación / composición, generalización / especialización, y tipos asociativos. (26)

AUP propone clasificar las clases en:

- Clases de interfaz: Modelan la interacción entre el sistema y sus actores.
- Clases de entidad: Modelan información que posee larga vida y que es a menudo persistente.

Clases de control: Coordinan la realización de uno o unos pocos casos de usos coordinando las actividades de los objetos que implementan la funcionalidad del caso de uso.

Los Diagramas de Clases del Análisis están organizados en cuatro paquetes, para mayor claridad, organización y comprensión, puesto que así se concibieron previamente (Diagrama de casos de uso del paquete de autentificación, Diagrama de casos de uso del paquete de actualización, Diagrama de casos de uso del paquete de registro y Diagrama de casos de uso del paquete de administración) los casos de uso del sistema que intervienen la creación del SSO:

- Diagrama de casos de uso del paquete de autentificación.
- Diagrama de casos de uso del paquete de actualización.
- Diagrama de casos de uso del paquete de registro.
- Diagrama de casos de uso del paquete de administración.

3.3 Diagrama de clases del análisis de los casos de uso acceder.

- Diagrama de clase del análisis del caso de uso “Acceder-CAS”.

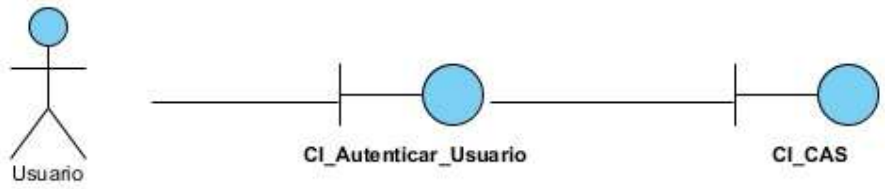

Figura 13. Diagramas de clases del análisis de los casos de uso de Acceder-CAS.

Entidad	Descripción
Usuario	Es el actor responsable de iniciar los casos de uso que engloba acciones.
CI_Autetificar_ Reservación_ Alimentos	Es la clase interfaz que representa el sistema e interactúa con la aplicación SSO. El usuario se autentifica mediante ella.
CI_CAS	Es la clase interfaz que le permite al usuario autenticarse y ser redirigido hacia la aplicación Reservación de Alimentos UCI.

Tabla 7. Descripción caso de uso Acceder-CAS.

3.4 Diagramas de clases del análisis de los casos de uso de autentificación.

- Diagrama de clase del análisis del caso de uso “Autentificar usuario Reservación de Alimentos”.

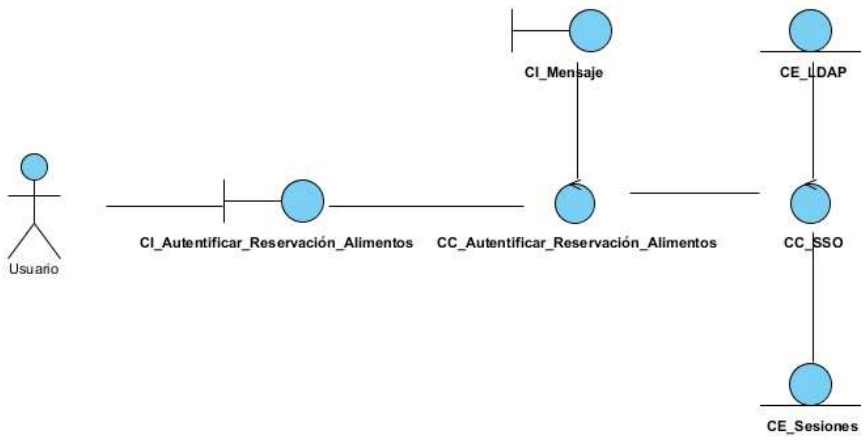

Figura 14. Diagramas de clases del análisis de los casos de uso Autentificación.

Entidad	Descripción
Usuario	Es el actor responsable de iniciar los casos de uso que engloba acciones.
CI_Autetificar_ Reservación_ Alimentos	Es la clase interfaz que representa el sistema e interactúa con la aplicación SSO. El usuario se autentifica mediante ella.
CC_Autetificar_ Reservación_ Alimentos	Es la clase control, que se encarga de realizar las operaciones de autentificación, para ello accede a la base de datos y se relaciona con la clase de control (CC_SSO) que verifica la existencia del usuario en el directorio activo de usuarios del sistema (LDAP).
CI_Mesaje	Clase interfaz que es creada cuando exista algún error en el proceso de autentificación como forma de notificación para el usuario.
CC_SSO	Clase control que realiza las operaciones del SSO para ello accede de forma

	directa al LDAP para verificar los datos del usuario.
CE_LDAP	Clase entidad que guarda datos de los usuarios de todo el sistema.
CE_Sesiones	Clase entidad que es una tabla de la base de datos del SSO, donde se guardara y consultara la información de las sesiones creadas para cada usuario.

