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

Título:

Sistema de Gestión de Sesiones.

Autoras:

- Elianys Hurtado Sola
- Mónica Rodríguez Carbajal

Tutor:

- Ing. Yudiel Alfredo Tamayo Agramonte.

Universidad De las Ciencias Informáticas.

Facultad #2

Cuidad de la Habana, Junio 2008.

"Año 50 de la Revolución"

"La sabiduría es el único bien que no se pueden llevar los ladrones."

Benjamin Franklin.

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 _____.

_____ _____

Elianys Hurtado Sola Mónica Rodríguez Carbajal

_____

Ing. Yudiel Alfredo Tamayo Agramonte

Datos de contacto

Síntesis del tutor: Ing. Yudiel Alfredo Tamayo Agramonte.

Profesión: Ingeniero Informático, Especialista Superior, Dirección de informatización.

Años de graduado**:** 3 años

Correo electrónico: yalfred@uci.cu

Agradecimientos

Durante la vida de estudiante son muchas las personas que contribuyen con nuestra formación profesional y para la vida, profesores, amigos y la familia fundamentalmente. En este momento cumbre de nuestros estudios, donde nos convertimos en profesionales les agradecemos sinceramente, con el mayor deseo de estar a la altura de las enseñanzas y principios que compartieron con nosotros.

A Yudiel Alfredo Tamayo Agramonte, nuestro tutor, por su ayuda, su paciencia y su apoyo infinitos. Por haber estado siempre dispuesto para nosotras y mostrarnos el camino.

A Ronny Zamora por toda la ayuda que nos prestó siempre que la necesitamos.

A mi madre y a mis abuelos, por su amor, dedicación y comprensión, porque lo que soy hoy, es gracias a ellos…son lo más grande que tengo en la vida.

A los amigos que me ayudaron cuando tenía algún problema, cuando no entendía algo y pude contar con ellos siempre.

A nuestros profesores y de manera especial agradezco a la Universidad de las Ciencias Informáticas.

Mónica.

A mi padre

Por ser mi mejor maestro y mi eterno guía.

A mi madre

Que siempre supo que llegaría y por enseñarme que siempre podemos ser mejores.

A mis amigos en general

Que supieron hacerme crecer en todo momento.

Elianys

Dedicatoria

A mis padres

Por su amor inmenso, su cariño, por el apoyo, su preocupación, su comprensión, sus acertados consejos, por la confianza que siempre me tuvieron y haber sabido hacer de mí, lo que hoy soy, en fin…
A ellos… Por todo.

A Gerdys

Que ha sido y es mi mejor apoyo, que me supo comprender en mis momentos más difíciles y me ayudó a salir de ellos de la mejor manera, por mostrarme la luz en los tiempos más oscuros. Por no decepcionarme en ningún instante, por estar siempre para mí...
Por ser mi amor.

A mis queridos abuelos

Por quererme y amarme tanto.

A Ody, a Nayi, a Marle, a la súper, a Gaby y a mis tías Yamila y Úrsula

Que siempre me han tenido presente, por su constante preocupación.

A Monikilla

Por no haberme defraudado, y porque mejor compañera de tesis nunca podría pedir.

A mis amigos

Aquellos que no me olvidan y que se que ahora, están pensando en mí.

Elianys.

A mi Alla, por ser mi dicha, por cuidarme siempre, por quererme mucho, por su preocupación constante para que no me pase nada malo…A mi abuelo, por ser mi guía, por su valiosa educación, por confiar en que puedo ser mejor, por enseñarme a ser fuerte y a no ponerme triste por nada…

A mi mamita preciosa, por su amor y fortaleza, por su ánimo, sus enseñanzas, por darme siempre lo mejor de ella para mi futuro.

A mis tíos Mauro, Zadi, Myle, Adita y Galle, por darme todo su apoyo y amor desde que nací…

A mi madrina Patricia, por tenerme siempre presente, por todo el cariño que me ha dado.

A Charles, por ser mi amor, mi alegría…porque en mis peores momentos ha estado allí, para cuidarme, para darme su apoyo y su amor….

A mis primos(a) lindos, para quienes quiero todo lo bueno del mundo.

A mis amigas Lissete y Raquel, que son mis hermanitas lindas, que siempre se acordaron de mi y estoy feliz de tenerlas.

A Elianys, mi compañera de tesis y de cada trabajo investigativo que siempre presentábamos juntas, a ella le deseo lo mejor en su futuro.

A mi grupo 2102, a Yelani, Yeisis, Yady…y a mis amigas que están ahora en Camagüey pero sé que me tienen presente.

Mónica.

Resumen

El tema de este trabajo es el desarrollo de un Sistema de Gestión de Sesiones, que se hace necesario en nuestra universidad ya que evita molestias y ahorra tiempo a los usuarios de la UCI y fortalece en gran medida los niveles de la seguridad de la organización. Este sistema se convierte en una iniciativa viable de protección y control que permite ofrecer, tanto internamente como a los usuarios, servicios de valor añadido con altos niveles de seguridad y confianza.

La realización de este sistema, dando cumplimiento a los objetivos planteados logrará mejorar la seguridad de la red en nuestra universidad, permitiendo a su vez a los usuarios disminuir su trabajo de autenticación y el tiempo que pierden los mismos en este proceso. Además en la universidad cada uno de los servicios o aplicaciones cuenta con su propio componente de seguridad, lo cual generalmente compromete la seguridad de todo el sistema, dado que el nivel de seguridad de todo un sistema es igual al nivel de seguridad del componente más inseguro que tenga.

Actualmente no existe ningún sistema implementado en la escuela y no hay conocimiento de que exista un sistema de este tipo en nuestro país, lo que trae consigo que los usuarios deban repetir el proceso de autenticación cada vez que van a acceder a un recurso de la red, gastando mayor tiempo y trabajo.

Este sistema se realiza con el objetivo de gestionar las sesiones de los usuarios en un único proceso de autenticación invisible a los ojos de los mismos, a través de un sistema con una fachada de servicios web que sea capaz de gestionar la apertura y cierre de sesiones por parte de las personas que trabajan en el dominio uci.cu.

Los resultados más relevantes que se recogen a los largo del desarrollo de esta investigación son la obtención e implementación la universidad de un sistema de nuevo que cuenta con una fachada de servicios y con una aplicación web para mostrar registros del sistema y configurar el mismo. Con el desarrollo de esta investigación se obtiene además el flujo de un nuevo proceso de autenticación, el que posteriormente utilizarán las aplicaciones del dominio.

Palabras claves:

Gestión, sesión, sistema, aplicación web, estadística, usuario.

Índice

Contenido

Índice de tablas

Contenido

Capítulo 2

Capítulo 5

Índice de imágenes

Capítulo 1

Capítulo 2

Capítulo 3

Capítulo 4

Introducción

Tradicionalmente se ha pensado que implementar Single Sign On (SSO) trae múltiples inconvenientes como son afrontar problemas de seguridad y tener que asumir altos costos de implementación y mantenimiento. Sin embargo las nuevas tecnologías desarrolladas, las infraestructuras de las aplicaciones que son cada vez más sólidas y la maduración de las diferentes técnicas de autenticación y seguridad, han permitido que se reconsidere la posibilidad de implementar Single Sign On como una estrategia para fortalecer la seguridad informática de las organizaciones.[1]

El concepto de Single Sign On hace referencia al acceso a múltiples recursos por medio de un único proceso de autenticación del usuario. La mayoría de las arquitecturas implementadas en diferentes organizaciones han sido diseñadas con el objeto de dar acceso a los usuarios a múltiples aplicaciones y/o servicios Web. La solución a este problema es la implantación de una estrategia Single Sign On. 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 SSO. En una arquitectura SSO, todos los mecanismos de seguridad se encuentran concentrados en el mismo, siendo éste el único punto de autenticación en el sistema. Otro beneficio de una arquitectura SSO es que los usuarios deben hacer el proceso de autenticación una sola vez, a pesar de que continúan interactuando con múltiples componentes de seguridad en el sistema. [1]

En nuestra universidad, y en el país, no existe ningún sistema de este tipo implementado y por tanto los usuarios cada vez que van a acceder a un recuso de la red deben introducir sus datos, entiéndase por estos datos el usuario y la contraseña, y cada vez que introducen sus datos pasar por el proceso de autenticación consiguiente. Implementar un sistema de este tipo evitaría todas estas molestias y ahorraría tiempo, en nuestro caso le llamaremos Sistema de Gestión de Sesiones, ya que este sistema solo se encargará del proceso de gestión de las sesiones de los usuarios.

Ante toda la ***situación problémica*** inicialmente referida se nos hace evidente que surge el siguiente ***problema científico***:

¿Cómo lograr que los usuarios realicen un único proceso de autenticación en la Universidad de Ciencias Informáticas?

A partir de lo explicado se puede afirmar que el ***objeto de estudio*** de este trabajo son los procesos de autenticación que trabajan con Sistemas SSO.

Definiendo así como ***campo de acción*** los Sistemas de Gestión de Sesiones desarrollados en la universidad.

Concretando como ***objetivo general*** de nuestro trabajo de tesis:

Desarrollar un sistema que garantice la gestión de sesiones de los usuarios de las aplicaciones y del dominio uci.cu al acceder a múltiples aplicaciones en la red.

Para el cumplimiento de nuestro objetivo general se han precisado los siguientes ***objetivos específicos:***

1. Analizar y diseñar un sistema de gestión de sesiones.
2. Diseñar la correspondiente base de datos del sistema.
3. Implementar el sistema.
4. Implementar una fachada de servicios para la comunicación con los demás sistemas.

A continuación de lo expresado la ***idea a defender*** que se plantea es la siguiente:

Con la implementación de un Sistema de Gestión de Sesiones los usuarios podrán introducir sus datos una sola vez al acceder a distintos recursos de la red, con esto se hace referencia a aplicaciones Web y accesos FTP.

Finalmente se determina que se deben cumplir con las siguientes ***tareas:***

- Investigar sobre el trabajo de los Sistemas de Gestión de Sesiones que ya se han implementado en otros países. (1)
- Recopilación de información relacionada con la manipulación de datos que se necesitan en la gestión de sesiones de usuarios. (1)

- Identificar las clases del análisis (clases de interfaz, clases de entidad, clases de control). Confeccionar un diagrama de clases del análisis por caso de uso.(1)
- Identificar las clases del diseño.(1)
- Confeccionar el diagrama de clases del diseño.(1)
- Confeccionar los diagramas de secuencia del diseño.(1)
- Construir el Diagrama de Entidad - Relación.(2)
- Identificar las extensiones del Modelo Entidad- Relación.(2)
- Transformar el Modelo Entidad-Relación al modelo relacional .(2)
- Generar la base de datos.(2)
- Construir las consultas SQL.(2)
- Investigar y escoger posteriormente, las herramientas y lenguajes a utilizar para el desarrollo del sistema.(3)
- Implementar el sistema con la utilización de las herramientas y lenguajes escogidos.(3)

Métodos Teóricos que se emplearon:

Inductivo – Deductivo: Se realizó un análisis del trabajo general de los SSO que nos permitió llegar al conocimiento particular de su aplicación en la Universidad de las Ciencias Informáticas.

Modelación: Se crearon modelos simplificados de la realidad, para estudiar las relaciones y cualidades de los métodos utilizados para el proceso de autenticación de usuarios. Principalmente se crearon y utilizaron los diagramas correspondientes la ingeniería de software de nuestro sistema, es decir los diagramas basados en la modelación UML.

La realización de este sistema, dando cumplimiento a los objetivos planteados logrará mejorar la seguridad de la red en nuestra universidad, permitiendo a su vez a los usuarios disminuir su trabajo de autenticación y el tiempo que pierden los mismos en este proceso.

El presente trabajo se compone de la siguiente estructura:

Capitulo 1: Fundamentación teórica: Descripción del estado del arte de los Sistemas de Gestión de Sesiones, en el mundo, en Cuba y en la UCI, de las tendencias, técnicas, tecnologías, metodologías y

software usados en la actualidad. Descripción de las herramientas y fundamentación de su utilización así como la de las tecnologías, el lenguaje de modelado y metodologías usadas.

Capítulo 2: Características del sistema: Fundamentación del problema y la situación problémica a través de la descripción de los objetivos estratégicos, el flujo de los procesos involucrados en el campo de acción, el análisis crítico de la realización actual de esos procesos, las causas que originan la situación problémica y las consecuencias. Descripción general del objeto de automatización, o sea de los procesos que serán objeto de automatización, así como de la propuesta de sistema y su comparación con otros sistemas de su tipo ya existentes. Contiene el desarrollo completo del Modelo de Dominio, la recogida de requerimientos del software tanto funcionales como los no funcionales y por último el diagrama completo de los casos de uso del sistema así como su descripción detallada.

Capítulo 3: Análisis y diseño del sistema: Realización de los diagramas de clases del análisis y del diseño por cada caso de uso, y de los diagramas de interacción del diseño. Se documenta el esbozo de la base de datos en su diagrama de clases persistentes como modelo lógico y en el modelo de datos como modelo físico. Queda ejecutado con su culminación el último paso hacia la implementación del sistema.

Capítulo 4: Implementación: Construcción y obtención del diagrama de despliegue por lo que queda definido los nodos físicos con los que contará la aplicación. Elaboración además del diagrama de componentes quedando documentado cada uno de los componentes que ejecutan código en el sistema.

Capítulo 5: Estudio de la factibilidad: Se realiza la planificación del proyecto quedando calculado luego de una serie de pasos y fórmulas el esfuerzo que trae consigo la implementación de la propuesta de solución, el tiempo de desarrollo para una cantidad de personas, el salario medio y el costo. Con esto se demuestra la factibilidad del sistema, o sea que el mismo no trae consigo pérdidas a la institución.

1 Capítulo

Fundamentación Teórica

1.1 Introducción.

En el capítulo que se presenta a continuación se aborda el estado del arte de los Sistemas de Gestiones en la UCI, en Cuba y en el mundo. Contiene además la descripción de las tendencias y tecnologías de desarrollo utilizadas en estos momentos en el análisis, diseño e implementación del sistema sobre las que se basa la propuesta de solución, así como las metodologías y software utilizados en la actualidad. Este capítulo trata los problemas que fundamentan la propuesta de solución al problema planteado y los objetivos que se persiguen.

1.2 Metodologías utilizadas

Metodología de Desarrollo de software utilizada:

Metodología RUP: Se utilizó la misma para asignar tareas y responsabilidades de forma disciplinada, buscando implementar las mejores prácticas de ingeniería de software. Usamos esta metodología además en la administración de requisitos, en el control de cambios del software, para modelar el software y para verificar su calidad. Logrando un desarrollo iterativo del mismo y el uso de una arquitectura basada en componentes.

1.3 Estado del arte de los Sistemas de Gestión de Sesiones

1.3.1 Estado actual de los Sistemas de Gestión de Sesiones en el mundo

A nivel mundial se han realizado Sistemas de Autenticación Única o Single Sign ON (SSO), pero estos no se aplican a nuestra universidad porque son software propietarios muy complejos con un alto costo en el mercado y no se ajustan a la arquitectura SOA que se quiere instaurar en la universidad.

Un ejemplo de sistema SSO implementado es la solución que le da CeRtiS Gull como corporación a la seguridad de otras empresas, un Single Sign ON, que autentica a los usuarios mediante sus huellas digitales, y en la UCI no aplica este sistema porque en la universidad no se desea autenticar a los usuarios a través de sus huellas digitales.

Otro ejemplo es la plataforma v-Go Single ON, producida por la compañía de Passlogix, uno de los sistemas SSO más populares y óptimos que se han implementado que tiene precios muy altos, la distribución de esta plataforma en conjunto con IBM tiene un costo de 75 USD por usuario que incluye un valor de licencia única y su mantenimiento por un año, si se compra su versión multilingüe y multiplataforma que incluye el CD-ROM de instalación costará entonces 140 USD por usuario y los servicios de instalación y configuración tendrán un valor de 3.60 USD por usuario.

1.3.2 Estado actual de los Sistemas de Gestión de Sesiones en Cuba.

No se tiene conocimiento de que en nuestro país se haya implementado algún sistema de este tipo, ni un SSO, ni una adaptación del mismo.

1.3.3 Estado actual de los Sistemas de Gestión de Sesiones en la UCI.

En la universidad este es el primer sistema de su tipo que se implementa, pues sucede lo mismo que en todo el país, no existe ningún sistema SSO, ni ninguna adaptación del mismo.

1.4 Tendencia de la solución del problema.

La tendencia actual de este trabajo es la de manejar las sesiones del dominio de forma que existan menores posibilidades de fallas en la seguridad y operar el sistema de forma que sea invisible al usuario.

1.5 Tendencias y tecnologías actuales

Teniendo en cuenta las necesidades vistas y las características del entorno donde se aplicará la solución propuesta, se realizó un estudio de las tendencias y tecnologías actuales posibles a emplear, descritas a continuación.

1.5.1 Aplicación Web

Una aplicación web es un sistema informático que los usuarios utilizan accediendo a un servidor web a través de Internet o de una intranet. Las aplicaciones web son populares debido a la practicidad del navegador web como cliente ligero. La facilidad para actualizar y mantener aplicaciones web sin distribuir e instalar software en miles de potenciales clientes es otra razón de su popularidad. [2]

¿Qué estructura tiene una aplicación de este tipo?

Una aplicación web está comúnmente estructurada como una aplicación de tres-capas. En su forma más común, el navegador web es la primera capa, un motor usando alguna tecnología web dinámica (ejemplo: CGI, PHP, Java Servlets o ASP) es la capa del medio, y una base de datos como última capa. El navegador web manda peticiones a la capa media, que la entrega valiéndose de consultas y actualizaciones a la base de datos generando una interfaz de usuario.

¿Qué ventajas nos ofrece una aplicación web?

Una ventaja significativa en la construcción de aplicaciones web que soporten las características de los browsers estándar es que deberían funcionar igual, independientemente de la versión del sistema operativo instalado en el cliente. En vez de crear clientes para Windows, Mac OS X, GNU/Linux, y otros sistemas operativos, la aplicación es escrita una vez y es mostrada casi en todos lados. Sin embargo, aplicaciones inconsistentes de HTML, CSS, DOM y otras especificaciones de browsers pueden causar problemas en el desarrollo y soporte de aplicaciones web. Adicionalmente, la habilidad de los usuarios a personalizar muchas de las características de la interfaz (como tamaño y color de fuentes, tipos de fuentes, inhabilitar Javascript) puede interferir con la consistencia de la aplicación web.

1.5.1.1 Arquitectura de una aplicación Web.

Las aplicaciones web se han convertido en pocos años en complejos sistemas con interfaces de usuario cada vez más parecidas a las aplicaciones de escritorio, dando servicio a procesos de negocio de considerable envergadura y estableciéndose sobre ellas requisitos estrictos de accesibilidad y respuesta. Esto ha exigido reflexiones sobre la mejor arquitectura y las técnicas de diseño más adecuadas.

1.5.1.1.1 Modelo Vista Controlador

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 la Lógica de negocio y el controlador es el responsable de recibir los eventos de entrada desde la vista. [3]

Descripción del modelo [3]:

- Modelo: Esta es la representación específica de la información con la cual el sistema opera. La lógica de datos asegura la integridad de estos y permite derivar nuevos datos; por ejemplo, no permitiendo comprar un número de unidades negativo, calculando si hoy es el cumpleaños del usuario o los totales, impuestos o importes en un carrito de la compra.
- Vista: Este presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario.
- Controlador: Este responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista.

Ventajas [3]:

- Hay una clara separación entre los componentes de un programa; lo cual nos permite implementarlos por separado
- Hay un API muy bien definido; cualquiera que use el API, podrá reemplazar el Modelo, la Vista o el Controlador, sin aparente dificultad.
- La conexión entre el Modelo y sus Vistas es dinámica; se produce en tiempo de ejecución, no en tiempo de compilación.

Al incorporar el modelo de arquitectura MVC a un diseño, las piezas de un programa se pueden construir por separado y luego unirlas en tiempo de ejecución. Si uno de los Componentes, posteriormente, se observa que funciona mal, puede reemplazarse sin que las otras piezas se vean afectadas.

Definición de las partes

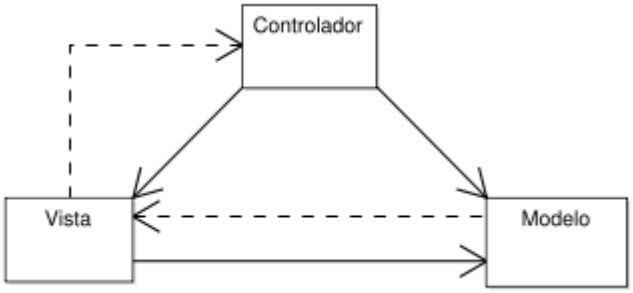

Imagen 1. 1: Estructura clásica del MVC

1.5.1.1.2 Modelo Cliente/Servidor

Con respecto a la definición de arquitectura cliente/servidor se encuentran las siguientes definiciones [4]:

- Cualquier combinación de sistemas que pueden colaborar entre sí para dar a los usuarios toda la información que ellos necesiten sin que tengan que saber donde está ubicada.
- Es una arquitectura de procesamiento cooperativo donde uno de los componentes pide servicios a otro.
- Es un procesamiento de datos de índole colaborativo entre dos o más computadoras conectadas a una red.
- El término cliente/servidor es originalmente aplicado a la arquitectura de software que describe el procesamiento entre dos o más programas: una aplicación y un servicio soportante.
- IBM define al modelo Cliente/Servidor. "Es la tecnología que proporciona al usuario final el acceso transparente a las aplicaciones, datos, servicios de cómputo o cualquier otro recurso del grupo de trabajo y/o, a través de la organización, en múltiples plataformas. El modelo soporta un medio ambiente distribuido en el cual los requerimientos de servicio hechos por estaciones de trabajo inteligentes o "clientes", resultan en un trabajo realizado por otros computadores llamados servidores".
- Es un modelo para construir sistemas de información, que se sustenta en la idea de repartir el tratamiento de la información y los datos por todo el sistema informático, permitiendo mejorar el rendimiento del sistema global de información.

Este modelo puede incluir múltiples plataformas, bases de datos, redes y sistemas operativos. Estos pueden ser de distintos proveedores, en arquitecturas propietarias y no propietarias y funcionando todos al mismo tiempo. Por lo tanto, su implantación involucra diferentes tipos de estándares: APPC, TCP/IP, OSI, NFS, DRDA corriendo sobre DOS, OS/2, Windows o PC UNIX, en TokenRing, Ethernet, FDDI o medio coaxial, sólo por mencionar algunas de las posibilidades.

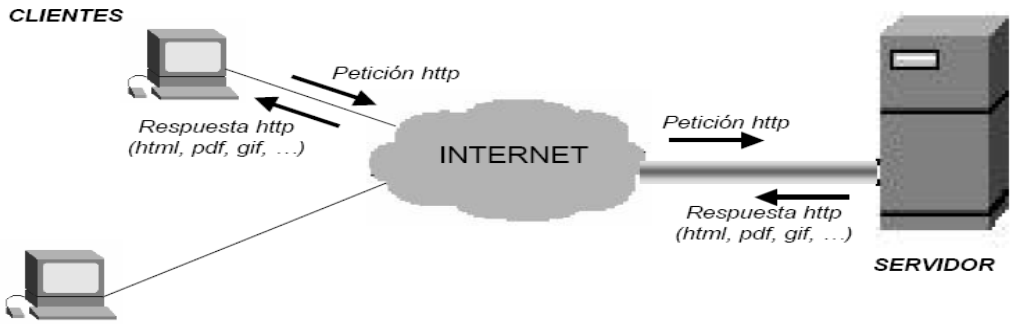

Imagen 1. 2: Arquitectura clásica del modelo Cliente/Servidor

Ventajas del Modelo Cliente/Servidor [4]:

- Centralización del control: Los accesos, recursos y la integridad de los datos son controlados por el servidor de forma que un programa cliente defectuoso o no autorizado no pueda dañar el sistema. Esta centralización también facilita la tarea de poner al día datos u otros recursos.

- Escalabilidad: Se puede aumentar la capacidad de clientes y servidores por separado. Cualquier elemento puede ser aumentado (o mejorado) en cualquier momento, o se pueden añadir nuevos nodos a la red (clientes y/o servidores).

- Fácil mantenimiento: Al estar distribuidas las funciones y responsabilidades entre varios ordenadores independientes, es posible reemplazar, reparar, actualizar, o incluso trasladar un servidor, mientras que sus clientes no se verán afectados por ese cambio (o se afectarán mínimamente). Esta independencia de los cambios también se conoce como encapsulación.

- Hay tecnologías maduradas diseñadas para el Modelo Cliente/Servidor que aseguran la seguridad en las transacciones, la amigabilidad del interfaz, y la facilidad de empleo.

1.5.2 Servicio Web (web service)

Un servicio web (en inglés web service) es una colección de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma, pueden utilizar los servicios web para intercambiar datos en redes de ordenadores como Internet. La interoperabilidad se consigue mediante la adopción de estándares abiertos. [5].

¿Qué ventajas ofrecen los web service?

- Aportan interoperabilidad entre aplicaciones de software independientemente de sus propiedades o de las plataformas sobre las que se instalen.
- Los servicios Web fomentan los estándares y protocolos basados en texto, que hacen más fácil acceder a su contenido y entender su funcionamiento.
- Al apoyarse en HTTP, los servicios Web pueden aprovecharse de los sistemas de seguridad firewall sin necesidad de cambiar las reglas de filtrado.
- Permiten que servicios y software de diferentes compañías ubicadas en diferentes lugares geográficos puedan ser combinados fácilmente para proveer servicios integrados.
- Permiten la interoperabilidad entre plataformas de distintos fabricantes por medio de protocolos estándar.

1.5.3 PHP (PHP: Hypertext PreProcessor).

PHP es un lenguaje de programación usado en la creación de páginas web dinámicas. PHP es un acrónimo recursivo que significa "PHP Hypertext Pre-Processor" (inicialmente PHP Tools, o, Personal Home Page Tools), y se trata de un lenguaje interpretado. [6]

PHP es una alternativa a las tecnologías de Microsoft ASP y ASP.NET (que utiliza C#/VB.NET como lenguajes), a ColdFusion de la compañía Adobe (antes Macromedia), a JSP/Java de Sun Microsystems, y a CGI/Perl. Aunque su creación y desarrollo se da en el ámbito de los sistemas libres, bajo la licencia GNU, existe además un IDE (entorno integrado de desarrollo) comercial llamado Zend Optimizer. Recientemente, CodeGear (la división de lenguajes de programación de Borland) ha sacado al mercado un entorno integrado de programación para PHP, denominado Delphi for PHP. [6]

¿Qué ventajas brinda PHP?

