Universidad de las Ciencias Informáticas

Facultad 2

Componente para facilitar el proceso de autenticación de usuarios en aplicaciones informáticas en instituciones de salud.

Trabajo de Diploma para optar por el Título de

Ingeniero en Ciencias Informáticas

Autores: Dania María Téllez Kindelan

Roberto Asiel Guevara Castañeda

Tutores: Ing. Alain Ramos Medina

Ing. Noel Rodríguez Arias

Cotutor: Ing. Miguel Alejandro Nicao Cepeda

Ciudad de La Habana, junio de 2015

"Año 57 de la Revolución"

DECLARACIÓN DE AUTORÍA

Declaramos que somos los únicos autores de este trabajo y autorizamos a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmamos la presente a los 23 días del mes de junio del año 2015.

Autores:

Dania María Téllez Kindelan	Roberto Asiel Guevara Castañeda

Tutores:

Ing. Alain Ramos Medina	Ing. Noel Rodríguez Arias

DATOS DE CONTACTO

Tutores:

Ing. Alain Ramos Medina.

Graduado de Ingeniero en Ciencias Informáticas, en la Universidad de las Ciencias Informáticas en el año 2007. Especialista, vinculado al Centro de Informática Médica en el Departamento de Desarrollo de Componentes. Actualmente se encuentra vinculado al desarrollo del Sistema de Información Hospitalaria del CESIM y es el jefe del proyecto Sistema de Información Hospitalaria.

Correo electrónico: aramos@uci.cu

Ing. Noel Rodríguez Arias.

Graduado de Ingeniero en Ciencias Informáticas del año 2011 en la Universidad de las Ciencias Informáticas de Cuba. Especialista, vinculado al Centro de Informática Médica en el Departamento de Desarrollo de Componentes. Actualmente se encarga de la asesoría comercial del Centro de Informática Médica.

Correo electrónico: nrarias@uci.cu

Cotutor:

Ing. Miguel Alejandro Nicao Cepeda

Ingeniero en Ciencias Informáticas, graduado en la Universidad de las Ciencias Informáticas en el año 2013. Especialista, vinculado al Centro de Informática Médica en el Departamento de Desarrollo de Componentes. Actualmente se encuentra vinculado al desarrollo del Sistema de Información Hospitalaria del CESIM.

Correo electrónico: manicao@uci.cu

DEDICATORIA

Danía:

Quiero dedicarle este trabajo especialmente a mi Madre por ser la persona a quien más amo y estimo, la cual es mi ejemplo a seguir.

Roberto:

Le dedico este trabajo a mi Padre pues gracias a él no estaría hoy en esta universidad, para él un abrazo donde quiera que esté.

AGRADECIMIENTOS

Queremos agradecer a nuestros tutores, Alain y Noel por el apoyo incondicional en el desarrollo de este trabajo.

A nuestros compañeros durante nuestros años de la carrera.

Dania:

Quiero agradecer a mi mamita Mercedes por siempre confiar en mí, apoyarme en todas las decisiones que he tomado en mi vida, por alentarme a superarme, por sacrificarse tanto por nuestra familia y por hacerme sentir orgullosa de ser su hija.

A mi hermana Annia por todo el apoyo que me ha brindado durante mis años de carrera y por hacer el papel de mi segunda mamita.

A mis sobrinitos Aneys, Cesar y Rey por ser mis bebes a los cuales considero mis hijitos del alma.

A mi abue Dulce por darme siempre su amor incondicional.

A mi mejor amiga Ismaray por ser como una hermana para mí, apoyarme, soportarme y decirme la verdad en todo momento.

A Migue por haber sido el guía que me mostro el camino a seguir durante mis años de carrera donde me encontraba alejada de mi familia, por siempre alentarme a superarme profesionalmente y por ser una persona merecedora de todo mi cariño y respeto.

A mi suegri Niurka por ser otra mamá para mí y apoyarme en estos momentos tan difíciles de mi vida.

A Urbano por hacer el papel de mi padre, al cual quiero y respeto mucho.

A mis profesores, compañeros de grupo y a todas las personas que de una forma u otra contribuyeron en mi formación.

Roberto:

Quisiera agradecerles primeramente a mis profesores en todos los cursos por su paciencia y dedicación, y agradecerles en especial a mis compañeros de cuarto por su ayuda y apoyo incondicional.

A mi mamá por apoyarme siempre en todas las decisiones que tome y por siempre estar ahí cuando más la necesité, para ti mami un beso del tamaño de tu corazón que es inmenso.