Tabla 8. Descripción caso de uso Autentificación.

3.5 Diagrama de clases del análisis de los casos de uso validar.

- Diagrama de clases del análisis del caso de uso “Validar datos de usuario”.

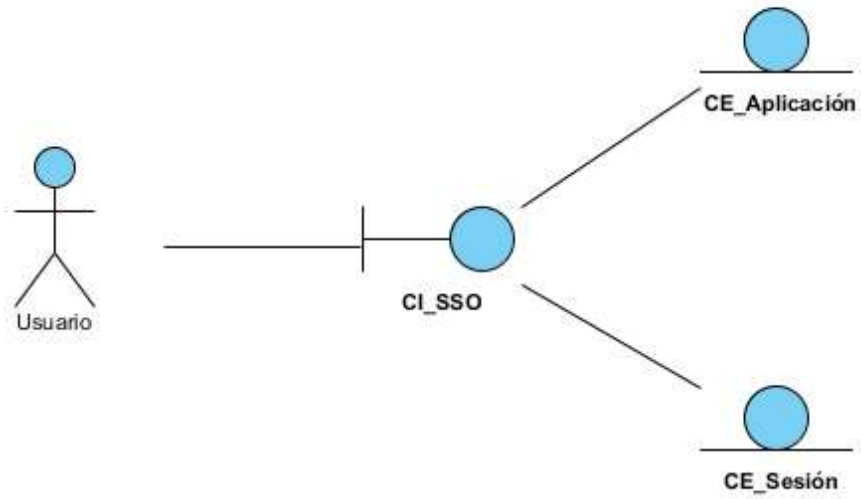

Figura 15. Diagrama de clases del análisis de los casos de uso Validar.

Entidad	Descripción
CI_SSO	1-La aplicación le solicita a la CI_SSO validar token de seguridad. La CI_SSO que a la misma vez se comparta como

	interfaz controladora, realiza un conjunto de operaciones como: ➢ Verificar si la aplicación tiene los permisos para utilizar el sistema, para esto busca en la base de datos. ➢ Luego la CI_SSO verifica que haya una sesión abierta y elimina las sesiones que hayan expirado o que no sean válidas de la base de datos (tabla sesión). ➢ Una vez eliminadas todas las sesiones fallidas, la CI_SSO verifica el token, para esto busca en la base de datos (tabla de sesión). ➢ La CI_SSO envía una respuesta a la aplicación.

Tabla 9. Descripción caso de uso Validar.

3.6 Diagrama de clases del análisis de los casos de uso registro.

- Diagrama de clases del análisis del caso de uso “Registrar usuario Reservación Alimentos”.

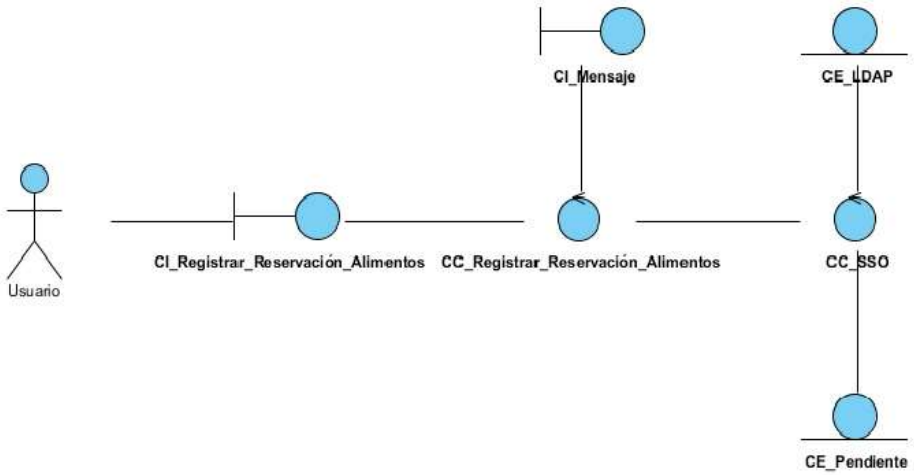

Figura 16. Diagrama de clases del análisis de los casos de uso Registrar.