PHP ofrece varias ventajas sobre otros lenguajes y las que hacen que sea el lenguaje más óptimo en la solución del problema planteado. Dichas ventajas se describen a continuación [6]:

- Es un lenguaje multiplataforma.
- Capacidad de conexión con la mayoría de los manejadores de base de datos que se utilizan en la actualidad.
- Capacidad de expandir su potencial utilizando la enorme cantidad de módulos (llamados ext's o extensiones).
- Posee una amplia documentación en su página oficial, entre la cual se destaca que todas las funciones del sistema están explicadas y ejemplificadas en un único archivo de ayuda.
- Es libre, por lo que se presenta como una alternativa de fácil acceso para todos. PHP es Open Source, lo cual significa que el usuario no depende de una compañía específica para arreglar cosas que no funcionan, además no está forzado a pagar actualizaciones anuales para tener una versión que funcione.
- Permite las técnicas de Programación Orientada a Objetos.
- Biblioteca nativa de funciones sumamente amplia e incluida
- No requiere definición de tipos de variables.
- Tiene manejo de excepciones.
- Rapidez. PHP generalmente es utilizado como módulo de Apache, lo que lo hace extremadamente veloz. Esta completamente escrito en C, así que se ejecuta rápidamente utilizando poca memoria.
- Se puede incrustar código PHP con etiquetas HTML.

1.5.4 Sistemas de Gestión de Bases de Datos.

Los Sistemas de gestión de base de datos son un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta. En los textos que tratan este tema, o temas relacionados, se mencionan los términos SGBD y DBMS, siendo ambos equivalentes, y acrónimos, respectivamente, de Sistema Gestor de Bases de Datos y DataBase Management System, su expresión inglesa.[7]

El propósito general de los sistemas de gestión de base de datos es el de manejar de manera clara, sencilla y ordenada un conjunto de datos que posteriormente se convertirán en información.

Entre los Sistemas de Gestión de Bases de Batos podemos encontrar los sistemas libres como por ejemplo PostgresSQL, MySQL y Firebird. Existen sistemas gratuitos como Microsoft SQL Server Compact Edition y Sybase ASE. También se encuentran los sistemas comerciales que son muchos, entre ellos se puede citar: Microsoft Access, Microsoft SQL Server, Oracle, Progress, Magic. Open Base y muchos más.

1.5.4.1 Gestor de base de datos: PostgreSQL.

PostgreSQL es un servidor de base de datos objeto-relacional libre, liberado bajo la licencia BSD. Como muchos otros proyectos open source, el desarrollo de PostgreSQL no es manejado por una sola compañía sino que es dirigido por una comunidad de desarrolladores. [8].

Para la propuesta de solución se utiliza este gestor de base de datos por sus características, algunas de ellas son [8]:

- Alta concurrencia:

Mediante un sistema denominado MVCC (Acceso Concurrente Multiversión, por sus siglas en inglés) PostgreSQL permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos. Cada usuario obtiene una visión consistente de lo último a lo que se le hizo *commit*.

- Amplia variedad de tipos nativos:

PostgreSQL provee nativamente soporte para:

- Números de precisión arbitraria.
- Texto de largo ilimitado.
- Figuras geométricas (con una variedad de funciones asociadas)
- Direcciones IP (IPv4 e IPv6).
- Bloques de direcciones estilo CIDR.
- Direcciones MAC.
- Arreglos.

Adicionalmente los usuarios pueden crear sus propios tipos de datos, los que pueden ser por completo indexables gracias a la infraestructura GiST de PostgreSQL.

Otras características [8]:

- Claves ajenas también denominadas Llaves ajenas o Llaves Foráneas (foreign keys).
- Disparadores (triggers).

Un disparador o triggers se define en una acción específica basada en algo ocurrente dentro de la base de datos. En PostgreSQL esto significa la ejecución de un procedimiento almacenado basado en una determinada acción sobre una tabla específica.

- Vistas.
- Integridad transaccional.
- Herencia de tablas.
- Tipos de datos y operaciones geométricas.

¿Cuáles son las ventajas y razones de uso que presenta PostgresSQL?

PostgreSQL ofrece muchas ventajas respecto a otros sistemas de bases de datos [9]:

Instalación ilimitada

> Es frecuente que las bases de datos sean instaladas en más servidores de lo que permite la licencia. Algunos proveedores comerciales consideran a esto la principal fuente de incumplimiento de licencia. Con PostgreSQL, nadie puede demandar por violar acuerdos de licencia, puesto que no hay costo asociado a la licencia del software.

Estabilidad y confiabilidad legendarias

> En contraste a muchos sistemas de bases de datos comerciales, es extremadamente común que compañías reporten que PostgreSQL nunca ha presentado caídas en varios años de operación de alta actividad. Ni una sola vez. Simplemente funciona.

Extensible

> El código fuente está disponible para todos sin costo. Si su equipo necesita extender o personalizar PostgreSQL de alguna manera, pueden hacerlo con un mínimo esfuerzo, sin

costos adicionales. Esto es complementado por la comunidad de profesionales y entusiastas de PostgreSQL alrededor del mundo que también extienden PostgreSQL todos los días.

Multiplataforma

PostgreSQL está disponible en casi cualquier Unix (34 plataformas en la última versión estable), y una versión nativa de Windows está actualmente en estado beta de pruebas.

Diseñado para ambientes de alto volumen

PostgreSQL usa una estrategia de almacenamiento de filas llamada MVCC para conseguir una mejor respuesta en ambientes de grandes volúmenes. Los principales proveedores de sistemas de bases de datos comerciales usan también esta tecnología, por las mismas razones.

Herramientas gráficas de diseño y administración de bases de datos

Existen varias herramientas gráficas de alta calidad para administrar las bases de datos (pgAdmin, pgAccess) y para hacer diseño de bases de datos (Tora, Data Architect).

1.5.5 Servidor Web: Apache 2

Un servidor web es un programa que implementa el protocolo HTTP (hypertext transfer protocol). Este protocolo está diseñado para transferir lo que llamamos hipertextos, páginas web o páginas HTML (hypertext markup language): textos complejos con enlaces, figuras, formularios, botones y objetos incrustados como animaciones o reproductores de música. [10]

El servidor HTTP Apache es un software (libre) servidor HTTP de código abierto para plataformas Unix (BSD, GNU/Linux, etc.), Windows, Macintosh y otras, que implementa el protocolo HTTP/1.1 y la noción de sitio virtual. [11]

Apache presenta entre otras características mensajes de error altamente configurables, bases de datos de autenticación y negociado de contenido, pero fue criticado por la falta de una interfaz gráfica que ayude en su configuración.

Apache tiene amplia aceptación en la red: en el 2005, Apache es el servidor HTTP más usado, siendo el servidor HTTP del 48% de los sitios web en el mundo y decreciendo su cuota de mercado.

Ventajas de Apache [11]

- Modular
- Open source
- Multi-plataforma
- Extensible
- Popular (fácil conseguir ayuda/soporte)
- Gratuito

1.6 Proceso de Desarrollo

Con el objetivo de lograr la productividad del sistema se hace necesario un proceso que integre las múltiples fases del desarrollo del mismo. Para esto se define una metodología a seguir que en el caso de este trabajo es la metodología RUP por ser la más adecuada para el cumplimiento de los objetivos planteados.

El Proceso Unificado Racional (Rational Unified Process en inglés, habitualmente resumido como RUP) es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. La metodología esta comprende tres frases claves. Dirigido por los casos de uso, centrado en la arquitectura, iterativo e incremental. En lo referente a dirigido por los casos de uso está enfocado hacia el cliente y se utilizan, con algunas modificaciones tal vez, hasta la fase de pruebas. Una de las arquitecturas que está vigente es la de 3 capas, y esta se le debería adaptar los casos de uso. Las capas son: presentación, reglas de negocio y datos. [12]

El RUP no es un sistema con pasos firmemente establecidos, sino un conjunto de metodologías adaptables al contexto y necesidades de cada organización. [12]

Principales características de RUP [12]

- Forma disciplinada de asignar tareas y responsabilidades (quién hace qué, cuándo y cómo)
- Pretende implementar las mejores prácticas en Ingeniería de Software
- Desarrollo iterativo
- Administración de requisitos

- Uso de arquitectura basada en componentes
- Control de cambios
- Modelado visual del software
- Verificación de la calidad del software

El RUP es un producto de Rational (IBM). Incluye artefactos (que son los productos tangibles del proceso como por ejemplo, el modelo de casos de uso, el código fuente, etc.) y roles (papel que desempeña una persona en un determinado momento, una persona puede desempeñar distintos roles a lo largo del proceso).

1.7 Lenguaje de modelado

El lenguaje de modelado de objetos es un conjunto estandarizado de símbolos y del modo de disponerlos para modelar parte de un diseño de software orientado a objetos, cuyo mejor ejemplo es Unified Modeling Languaje (UML).

1.7.1 Unified Modeling Languaje (UML).

Lenguaje Unificado de Modelado (UML, por sus siglas en inglés, *Unified Modeling Language*) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; aún cuando todavía no es un estándar oficial, está respaldado por el OMG (Object Management Group). Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocios y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables.[13]

Es importante resaltar que UML es un "lenguaje" para especificar y no para describir métodos o procesos. Se utiliza para definir un sistema de software, para detallar los artefactos en el sistema y para documentar y construir. En otras palabras, es el lenguaje en el que está descrito el modelo. Se puede aplicar en una gran variedad de formas para dar soporte a una metodología de desarrollo de software (tal como el Proceso Unificado de Rational) -pero no especifica en sí mismo qué metodología o proceso usar. [13]

Entre sus características más generales se encuentran:

- Corrección de errores

- Tecnología orientada a objetos.
- Desarrollo iterativo e incremental.
- Participación del cliente en todas las etapas del proyecto.

1.8 Herramientas Utilizadas

1.8.1 Herramienta de desarrollo: Framework Code Igniter

Un framework simplifica el desarrollo de una aplicación mediante la automatización de algunos de los patrones utilizados para resolver las tareas comunes. Además, un framework proporciona estructura al código fuente, forzando al desarrollador a crear código más legible y más fácil de mantener. Por último, un framework facilita la programación de aplicaciones, ya que encapsula operaciones complejas en instrucciones sencillas. [14]

Code Igniter es un framework para PHP, está pensado para ofrecer un alto rendimiento, ser ligero y fácilmente instalable (puede usarse en un alojamiento compartido y no es necesario tener acceso a la línea de comando). [14]

Características más destacadas: [14]

1. Escritura escueta.
2. Rendimiento excelente.
3. Compatible con varias versiones de PHP y MySQL.
4. Casi no tiene configuraciones.
5. No requiere usar la línea de comandos para nada.
6. No requiere reglas de codificación estrictas.
7. No fuerza a aprender un lenguaje por plantillas, aunque es posible usarlas si las necesitas.
8. Sin complejidad, favoreciendo las soluciones simples.
9. Una documentación muy explicita

El Code Igniter tiene una característica llamada ActiveRecords que es una forma muy interesante de interactuar con un servidor de base de datos, permitiendo que se puedan obtener resultados de una BD (base de datos) sin muchos conocimientos de SQL. [14]

Otro punto muy importante es que el CodeIgniter es compatible con PHP4/5 y no necesita instalar nada en el servidor, es decir, se implementa, se carga un sistema con todos los archivos necesarios y ya el mismo funciona. [14]

1.8.2 Zend Studio

La versión que se utiliza en esta solución es Zend Studio 5 ya que es el soporte en desarrollos y pruebas de PHP con el set más completo de herramientas para la creación de aplicaciones altamente fiables. Asegura el desarrollo de software mediante la combinación del IDE líder para PHP con un entorno de prueba que agiliza la seguridad de la calidad, integración y las etapas de los procesos.

Zend Studio 5 se ha diseñado para una amplia gama de programadores y existen dos ediciones: Standard y Professional. Zend Studio 5, concebido con el fin de crear aplicaciones altamente fiables, proporciona una facilidad de uso inigualable, escalabilidad, fiabilidad, y la extensión que los programadores profesionales y de empresas requieren para desarrollar, distribuir, depurar y administrar aplicaciones PHP críticas de negocios.

Características de la herramienta que hacen de ella la opción más factible para el desarrollo de la propuesta de solución [15]:

- EL ENTORNO DE DESARROLLO MÁS PODEROSO PARA PHP:

• Integración del uso y completado de código perzonalizado de Zend Framework y vista de la lista de las funciones del framework desde la Visualización de Funciones PHP.

• Visualización los eventos de Zend Platform en una ventana de lista de eventos personalizada y dedicada.

• Aumenta la productividad con: Soporte PHP 5 completo, Analizador de Código, carpeta de Código, completado de Código, coloreado de Sintaxis, Administrador de Proyecto, Editor de Código, Depurador de gráficos y asistentes.

• Documentación del código de forma más sencilla, aplicaciones, y proyectos con PHPDocumentor, la herramienta de documentación standard para PHP.

• Simplifica el despliegue con la integración FTP y SFTP de forma tal que permita a los programadores en forma segura subir y descargar archivos de proyectos de modo transparente hacia y desde servidores remotos.

DESARROLLO DE APLICACIONES DE NEGOCIOS SUPERIORES

• Conectarse directamente con la bases de datos profesionales más utilizadas tales como IBM DB2/Cloudscape/ Derby/, MySQL, Oracle, Microsoft SQL Server, PostgreSQL y SQlite.

• Permite escribir y realizar consultas a servidores conectados usando el editor de consultas SQL de Zend con SQL92 y soporte de coloreado de Sintaxis.
• Permite visualizar las estructuras de la base de datos y administrar el contenido con el explorador SQL de Zend.
ENTORNO DE DEPURACIÓN COMPLETO PARA PHP
Características de depuración avanzadas, incluyendo: condiciones límites, visualización de errores, vistas avanzadas, variables y buffer de salida.
• Depura en forma local y remota en un medio conocido utilizando el depurador PHP más poderoso.
• Asegura la protección máxima de ubicaciones de proyectos o en Internet con depuradores remotos seguros.
• Depura y analiza su código directamente desde el nuevo Navegador IE (internet explorer) con “un click en el depurador de navegador”.
EQUIPO DE DESARROLLO Y HERRAMIENTAS DE PRUEBA SÓLIDAS
• Facilita el desarrollo y colaboración en equipo mediante la administración efectiva de su código fuente utilizando CVS o Subversión directamente desde Zend Studio.
• Obtiene una respuesta inmediata en forma de código o en rendimiento script cuando utiliza Zend Studio 5 Control de Calidad/ herramientas de prueba.
• Entrega aplicaciones altamente fiables utilizando herramientas PHP Intelligence que lo ayudarán a solventar las debilidades de la aplicación antes de que aparezcan en la producción.

1.8.3 Herramienta Case: Rational Rose

Las Herramientas CASE (Computer Aided Software Engineering, Ingeniería de Software Asistida por Ordenador) 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 de dinero. Estas herramientas nos pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del software en tareas como el proceso de realizar un diseño del proyecto, cálculo de costes, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores, entre otras. Dentro de estas herramientas se encuentra el Rational Rose que es la mejor y por tanto más utilizada en el mundo y la que se utiliza en el modelado de esta propuesta de solución. [16]

Rational Rose es la herramienta CASE que comercializan los desarrolladores de UML y que soporta de forma completa la especificación del UML 1.1. [17]

Esta herramienta propone la utilización de cuatro tipos de modelo para realizar un diseño del sistema, utilizando una vista estática y otra dinámica de los modelos del sistema, uno lógico y otro físico. Permite crear y refinar estas vistas creando de esta forma un modelo completo que representa el dominio del problema y el sistema de software. [17]

Algunas características: [17]

Desarrollo Iterativo

Rational Rose utiliza un proceso de desarrollo iterativo controlado (controlled iterative process development), donde el desarrollo se lleva a cabo en una secuencia de iteraciones.

Trabajo en Grupo

Rose permite que haya varias personas trabajando a la vez en el proceso iterativo controlado, para ello posibilita que cada desarrollador opere en un espacio de trabajo privado que contiene el modelo completo y tenga un control exclusivo sobre la propagación de los cambios en ese espacio de trabajo.

Generador de Código

Se puede generar código en distintos lenguajes de programación a partir de un diseño en UML.

Ingeniería Inversa

Rational Rose proporciona mecanismos para realizar la denominada Ingeniería Inversa, es decir, a partir del código de un programa, se puede obtener información sobre su diseño.

Se utiliza el Rational Rose Enterprise Edition 2003, para sustentar la documentación, como modelador visual de la notación UML (Unified Modeling Language) para la confección de los diagramas.

1.8.4 EMS PostgreSQL

EMS SQL Management Studio para PostgreSQL es una solución integral para la administración y desarrollo de bases de datos. Con componentes que hacen foco especialmente a tareas más críticas de administración de bases de datos, SQL Studio es una simple plataforma de trabajo que te provee herramientas esenciales para administrar bases de datos y gestionar sus objetos tales como la migración de bases de datos, extracción, importación, exportación y comparación de datos. SQL

Studio integra estas herramientas en un entorno potente y sencillo, que pueden trabajar en cualquier momento.

¿Por qué utilizar EMS PostgreSQL?

EMS PostgreSQL posee características adicionales que responden a las necesidades de los usuarios, como son la ejecución manual de utilitarios y servicios, Almacenamiento de plantillas en un solo repositorio, lista común de origen de datos para todos los servicios y utilidades, creación de tareas complejas y ejecución de tareas programadas, capacidad para correr aplicaciones externas desde el programador horario (scheduler), almacenamiento de todos los sucesos (logs) de ejecución de tareas en una base de datos, creación de notificaciones personalizadas con una gran variedad de opciones de entrega, capacidad para configurar el estilo de la interfaz visual únicamente para las aplicaciones de SQL Studio y acceso rápido al Servicio de Soporte Técnico de EMS.

1.9 Conclusiones

En el desarrollo de este capítulo se muestra la necesidad actual de implementar el sistema que se propone como solución al problema planteado, sobre todo en la universidad. Se determinan las tecnologías que se utilizan en el desarrollo del sistema, definiéndose finalmente que la tecnología a utilizar es PHP 5 en el desarrollo de páginas dinámicas, utilizando como herramienta de desarrollo el framework Code Igniter y como el Gestor de Base de Datos más apropiado al sistema que da solución al problema, PostgreSQL. Se define además que se va a utilizar la metodología RUP en el proceso de desarrollo de software por su orientación a objetos y el modelado visual en UML, permitiendo al proceso de desarrollo del sistema un control de cambios y requerimientos. De forma general en el capítulo se exponen las condiciones y problemas que rodean al objeto de estudio, o sea en esta sección se fundamentan teóricamente las razones por las que hay que crear un sistema de gestión de sesiones que cumpla con los requisitos pertinentes.

2 Capítulo

Características del sistema.

2.1 Introducción.

En este capítulo se describe la propuesta de solución para la situación problémica planteada, se muestran los procesos del negocio mediante el Modelo de Dominio y se brinda una visión general del sistema propuesto, realizando el análisis y definiendo las reglas del negocio, además de especificar las funcionalidades y características que el sistema a desarrollar debe tener, que son los requisitos del software. Se muestra el Diagrama de Casos de Uso del Sistema y la especificación para cada Caso de Uso respectivamente.

2.2 Problema y situación problémica

En nuestra universidad los usuarios cada vez que van a acceder a un recuso de la red deben repetir el proceso de autenticación una y otra vez, gastando mayor tiempo y trabajo. Ni en la UCI ni en el país se tiene conocimiento de que exista sistema alguno que evite esta cuestión y que a su vez fortalezca los niveles de seguridad de la organización.

2.3 Objeto de automatización

Implementar un sistema de este tipo evitaría todas estas molestias y ahorraría tiempo. El Sistema de Gestión de Sesiones solo se encargará del proceso de gestión de las sesiones de los usuarios, logrará mejorar la seguridad de la red de nuestra universidad, y a su vez permitirá a los usuarios disminuir su trabajo de autenticación y el tiempo que se pierde al realizar este proceso repetidas veces.

2.4 Propuesta de sistema

Se propone un sistema que gestione las sesiones de los usuarios, mediante procesos invisibles a los ojos de los mismos, de manera que solo tengan que realizar un único proceso de autenticación. El sistema contará con una fachada de servicios web que será capaz de gestionar la apertura y cierre de las sesiones de cada uno de los usuarios del dominio uci.cu.

2.5 Descripción del negocio

RUP define en su primera fase de desarrollo la realización del modelo de negocio, con el objetivo de comprender el entorno del cliente y detectar las mejoras potenciales en los procesos de la organización. Cuando no es posible identificar claramente los procesos del negocio, RUP propone realizar un modelo del dominio, que es un subconjunto del modelo de negocio.

Un modelo de dominio captura los tipos más importantes de objetos en el contexto del sistema. Los objetos del dominio representan las cosas que existen o eventos que ocurren en el entorno en el que trabaja el sistema. El modelo de dominio se describe específicamente mediante diagramas de clases, utilizando el lenguaje de modelado UML. [20].

2.6 Modelo de dominio

Se hace Modelo de Dominio porque cuando se hace el análisis del negocio resulta imposible obtener actores, trabajadores y la especificación detallada de los procesos que se llevan a cabo en el Sistema de Gestión de Sesiones, es decir, no están bien definidos.

El Modelo de Dominio (o Modelo Conceptual) es una representación visual de los conceptos u objetos del mundo real significativos para un problema o área de interés. Representa clases conceptuales del dominio del problema. Representa conceptos del mundo real, no de los componentes del software.

2.6.1 Glosario de Términos.

El propósito de este glosario es para dar respuesta a las dudas que se le pueda presentar al usuario a la hora de navegar por el sitio, o sea palabras o términos que no sean de su conocimiento. El glosario tiene un gran alcance, ya que por su utilidad se considera un elemento muy importante en el contenido del sitio. También a través del glosario el cliente puede obtener mayor conocimiento acerca del software, documentarse, en fin, alcanzar una amplia información sobre los términos utilizados en el sitio antes mencionado y que además le podrían causar dificultad al navegar por el mismo.

2.6.1.1 SGS: Sistema de Gestión de Sesiones

Single sign-on (SSO) es un procedimiento de autenticación que habilita al usuario para acceder a varios sistemas con una sola instancia de identificación.

2.6.1.2 Token

El token no es más que el es ticket que se le da al usuario al crear su sesión en el sistema, este guardará su hora y fecha de creado, el id del usuario, el TTL de la sesión y el id de la aplicación a la cual accede.

2.6.1.3 TTL Sesión: Time To Live

El "Time To Live" o Tiempo de Vida, es el tiempo de duración de una sesión de usuario en el sistema. Al cerrarse la sesión del sistema, el usuario debe repetir el proceso de autenticación. El TTL Sesión es predeterminado por el administrador del sistema.

2.6.1.4 TTL Aplicación

Es el tiempo de duración de cada aplicación en cada token o sesión del sistema. Mientras que haya una aplicación abierta con tiempo de vida, no expira el token, siempre y cuando este tenga todavía TTL de Sesión. Si a una aplicación se le acaba su TTL de aplicación y hay otra que todavía tiene TTL de aplicación, entonces se desactiva la misma, siempre y cuando el TTL Sesión no haya expirado todavía. Este TTL también lo define el administrador del sistema.

2.6.1.5 TTL Salvas del Historial:

Es el tiempo que estarán guardados las trazas de los usuarios en el sistema. Lo define también el administrador del sistema.

2.6.1.6 Traza:

Acciones que realiza un usuario, aplicaciones que ejecuta en un orden y en una fecha determinada.

2.7 Visión

El propósito general de la visión es brindar una perspectiva del dominio, y el modelo planteado para dar la solución al problema.

Este sistema está encaminado a resolver un problema real, y se pretende un riguroso cumplimiento de los todos los requisitos propuestos, aspirando a la máxima calidad y satisfacción.

2.8 Reglas del Negocio

- ✓ Un usuario no puede configurar el sistema.
- ✓ Solo el administrador podrá configurar el sistema.
- ✓ Un usuario solo podrá ver su información.
- ✓ El administrador podrá ver y eliminar la información referente a todos sus los usuarios.