A mi abuelita querida por aconsejarme siempre cuando veía que estaba haciendo las cosas mal. Para ti "Tía" un beso igual de grande.

A mi hermanita "Dalín" por ser la mejor hermana del mundo y siempre estar de mi lado en las buenas y en las malas. Un beso mi hermana.

A mi sobrinita que aunque no me reconozca casi sé que me va a querer muchísimo y estará orgullosa de su tío.

A mi familia en general por su preocupación y por estar presentes en los momentos más difíciles de mi vida.

Y quisiera agradecerle infinitamente a mi Papá que, aunque hoy no este entre nosotros, siempre supo que era lo mejor para sus hijos y siempre quiso que nosotros fuéramos profesionales y tuviéramos las oportunidades que él nunca tu cuando tenía mi edad. Para él va esta tesis, este título y por qué no, esta ingeniería en ciencias informáticas. Para mi papá de su hijo que lo quiso, lo quiere y lo querrá por siempre va este trabajo, un beso y un abrazo grande.

RESUMEN

Para el proceso de autenticación en sistemas informáticos relacionados con el ámbito de la salud, la Alternativa de Integración Sanitaria propone los perfiles de integración *Enterprise User Authentication* y *Cross Enterprise User Assertion*, los cuales son los encargados de centralizar y hacer única la autenticación de usuarios, a nivel empresarial y entre empresas respectivamente. El Centro de Informática Médica de la Universidad de las Ciencias Informáticas desarrolla diversos sistemas para la salud. Estos pueden converger en una misma estación de trabajo lo que resulta engorroso para la autenticación de usuario pues deben ingresar sus credenciales constantemente. Para tratar de resolver este aspecto, existen algunas soluciones que resuelven parcialmente el problema pero presentan diversos inconvenientes.

Por tales motivos el propósito de este trabajo es desarrollar un componente que centralice y unifique la autenticación del usuario reduciendo la cantidad de acciones de autenticación en las aplicaciones que el profesional de la salud desee emplear en su estación de trabajo y agilizando la atención al paciente. Para llevar a cabo la implementación del componente se hizo uso del perfil *Enterprise User Authentication* de la Iniciativa de Integración Sanitaria para el Centro de Informática Médica enfocándonos específicamente en el Sistema de Información Hospitalaria. Este componente estuvo guiado por el Proceso Unificado de Desarrollo, se basó en tecnologías libres, se utilizó Java como lenguaje de programación y como servidor de aplicaciones WSO2 AS. Al concluir el desarrollo del componente se obtuvo un producto totalmente funcional donde el Sistema de Información Hospitalaria logró integrarse correctamente.

Palabras clave: *Enterprise User Authentication*, IHE, Kerberos, autenticación única.

TABLA DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

El desarrollo de las tecnologías en el campo de la informática ha propiciado el avance en distintas esferas de la sociedad. La medicina es una esfera donde el desarrollo de esta ciencia ha tenido un gran impacto. La informática médica (IM) se encarga de prestar servicios a los profesionales de la salud para mejorar los procesos asistenciales y administrativos de las instituciones hospitalarias.

La IM es la aplicación en el área de la salud de las tecnologías de la información mediante el uso del *software* médico, para el almacenamiento, procesamiento, análisis y recuperación de datos, asociados a la prestación de servicios sanitarios, la investigación, la elaboración de diagnósticos y toma de decisiones en la asistencia a pacientes (1). La misma tiene como objetivo fundamental, brindar determinados servicios a los profesionales de la salud para mejorar la calidad de la atención sanitaria en sus tres niveles[1] y contribuir al bienestar de las personas brindándoles un mejor servicio, a través del uso de sistemas informáticos.

Los sistemas informáticos desarrollados para las instituciones de salud tienen el fin de mejorar los servicios que allí se prestan, entre estos se encuentran los Sistemas de Almacenamiento, Transmisión y Visualización de Imágenes Médicas (*Picture Archiving and Communication System* PACS, por sus siglas en inglés) (2), Sistemas de Información Radiológica (*Radiology Information System* RIS, por sus siglas en inglés) (3), Sistemas de Información de Laboratorio (*Laboratory Information System* LIS, por sus siglas en inglés) (4), Sistemas de Información Hospitalaria (*Health Information System* HIS, por sus siglas en inglés) (5). Los mismos tienen como propósito satisfacer las necesidades de generación de información, para almacenar y procesar datos médico-administrativos de cualquier institución hospitalaria, permitiendo la optimización de los recursos humanos y materiales, además de minimizar los inconvenientes de trámites administrativos que enfrentan los pacientes.