Entidad	Descripción
CI_Registrar_ Reservación_Alimentos	Clase que brinda una interfaz para el registro de la información del sistema Reservación Alimentos.
CC_Registrar_ Reservación_ Alimentos	Clase de control que maneja los datos para realizar el registro del usuario en la base de datos del sistema Reservación Alimentos y la conexión con la clase control CC_SSO quien registra, al mismo tiempo la información modificada en el LDAP.
CI_Enviar_Notificación	Clase interfaz que provee al usuario, una manera de registrarse en el sistema, la notificación del envío.
CI_Pendiente	Clase entidad que almacenara información delas cuentas del usuario en la base de datos SSO.

Tabla 10. Descripción caso de uso Registrar.

3.7 Modelo de Diseño del SSO:

El modelo de diseño de un sistema:

- Caracteriza la realización de los casos de uso.
- Se concentra en cómo los requisitos funcionales y no funcionales tienen impacto en el sistema.

AUP propone que el artefacto Modelo de Diseño básicamente contenga:

- Introducción: una descripción textual que sirve como breve introducción al modelo.
- Diagramas: los diagramas de clases del diseño y diagramas de interacción (colaboración y/o secuencia) del diseño. Estos últimos también llamados realización de casos de uso.
- Clases, interfaces, relaciones: contenidas en los casos de uso y una breve descripción de ellos.

3.8 Diagramas de interacción:

Los diagramas de interacción muestran una conexión concreta: un conjunto de objetos y sus relaciones, junto con los mensajes que se envían entre ellos.

- Forman el comportamiento dinámico del sistema, el camino de control en una acción.
- Caracterizan la interacción entre objetos, los objetos interactúan a través de mensajes para cumplir tareas.
- Las interacciones brindan un <<comportamiento>> e implementan un caso de uso.

Constan dos tipos de diagramas de interacción en UML:

- Diagramas de Secuencia (dimensión temporal).
- Diagramas de colaboración (dimensión estructural). (27)

3.9 Pruebas.

El desarrollo de software necesita de la realización de pruebas para comprobar el cumplimiento de los requerimientos identificados y el correcto funcionamiento del software para garantizar la calidad del mismo y lograr su aceptación por parte del cliente.

Las pruebas son uno de los pilares fundamentales. Estas permiten aumentar la calidad de los sistemas reduciendo el número de errores no detectados y disminuyendo el tiempo

transcurrido entre la aparición de un error y su detección. También permiten aumentar la seguridad de evitar efectos colaterales no deseados a la hora de realizar modificaciones y refactorizaciones.

3.9.1 Pruebas de aceptación.

Las pruebas de aceptación son creadas en base a los casos de uso, en cada ciclo de la iteración del desarrollo. El cliente debe especificar uno o varios escenarios para comprobar que un caso de uso ha sido correctamente implementado. Son consideradas como "pruebas de caja negra" ("*Black box system tests*"). Se definen para cada caso de uso y los clientes son responsables de verificar que los resultados de estas pruebas sean correctos. Asimismo, en caso de que fallen varias pruebas, deben indicar el orden de prioridad de resolución. Un caso de uso no se puede considerar terminado hasta tanto pase correctamente todas las pruebas de aceptación.

Las pruebas se realizan por un cliente en un entorno controlado por el equipo de desarrollo. Para que tenga validez, se debe primero crear un ambiente con las mismas condiciones que se encontrarán en las instalaciones del cliente. Una vez logrado esto se procede a realizar las pruebas y a documentar los resultados. (28)

Estas pruebas tienen una importancia crítica para el éxito de una iteración. Por lo tanto, el encargado de las pruebas (*tester*) debe tenerlas listas lo antes posible a partir del comienzo de la iteración, y lograr que el cliente las apruebe para poder presentarlas cuanto antes al equipo de desarrollo.

Prueba de aceptación	
Código: CU-1	RF-1
Nombre: Acceso al CAS mediante el Sistema de Reservación de Alimentos.	
Descripción: Esta funcionalidad permitirá el acceso rápido al Sistema de Reservación de Alimentos UCI.	
Condiciones de ejecución:	

Entrada/Pasos de ejecución: Acceder al logotipo del servidor CAS.
Resultado esperado: Redireccionar al Sistema de Reservación de Alimentos UCI.
Resultado obtenido: El sistema re direcciona al Sistema de Reservación de Alimentos UCI.
Evaluación de la prueba: Satisfactoria.