2.9 Modelo de Objetos

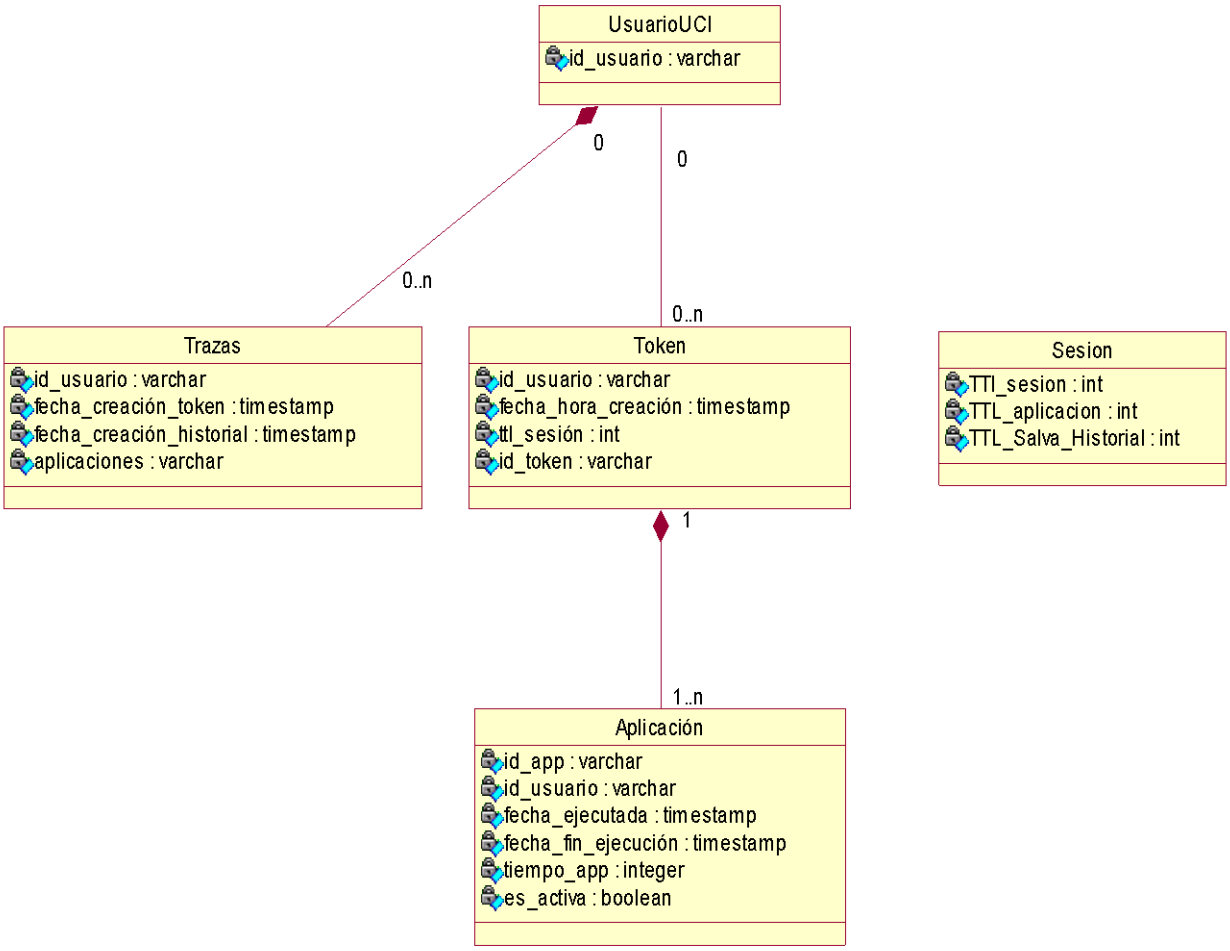

Imagen 2. 1: Modelo de objetos

2.10 *Modelación del sistema.*

Especificación de los requisitos del software:

Todas las ideas que los clientes, usuarios y miembros del equipo de proyecto tengan acerca de lo que debe hacer el sistema, deben ser analizadas como candidatas a requisitos. Los requisitos se pueden clasificar en: funcionales y no funcionales. Los requerimientos funcionales son capacidades o condiciones que el sistema debe cumplir, en cambio, los no funcionales se refiere a cualidades del sistema.

2.10.1 Requerimientos funcionales

En la realización de los casos de uso del negocio, se obtienen las actividades que serán objeto de automatización. Estas actividades no son exactamente los requerimientos funcionales, pero si son el punto de partida para identificar qué debe hacer el sistema.

Los requerimientos funcionales no alteran la funcionalidad del producto, esto quiere decir que los requerimientos funcionales se mantienen invariables sin importarle con que propiedades o cualidades se relacionen. Los requerimientos no funcionales también añaden funcionalidad al producto, pues hacen que un producto sea fácil de usar, seguro, o interactivo demanda cierta cantidad de procesamiento. Sin embargo la razón fundamental de que esta funcionalidad sea parte del producto es brindarle a este las características deseadas.

Para el caso del Sistema de Gestión de Sesiones, podrían definirse como requerimientos funcionales, entre otros, a los siguientes:

R1. Gestionar sesiones.

R1.1. Crear Sesión.

R1.2. Guardar Sesión.

R1.3. Actualizar Sesión.

R1.4. Cerrar Sesión.

R2. Registrar estadísticas.

R3. Visualizar estadísticas de usuario.

R4. Autenticar Usuario en la aplicación Web del sistema.

R5. Modificar TTL Aplicación, TTL Sesión y TTL de Salva del historial.

R6. Visualizar estadísticas generales.

R7. Visualizar y cerrar token.

2.10.2 Requerimientos no funcionales

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable. En muchos casos los requerimientos no funcionales son fundamentales en el éxito del producto. Normalmente están vinculados a requerimientos funcionales, es decir una vez se conozca lo que el sistema debe hacer podemos determinar cómo ha de comportarse, qué cualidades debe tener o cuán rápido o grande debe ser.

Los requerimientos no funcionales forman una parte significativa de la especificación. Son importantes para que clientes y usuarios puedan valorar las características no funcionales del producto, pues si se conoce que el mismo cumple con la toda la funcionalidad requerida, las propiedades no funcionales, como cuán usable, seguro, conveniente y agradable, pueden marcar la diferencia entre un producto bien aceptado y uno con poca aceptación.

Los requerimientos no funcionales incluyen:

- Conjunto de facilidades.
- Capacidades.
- Seguridad.

Existen múltiples categorías para clasificar a los requerimientos no funcionales, a continuación se muestran los requerimientos no funcionales que el Sistema de Gestión de Sesiones necesita:

2.10.2.1 Requerimientos de apariencia o interfaz externa

Este tipo de requerimiento describe la apariencia del producto. El sistema esta constituido por un servicio web y por una aplicación, el servicio web evidentemente no cuenta con una interfaz externa. La aplicación por su parte tendrá una interfaz que será fácil de usar por el administrador del sistema y por los usuarios para que estos puedan ver todas las estadísticas que estarán a su disposición y realizar sus respectivas funcionalidades. Será un interfaz comprensible, amigable, profesional y fácil de usar para aquellos que no sean muy ágiles en el uso de la computadora. Además no contiene muchas

imágenes para que la respuesta al usuario sea rápida que es la característica fundamental del sistema. Esta interfaz además será adaptable a los patrones de diseño de la UCI.

2.10.2.2 Requerimientos de Rendimiento

Imponen condiciones a los Requerimientos Funcionales. Por ejemplo, para una acción especifica pueden definirse parámetros tales como: Velocidad de procesamiento o cálculo, Eficiencia, Disponibilidad, Precisión, Tiempo de respuesta, Tiempo de recuperación y Aprovechamiento de los recursos. La necesidad de velocidad debe ser genuina, el éxito del producto puede depender de la velocidad, y, en el caso de los sistemas en tiempo real, muchos de los requerimientos de rendimientos son indispensables para su funcionamiento como es el caso del SGS, cuya eficiencia va a estar dada por la velocidad de procesamiento, disponibilidad y su tiempo de respuesta que debe de ser inmediato.

2.10.2.3 Requerimientos de Soporte

Este producto de software podrá ser usado en diferentes plataformas, por ejemplo: Windows, Linux, no tiene ningún tipo de requisitos especial en cuanto a soporte, solo depende del contrato que establezca con el cliente.

2.10.2.4 Requerimientos de Seguridad

Este es quizás el tipo de requerimiento más difícil, que provocará los mayores riesgos si no se maneja correctamente. La seguridad puede ser tratada en tres aspectos diferentes:

- ***Confidencialidad:*** La información manejada por el sistema debe estar protegida de acceso no autorizado y divulgación.

- ***Integridad:*** La información manejada por el sistema será objeto de cuidadosa protección contra la corrupción y estados inconsistentes, de la misma forma será considerada igual a la fuente o autoridad de los datos. Pueden incluir también mecanismos de chequeo de integridad y realización de auditorías.

- ***Disponibilidad:*** Significa que el o los administradores del sistema se les garantizará el acceso a la información y que los dispositivos o mecanismos utilizados para lograr la seguridad no ocultarán o retrasarán a los usuarios para trabajar en sus respectivas sesiones.

- ***Seguridad:*** La seguridad de un sistema no solo tiene en cuenta la seguridad del sistema propiamente dicho sino, además, el ambiente en el que se usará el sistema., por lo que se tiene que

contemplar la seguridad física del lugar donde se usa la aplicación, los controles administrativos que se establecen de acceso al sistema y las regulaciones legales que afecta o determinan el uso del sistema y que serán tenidas en cuenta si se incumple.

La seguridad es un requerimiento no funcional que genera posiblemente requerimientos funcionales. Esto depende de la propuesta de seguridad que se tenga para el sistema.

Supongamos que la propuesta incluye que a la aplicación solo tienen acceso determinando administrador a determinadas opciones del sistema. Esto implica que el sistema tiene que permitir:

1. Validar el ingreso al sistema de un usuario.
2. Actualizar sesiones.
3. Actualizar perfiles de usuario del sistema.
4. Asignar perfil a usuario del sistema.
5. Asignar opciones a perfil.
6. Renovar certificados (tokens).

2.10.2.5 Requerimientos de confiabilidad

Estos requerimientos caracterizan la respuesta del sistema ante los fallos o indican cuán robusto de este. El sistema debe estar disponible en todo momento permitiendo el trabajo de los usuarios y las acciones de mantenimiento, previendo también los siguientes posibles factores:

- Frecuencia y severidad de los fallos.
- Protección contra fallos.
- Recuperación.
- Predicción de fallos.
- Tiempo medio entre fallos.

2.10.2.6 Requerimientos de Software

Se debe disponer de un Sistema Operativo Windows 95 o Superior, Mac OS, Linux, entre otros, y un servidor de base de datos con el Gestor Apache2.

2.10.2.7 Requerimientos de Hardware

Se requiere un procesador Pentium 3 o superior, memoria RAM de 512 MB o superior.

El servidor debe tener las siguientes características: capacidad de disco duro superior a 80.0 GB, microprocesador Pentium IV superior a 2.0 GHz y como mínimo 1.0 GB de RAM.

2.10.2.8 Extensibilidad

Se debe lograr un diseño adaptable, con la capacidad de poder soportar funcionalidades adicionales o modificar las funcionalidades existentes sin impactar el resto de los requerimientos contemplados en el sistema.

2.10.2.9 Restricciones en el diseño y la implementación

El componente debe brindar la posibilidad de ser configurado según el cliente donde se instale. Su diseño debe ser flexible y adaptable a nuevos requisitos de tecnología.

El análisis y diseño de la aplicación será basado en la Metodología RUP con el uso del lenguaje de modelado UML.

Se usará como herramienta Rational Rose para el modelado de los artefactos que se generan en cada uno de los flujos de trabajo definidos por RUP.

Se usará como lenguaje de programación PHP 5.

El framework de desarrollo a utilizar será Code Igniter.

Se utilizarán herramientas de desarrollo que garanticen la calidad de todo el ciclo de desarrollo del producto.

2.10.2.10 Rendimiento

Se trata de garantizar la rapidez de respuesta del componente ante las solicitudes de las aplicaciones, al igual que la velocidad de procesamiento. Para lo cual recibe los datos validados y los envía al servidor. Lográndose una mayor velocidad de procesamiento, y un mayor aprovechamiento de los recursos.

2.10.2.11 Ayuda

Debe estar totalmente documentado, y contar con una guía de ayuda para el cliente.

2.11 Definición de los Casos de Uso del Sistema

2.11.1 Actores del sistema

Actores	Justificación
Usuario UCI	Es quien hace las llamadas al sistema, e inicializa casos de uso tales como “Gestionar Sesiones”, dando paso a la creación, actualización y almacenamiento de su respectiva sesión dentro del sistema. También puede visualizar sus estadísticas como usuario.
Administrador del sistema	Es el encargado de configurar el sistema, gestionando usuarios y roles, además de los tiempos de vida de las sesiones y aplicaciones. También tiene como función configurar los certificados o tokens, y tiene acceso a la visualización de las estadísticas generales, entiéndase por esto último, todos los registros de cada usuario en el sistema.

Tabla 2. 1: Justificación de actores del sistema

2.11.2 Diagrama de Casos de Uso del Sistema

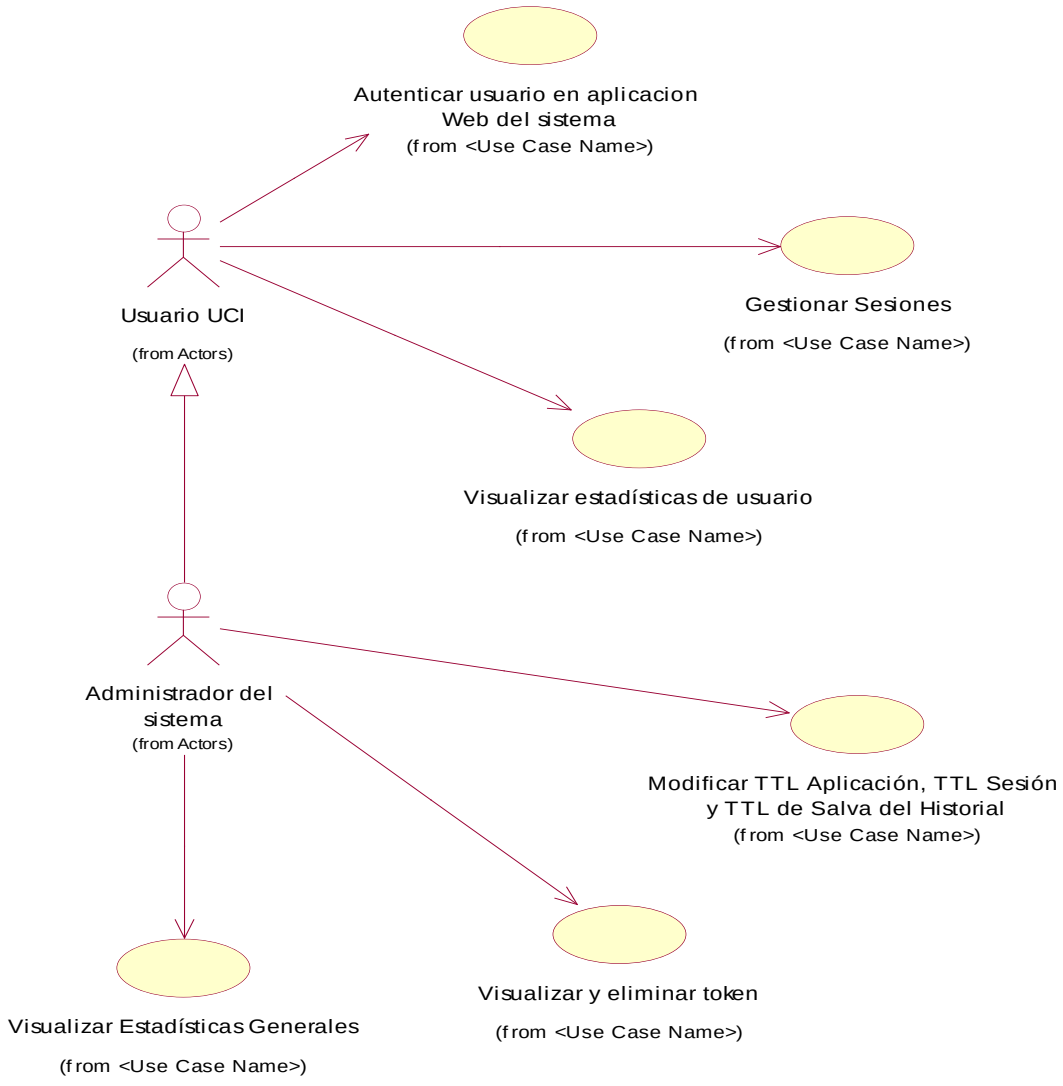

Imagen 2. 2: Diagrama de casos de uso del sistema

2.11.3. Descripción de los casos de uso (Anexo I).

2.12 Conclusiones

En este capítulo fueron elaborados y descritos brevemente algunos de los artefactos propuestos por RUP para el desarrollo de software tales como el modelo de dominio, definición de los requisitos funcionales y no funcionales, diagrama de casos de uso del sistema y la descripción textual de los casos de uso del sistema.

3 Capítulo

Análisis y diseño del sistema.

3.1 Introducción

El presente capítulo tiene como objetivo principal el desarrollo del flujo de trabajo de Análisis y Diseño de la propuesta de solución para lograr un entendimiento del funcionamiento futuro del sistema en general. Se describe como será realizado el Sistema de Gestión de Sesiones a partir de los requisitos vistos en el capítulo 2, indicando lo que se debe programar. Se define el modelo de análisis llevándose a cabo la realización del modelo de clases del análisis. Además a partir de dichos modelo se despliega el modelo de diseño que describe con precisión las clases del sistema, con las especificaciones del lenguaje a utilizar. Se detalla el flujo de eventos de cada caso de uso anteriormente descrito en sus diagramas de secuencia, en sus flujos normales y alternativos. En este capítulo se realiza también el diseño de la base de datos del sistema, desplegado en su diagrama de entidad- relación y la descripción de sus tablas. Por último se abordarán temas tales como las definiciones de diseño, el tratamiento de errores, la seguridad, interfaz gráfica del sistema y las concepciones de ayudas de la propuesta de solución.

3.2 Análisis

Durante el análisis se analizan los requisitos obtenidos y descritos en el capítulo anterior, su objetivo principal es conseguir una comprensión más precisa de los requisitos y una comprensión de los mismos que sea fácil de mantener y ayude a estructurar la propuesta de solución.

La estructura de los requisitos que proporciona el análisis también sirve como entrada fundamental para dar forma al sistema en su totalidad (incluyendo su arquitectura), esto es debido al propósito de construir el sistema como un todo mantenible y no sólo describir sus requisitos.[20]

3.2.1 Modelo conceptual de clases del análisis

Analizar los requisitos en forma de modelo de análisis es muy importante por varios motivos, algunas de sus características son [20]:

- ✓ Descrito con el lenguaje del desarrollador.
- ✓ Vista interna del sistema.
- ✓ Estructurado por clases y paquetes estereotipados, proporciona la estructura de la vista interna del sistema.
- ✓ Utilizado por los desarrolladores al comprender como debería darse forma al sistema, es decir como debería ser diseñado e implementado.
- ✓ No debería tener redundancias, inconsistencias, etc., entre requisitos.
- ✓ Esboza como llevar a cabo la funcionalidad dentro del sistema incluida la funcionalidad significativa para la arquitectura, sirve como una aproximación al diseño.
- ✓ Define realización de casos de uso y cada una de ellas representa el análisis de un caso de uso del modelo de casos de uso.

3.2.2 Clases del análisis

Las clases del análisis representa una atracción de una o varias clases y/o subsistemas del diseño del sistema. Se centran en el tratamiento de los requisitos funcionales y los no funcionales, denominándolos requisitos especiales, hasta llegar a las actividades de diseño e implementación subsiguientes, haciendo que estas clases sean conceptuales, de mayor granulidad que sus contrapartidas en el diseño e implementación. [20]

Una clase del análisis raramente define u ofrece una interfaz en términos de operaciones y de sus signaturas. Sus comportamientos se definen mediante responsabilidades en un nivel más alto y menos formal (una responsabilidad es una descripción textual de un conjunto cohesivo del comportamiento de una clase). Adema define atributos aunque éstos son de un nivel bastante alto y sus tipo por lo general son conceptuales y reconocibles en el dominio del problema (estos pasan a ser con frecuencia clases en el diseño y en la implementación). [20]

Dichas clases participan en relaciones más conceptuales en sus contrapartidas en el diseño y la implementación. También encajan siempre dentro de uno de tres estereotipos: de interfaz, control o entidad.

3.2.2.1 Clases de interfaz

Las clases interfaz se utilizan para modelar la interacción entre el sistema y sus actores (usuarios en este caso), a menudo implica recibir, y presentar, información y peticiones de, y hacia, dichos actores. Modelan las partes del sistema que dependen de sus actores, clarificando y reuniendo los requisitos en los límites del sistema. Un cambio en una interfaz usuario queda aislado en una, o más, clases interfaz. [20]

3.2.2.2 Clases de entidad

Las clases entidad se utilizan para modelar información que posee una vida larga y es a menudo persistente, modelan la información y el comportamiento asociado de algún fenómeno o concepto, como una persona, un objeto del mundo real o un suceso del mundo real. Además reflejan la información de un modo que beneficia a los desarrolladores a diseñar e implementar el sistema, incluyendo su soporte de persistencia. [20]

3.2.2.3 Clases de control

Las clases de control representan coordinación, secuencia y transacciones y control de otros objetos y se usan con frecuencia para encapsular el control de un caso de uso en concreto. [20]

Los aspectos dinámicos del sistema se manejan con clases de control debido a que ellas manejan y coordinan las acciones y los flujos de control principales y delegan trabajo a otros objetos (objetos de interfaz y de entidad).[20]

3.2.3 Diagramas de clases del análisis

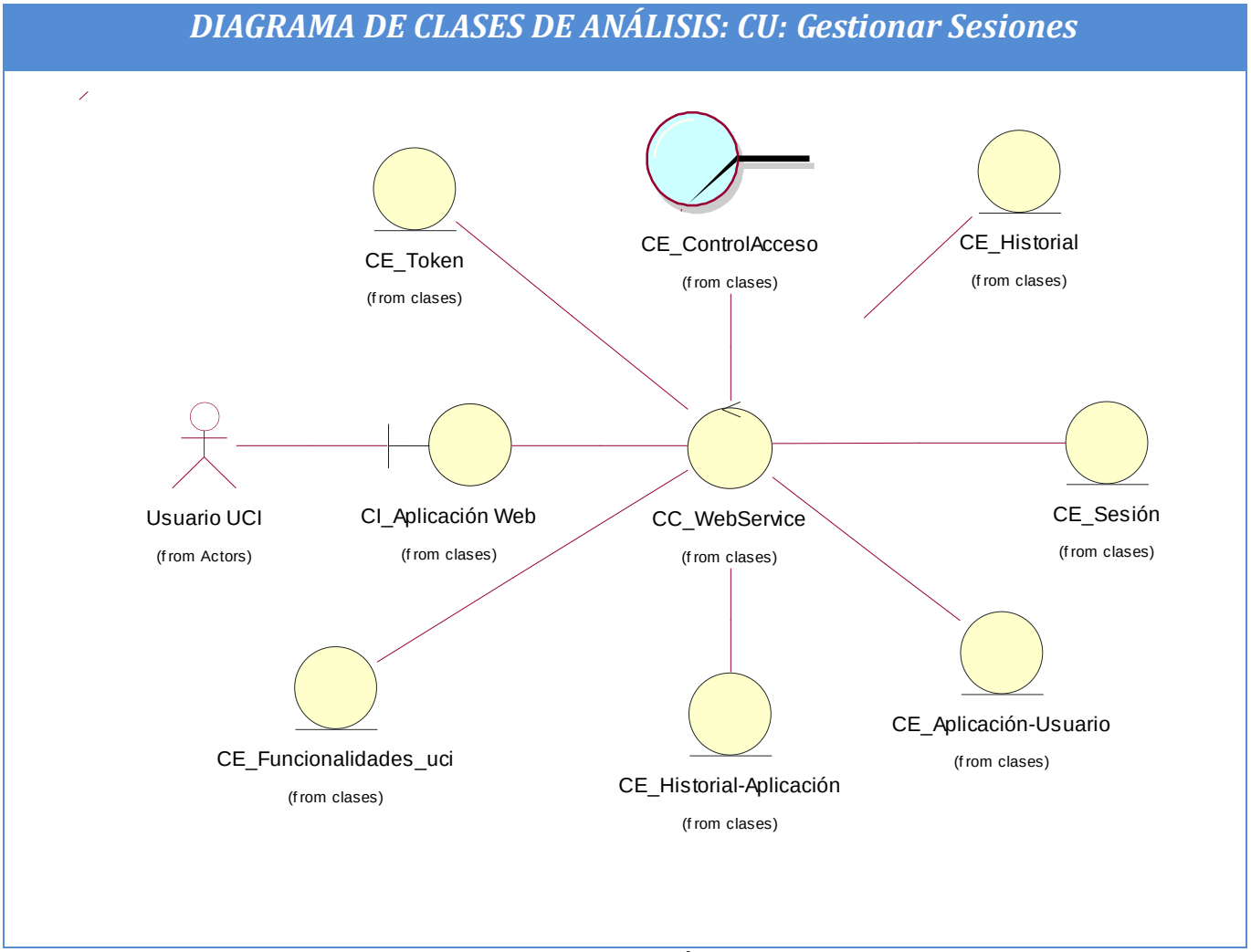

Imagen 3. 1: DIAGRAMA DE CLASES DE ANÁLISIS: CU: Gestionar Sesiones

DIAGRAMA DE CLASES DE ANÁLISIS: CU: Visualizar estadísticas de usuario

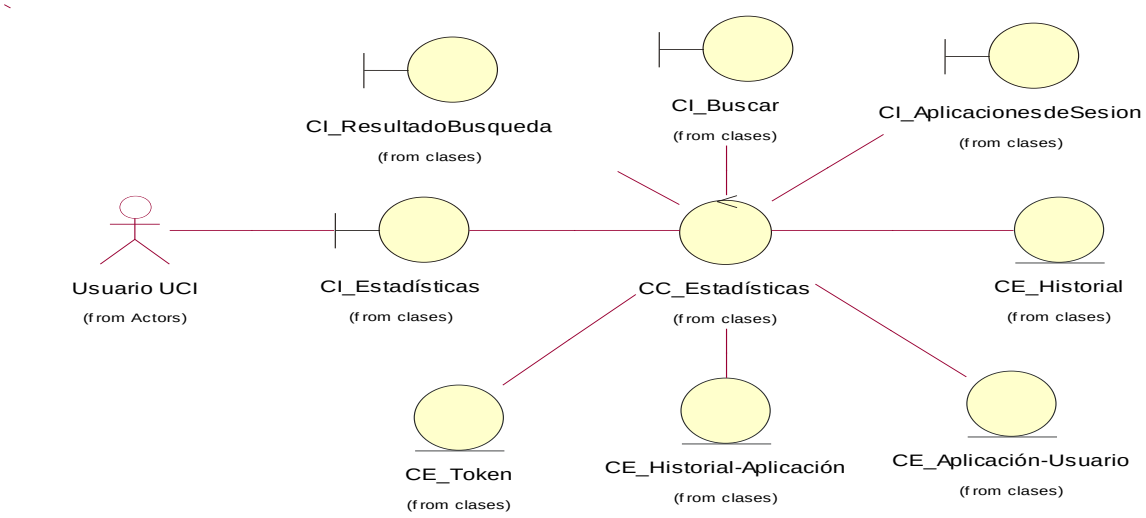

Imagen 3. 2: DIAGRAMA DE CLASES DE ANÁLISIS: CU: Visualizar estadísticas de usuario

DIAGRAMA DE CLASES DE ANÁLISIS: CU: Autenticar usuario en la aplicación Web del sistema.