[1] Niveles de la atención sanitaria: el primer nivel se denomina Atención Primaria que constituye el más cercano a la población específicamente para la medicina preventiva, el segundo nivel se denomina Atención Secundaria que brinda asistencia médica en los hospitales y el tercer nivel se denomina Atención Terciaria formado por institutos especializados.

En la actualidad diversos países están inmersos en el desarrollo de sistemas informáticos para la salud. Entre estos países se encuentra Cuba, el cual está llevando a cabo un proceso de informatización del Sistema Nacional de Salud, donde participan empresas tales como Softel, Desoft y la Universidad de las Ciencias Informáticas (UCI).

La UCI tiene 14 centros productivos dentro de los que se encuentra el Centro de Informática Médica (CESIM), que desarrolla aplicaciones para la salud. Entre sus productos cuenta con un HIS, un RIS, un Sistema de Gestión para la Ingeniería Clínica y Electromedicina, un Sistema de Gestión de Ensayos Clínicos, un Sistema de Información Hospitalaria para Clínicas Estomatológicas, un Sistema de Telemedicina y un PACS. Estas soluciones informáticas para su comunicación necesitan implementar elementos de los estándares internacionales tales como HL7 (6) y DICOM (7) que facilitan el intercambio de información médica entre dichos sistemas.

Estos estándares por si solos no garantizan que toda la información necesaria en la atención del paciente sea correcta y disponible para los profesionales de la salud. Por tal motivo surge la necesidad de implementar un mecanismo de integración que asegure la interoperabilidad entre Sistemas de información en organizaciones e instituciones de salud.

Los sistemas anteriormente mencionados desarrollados por el CESIM cuentan con:

- Un servicio independiente de autenticación.
- Convergen en una misma institución y/o estación de trabajo.
- Se hace necesario tomar información referente a cada uno de los usuarios que harán uso de los mismos, obligándolos a manejar el par: usuario y contraseña por cada aplicación.
- La mayoría de los usuarios aplican varios métodos poco seguros para manejar la situación de aprenderse varias contraseñas, como contraseñas en notas adhesivas, contraseñas muy fáciles, contraseñas debajo del teclado, entre otros.

Aun lográndose la autenticación por medio de Protocolo Ligero de Acceso a Directorio (*Lightweight Directory Access Protocol* LDAP, por sus siglas en inglés) (8), u otra vía que centralice este proceso, al cambiar de aplicación en una misma estación de trabajo, el usuario se ve obligado a autenticarse una y otra vez según la cantidad de aplicaciones disponibles. La introducción manual de estos datos toma mucho tiempo y es difícil recordar múltiples contraseñas, lo que entorpece o ralentiza la atención al

paciente. Estos elementos se evidencian con las frecuentes llamadas de los usuarios al soporte técnico para solucionar el problema de contraseñas olvidadas.

Por lo anteriormente planteado se define como **problema de la investigación**: ¿Cómo facilitar el proceso de autenticación de usuarios en aplicaciones informáticas en instituciones de salud?

El **objeto de estudio** definido para dar solución al problema planteado es: la interacción entre aplicaciones informáticas en el área de la salud. Enmarcado en el **campo de acción** proceso de autenticación de usuarios en aplicaciones informáticas en instituciones de salud.

Como **objetivo general** se plantea: desarrollar un componente de *software* para facilitar el proceso de autenticación de usuarios en aplicaciones informáticas en instituciones de salud.

Para dar cumplimiento al objetivo planteado se proponen las siguientes **tareas de la investigación**:

- Analizar los estándares o procedimientos de comunicación y perfiles que permitan el inicio de sesión único a varias aplicaciones de una misma organización.
- Identificar las dependencias y/o requerimientos de los perfiles que permitan el inicio de sesión único a varias aplicaciones de una misma organización.
- Implementar el perfil de autenticación empresarial de usuarios.
- Validar la correcta integración del Sistema de Información Hospitalaria al componente de autenticación.

Para la realización de la presente investigación se hace uso de los siguientes métodos científicos:

Analítico – Sintético: se realizó el estudio (análisis) de documentos especializados con la problemática planteada, para así poder resumir (síntesis) la información que es relevante, permitiendo efectuar una correcta investigación.