Tabla 11.Prueba de aceptación Acceso- CAS.

Prueba de aceptación	
Código:CU-2	RF-2
Nombre: Autenticar usuario Reservación de Alimentos UCI.	
Descripción: Esta funcionalidad le permite al usuario autenticarse en el sitio, para acceder a los servicios brindados de acuerdo al rol que representa.	
Condiciones de ejecución: El usuario debe de estar autenticado para tener acceso a este servicio.	
Entrada/Pasos de ejecución: Acceder a la opción accede en el menú principal. 1. Se entran los datos del usuario de manera correcta. 2. Se entran los datos del usuario de manera incorrecta.	
Resultado esperado: 1. Ingresar al sistema. 2. Mostrar un mensaje indicando el intento fallido de la operación.	
Resultado obtenido: 1. Ingresa el sitio. 2. El sistema muestra un mensaje indicando el intento fallido de la operación.	
Evaluación de la prueba: Satisfactoria.	

Tabla 12.Prueba de aceptación Autenticar usuario.

Prueba de aceptación	
Código: CU-3	RF-3
Nombre: Validar datos del usuario en el LDAP.	
Descripción: Esta funcionalidad permita la autenticación centralizada al usuario.	
Condiciones: Las credenciales del usuario deben estar almacenadas en el LDAP-UCI.	
Entrada/Pasos de ejecución: Acceder a la opción accede en el menú principal. 1. Se entran los datos del usuario de manera correcta. 2. Se entran los datos del usuario de manera incorrecta.	
Resultado esperado: 1. Validar usuario. 2. Mostrar un mensaje indicando el intento fallido de la operación.	
Resultado obtenido: 1. Validación correcta de usuario. 2. El sistema muestra un mensaje indicando el intento fallido de la operación.	
Evaluación de la prueba: Satisfactoria.	

Tabla 13.Prueba de aceptación Validar datos de usuario.

Prueba de aceptación	
Código: CU-4	RF-4
Nombre: Registrar usuario Reservación de Alimentos UCI.	
Descripción: Esta funcionalidad le permite al usuario registrarse en el sitio.	
Condiciones de ejecución: El usuario debe introducir sus datos los cuales deben estar validados correctamente.	
Entrada/Pasos de ejecución: Acceder a la opción accede en el menú principal. 1. Se entran los datos del usuario de manera correcta. 2. Se entran los datos del usuario de manera incorrecta.	
Resultado esperado: 1. Registrar el usuario.	

2. Mostrar un mensaje indicando el intento fallido de la operación.
Resultado obtenido: 1. Registro del usuario. 2. El sistema muestra un mensaje indicando el intento fallido de la operación.
Evaluación de la prueba: Satisfactoria.

Tabla 14.Prueba de aceptación Registrar usuario.

3.10 Conclusiones parciales.

En este capítulo se establecieron las pautas de diseño necesarias para la construcción del sistema obtenida en el capítulo anterior. Se modelaron los diagramas de clases del análisis correspondiente a cada caso de uso del sistema, la descripción de todas las clases que integraron la estructura del sistema *Single Sign-On,* lo que facilitó un mejor entendimiento de la estructura del sistema como base para la realización de la ingeniería del producto.

Conclusiones

Después de una intensa investigación, la cual derivó en la tarea de recopilar información de sistemas (SSO), se llegó a la conclusión que esta era la vía correcta para la solución inmediata para lograr la implementación de un mecanismo de autentificación centralizada mediante SSO haciendo uso del CAS en el Sistema de Reservación de Alimentos UCI.

1. Las herramientas y tecnologías puestas en práctica permitieron el desarrollo claro y fluido de un sistema construido sobre bases sólidas y un entorno bien definido.
2. *Single Sign-On (SSO)* es un procedimiento de autentificación que habilita al usuario para acceder a varios sistemas con una sola instancia de autentificación.
3. Se realizó la implementación de un sistema SSO para unificar el sistema de Reservación de Alimentos UCI que garantiza un proceso de autenticación fácil, fluido y con mayor grado de seguridad.

Recomendaciones.

Los objetivos propuestos para el presente trabajo de diploma han sido cumplidos satisfactoriamente, no obstante, se incluyen dos recomendaciones a tomar en cuenta para futuras implementaciones de sistemas SSO.

- Crear un manual de usuario para garantizar el soporte a los administradores que trabajan en la configuración y mantenimiento del sistema.
- Implementar un sistema de este tipo en la Universidad de las Ciencias Informáticas para lograr una mayor coherencia, facilidad de uso, y seguridad en los servicios Web.