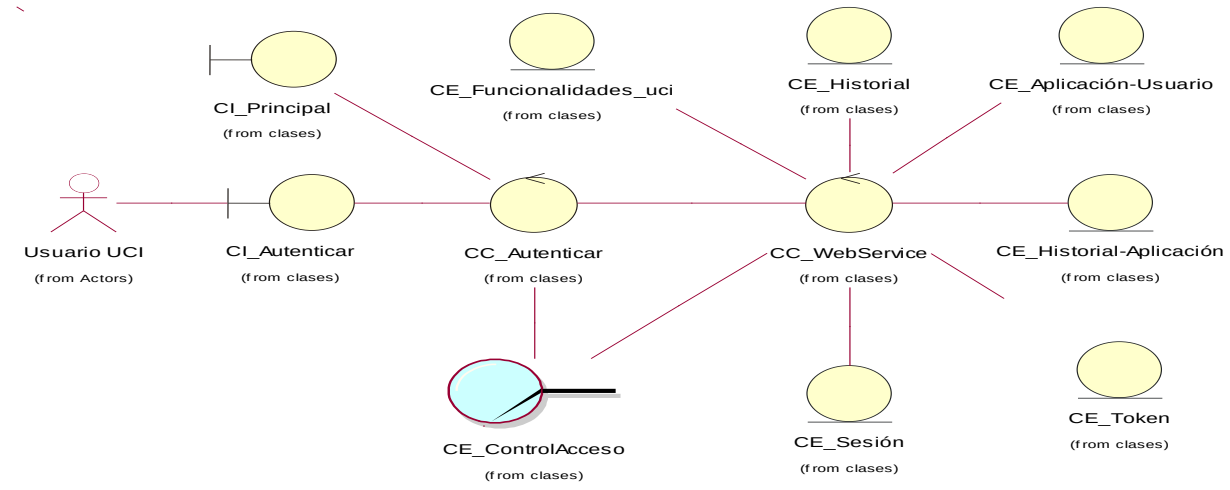

Imagen 3. 3: DIAGRAMA DE CLASES DE ANÁLISIS: CU: Autenticar usuario en la aplicación Web del sistema

DIAGRAMA DE CLASES DE ANÁLISIS: CU: Visualizar y cerrar token

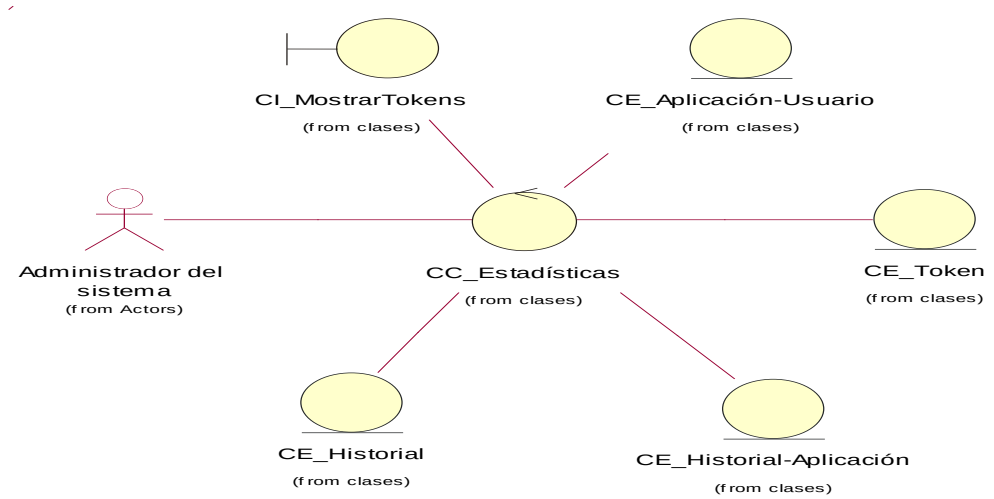

Imagen 3. 4: DIAGRAMA DE CLASES DE ANÁLISIS: CU: Visualizar y cerrar token

DIAGRAMA DE CLASES DE ANÁLISIS: CU: Modificar TTL Aplicación, TTL Sesión y TTL de Salva del Historial

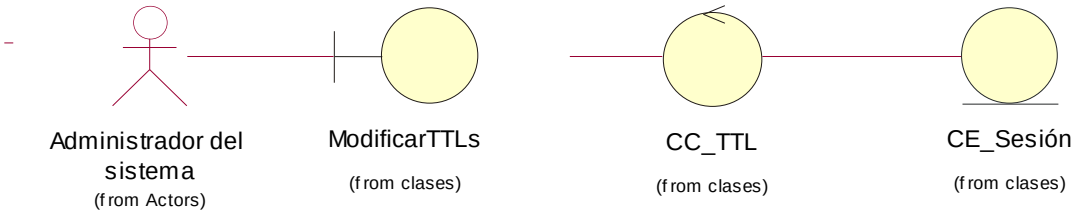

Imagen 3. 5: DIAGRAMA DE CLASES DE ANÁLISIS: CU: Modificar TTL Aplicación, Sesión y de Salva del Historial

DIAGRAMA DE CLASES DE ANÁLISIS: CU: Visualizar estadísticas generales

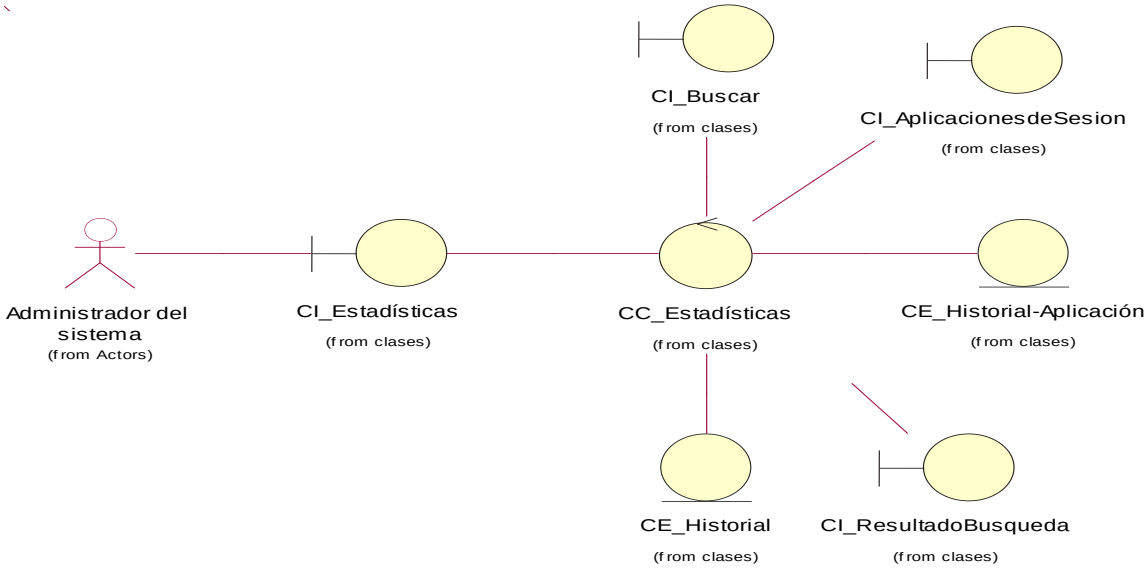

Imagen 3. 6: DIAGRAMA DE CLASES DE ANÁLISIS: CU: Visualizar estadísticas generales

3.3 Diseño

El modelo de diseño es un modelo de objetos que describe la realización física de los casos de uso, centrándose en cómo los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de programación, tienen impacto en el sistema a considerar, siendo una entrada fundamental para las actividades de implementación.

3.3.1 Diagramas de Interacción (Anexo II)

Los diagramas de secuencia muestran la sucesión de mensajes entre objetos durante un escenario determinado. Cada objeto viene dado por una barra vertical y el tiempo transcurre de arriba hacia abajo. Estos diagramas constituyen una guía importante para el programador; a continuación los diagramas de secuencia de la aplicación:

Ver diagramas de interacción **Anexo II**.

3.3.2 Diagramas de clases del diseño (Diagrama de clases Web).

La forma tradicional de modelar clases no es aplicable a la hora de diseñar una aplicación Web, es por esto que Rational creó una extensión para UML que se adapta a la arquitectura de este tipo de sistemas.

El diseño es el centro de atención al final de la fase de elaboración y el comienzo de las iteraciones de construcción. Este contribuye a una arquitectura estable y sólida y a crear un plano del modelo de implementación. [20]

En el diseño se modela el sistema y encontramos su forma (incluida la arquitectura) para que soporte todos los requisitos incluyendo los requisitos no funcionales y otras restricciones- que se le supone. En concreto, los propósitos del diseño son [20]:

- ✓ Adquirir una comprensión en profundidad de los aspectos relacionados con los requisitos no funcionales y restricciones relacionadas con los leguajes de programación, componentes reutilizables, sistemas operativos, tecnologías de distribución y concurrencia, tecnologías de interfaz de usuario, tecnologías de interfaz de transacciones, etc.
- ✓ Crear una entrada apropiada y un punto de partida para actividades de implementación subsiguientes capturando los requisitos o subsistemas individuales, interfaces y clases.
- ✓ Ser capaces de descomponer los trabajos de implementación en partes mas manejables que puedan ser llevadas a cabo por diferentes equipos de desarrollo, teniendo en cuenta la posible concurrencia. Esto resulta útil en los casos en los que la descomposición no puede ser hecha basándose en los resultados de la captura de requisitos.
- ✓ Capturar las interfaces entre los subsistemas antes del ciclo de vida del software.
- ✓ Ser capaces de visualizar y reflexionar sobre el diseño utilizando una notación común.
- ✓ Crear una abstracción de la implementación del sistema.

A continuación los diagramas de clases del diseño:

DIAGRAMA DE CLASES DEL DISEÑO: CU: Gestionar Sesiones

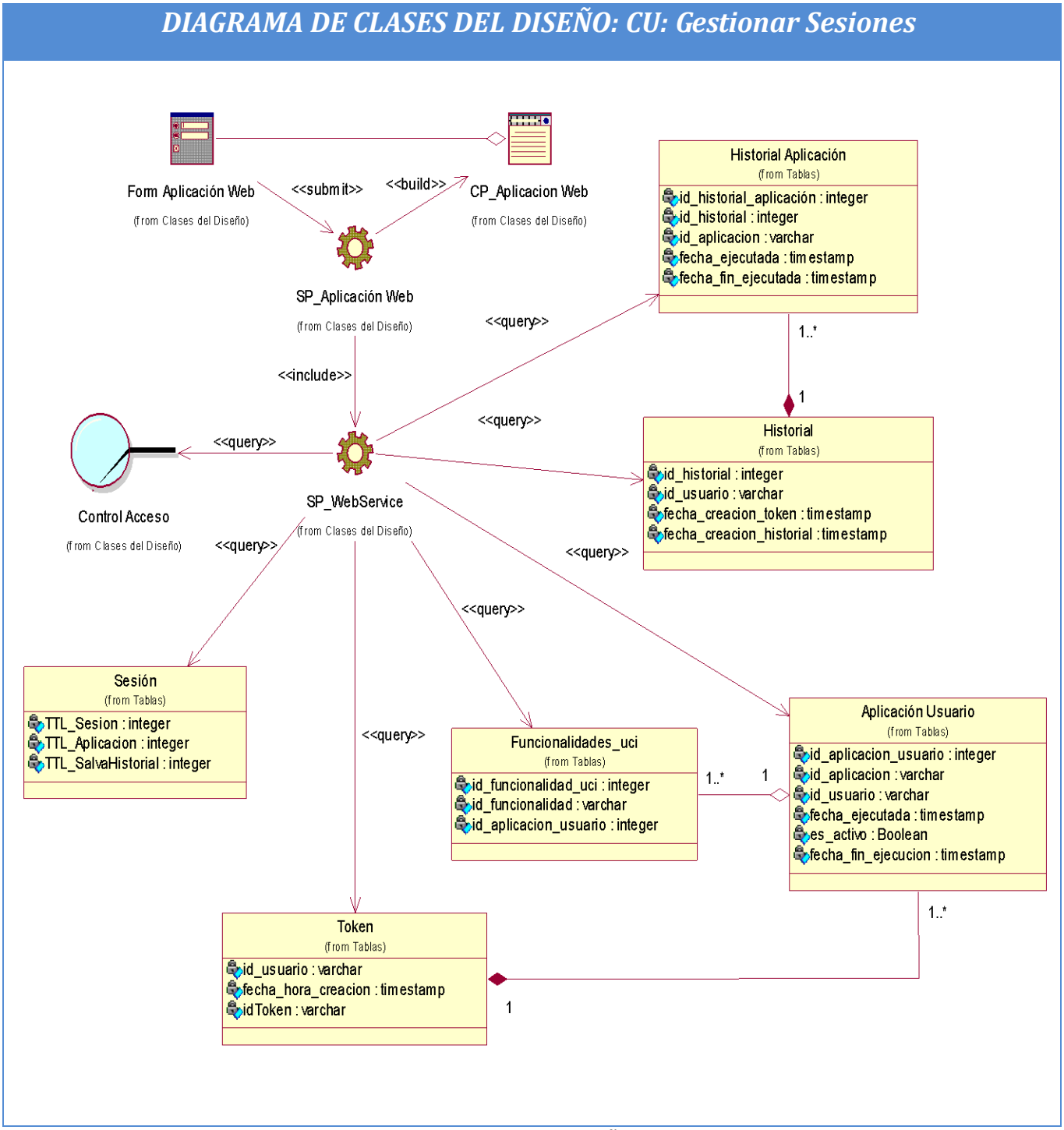

Imagen 3. 7: DIAGRAMA DE CLASES DEL DISEÑO: CU: Gestionar Sesiones

DIAGRAMA DE CLASES DEL DISEÑO: CU: Visualizar estadísticas de usuario

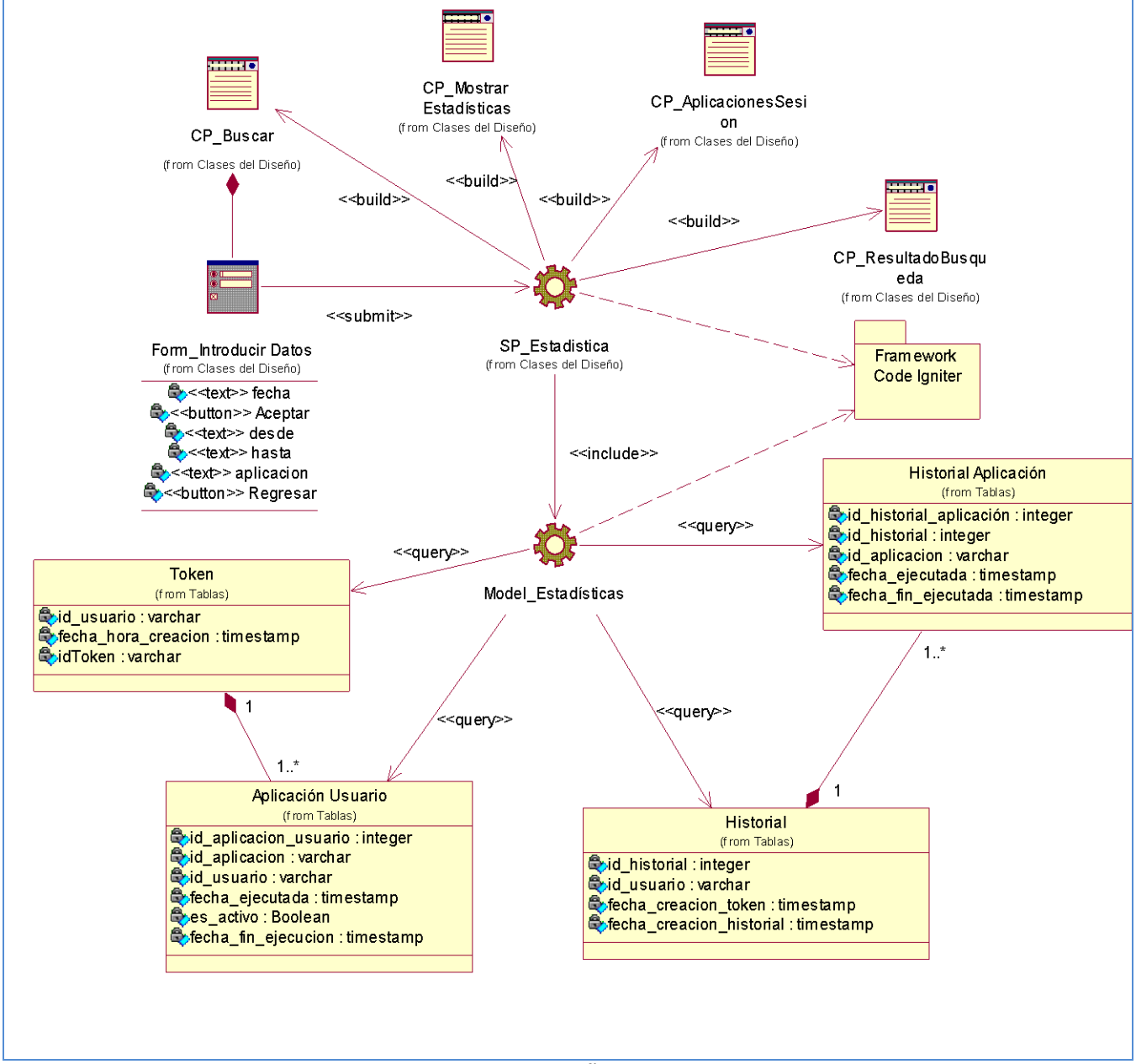

Imagen 3. 8: DIAGRAMA DE CLASES DEL DISEÑO: CU: Visualizar estadísticas de usuario

DIAGRAMA DE CLASES DEL DISEÑO: CU: Visualizar y cerrar token

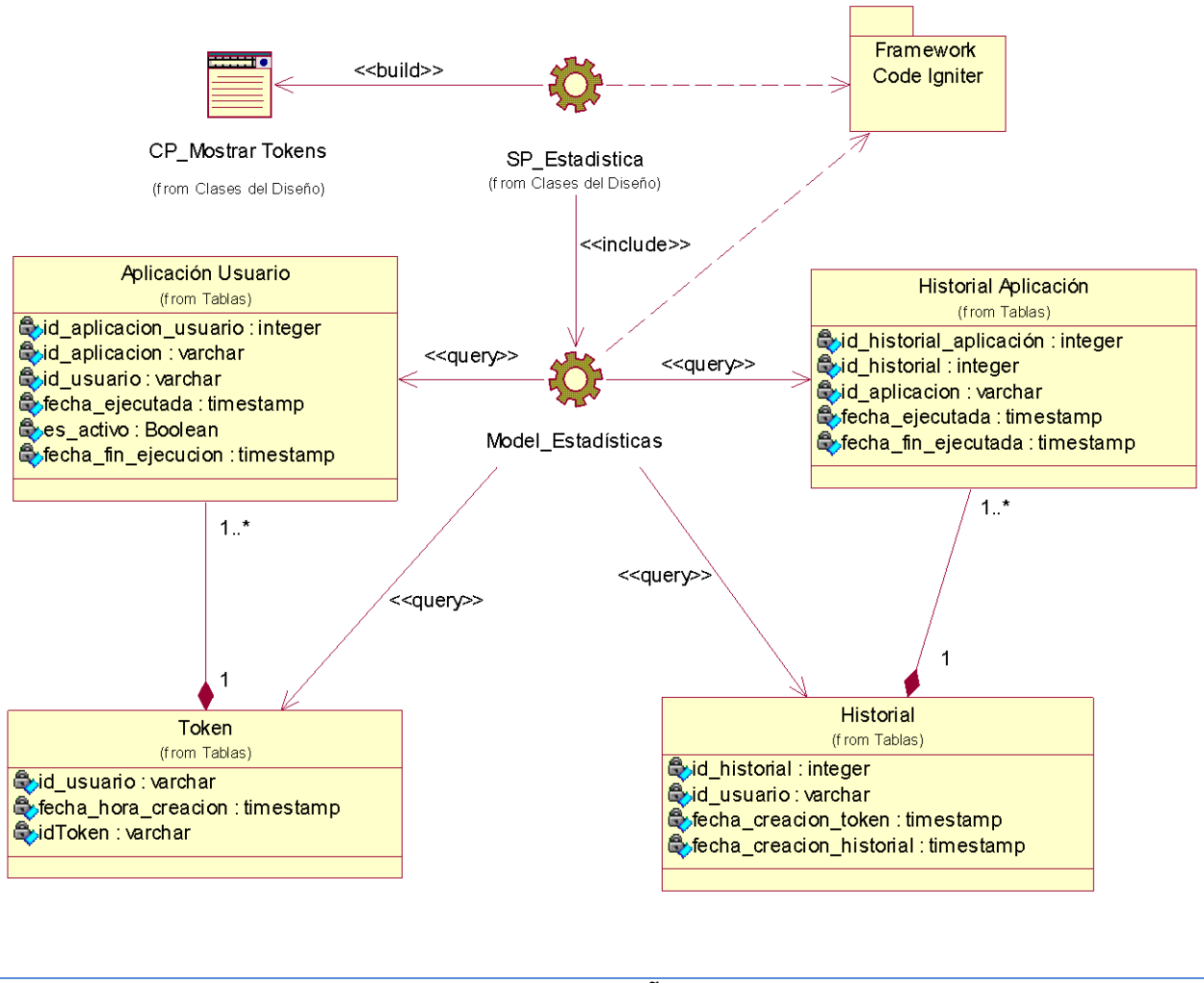

Imagen 3. 9: DIAGRAMA DE CLASES DEL DISEÑO: CU: Visualizar y cerrar token

DIAGRAMA DE CLASES DEL DISEÑO: CU: Autenticar usuario en la aplicación.

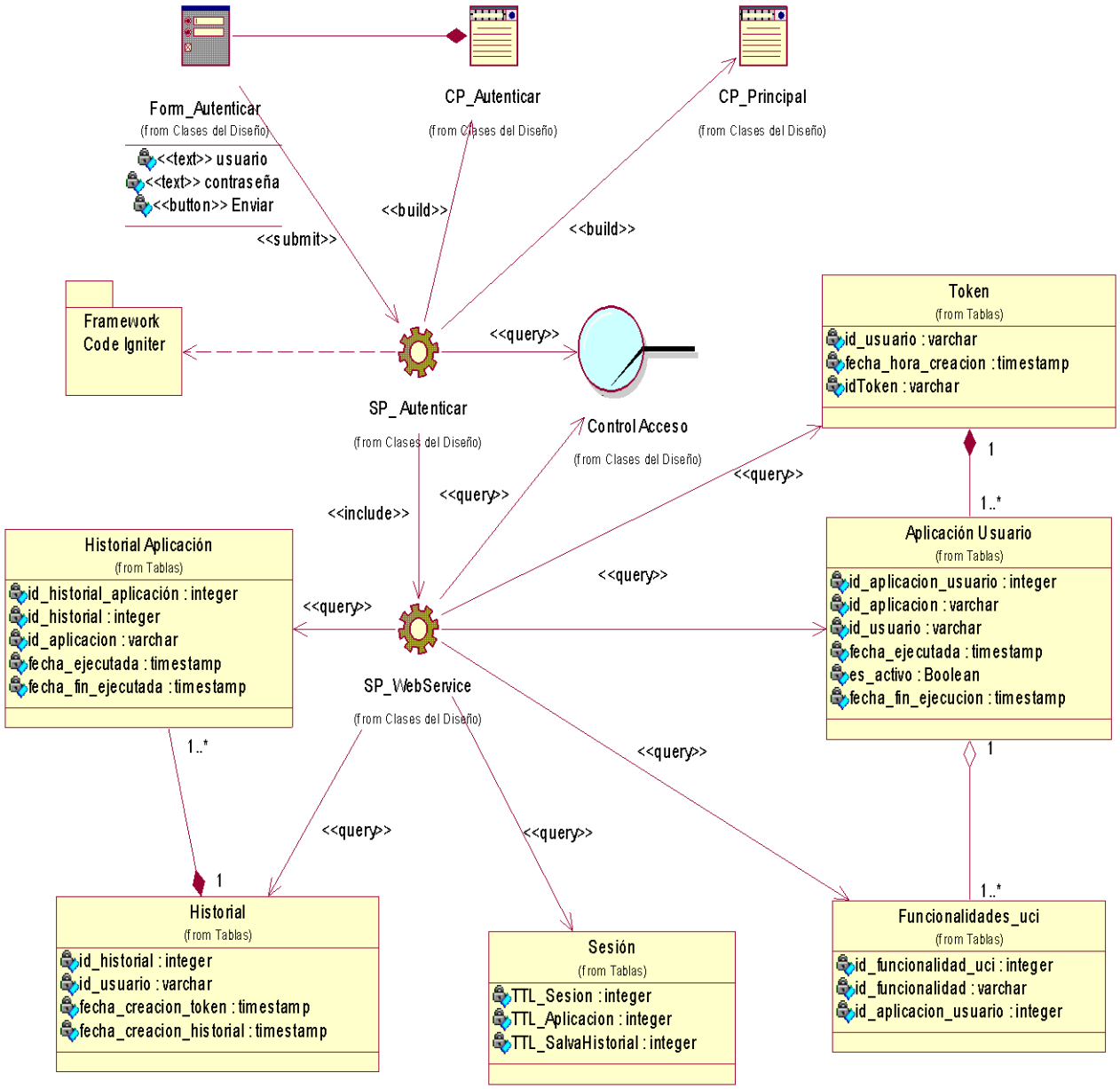

Imagen 3. 10: DIAGRAMA DE CLASES DEL DISEÑO: CU: Autenticar usuario en la aplicación Web