Glosario de términos.

Autentificación: se llama autentificación a la comprobación de la identidad de una persona o un objeto. La autentificación mediante identificador y contraseña es el sistema más común ya que viene incorporado en los sistemas operativos modernos de todos los ordenadores.

Autenticar: este verbo se documenta ya en el español medieval y sigue plenamente vigente en el lenguaje legal y administrativo, especialmente en América. Con este mismo sentido se ha creado modernamente el verbo *autentificar*, que se considera también valido y es el usado con preferencia en el español de España y de buena parte de América.

Credenciales: contienen información que le permite al usuario acceder a las diferentes aplicaciones (nombre de usuario, contraseña).

Framework: es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Puede ser considerado como el conjunto de procesos y tecnologías usadas para resolver un problema complejo. Es el esqueleto sobre el cual varios objetos son integrados para una solución dada.

Kerberos: es un protocolo de red que utiliza la criptografía simétrica para proporcionar autenticación para aplicaciones cliente-servidor.

Latencia: la latencia de una red es sinónimo de retardo. Es una medida de cuánto tiempo le toma ir de un punto a otro, a un paquete de datos. En algunos casos se acostumbra definir la latencia como el tiempo que le toma a un paquete de datos recorrer el trayecto cerrado (ida hasta algún punto y regreso hasta su remitente). Los factores que contribuyen a la latencia de la red son: Propagación, Transmisión, Enrutamiento, Procesamiento, y algunos retardos asociados a otros dispositivos (switches, discos duros).

OpenID: es un sistema de identificación digital descentralizado, con el que un usuario puede identificarse en una página Web a través de un URL y puede ser verificado por cualquier servidor que soporte el protocolo.

SSO: Sistema de Autenticación Única. *Single Sign-On (SSO)* es un procedimiento de autenticación que habilita al usuario para acceder a varios sistemas con una sola instancia de identificación.

Referencia

1. **Sánchez, Jordi.** *Sistemas de Autenticación y Autorización en Internet.* Universidad de Lleida : http://jordisan.net/proyectos/Autent_y_auth-J_Sanchez.pdf, junio 2011.

2. **Carter, Michael Fleming Grubb and Rob.** *Single Sign-On and the System Administrator.* Boston, Massachusetts : http://www.usenix.org, December 6-11, 1998.

3. **Eliurkis.** *Single Sign On: Sistema de Autenticación Único.* s.l. : http://www.deepinphp.com/2007/09/01/single-sign -on -sistema -de-autenticacion- unico, 2007.

4. **Trilla, Miqel.** *Single Sign-On.* 2006.

5. **Gamboa, Harold González.** *Análisis Teórico-Técnico de Single Sign-On.* Universidad Latinoamericana de Ciencia y Tecnología : Licenciatura en Informática con Énfasis en Desarrollo de Software, 2005 .

6. **GSA, Notas.** *Google Search Appliance.* s.l. : http://static.googleusercontent.com/media/www.google.com/es//support/enterprise/static/gsa/docs/deployment/es/GSAUserExperienceGuide_es.pdf, Agosto de 2014.

7. **Informática, Servicio de.** *Manual de Windows Live Messenger.* Universidad de Jaén : https://www10.ujaen.es/sites/default/files/users/sinformatica/guiaspracticas/guia_messenger.pdf, noviembre de 2008.

8. **Josefsson, Simon.** *OpenID.* s.l. : https://josefsson.org/talks/fscons-openid.pdf, 2007.

9. **Zwanziger, Iván Fernández.** *Diseño e implementación de un sistema para la gestión y publicación de videos.* s.l. : MTI, 14 de Septiembre del 2009.

10. **Fernández, Oscar Belmonte.** *Introducción al lenguaje de programación Java.: Una guía básica.* s.l. : http://www3.uji.es/~belfern/pdidoc/IX26/Documentos/introJava.pdf, 25-10-2004.

11. **Mestras, Juan Pavón.** *Java EE – Apache Tomcat, Aplicaciones Web/Sistemas Web.* Dep. Ingeniería del Software e Inteligencia Artificial, Facultad de Informática, Universidad Complutense Madrid : https://www.fdi.ucm.es/profesor/jpavon/web/42-tomcat.pdf, 2011.

12. **Ferrer.** *Diseño de Sistemas.* s.l. : http://www.daedalus.es/inteligencia-de-negocio/sistemas-complejos/ingenieria-de-sistemas/diseño-de-sistemas/, 2006.