DIAGRAMA DE CLASES DEL DISEÑO: CU: Visualizar estadísticas generales

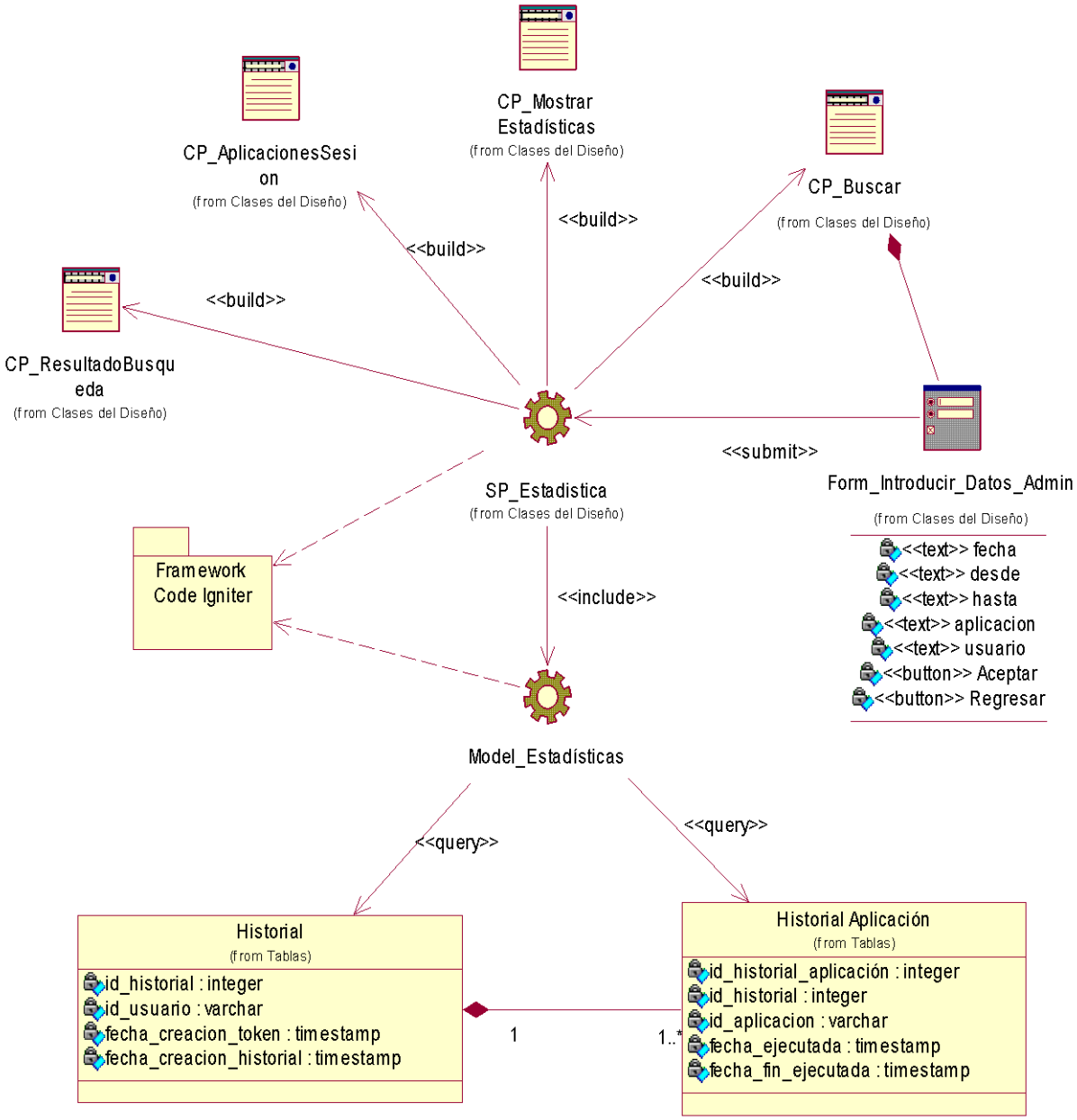

Imagen 3. 11: DIAGRAMA DE CLASES DEL DISEÑO: CU: Visualizar estadísticas generales

DIAGRAMA DE CLASES DEL DISEÑO: CU: Modificar TTL Aplicación, de Sesión y de Salva del Historial

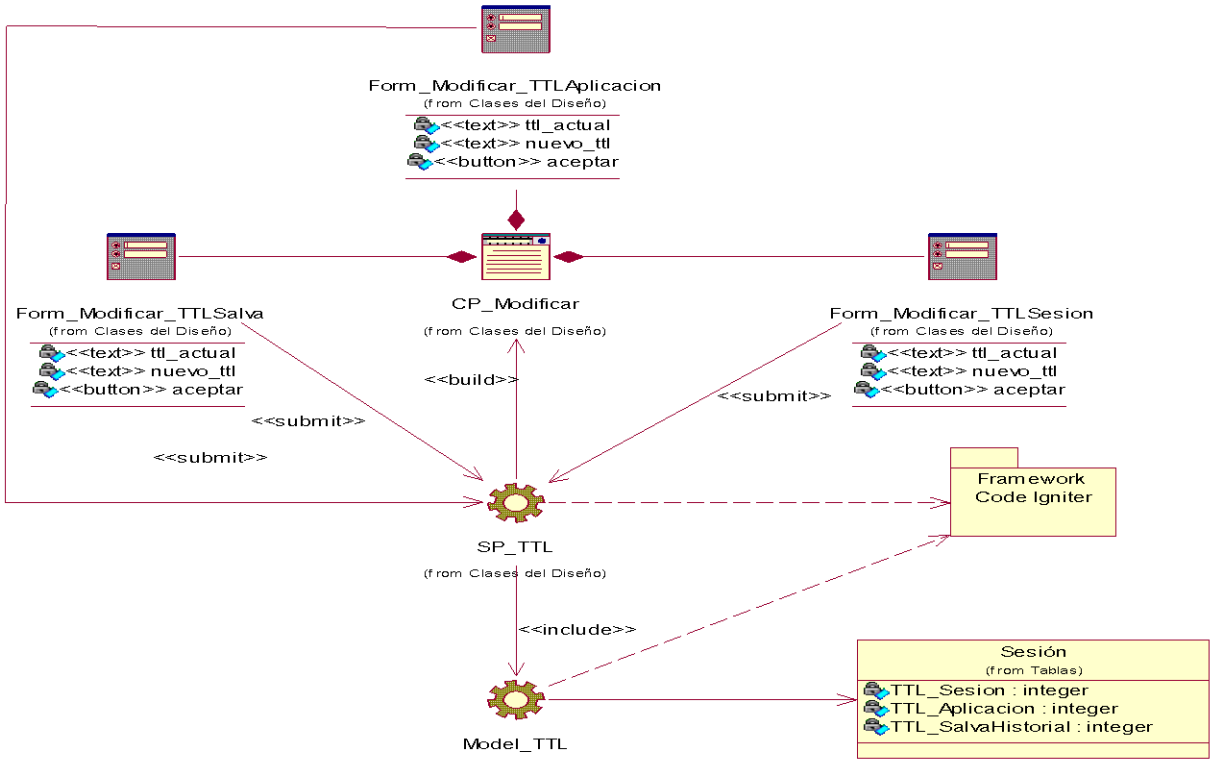

Imagen 3. 12: DIAGRAMA DE CLASES DEL DISEÑO: CU: Modificar TTL Aplicación, Sesión y de Salva del Historial

3.4 Diseño de la BD

La base de datos es un artefacto fundamental en este trabajo, su diseño por lo tanto es de vital importancia para el desarrollo de la propuesta de solución, buscando siempre un entendimiento de la forma en que se almacena la información en el sistema.

3.4.1 Modelo lógico de datos (diagrama de clases persistentes)

Las clases persistentes son las clases que tienen la capacidad de mantener su valor en el espacio y en el tiempo, la necesidad de guardar su estado esta dado por al almacenamiento físico permanente de la información de la clase, para la copia de seguridad en caso del fracaso del sistema, o para el intercambio de información.

El diagrama de clases persistentes es usado para modelar la estructura lógica de la base de datos con clases representando tablas, y atributos de clase representando columnas. Dicho diagrama se muestra a continuación:

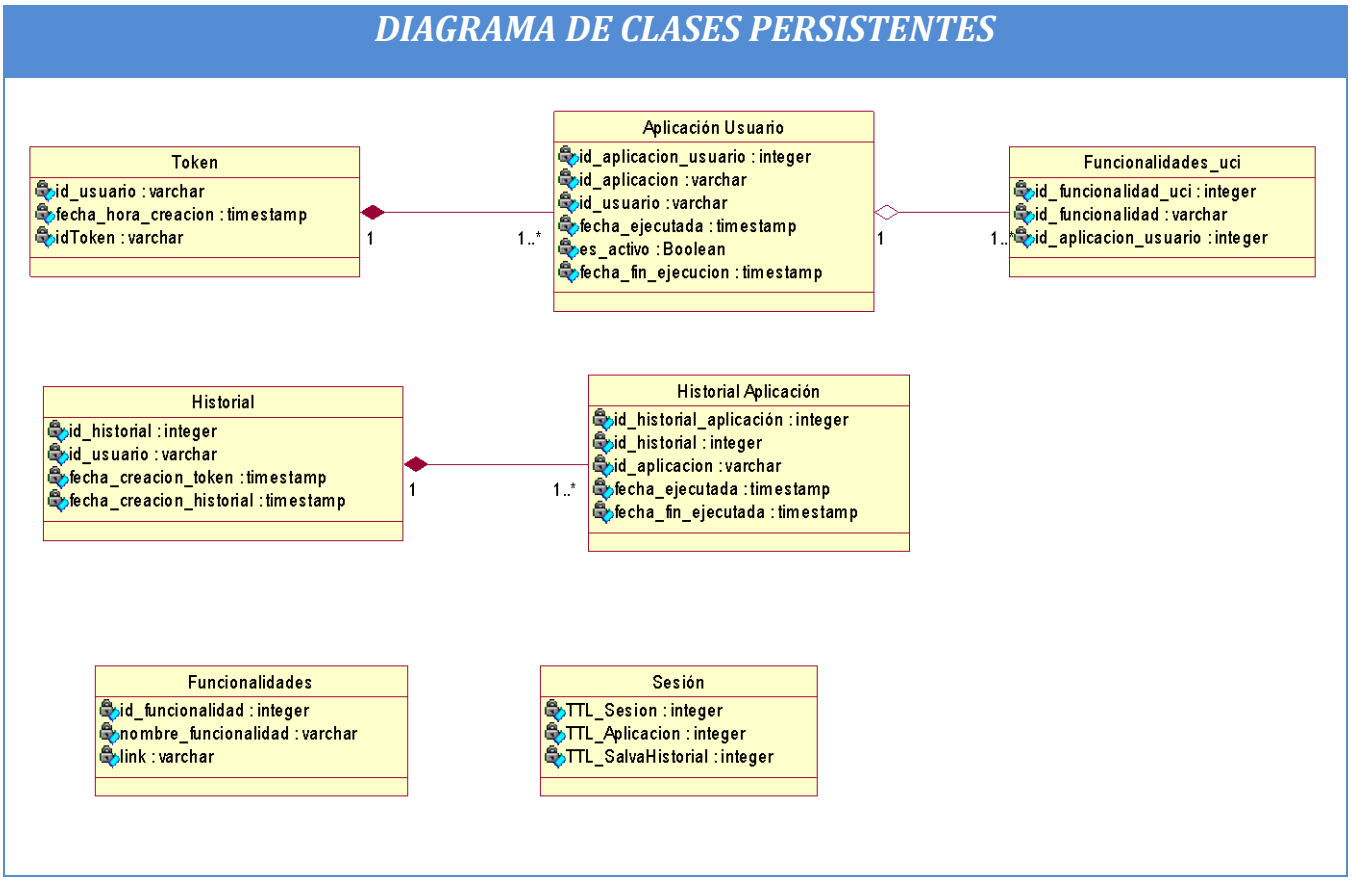

Imagen 3. 13: DIAGRAMA DE CLASES PERSISTENTES

3.4.2 Modelo físico de datos (modelo de datos)

El diagrama del modelo de datos se corresponde con la representación física de la base de datos.

DIAGRAMA ENTIDAD RELACION

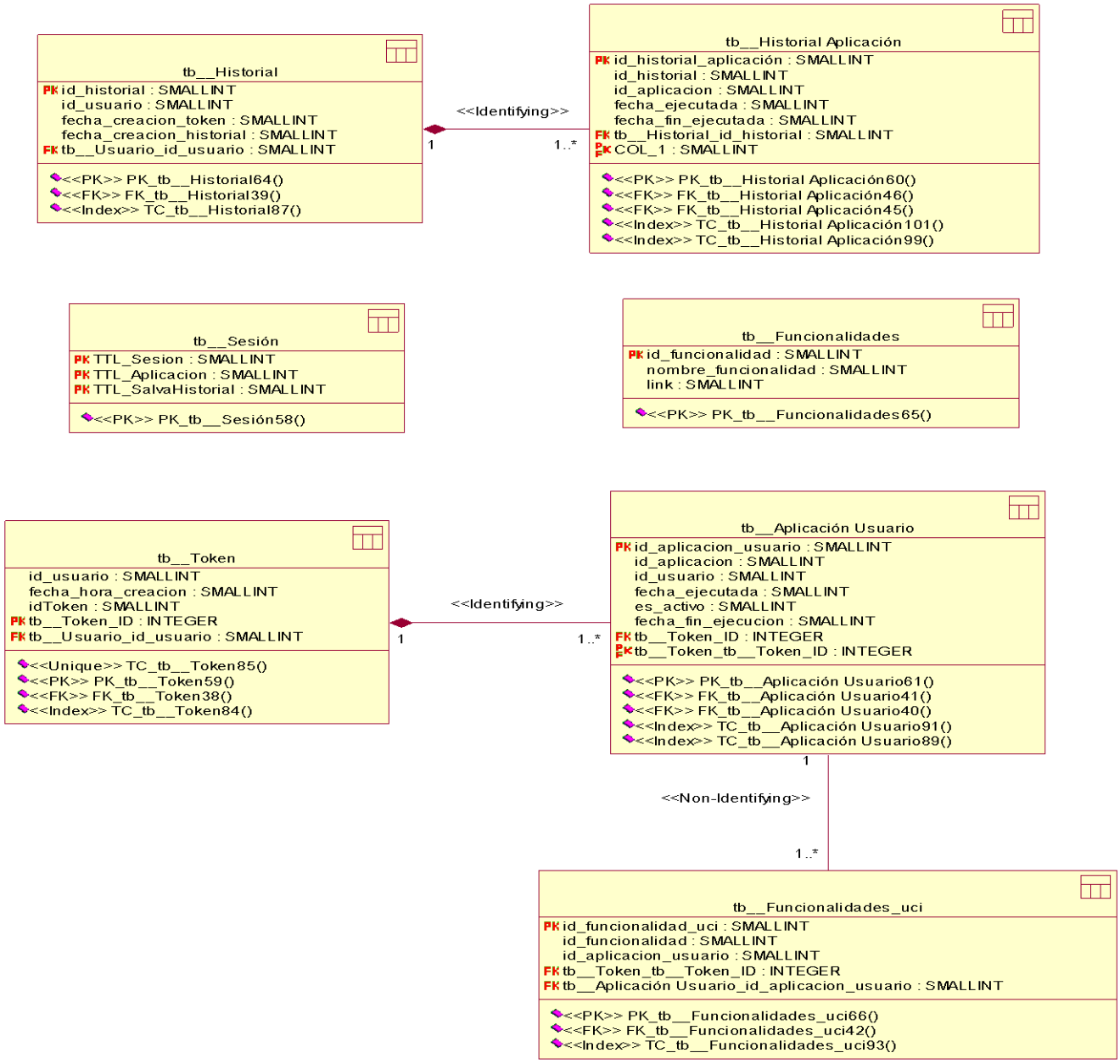

Imagen 3. 14: DIAGRAMA ENTIDAD RELACION

3.5 Conclusiones

Al término de este capítulo se puede concluir que el sistema consta de una parte fundamental y es todo lo relacionado con el caso de uso “Gestionar Sesiones” que el más importante, pues sin este caso de uso el sistema como tal no podría funcionar en toda su totalidad y no cumpliría su objetivo principal. Además se observa un avance del sistema y una definición mucho mayor de cómo será el mismo en un futuro y sus características fundamentales. En el desarrollo del capitulo se registra un evolución del sistema desde un punto de vista general hacia una visión específica de cada elemento y detalle del mismo, alcanzando un sistema listo para su implementación posterior.

Este capitulo es esencial en ciclo de desarrollo del software que facilita al programador la obtención de un sistema eficiente y que cumpla con los requerimientos buscados y expuesto en secciones anteriores.

4 Capítulo

Implementación.

4.1 Introducción

Este capítulo tiene como objetivo el desarrollo del flujo de trabajo de Implementación de la propuesta de solución. La implementación se realiza construyendo la codificación, es simplemente la programación del sistema, siguiendo el modelo de diseño. La implementación es el centro durante las iteraciones de construcción, aunque también se lleva a cabo trabajo de implementación durante la fase de elaboración, para crear la línea base ejecutable de la arquitectura, y durante la fase de transición, para tratar defectos tardíos como los encontrados con distribuciones beta del sistema. Empezamos con el resultado del diseño e implementamos el sistema en términos de componentes, es decir, ficheros de código fuente, *scripts,* ficheros de código binario, ejecutables y similares.

4.2 Diagrama de despliegue

Un diagrama de despliegue muestra las relaciones físicas entre los componentes hardware y software en el sistema final, es decir, la configuración de los elementos de procesamiento en tiempo de ejecución y los componentes software (procesos y objetos que se ejecutan en ellos). Estarán formados por instancias de los componentes software que representan manifestaciones del código en tiempo de ejecución (los componentes que sólo sean utilizados en tiempo de compilación deben mostrarse en el diagrama de componentes).

Un diagrama de despliegue es un grafo de nodos unidos por conexiones de comunicación. Un nodo puede contener instancias de componentes software, objetos, procesos (caso particular de un objeto). En general un nodo será una unidad de computación de algún tipo, desde un sensor a un mainframe. Las instancias de componentes software pueden estar unidas por relaciones de dependencia, posiblemente a interfaces (ya que un componente puede tener más de una interfaz).

Los diagramas de despliegue muestran la configuración en funcionamiento del sistema, incluyendo su hardware y su software. Para cada componente de un diagrama de despliegue se deben

documentar las características técnicas requeridas, el tráfico de red esperado, el tiempo de respuesta requerido, etc.

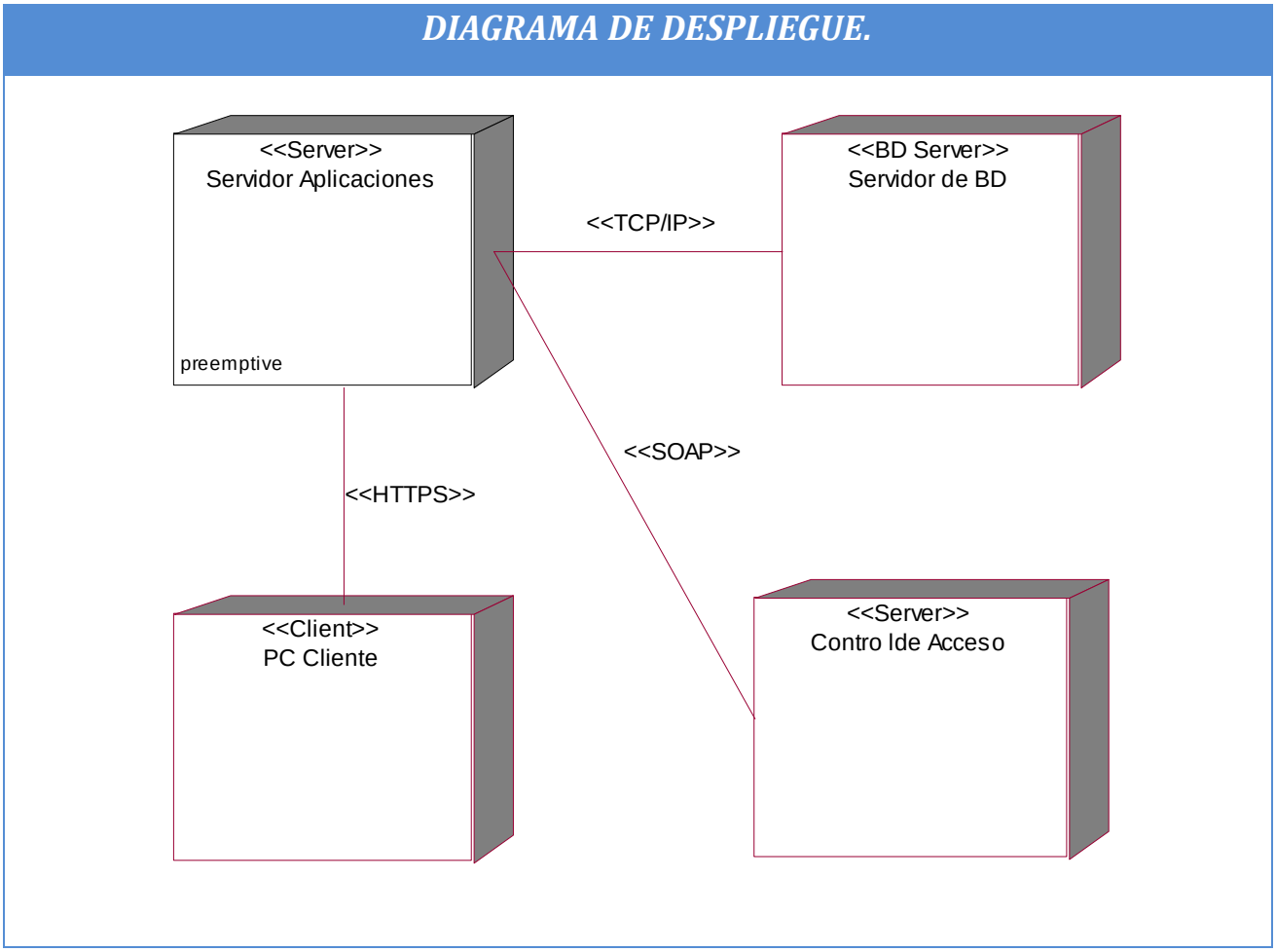

Imagen 4. 1: Diagrama de despliegue

4.3 Diagrama de componentes

Un diagrama de componentes muestra las organizaciones y dependencias lógicas entre componentes software, sean éstos componentes de código fuente, binarios o ejecutables. Desde el punto de vista del diagrama de componentes se tienen en consideración los requisitos relacionados con la facilidad de desarrollo, la gestión del software, la reutilización, y las restricciones impuestas por los lenguajes de programación y las herramientas utilizadas en el desarrollo. Los elementos de modelado en un diagrama de componentes son componentes y paquetes.

Componentes:

De los componentes, sólo aparecen tipos de componentes, ya que las instancias específicas de cada tipo se encuentran en el diagrama de despliegue. Los diagramas de componentes muestran los componentes software que constituyen una parte reusable, sus interfaces, y su interrelaciones, en muchos aspectos se puede considerar que son diagramas de clases a gran escala. Cada componente en el diagrama debe ser documentado con un diagrama de componentes más detallado, un diagrama de clases, o un diagrama de casos de uso.

Paquetes:

Un paquete en un diagrama de componentes representa una división física del sistema. Los paquetes se organizan en una jerarquía de capas donde cada capa tiene una interfaz bien definida. Un ejemplo típico de una jerarquía en capas de este tipo es: Interfaz de usuario; Paquetes específicos de la aplicación; Paquetes reusables; Mecanismos claves; y Paquetes hardware y del sistema operativo.

DIAGRAMA DE COMPONENTES.

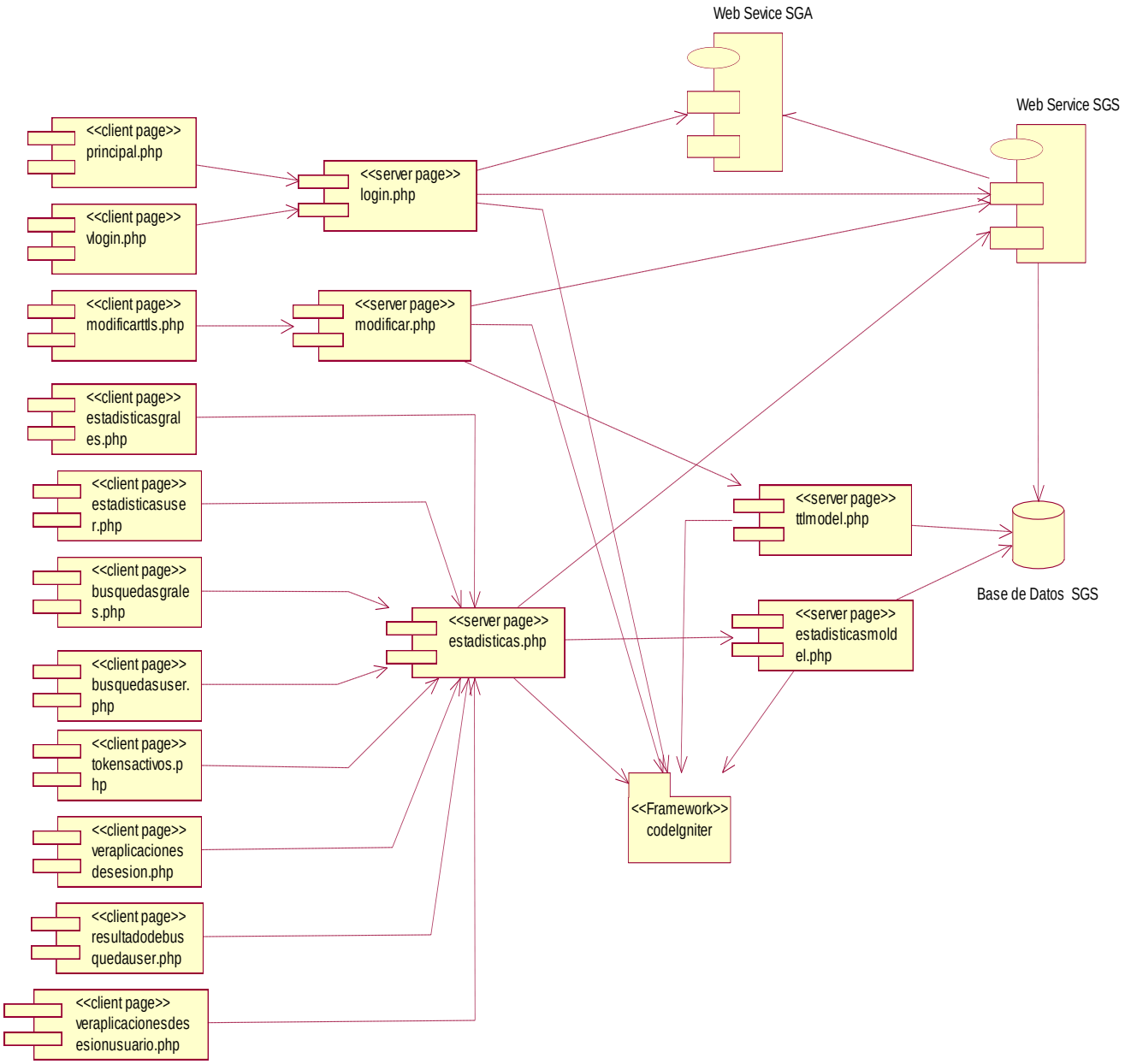

Imagen 4. 2: Diagrama de componentes

4.4 Conclusiones

Como vimos en este capítulo, el modelo de implementación es la entrada principal de las etapas de prueba que siguen a la implementación. El resultado principal de la implementación es el modelo de implementación, el cual incluye los subsistemas de implementación, sus dependencias, interfaces y contenido; los componentes, incluyendo componentes fichero y ejecutables, y las dependencias entre ellos. Los componentes son sometidos a pruebas de unidad; y por último la vista de la arquitectura del modelo de implementación, incluyendo sus elementos arquitectónicamente significativos.

5 Capítulo

Estudio de factibilidad.

5.1 Introducción

La investigación de factibilidad en un proyecto consiste en descubrir cuales son los objetivos de la organización, luego determinar si el proyecto es útil para que la empresa logre sus objetivos. En este capítulo se detalla el método estimación aplicado al sistema propuesto, ya que el análisis del costo, el esfuerzo y los beneficios, es de suma importancia a la hora de implementar un sistema. El estudio de factibilidad se realiza mediante el análisis de Puntos de Casos de Uso, método de estimación de tiempo de desarrollo del proyecto, a partir de las características de sus requisitos, expresados en los casos de uso.

5.2 Método de estimación basado en casos de usos.

El análisis de Puntos de Casos de Uso es una de las alternativas posibles para la estimación del esfuerzo en proyectos basados en Casos de Uso, se trata de un método de estimación del tiempo de desarrollo de un proyecto mediante la asignación de "pesos" a un cierto número de factores que lo afectan, para finalmente, contabilizar el tiempo total estimado para el proyecto a partir de esos factores. A continuación se detallan los pasos a seguir mediante la aplicación de este método para el sistema propuesto:

Paso 1. Cálculo de Puntos de Casos de Uso sin ajustar.

UUCP = UAW + UUCW

UUCP = 6+60=66

Donde:

UUCP: Puntos de Casos de Uso sin ajustar.

UAW: Factor de Peso de los Actores sin ajustar.

UUCW: Factor de Peso de los Casos de Uso sin ajustar.

Tabla: Factor de Peso de los Actores sin ajustar.

Tipo de Actor	Descripción	Factor de peso	Actores	Total
Simple	Sistema con sistema a través de interfaz de programación.	1	0	0
Medio	Sistema con sistema mediante protocolo de interfaz basada en texto.	2	0	0
Complejo	Persona que interactúa con el sistema mediante una interfaz gráfica.	3	2	6

Tabla 5. 1: Factor de Peso de los Actores sin ajustar

UAW = ∑ (cant actores * peso)

UAW = 3*2 =6

Tabla: Factor de Peso de los Casos de Uso sin ajustar.

Tipo de CU	Descripción	Peso	Cantidad de CU	Total
Simple	El caso de uso tiene de 1 a 3 transacciones.	5	2	10
Medio	El caso de uso tiene de 4 a 7 transacciones.	10	2	20
Complejo	El caso de uso tiene más de 8 transacciones.	15	2	60

Tabla 5. 2: Factor de Peso de los Casos de Uso sin ajustar

UUCW = ∑ (cant CU * peso)

UUCW = (5*2)+ (10*2)+ (15*2)=60

Paso 2. Cálculo de Puntos de Casos de Uso ajustados.

UCP = UUCP * TCF * EF

UCP =66*0.82*0.8= 43.296

Donde:

UCP: Puntos de Casos de Uso ajustados.

UUCP: Puntos de Casos de Uso sin ajustar.

TCF: Factor de complejidad técnica.

EF: Factor de ambiente.

El factor de complejidad técnica (TCF) se calcula mediante la cuantificación de un conjunto de factores que determinan la complejidad técnica del sistema. Cada factor se cuantifica en un valor desde 0 (aporte irrelevante) hasta 5 (aporte muy relevante).

Tabla: Factor de Complejidad Técnica.

Factor	Descripción	Peso	Valor asignado	Total
T1	Sistema distribuido.	2	0	0
T2	Tiempo de respuesta.	1	1	1
T3	Eficiencia del usuario final.	1	1	1
T4	Procesamiento interno complejo.	1	1	1
T5	El código debe ser reutilizable.	1	0	0
T6	Facilidad de instalación.	0.5	0	0
T7	Facilidad de uso.	0.5	4	2
T8	Portabilidad	2	1	2
T9	Facilidad de cambio.	1	2	2
T10	Concurrencia.	1	4	4
T11	Incluye objetivos especiales de seguridad.	1	3	3
T12	Provee acceso directo a terceras partes	1	5	5
T13	Se requieren facilidades especiales de entrenamiento de usuarios	1	1	1

Tabla 5. 3: Factor de Complejidad Técnica.

TCF = 0.6 + 0.01 * $\sum$ (peso * valor asignado)

TCF =0.61 +0.01*22= 0.82

El factor de ambiente (EF) está relacionado con las habilidades y entrenamiento del grupo de desarrollo que realiza el sistema. Cada factor se cuantifica con un valor desde 0 (aporte irrelevante) hasta 5 (aporte muy relevante).

Tabla: Factor de Ambiente.

Factor	Descripción	Peso	Valor asignado	Total
E1	Familiaridad con el modelo de proyecto utilizado.	1.5	3	4.5
E2	Experiencia en la aplicación.	0.5	0	0
E3	Experiencia en la orientación a objetivos.	1	5	5
E4	Capacidad del analista líder.	0.5	5	2.5
E5	Motivación.	1	5	5
E6	Estabilidad de requerimientos.	2	3	6
E7	Personal Part-Time.	-1	0	0
E8	Dificultad del lenguaje de programación.	-1	3	-3

Tabla 5. 4: Factor de Ambiente.

EF = 1.4 - 0.03 * ∑ (peso * valor asignado)

EF =1.4-0.03*20=0.8

Paso 3. Estimación del esfuerzo.

E = UCP * CF

E = 43.296 * 20

E = 865.92 Horas-Hombres

Donde:

E: Esfuerzo estimado en horas-hombre.

UCP: Puntos de Casos de Uso ajustados.

CF: Factor de conversión.

Para obtener el factor de conversión (CF) se cuentan cuantos valores de los que afectan el factor ambiente (E1...E6) están por debajo de la media (3), y los que están por arriba de la media para los restantes (E7, E8). Si el total es 2 o menos se utiliza el factor de conversión 20 Horas-Hombre / Punto de Casos de Uso. Si el total es 3 o 4 se utiliza el factor de conversión 28 Horas-Hombre / Punto de Casos de uso. Si el total es mayor o igual que 5 se recomienda efectuar cambios en el proyecto ya que se considera que el riesgo de

fracaso del mismo es demasiado alto.

En este caso se puede decir que:

CF = 20 Horas-Hombre

Paso 4. Calcular esfuerzo total del proyecto.

Tabla: Esfuerzo del proyecto.

Actividad	Porcentaje %	Horas-Hombre
Análisis	10	216.48
Diseño	20	342.96
Implementación	40	865.92
Pruebas	15	324.72
Sobrecarga(otras actividades)	15	324.72
Total	100	2164.8

Tabla 5. 5: Esfuerzo del proyecto.

Si el esfuerzo total (ET1) es de 2164.8 horas-hombre y se estima que cada mes tiene como promedio 192 horas laborables, daría un esfuerzo total por hombre (ET2) de 11.275 mes-hombre.

Esto quiere decir que una persona puede realizar la propuesta de solución analizada en 11 meses y 3 semanas aproximadamente.

Costo del proyecto:

Se asume un salario promedio mensual (SM) de $150.00.

CH: Cantidad de hombres.

Tiempo: Tiempo total del proyecto

CHM: Costo de hombres-mes.

CH: 2

CHM= 2 * SM

CHM= 150.00 * 2 = 300.00 $/mes

Costo= CHM * ET2 / CH

Costo= 300.00 * 11.275 / 2

Costo= $1691.65

Tiempo= ET2 / CH= 11.275 / 2 = 5.6375 ˜ 5.64 meses

Resumen de los resultados:

Esfuerzo Total (Horas--Hombre)	ET1	2164.80
Esfuerzo Total (Mes--Hombre)	ET2	11.275
Salario Promedio	SM	150.00
Cantidad de Hombres	CH	2
Costo Hombres--Mes	CHM	300.00
Costo Total	Costo	1691.65
Tiempo Total	Tiempo	5.64

De los resultados obtenidos se interpreta que con 2 hombres trabajando en el proyecto el mismo se desarrolla en 5.64 meses y su costo total se estima que sea en $1691.65.

5.3 Beneficios Tangibles e Intangibles.

El Sistema de Gestión de Sesiones no es un producto utilizado con fines comerciales u otros intereses similares, ha sido desarrollado para transferir la funcionalidad y complejidad de todos los componentes de seguridad a un solo servicio de SSO, en nuestra universidad. El beneficio principal que reporta el sistema es contar una solución informática que permita acceder a múltiples recursos de la red mediante un único proceso de autenticación por parte de todos los usuarios de la UCI. Si se habla de beneficios intangibles que reporta el sistema, se puede mencionar la disminución considerable del tiempo que demoran los usuarios realizando una y otra vez mismo proceso de autenticación cada vez que quieran acceder a un recurso de la red, además de ofrecer un servicio que proporciona un alto nivel de seguridad y confianza.

5.4 Análisis de costos y beneficios.

El sistema de Gestión de Sesiones no requiere de inversión de software porque las herramientas y la tecnología empleadas para su desarrollo son totalmente libres, por lo que una vez analizado el costo del proyecto y los beneficios que este reporta se puede concluir que el sistema desarrollado es factible y que su uso contribuirá a que se lleven a cabo eficientemente los principios de la aplicación del Sistema de Gestión de Sesiones en la UCI, por parte de todos los usuarios de la universidad.

5.5 Conclusiones

En este capítulo se desarrolló la estimación por Puntos de Caso de Uso que resultó muy efectiva para estimar el esfuerzo del proyecto teniendo en cuenta los factores que influyen en el desarrollo del software. Se realizó el análisis de los beneficios tangibles e intangibles así como sus costos y los beneficios que nos proporciona como sistema, permitiéndonos valorar cuán factible sería la realización del mismo, así como optimizar los recursos empleados para su desarrollo.

Conclusiones Generales

Al término de este trabajo se ha podido comprobar que cada día se hace más necesario la implementación en las redes de nuevos mecanismos de seguridad que incrementen el grado de complejidad de la misma para evitar ataques contra las entidades. Un SSO es una manera eficiente de asegurar que ninguna aplicación que funcione dentro de una red contenga un mecanismo de seguridad erróneo y poco eficiente, estos sistemas se hacen precisos además porque aporta a ambas partes implicadas ya que a los usuarios tendrán una mayor comodidad pues también les evita tiempo y trabajo ya que realizará un solo proceso de autenticación en la ejecución de varios recursos de la red.

Con la culminación de este sistema se da un paso más por parte de la Dirección de Informatización hacia la soñada Ciudad Digital, pues estos software constituyen una tecnología de gran categoría que en el mundo han tenido una gran aceptación por parte de usuarios y las grandes compañías.

Se logra un software que también será capaz de guardar todos los registros de los accesos a las aplicaciones del dominio, dichos registros contienen el usuario, en la fecha y la hora que accedió a cuales aplicaciones.

Con la realización de este sistema se ha aprendido el trabajo de web services, del framework Code Igniter, se adquiere experiencia en cuanto a la realización de sistemas y las distintas vías de hacer un mismo mecanismo buscando siempre la mejor forma. Se adquieren conocimientos del funcionamiento de una red y todo lo que se gestiona detrás de la entrada un usuario y contraseña en la red de la universidad.

Recomendaciones

Se le recomienda al interesado en este tipo de sistemas seguir pensando en mejoras que aún se le puedan realizar, implementarlas y distribuirlas, o hacer llegar dichas ideas a las autoras de este trabajo buscando siempre progresos en el mismo.

Agregar seguridad a los servicios web con mecanismos de llaves privadas y públicas para evitar ejecuciones a los mismos sin la debida autorización.

Por ultimo recomendamos agregar nuevas funcionalidades en versiones posteriores de este sistema, como por ejemplo que la aplicación pueda brindar estadísticas graficas del número de sesiones que se abrieron en un rango de tiempo que defina el administrador para que el mismo tenga una mejor noción de la cantidad de accesos por parte de los usuarios a las aplicaciones.

Referencias Bibliográficas

[1] NetSignia. *Single Sign On*. [Disponible en:

<http://www.netsignia.es/docs/SSO_final.pdf >]

[2] Wikipedia. *Aplicación Web*. 26 de noviembre 2007, [Disponible en:

< http://es.wikipedia.org/wiki/PHP >]

[3] Wikipedia. *Modelo Vista Controlador.* 12 de mayo del 2008. [Disponible en: <http://64.233.169.104/search?q=cache:XfVFs2DsyKMJ:es.wikipedia.org/wiki/Modelo_Vista_Controlador+modelo+vista+controlador&hl=es&ct=clnk&cd=1&gl=cu&client=firefox-a >]

[4] Guillermo Valle, José y Gildardo Gutiérrez, James. *Definición arquitectura cliente servidor.* Año 2005 [Disponible en: <http://www.monografias.com/trabajos24/arquitectura-cliente-servidor/arquitectura-cliente-servidor.shtml >]

[5] Wikipedia. *Servicio Web*.12 de noviembre 2007, [Disponible en: < http://es.wikipedia.org/wiki/Servicio_Web >]

[6] Wikipedia. *PHP*.28 de noviembre 2007, [Disponible en:

< http://es.wikipedia.org/wiki/PHP >]

[7] Wikipedia. *Sistema de gestión de Base de Datos*. 12 de noviembre 2007, [Disponible en: < http://es.wikipedia.org/wiki/Sistema_de_gesti%C3%B3n_de_base_de_datos >]

[8] Wikipedia. *PostgreSQL*. 19 de septiembre 2007, [Disponible en: < http://es.wikipedia.org/wiki/PostgreSQL >]

[9] Tienda Linux. *Ventajas de PostgreSQL*. 31 de mayo 2003 [Disponible en:

< http://soporte.tiendalinux.com/portal/Portfolio/postgresql_ventajas_html >]

[10] Wikipedia. *Servidor Web*. 22 de noviembre 2007, [Disponible en: < http://es.wikipedia.org/wiki/Servidor_web>]

[11] Wikipedia. *Servidor HTTP Apache*. 29 de noviembre 2007, [Disponible en: < http://es.wikipedia.org/wiki/Servidor_web>]

[12] Wikipedia. *Proceso Unificado de Rational*. 29 de noviembre 2007, [Disponible en:

< **http://es.wikipedia.org/wiki/Proceso_Unificado_de_Rational** >]

[13] Wikipedia. *Lenguaje Unificado de Modelado*.29 de noviembre 2007, [Disponible en:

< **http://es.wikipedia.org/wiki/Lenguaje_Unificado_de_Modelado** >]

[14] Nideaderedes. Code Igniter – framework PHP. [Disponible en:

< **http://nideaderedes.urlansoft.com/2007/02/27/code-igniter-framework-php/** >]

[15] *Zend Studio 5 I LAS SOLUCIONES MÁS COMPLETAS PARA EL DESARROLLO DE PHP.* [Disponible en:
< **www.cv-team.com/comunes/ficheros/fileDownload.php?OTA1OQ%3D%3D**>]

[16] Wikipedia. *Herramienta Case*. 30 de noviembre 2007, [Disponible en:

< **http://es.wikipedia.org/wiki/Herramienta_CASE**>]

[17] *Rational Rose Enterprise.* [Disponible en:
<**http://www111.ibm.com/ecatalog/Detail.wss?locale=es_ES&synkey=M221280M46834Z27**>]

Anexos

Anexo I: Especificación de los Casos de Uso del Sistema

2.11.3.1 CU: Gestionar de Sesiones.

Caso de Uso: Gestionar Sesiones	
Actores:	Usuario UCI
Resumen:	El caso de uso se inicia cuando el usuario ejecuta una aplicación y el sistema verifica si ya tiene sesión creada, en caso de una respuesta positiva, verifica que el usuario tiene acceso a esta aplicación y actualiza el certificado. Si la respuesta es negativa, el sistema le crea la sesión y guarda el certificado, validando después que el usuario tenga acceso a la aplicación. Cuando el usuario ha cerrado todas sus aplicaciones el sistema cierra su sesión.
Precondiciones:	
Referencia:	R 1, R 1.1, R 1.2, R 1.3, R 2.
Prioridad:	Crítico

Flujo Normal de Eventos	
Acción del Actor	Respuesta del Sistema
1 El caso de uso inicia cuando el usuario ejecuta una aplicación.	1.1 El sistema verifica que el usuario tiene sesión creada en el sistema.
	1.2 El sistema verifica que la sesión no haya expirado aún, en caso de que el usuario si tenga sesión en el sistema.

	1.3 El sistema verifica que el usuario tenga permiso para acceder a la aplicación, en el caso que la sesión ya esté creada en el sistema y sin haber expirado.
	1.4 Si el usuario tiene acceso, el sistema registra las estadísticas que son: hora y fecha de inicio en la que el usuario entró a la aplicación, también se registra el usuario y aplicación que ejecutó, y además las funcionalidades del usuario en la aplicación ejecutada.
	1.5 Se le da una respuesta a la aplicación con los datos del usuario, como el estado de la sesión y el estado de la aplicación, la fecha en abrió la sesión, etc., y termina el caso de uso.
Flujos Alternos	
Acción del Actor	Respuesta del Sistema
1ª El caso de uso se inicia cuando el usuario se autentica directamente en el Sistema de Gestión de Sesiones.	1ª1 El sistema verifica el estado del usuario.
	1ª2 El sistema realiza el proceso de autenticación de usuario para comprobar que sus datos fueron introducidos correctamente en el caso que el usuario no tenga sesión en el sistema y termina el caso de uso.
Flujos Alternos	
Acción del Actor	Respuesta del Sistema

	1.1ª Si la sesión no está creada, el sistema da una respuesta a la aplicación que se ejecutó de que el usuario no se ha autenticado para que la misma realice el proceso de autenticación.
1.1ª1 Realiza el proceso de autenticación.	
	1.1ª2 El sistema crea la sesión con los datos de usuario, el sistema registra las estadísticas que son: hora y fecha de inicio en la que el usuario entró a la aplicación y se creo la sesión, también se registra el usuario y aplicación que ejecutó. Termina el caso de uso.
Flujos Alternos	
Acción del Actor	Respuesta del Sistema
	1.2ª Si la sesión del usuario ya expiró se cierra la sesión del usuario, se le da una respuesta a la aplicación de que ya el usuario no tiene sesión en el sistema y el mismo debe realizar el proceso de autenticación nuevamente y termina el caso de uso.
Flujos Alternos	
Acción del Actor	Respuesta del Sistema
	1.4ª Si el usuario entró a dicha aplicación con anterioridad, el sistema procede a verificar si el usuario aún tiene acceso a la aplicación o si sus funcionalidades han cambiado.
	1.4ª 1 Si aún tiene acceso y cambiaron sus funcionalidades, el sistema procede a actualizar

	sus funcionalidades.
	1.4ª 2 El sistema actualizar las estadísticas que son: hora y fecha de inicio en las que el usuario entró a la aplicación.
	1.4ª 3 Se le da una respuesta a la aplicación con los datos del usuario en el sistema y termina el caso de uso.
Flujos Alternos	
Acción del Actor	Respuesta del Sistema
	1.4ª 1ª Si el usuario ya no tiene acceso a dicha aplicación se cierra su sesión de aplicación y se verifica si aún quedan en la sesión otras aplicaciones abiertas. Si no quedan se cierra la sesión completa del usuario.
	1.4ª 1ª 1 Se le da una respuesta a la aplicación de que este usuario ya no tiene acceso y termina el caso de uso.
Flujos Alternos	
Acción del Actor	Respuesta del Sistema
1b El caso de uso se inicia cuando el actor cierra todas las aplicaciones que abrió hasta el momento.	1b1 El sistema procede a registrar o actualizar las estadísticas que son: hora y fecha de fin en las que el usuario salió tanto de las aplicaciones como de la sesión.
	1b2 El sistema elimina este token y todas las trazas de este usuario pasan a los historiales. Termina así el caso de uso.

Poscondiciones	La sesión en el sistema queda creada, actualizada, guardada y/o eliminada para el usuario.

Tabla 2. 2: CU: Gestionar sesiones

2.11.3.2 CU: Visualizar estadísticas de usuario.

Caso de Uso: Visualizar estadísticas de usuario	
Actores:	Usuario UCI
Resumen:	El caso de uso se inicia cuando el usuario ejecuta la opción ¨Visualizar Estadísticas¨ en la aplicación Web del sistema y este le muestra las estadísticas correspondientes a ese usuario.
Precondiciones:	El usuario debe haber realizado el proceso de autenticación.
Referencia:	R3
Prioridad:	Auxiliar
Flujo Normal de Eventos	
Acción del Actor	Respuesta del Sistema
1 El usuario ejecuta la opción ¨Visualizar Estadísticas¨.	1.1 El sistema le muestra las estadísticas correspondientes a ese usuario en el día actual. Dichas estadísticas muestran todas las aplicaciones a las que el mismo ha entrado con sus respectivas fecha y hora de ejecución.
	1.2 El sistema también le muestra la posibilidad de buscar sus estadísticas por criterios de fechas (fecha y rango) y por nombre de aplicación, terminando así el caso de uso.
Escenario 1	

Acción del Actor	Respuesta del Sistema
2. El usuario introduce una fecha o el rango de los días en que los que desee busca sus estadísticas y/o la aplicación para la que desea buscar sus registros.	2.1 El sistema le muestra las estadísticas y termina el caso de uso.
Escenario 2	
Acción del Actor	Respuesta del Sistema
3. El usuario escoge la opción “Más” de la estadística de una sesión determinada.	3.1 El sistema le muestra las aplicaciones que se abrieron en la sesión y termina el caso de uso.
Escenario 3	
Acción del Actor	Respuesta del Sistema
3.1ª El usuario escoge la opción “Más” de la sesión que tiene activa en ese momento.	3.1ª 1 El sistema le muestra las aplicaciones que el usuario ha abierto hasta el momento en la sesión y termina el caso de uso.
Poscondiciones	El usuario visualiza sus estadísticas en el sistema.

Tabla 2. 3: CU: Visualizar estadísticas de usuario

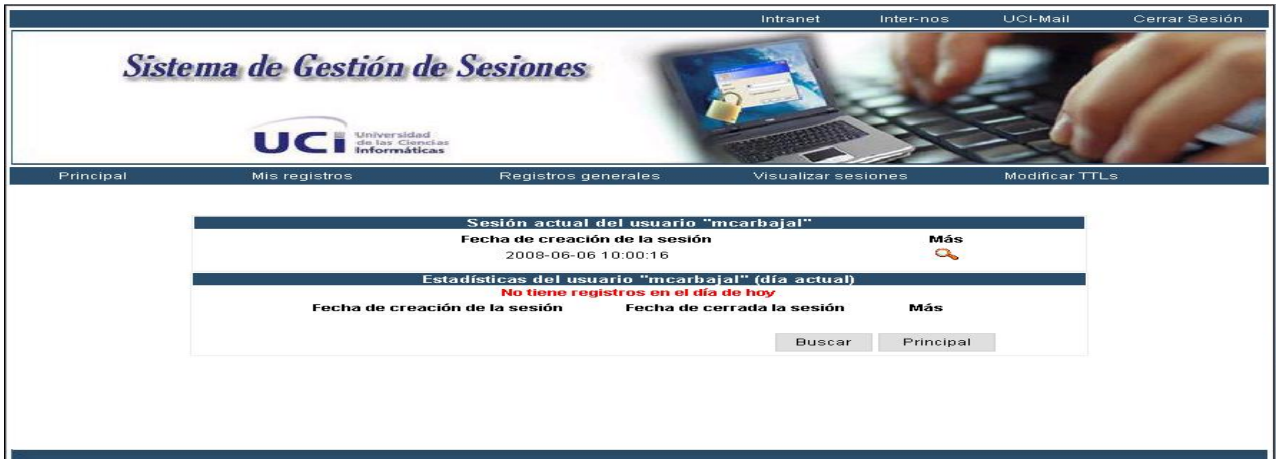

Imagen 2. 3: Interfaz del CU: Visualizar estadísticas de usuario

2.11.3.3 CU: Autenticar usuario en la aplicación Web.

Caso de Uso: Autenticar usuario en la aplicación Web del sistema.	
Actores:	Usuario UCI
Resumen:	El caso de uso inicia cuando el usuario ejecuta la aplicación Web del sistema y este le pide que realice el proceso de autenticación.
Precondiciones:	
Referencia:	R4
Prioridad:	Crítico
Flujo Normal de Eventos	
Acción del Actor	Respuesta del Sistema
1 El usuario ejecuta la aplicación Web.	1.1 El sistema verifica que no haya sesión guardada en el explorador.
1.2 El usuario introduce su nombre de usuario y su contraseña si no hay sesión abierta en el explorador.	1.3 El sistema verifica que ese usuario no tiene sesión en el sistema.
	1.4 Si los datos están correctos y el usuario no tiene sesión en el sistema se registran los datos y termina el caso de uso.
Flujos alternos	
Acción del Actor	Respuesta del Sistema
	1.1ª Si existe sesión guardada en el explorador el sistema verifica que no haya expirado esta sesión
	1.1ª 1 Si no ha expirado la sesión el usuario, se registran los datos de fecha y hora de ejecutada la

	aplicación y el usuario entra al sistema. Termina el caso de uso.
Flujos alternos	
Acción del Actor	Respuesta del Sistema
	1.3ª Si el usuario que se autenticó sí tiene sesión en el sistema y por lo tanto la misma no ha expirado se registran los datos de fecha y hora de ejecutada la aplicación y el usuario entra al sistema. Termina el caso de uso.
Flujos alternos	
Acción del Actor	Respuesta del Sistema
	1.1ª 1ª Si la sesión del usuario expiró se ejecuta el paso 1.2.
Flujos alternos	
Acción del Actor	Respuesta del Sistema
	1.4ª En caso de que los datos que entró el usuario estén incorrectos se ejecuta el paso 1.2.
Poscondiciones	El usuario queda autenticado en el sistema.