13. **Fernandez, Norberto.** *Introducción a Spring.* Departamento de Ingeniería Telemática Universidad Carlos III de Madrid : http://www.it.uc3m.es/mario/si/Introduccion_a_Spring.pdf, 2008/2009.

14. **M.C.C. Valentin Roldan, Eduardo Cabral Mendoza.** *Mapeo Objeto Relacional.* s.l. : http://es.slideshare.net/poropunk/mapeo-objeto-relacional, 8 de octubre 2012.

15. **Gavin King, Christian Bauer, Max Rydahl Andersen, Emmanuel Bernard, y Steve Ebersole.** *HIBERNATE - Persistencia relacional para Java idiomático.* s.l. : https://docs.jboss.org/hibernate/orm/3.5/reference/es-ES/html_single, September 15, 2010.

16. **BERTINO, E. A. y MARTINO.** *Sistemas de bases de datos orientadas a objetos.* Los Angeles,EE.UU : https://books.google.com.cu/books, 1995.

17. **Denzer, Patricio.** *PostgreSQL.* s.l. : http://profesores.elo.utfsm.cl/~agv/elo330/2s02/projects/denzer/informe.pdf, 23 de octubre de 2002.

18. **Gómez, Ruth Priscila Landeros.** *Herramientas Case.* Universidad Veracruzana : http://www.monografias.com/trabajos14/herramicase/herramicase.shtml.

19. **Lianet Cabrera González, Enrique Roberto Pompa Torres.** *Extensión de Visual Paradigm for UML para el desarrollo dirigido por modelos de aplicaciones de gestión de información.* La Hab ana, Cuba : s.n., 15/10/2012.

20. **LIC. ERVIN FLORES, JORGE LUIS CORDERO L.** *METODOLOGIAS AGILES PROCESO UNIFICADO AGIL (AUP).* LA PAZ, EL ALTO – BOLIVIA : s.n.

21. **López, Ing. Febe Ángel Ciudad Ricardo - Ing. Ninet Soto.** *Fase de inicio. modelo del negocio.* s.l. : Conferencia 2, 2006.

22. **López., Ing. Febe Ángel Ciudad Ricardo - Ing. Ninet Soto.** *Fase de inicio. Levantamiento de requisitos.* s.l. : Conferencia 3, 2006.

23. **Kaiser, Andreas.** *Software Libre.* s.l. : http://www.atela.net/contenido/articulos/softwarelibre.html] , 2000.

24. **Mestras, Juan Pavón.** *El patrón Modelo-Vista-Controlador (MVC).* Universidad Complutense Madrid : https://www.fdi.ucm.es/profesor/jpavon/poo/2.14.MVC.pdf, 2008-2009.

25. **Donnelly, Michael.** *Una Introducción a LDAP.* s.l. : http://www.ldapman.org , 28 abril 2000.

26. **López, Ing. Febe Ángel Ciudad Ricardo - Ing. Ninet Soto.** *Fase de Elaboración. Análisis - Diseño.* s.l. : Conferencia 5, 2006.

27. **Ferrer, F.** *Diseño de Sistemas.* s.l. : http://www.daedalus.es/inteligencia-denegocio/sistemas-complejos/ingenieria-de-sistemas/diseno-de-sistemas/, 2006.

28. **Pablo Suárez, Carlos Fontela.** *Documentación y pruebas.* s.l. : http://materias.fi.uba.ar/7507/content/20101/lecturas/documentacion_pruebas.pdf, 2003.

Anexos.

Diagramas de clase de diseño.

Autenticar usuario Reservación de Alimentos.