Tabla 2. 4: CU: Autenticar usuario en la aplicación Web

Imagen 2. 4: Interfaz del CU: Autenticar usuario en la aplicación Web

2.11.3.4 CU: Modificar TTL Aplicación, de Sesión y de Salva del Historial.

Caso de Uso: Modificar TTL Aplicación, de Sesión y de Salva del Historial.	
Actores:	Administrador del sistema.
Resumen:	El caso de uso se inicia cuando el administrador del sistema escoge la opción ¨Modificar TTL ¨, el sistema le dará la opción de modificar TTL Aplicación, TTL Sesión ó TTL de Salva del Historial según el usuario escoja.
Precondiciones:	El administrador debe haber realizado el proceso de autenticación.
Referencia:	R5
Prioridad:	Crítico

Flujo Normal de Eventos	
Acción del Actor	Respuesta del Sistema
1 El administrador escoge la opción ¨Modificar TTL¨.	1.1 El sistema le muestra los formularios de ¨Modificar TTL Aplicación¨, ¨Modificar TTL Sesión¨ y “Modificar TTL de Salva del Historial” donde el administrador podrá introducir el nuevo TTL deseado según el formulario que llene.

Escenario 1	
Acción del Actor	Respuesta del Sistema
2.1 El administrador llena el formulario de “Modificar TTL Aplicación” y da clic en el botón ¨Aceptar¨.	2.2 El sistema guarda y actualiza el TTL Aplicación y termina el caso de uso.
Escenario 2	
Acción del Actor	Respuesta del Sistema
3.1 El administrador llena el formulario de “Modificar TTL Sesión” y da clic en el botón ¨Aceptar¨.	3.2 El sistema guarda y actualiza el TTL Sesión y termina el caso de uso.
Escenario 3	
Acción del Actor	Respuesta del Sistema
4.1 El administrador llena el formulario de “Modificar TTL de Salva del Historial” y da clic en el botón ¨Aceptar¨.	4.2 El sistema guarda y actualiza el TTL de Salva del Historial y termina el caso de uso.
Poscondiciones	1. TTL Aplicación actualizado y guardado en la base de datos 2. TTL Sesión actualizado y guardado en la base de datos. 3. TTL de Salva del Historial actualizado y guardado en la base de datos.

Tabla 2. 5: CU: Modificar TTL Aplicación, TTL Sesión y TTL de Salva del Historial

Imagen 2. 5: Interfaz del CU: Modificar TTL de Aplicación, de Sesión y de Salva del Historial

2.11.3.5 CU: Visualizar estadísticas generales.

Caso de Uso: Visualizar estadísticas generales.	
Actores:	Administrador del sistema.
Resumen:	El caso de uso inicia cuando el administrador del sistema escoge la opción ¨Ver estadistas generales¨ y luego selecciona el criterio según va a buscar las estadísticas, y el sistema se las muestra.
Precondiciones:	El administrador debe haber realizado el proceso de autenticación.
Referencia:	R6.
Prioridad:	Auxiliar
Flujo Normal de Eventos	
Acción del Actor	Respuesta del Sistema
1 El administrador del sistema escoge la opción ¨Ver estadísticas¨.	1.1 El sistema le muestra las estadísticas guardadas para el día actual
Escenario 1	
Acción del Actor	Respuesta del Sistema

2 El administrador escoge la opción de buscar estadísticas	2.1 El sistema le muestra los criterios para ver dichas estadísticas, que pueden ser: por usuario, por fecha, por aplicación y las posibles combinaciones de estas opciones, o todas las estadísticas.
2.2 El administrador introduce los datos de los criterios de búsqueda y da clic en el botón “Aceptar”.	2.3 El sistema le muestra las estadísticas y termina el caso de uso.
Escenario 2	
Acción del Actor	Respuesta del Sistema
3 El administrador del sistema escoge la opción “Eliminar todas las estadísticas ”	3.1 El sistema elimina todas las estadísticas o historiales que están registradas en ese momento y termina el caso de uso.
Escenario 3	
Acción del Actor	Respuesta del Sistema
4 El administrador da clic en el botón “Eliminar” de una estadística	4.1 El sistema elimina la estadística seleccionada y termina el caso de uso.
Escenario 4	
Acción del Actor	Respuesta del Sistema
5 El administrador del sistema escoge la opción “Eliminar estadísticas vencidas”	5.1 El sistema elimina todas las estadísticas cuyo tiempo de salva del historial haya expirado y termina el caso de uso.
Escenario 5	
Acción del Actor	Respuesta del Sistema

6 El administrador escoge la opción "Más" de la estadística de una sesión determinada.	6.1 El sistema le muestra todas las aplicaciones que fueron abiertas en la sesión y termina el caso de uso
Poscondiciones	1- El sistema muestra al administrador las estadísticas generales. 2- Quedan eliminadas las estadísticas deseadas

Tabla 2. 6: CU: Visualizar estadísticas generales

Imagen 2. 6: Interfaz del CU: Visualizar estadísticas generales

2.11.3.6 CU: Visualizar y cerrar token.

Caso de Uso: Visualizar y cerrar token.	
Actores:	Administrador del sistema.
Resumen:	El caso de uso se inicia cuando el administrador de sistema escoge la opción "Ver listado de tokens", de allí el sistema le dará la opción para eliminar el token o sesión deseada.
Precondiciones:	El administrador debe haber realizado el proceso de autenticación.
Referencia:	R7.
Prioridad:	Crítico.

Flujo Normal de Eventos	
Acción del Actor	Respuesta del Sistema
1. El administrador escoge la opción “Visualizar sesiones”.	1.1 El sistema le muestra la lista de todas los tokens o sesiones activa que tiene.
1.2 El administrador da clic en el botón “Cerrar sesión” de una sesión.	1.3 El sistema elimina el token, cerrándole la sesión al usuario y termina el casi de uso.
Poscondiciones	Se visualizan los tokens y se elimina el token deseado.

Tabla 2. 7: CU: Visualizar y cerrar token

Imagen 2. 7: Interfaz del CU: Visualizar y cerrar token

Anexo II: Diagramas de secuencia del diseño

DIAGRAMA DE SECUENCIA DEL DISEÑO: CU: Autenticar usuario en la aplicación web

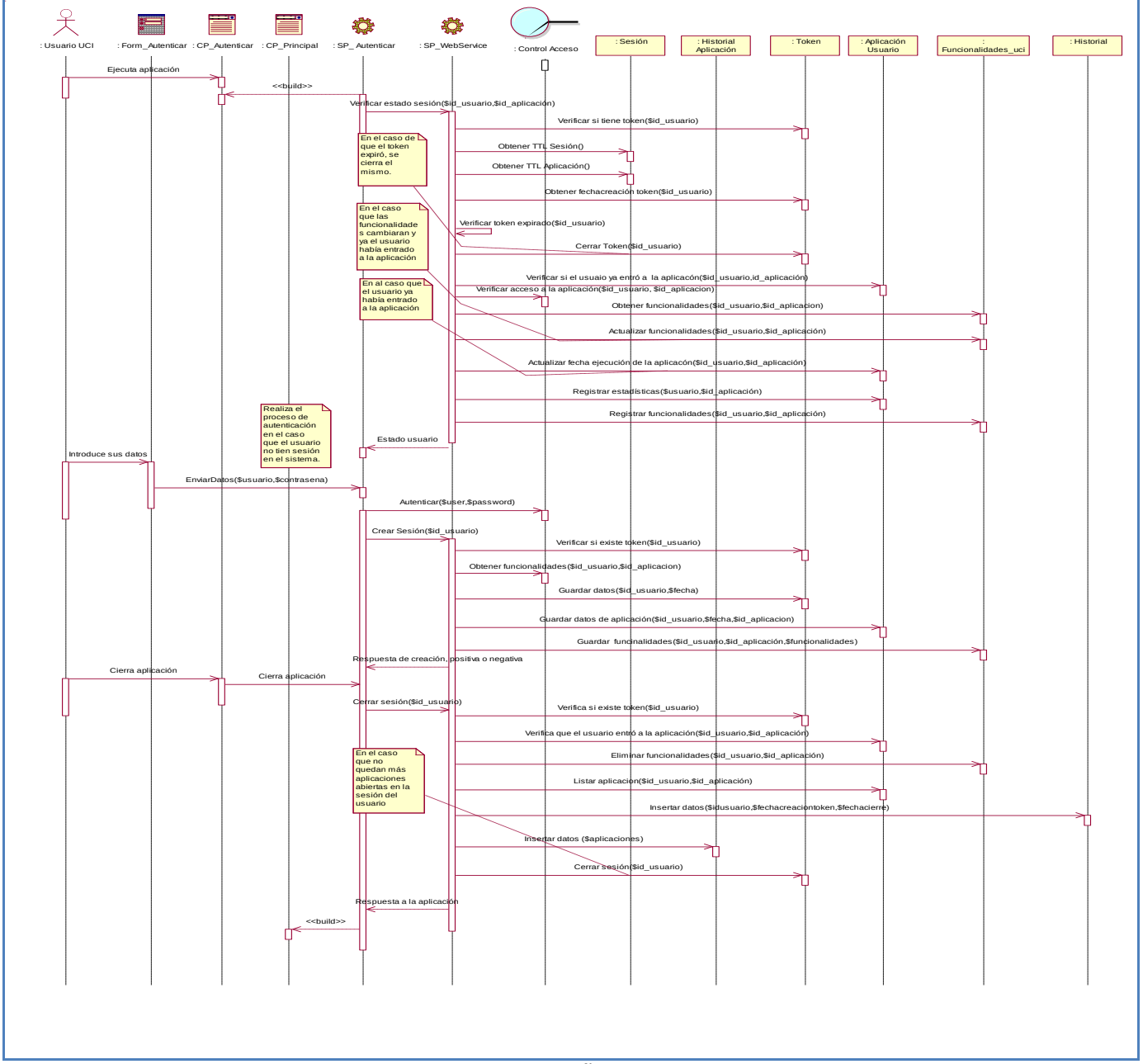

Imagen 3. 15: DIAGRAMA DE SECUENCIA DEL DISEÑO: CU: Autenticar usuario en la aplicación web

DIAGRAMA DE SECUENCIA DEL DISEÑO: CU: Gestionar Sesiones

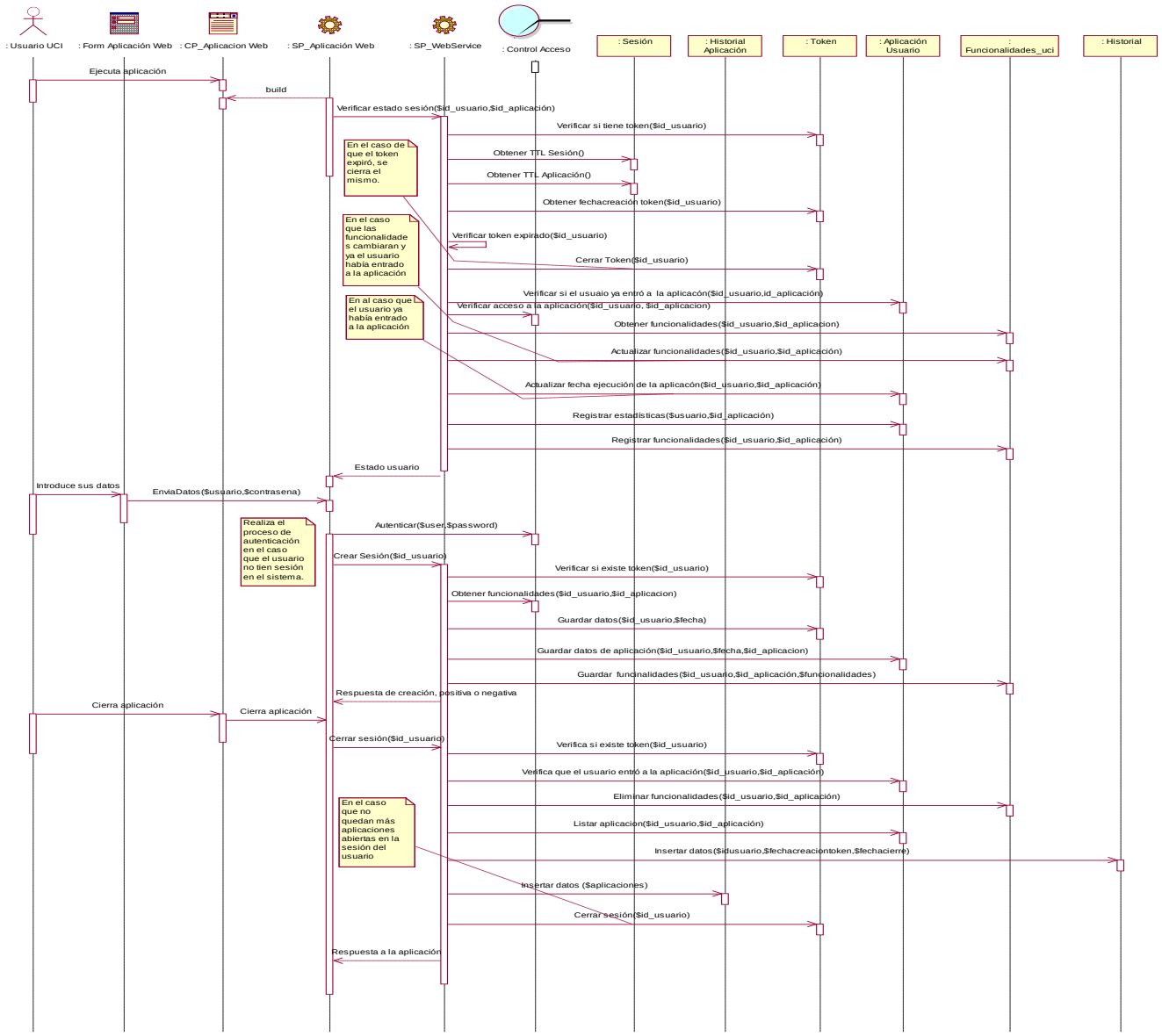

Imagen 3. 16: DIAGRAMA DE SECUENCIA DEL DISEÑO: CU: Gestionar Sesiones

DIAGRAMA DE SECUENCIA DEL DISEÑO: CU: Modificar TTL Aplicación, de Sesión y de Salva del Historial. Escenario 1

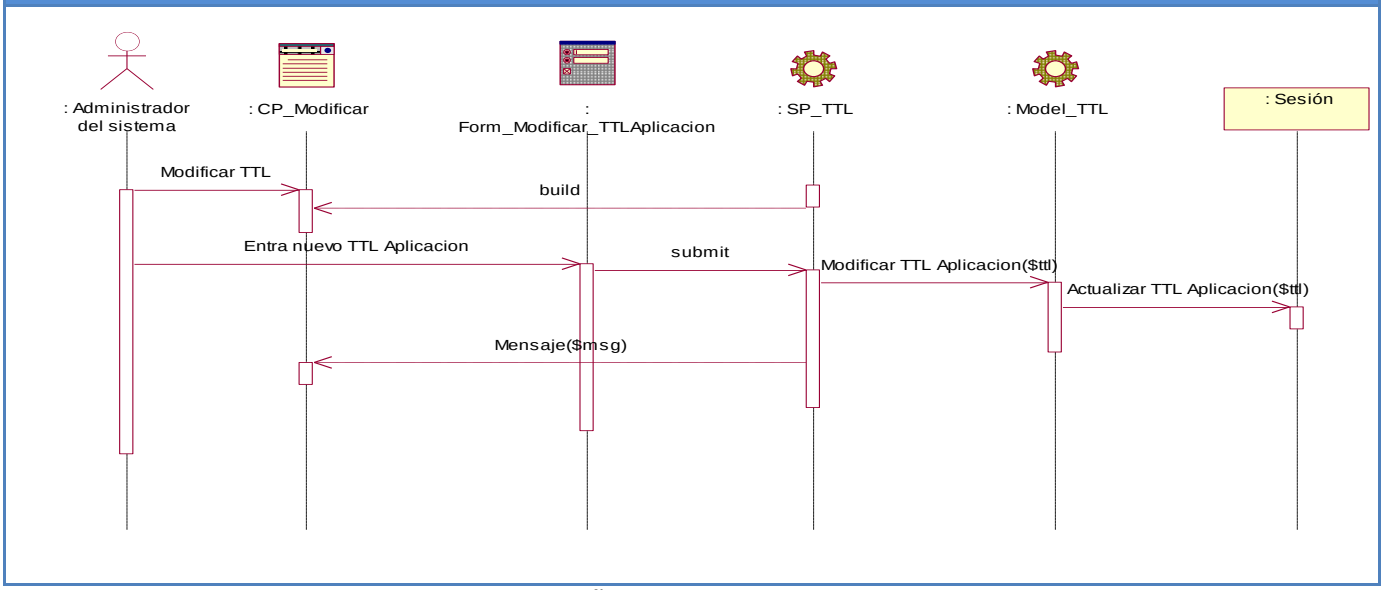

Imagen 3. 17: DIAGRAMA DE SECUENCIA DEL DISEÑO: CU: Modificar TTL Aplicación, de Sesión y de Salva del Historial. Escenario 1

DIAGRAMA DE SECUENCIA DEL DISEÑO: CU: Modificar TTL Aplicación, de Sesión y de Salva del Historial. Escenario 2

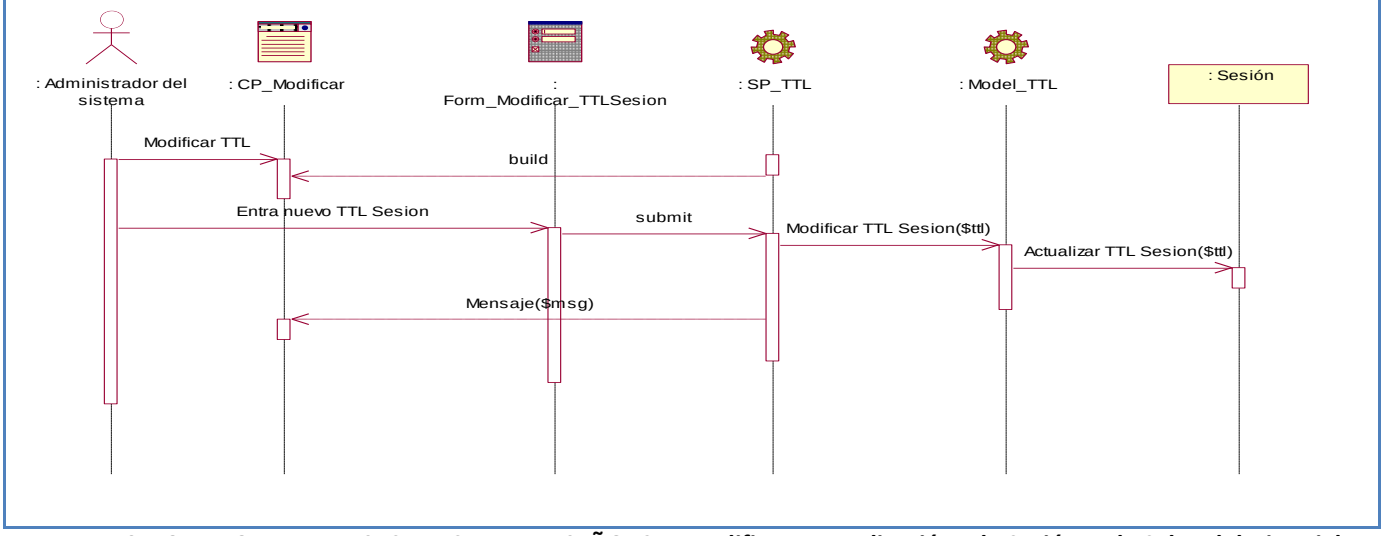

Imagen 3. 18: DIAGRAMA DE SECUENCIA DEL DISEÑO: CU: Modificar TTL Aplicación, de Sesión y de Salva del Historial. Escenario 2

DIAGRAMA DE SECUENCIA DEL DISEÑO: CU: Modificar TTL Aplicación, de Sesión y de Salva del Historial. Escenario 3

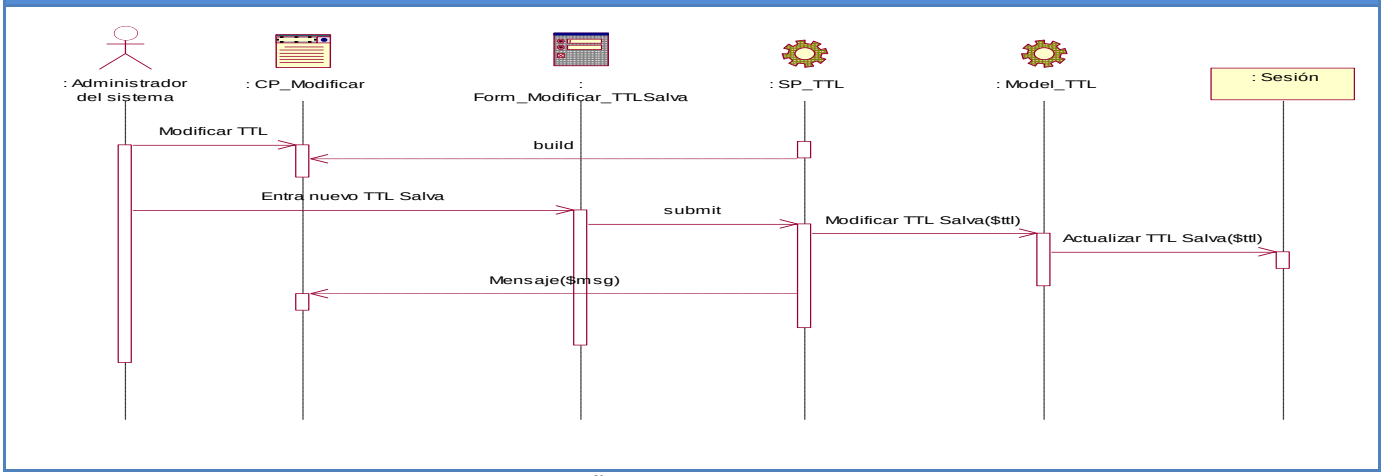

Imagen 3. 19: DIAGRAMA DE SECUENCIA DEL DISEÑO: CU: Modificar TTL Aplicación, de Sesión y de Salva del Historial. Escenario 3

DIAGRAMA DE SECUENCIA DEL DISEÑO: CU: Visualizar y cerrar token

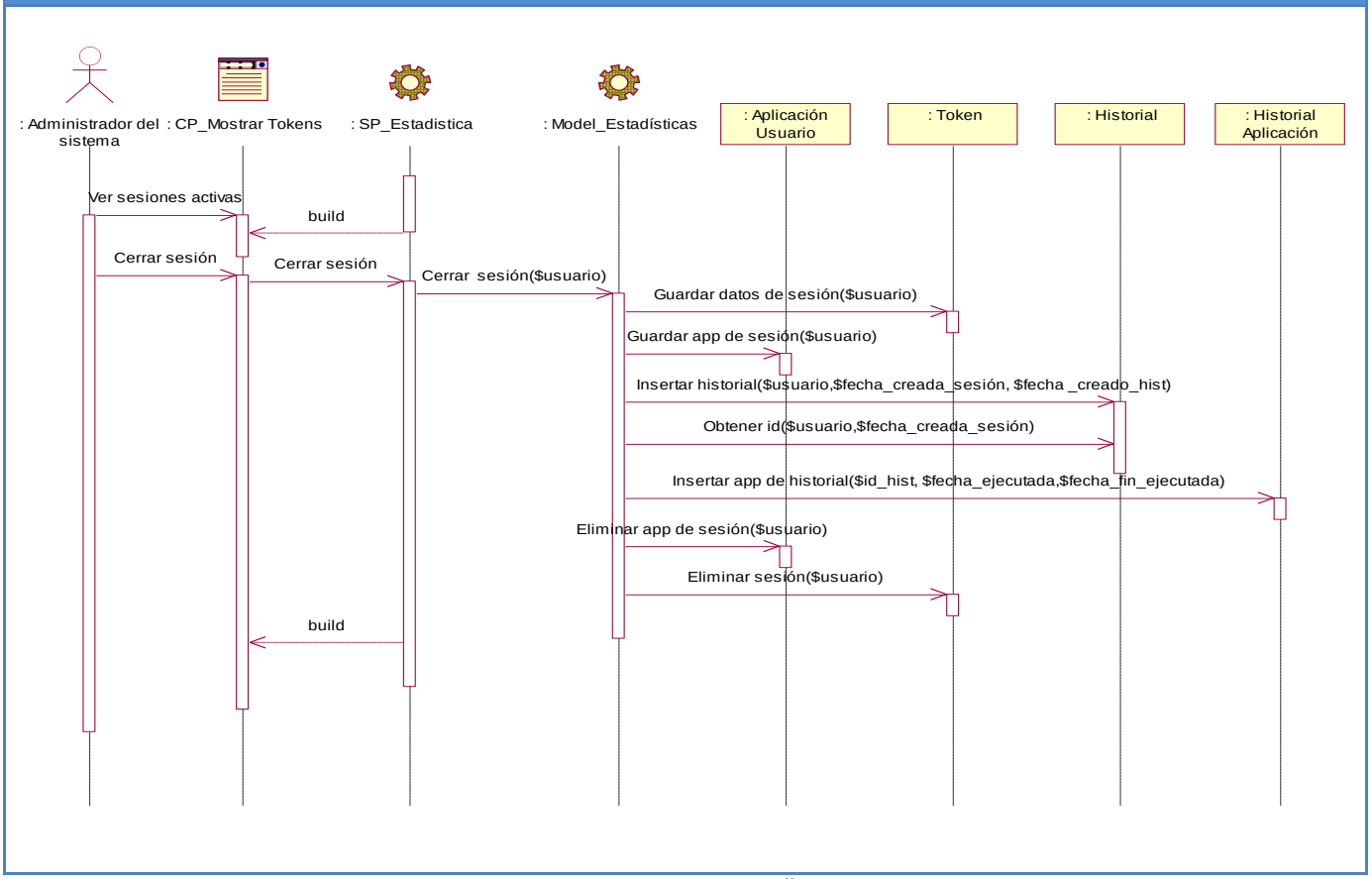

Imagen 3. 20: DIAGRAMA DE SECUENCIA DEL DISEÑO: CU: Visualizar y cerrar token

DIAGRAMA DE SECUENCIA DEL DISEÑO: CU: Visualizar estadísticas generales. Escenario 1

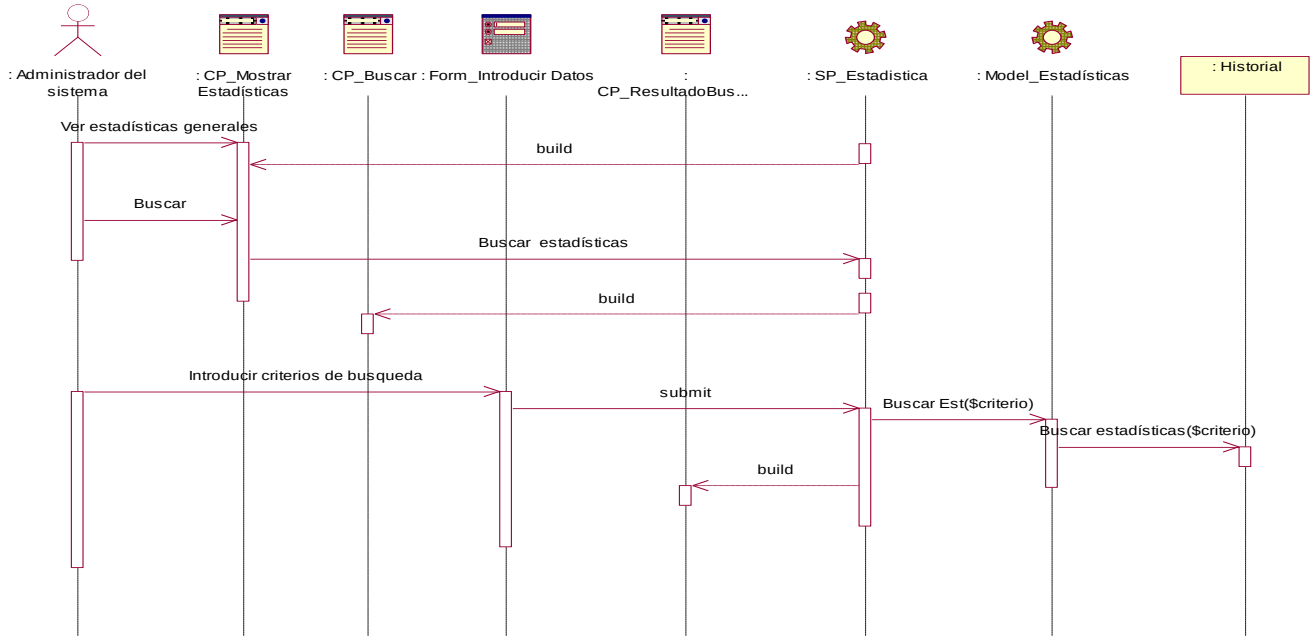

Imagen 3. 21: DIAGRAMA DE SECUENCIA DEL DISEÑO: CU: Visualizar estadísticas generales. Escenario 1

DIAGRAMA DE SECUENCIA DEL DISEÑO: CU: Visualizar estadísticas generales. Escenario 2

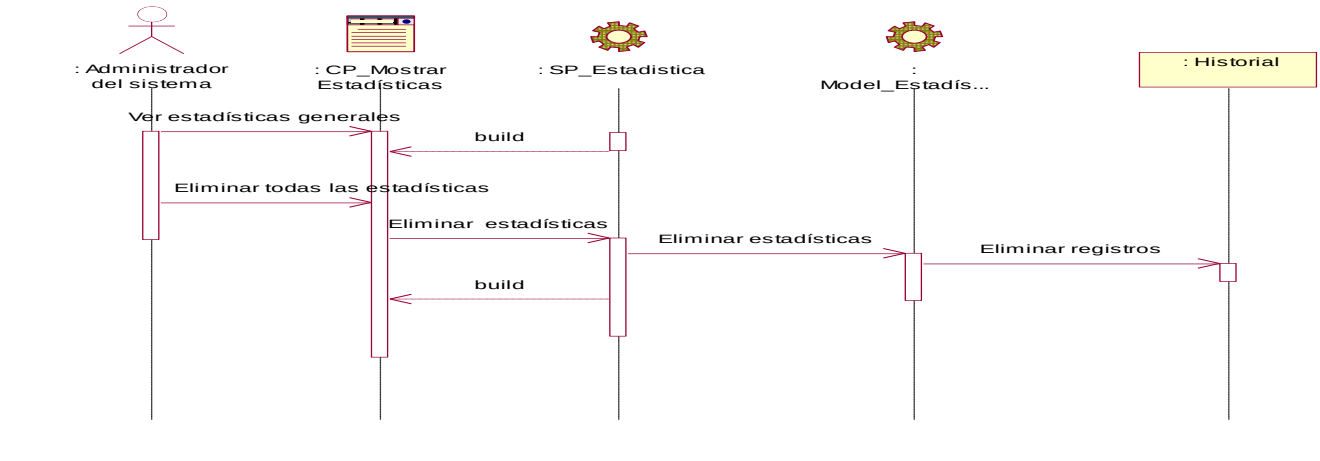

Imagen 3. 22: DIAGRAMA DE SECUENCIA DEL DISEÑO: CU: Visualizar estadísticas generales. Escenario 2

DIAGRAMA DE SECUENCIA DEL DISEÑO: CU: Visualizar estadísticas generales. Escenario 3

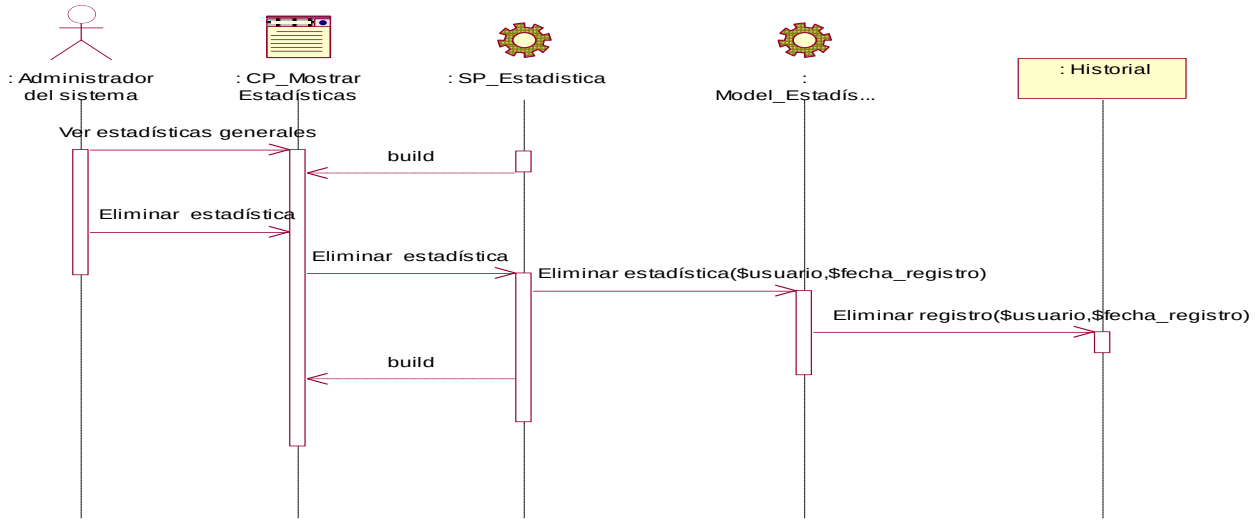

Imagen 3. 23: DIAGRAMA DE SECUENCIA DEL DISEÑO: CU: Visualizar estadísticas generales. Escenario 3

DIAGRAMA DE SECUENCIA DEL DISEÑO: CU: Visualizar estadísticas generales. Escenario 4

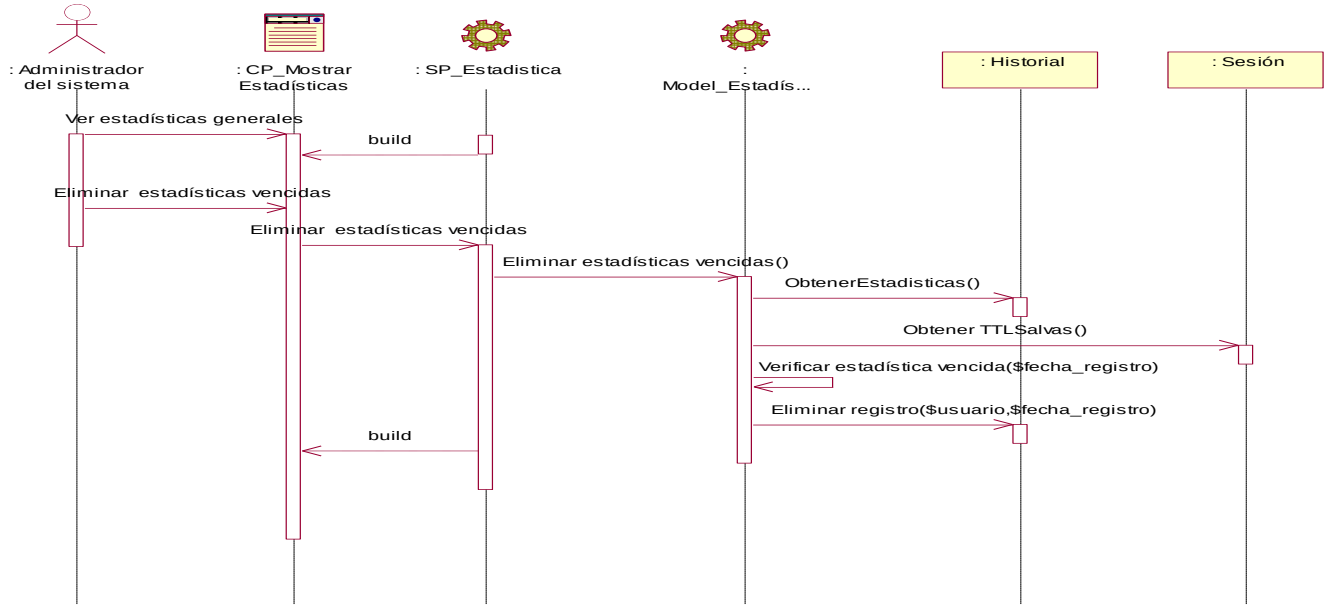

Imagen 3. 24: DIAGRAMA DE SECUENCIA DEL DISEÑO: CU: Visualizar estadísticas generales. Escenario 4

DIAGRAMA DE SECUENCIA DEL DISEÑO: CU: Visualizar estadísticas generales. Escenario 5

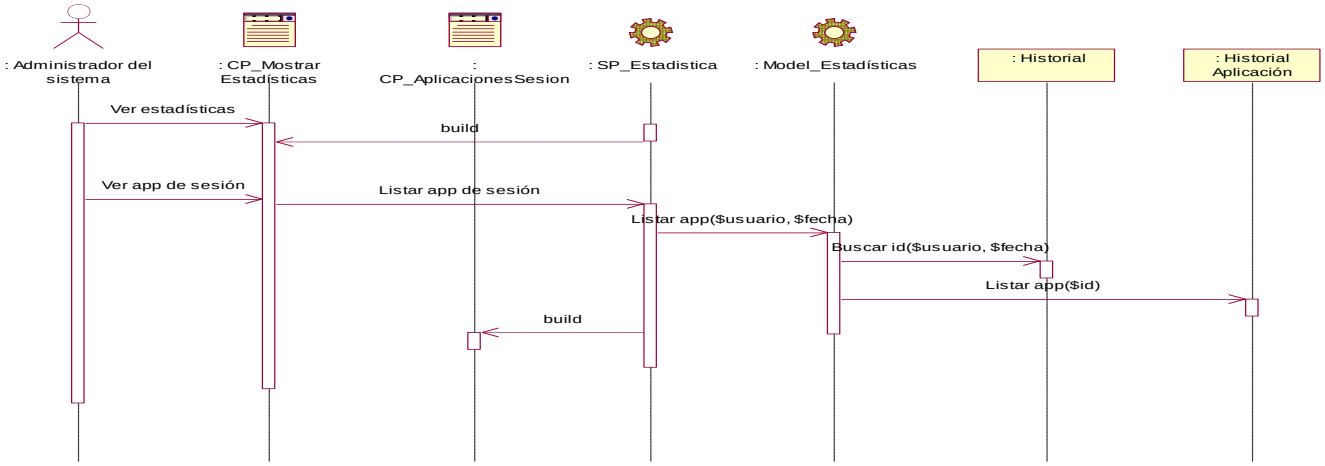

Imagen 3. 25: DIAGRAMA DE SECUENCIA DEL DISEÑO: CU: Visualizar estadísticas generales. Escenario 5

DIAGRAMA DE SECUENCIA DEL DISEÑO: CU: Visualizar estadísticas de usuario. Escenario 1

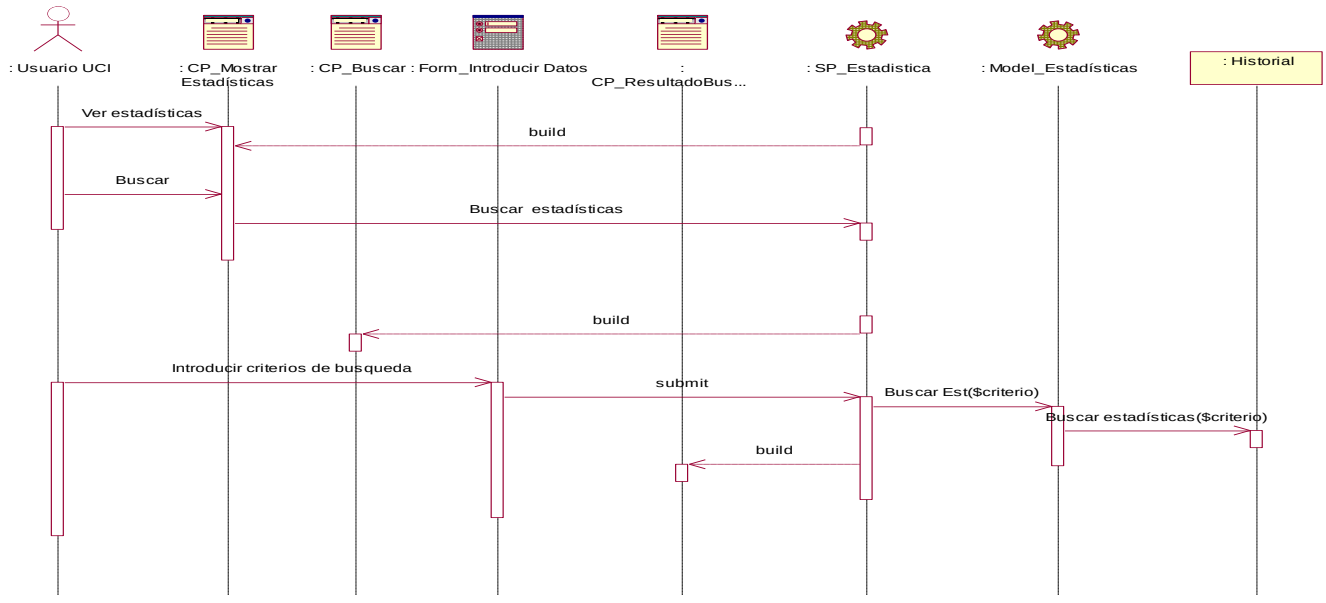

Imagen 3. 26: DIAGRAMA DE SECUENCIA DEL DISEÑO: CU: Visualizar estadísticas de usuario. Escenario 1

DIAGRAMA DE SECUENCIA DEL DISEÑO: CU: Visualizar estadísticas de usuario. Escenario 2

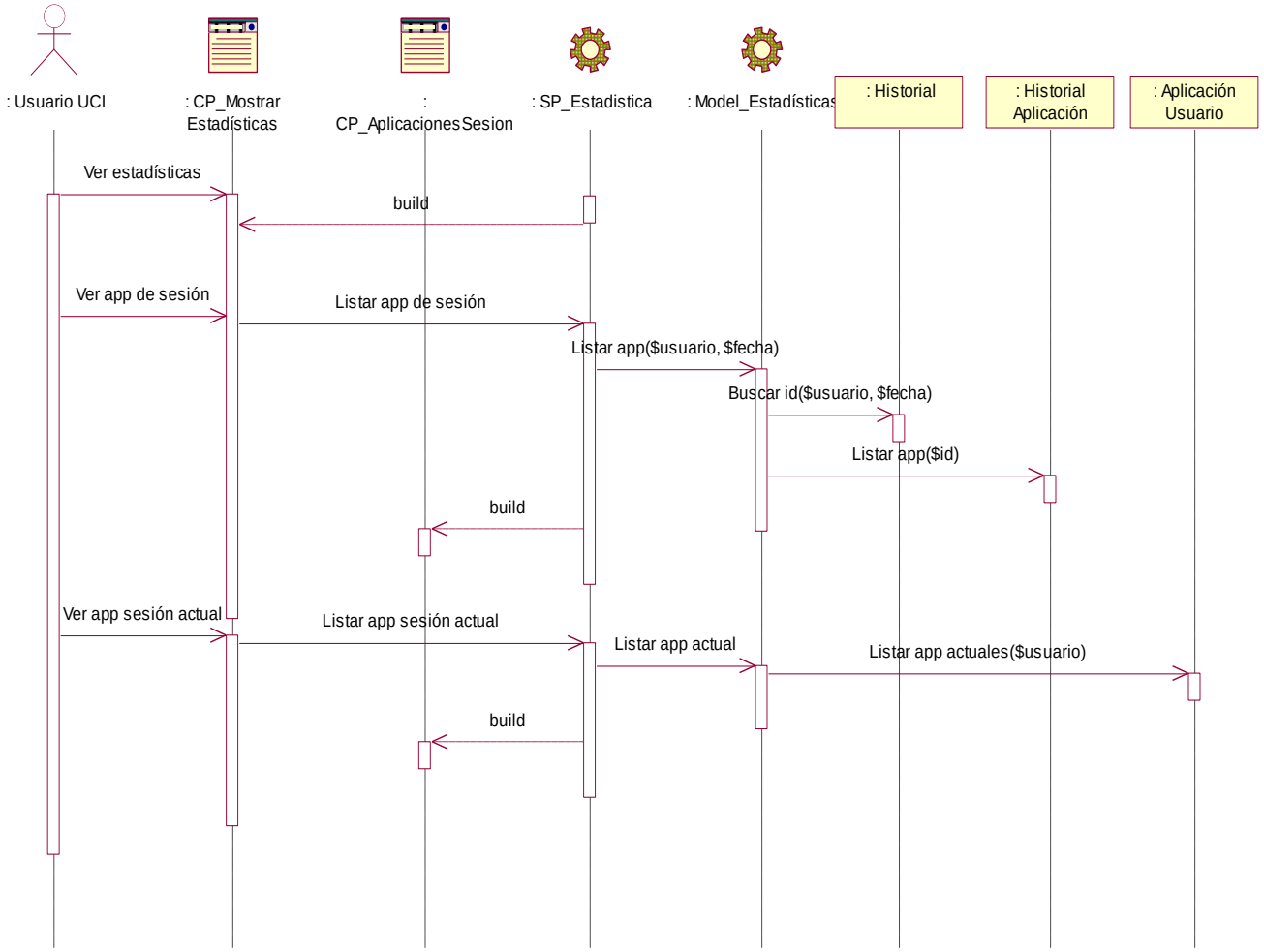

Imagen 3. 27: DIAGRAMA DE SECUENCIA DEL DISEÑO: CU: Visualizar estadísticas de usuario. Escenario 2

Glosario de Términos

SSO: Single Sign On. Sistema de autenticación única.

SGS: Sistema de Gestión de Sesiones: es un procedimiento de autenticación que habilita al usuario para acceder a varios sistemas con una sola instancia de identificación. Implementa una parte de los SSO.

Token: El token no es más que el es ticket que se le da al usuario al crear su sesión en el sistema, este guardará su hora y fecha de creado, el id del usuario, el TTL de la sesión y el id de la aplicación a la cual accede.

TTL Sesión: Time To Live: El "Time To Live" o Tiempo de Vida, es el tiempo de duración de una sesión de usuario en el sistema. Al cerrarse la sesión del sistema, el usuario debe repetir el proceso de autenticación. El TTL Sesión es predeterminado por el administrador del sistema.

TTL Aplicación: Es el tiempo de duración de cada aplicación en cada token o sesión del sistema. Mientras que haya una aplicación abierta con tiempo de vida, no expira el token, siempre y cuando este tenga todavía TTL de Sesión. Si a una aplicación se le acaba su TTL de aplicación y hay otra que todavía tiene TTL de aplicación, entonces se desactiva la misma, siempre y cuando el TTL Sesión no haya expirado todavía. Este TTL también lo define el administrador del sistema.

TTL Salvas del Historial: Es el tiempo que estarán guardados las trazas de los usuarios en el sistema. Lo define también el administrador del sistema.

MVC: Modelo Vista Controlador

Estadística: Registros que guarda el sistema de las trazas de un usuario.

Traza: Acciones que realiza un usuario, aplicaciones que ejecuta en un orden y en una fecha determinada.

CD-ROM (Compact Disc-Read Only Memory): Discos compactos de memoria legible que son solo de lectura y por lo tanto es imposible grabarlos.

USD (United States Dollar): Código ISO del dólar americano o estadounidense.

IBM (International Business Machines): Conocida coloquialmente como el Gigante Azul es una empresa que fabrica y comercializa herramientas, programas y servicios relacionados con la informática. Tiene su sede en Armonk (Estados Unidos). IBM es la segunda empresa relacionada con la informática en el mundo, detrás de Hewlett-Packard, y una de las pocas que lleva operando desde el siglo XIX hasta la actualidad.

SOA (Arquitectura Orientada a Servicios): Una arquitectura orientada a servicios (SOA) es un patrón en el que los recursos que están interconectados en una red se conciben como servicios accesibles por terceros a través de una interfaz estándar. En este modelo existen tres actores principales: el proveedor del servicio, el registro del servicio y el solicitante del servicio. Un componente en esta arquitectura podría considerarse como un servicio que puede ser publicado, descubierto e invocado de forma dinámica. La característica más importante de este modelo es el bajo grado de acoplamiento entre componentes junto a una mayor flexibilidad ante cambios futuros.

CGI (Interfaz de entrada común): Importante tecnología de la World Wide Web que permite a un cliente (explorador web) solicitar datos de un programa ejecutado en un servidor web. CGI especifica un estándar para transferir datos entre el cliente y el programa. Es un mecanismo de comunicación entre el servidor web y una aplicación externa.

Java Servlet: La palabra servlet deriva de otra anterior, applet, que se refería a pequeños programas escritos en Java que se ejecutan en el contexto de un navegador web. Por contraposición, un servlet es un programa que se ejecuta en un servidor. El uso más común de los servlets es generar páginas web de forma dinámica a partir de los parámetros de la petición que envíe el navegador web.

Active Server Pages (ASP): Tecnología del lado servidor de Microsoft para páginas web generadas dinámicamente, que ha sido comercializada como un anexo a Internet Information Server (IIS). La tecnología ASP está estrechamente relacionada con el modelo tecnológico de su fabricante. Intenta ser solución para un modelo de programación rápida.

CSS (Hojas de Estilo en Cascada (Cascading Style Sheets)): Mecanismo simple que describe cómo se va a mostrar un documento en la pantalla, o cómo se va a imprimir, o incluso cómo va a ser pronunciada la información presente en ese documento a través de un dispositivo de lectura. Esta

forma de descripción de estilos ofrece a los desarrolladores el control total sobre estilo y formato de sus documentos.

DOM (Document Object Model): Ofrece un modelo orientado a objetos para el tratamiento y manipulación en tiempo real (o en forma dinámica) a la vez que de manera estática de páginas de internet.

API (Application Programming Interface - Interfaz de Programación de Aplicaciones): Conjunto de funciones y procedimientos (o métodos si se refiere a programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

Paradigma P2P: Nuevo paradigma comunicativo caracterizado por una comunicación simétrica y democrática, sin ningún tipo de jerarquía que prime el papel del emisor como en los modelos comunicativos tradicionales. Este nuevo modelo de comunicación contempla una serie de características que permiten su implementación en sistemas audiovisuales infra desarrollados como el vídeo a la carta, aunque para ello aún deba superar algunos obstáculos tecnológicos y otros relacionados con los modelos de uso.

CIDR (Classless Inter-Domain Routing): Última mejora en el modo como se interpretan las direcciones IP. Permite una mayor flexibilidad al dividir rangos de direcciones IP en redes separadas.

Direcciones MAC (Media Access Control address): Identificador de 48 bits (6 bytes) que corresponde de forma única a una tarjeta o interfaz de red. Es individual, cada dispositivo tiene su propia dirección MAC determinada y configurada. Las direcciones MAC son únicas a nivel mundial, puesto que son escritas directamente, en forma binaria, en el hardware en su momento de fabricación. Debido a esto, las direcciones MAC son a veces llamadas Quemadas En Las Direcciones (BIA).

IPv4: Versión 4 del Protocolo IP (Internet Protocol). Esta fue la primera versión del protocolo que se implementó extensamente, y forma la base de Internet.

IPv6: Versión 6 del Protocolo de Internet (Internet Protocol), un estándar en desarrollo del nivel de red encargado de dirigir y encaminar los paquetes a través de una red. Sustituye al IPv4.

OMG (Object Management Group): Consorcio dedicado al cuidado y el establecimiento de diversos estándares de tecnologías orientadas a objetos, tales como UML, XMI, CORBA. Es una organización sin ánimo de lucro que promueve el uso de tecnología orientada a objetos mediante guías y especificaciones para las mismas.

FTP (File Transfer Protocol): Protocolo de transferencia de archivos entre sistemas conectados a una red TCP basado en la arquitectura cliente-servidor, de manera que desde un equipo cliente nos podemos conectar a un servidor para descargar archivos desde él o para enviarle nuestros propios archivos.

SFTP (Security File Transfer Protocol): Protocolo de red que proporciona la transferencia de archivos y la funcionalidad de manipulación de datos fiables sobre cualquier secuencia ya que los datos circulan cifrados por la red. Se utiliza normalmente con SSH a fin de asegurar la transferencia de archivos.