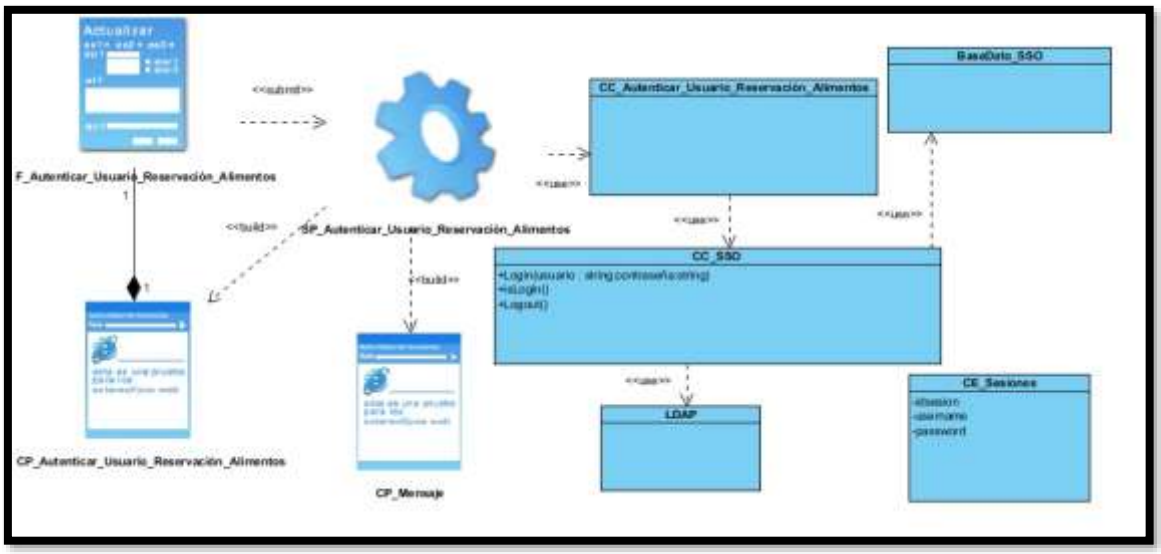

Figura 17.Diagramas de clase de diseño. Autenticar usuario.

Registrar usuario Reservación de Alimentos.

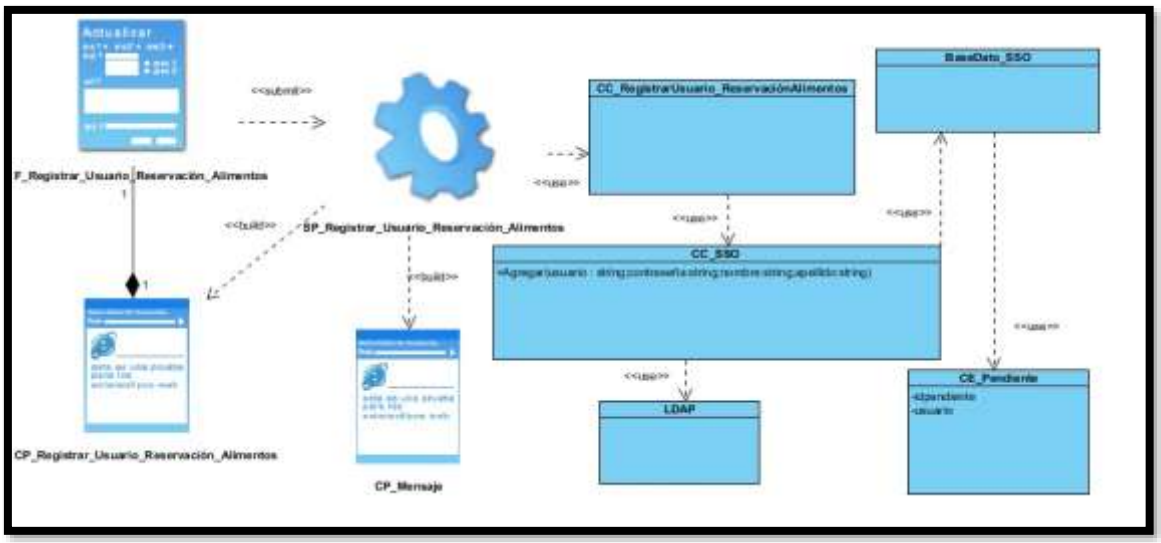

Figura 18.Diagramas de clase de diseño. Registrar usuario.

Validar datos de usuario.

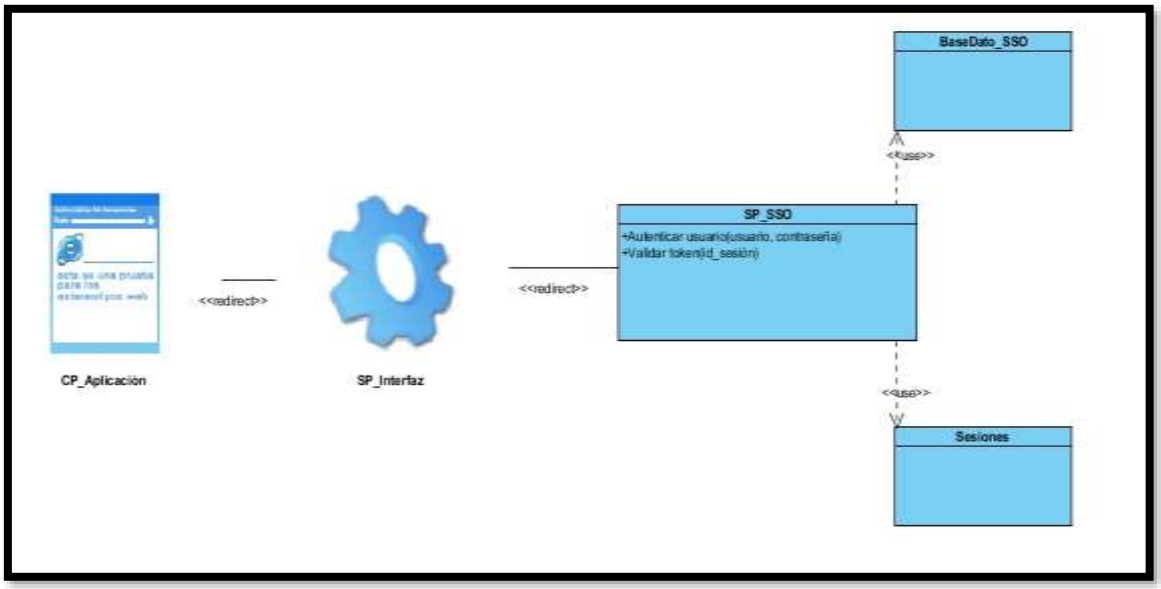

Figura 19.Diagramas de clase de diseño. Validar datos usuarios.

Open ID

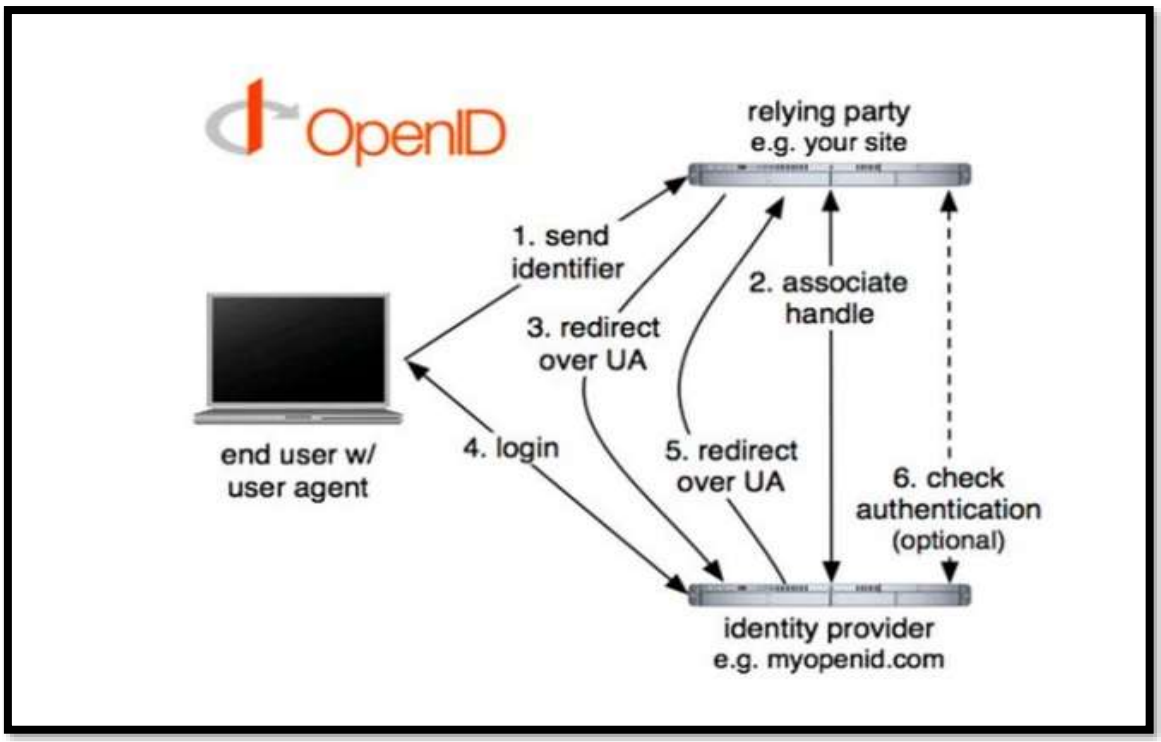

Figura 20.Esquema de identificación Open ID.

Interfaz CAS.

Figura 21.Interfaz CAS