Histórico – Lógico: se realizó el estudio del arte de cómo se llevaba a cabo actualmente el proceso de autenticación de usuarios en las aplicaciones de las instituciones de salud.

Modelación: se emplea este método científico pues a través de él se realizaron los modelos y diagramas correspondientes que servirán de apoyo para la implementación del componente de autenticación.

Beneficios

Con el desarrollo del componente para facilitar el proceso de autenticación de usuarios en aplicaciones informáticas para la salud se esperan los siguientes beneficios:

- Sincronización de autenticación en las aplicaciones que el profesional de la salud utilice en la misma estación de trabajo, ya que solo la primera aplicación iniciada en dicha estación de trabajo solicitará información de autenticación.
- Mayor rapidez en el proceso de atención al paciente que requiera para su cumplimiento el empleo de más de un sistema informático.

El trabajo de diploma se divide en cuatro capítulos, los cuales estarán estructurado de la siguiente forma:

Capítulo 1. Fundamentación teórica del componente para facilitar el proceso de autenticación de usuarios en aplicaciones informáticas en instituciones de salud: se describen los principales conceptos y las tendencias actuales referentes a la autenticación de usuario centralizada y única. Son fundamentadas las tecnologías, metodologías y herramientas de desarrollo a utilizar.

Capítulo 2. Características del componente para facilitar el proceso de autenticación de usuarios en aplicaciones informáticas en instituciones de salud: se describen las principales características del componente que permitirá facilitar el proceso de autenticación de usuarios en las aplicaciones informáticas en las instituciones de salud. Se presenta el Modelo de Dominio, se especifican los Requisitos Funcionales y No Funcionales del *software*, además se define el modelo de Casos de Uso del Sistema.

Capítulo 3. Análisis y diseño del componente para facilitar el proceso de autenticación de usuarios en aplicaciones informáticas en instituciones de salud: se realiza la descripción y análisis de la estructura del componente que permitirá facilitar el proceso de autenticación de usuarios en las aplicaciones informáticas en las instituciones de salud, el cual se propone para dar respuesta a la problemática planteada. Se especifican los elementos del diseño tales como el Diagrama de Paquetes y Diagramas de Clases del Diseño con sus correspondientes descripciones de clases del diseño. Se fundamenta la arquitectura empleada así como la estrategia de integración a tener en cuenta.

Capítulo 4. Implementación del componente para facilitar el proceso de autenticación de usuarios en aplicaciones informáticas en instituciones de salud: partiendo de los resultados obtenidos en el diseño se avanza a la fase de implementación donde se exponen aspectos tales como el tratamiento de

errores, estrategias de codificación y la elaboración del Diagrama de Despliegue y del Diagrama de Componentes.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA DEL COMPONENTE PARA FACILITAR EL PROCESO DE AUTENTICACIÓN DE USUARIOS EN APLICACIONES INFORMÁTICAS EN INSTITUCIONES DE SALUD

En el presente capítulo se describen los conceptos fundamentales asociados al proceso de autenticación de usuarios en aplicaciones informáticas en instituciones de salud. Se realiza el estudio del arte de los estándares utilizados para la interoperabilidad entre aplicaciones informáticas y los perfiles que permiten facilitar el proceso de autenticación de usuarios en aplicaciones informáticas para la salud. Son analizados los antecedentes referentes al tema en cuestión, además son fundamentadas las tecnologías, metodología y herramientas a utilizar.

1.1 Conceptos asociados al dominio del problema

La **seguridad de la información** es la preservación de las siguientes características: confidencialidad, integridad y disponibilidad y se logra implementando un conjunto adecuado de controles, que abarcan políticas, prácticas, procedimientos, estructuras organizacionales y funciones del *software*. (9)

Control de acceso: es el conjunto de políticas que definen las normas en todo lo que el usuario se le permite hacer en un sistema. Estas políticas se definen en un alto nivel independiente de los sistemas específicos, por lo que se pueden aplicar a cualquier sistema, ya que no dependen del mismo. (9)

El control de acceso consta de tres pasos:

Paso 1. Identificación: es la acción mediante la cual el usuario dice ser quién es realmente.

Paso 2. **Autenticación**: verificación de la identidad de un usuario cuando trata de acceder a un sistema. Es la comprobación de que esa entidad es quien dice ser en realidad. Existen cuatro tipos de técnicas que permiten realizar la autenticación de la identidad del usuario, las cuales pueden ser utilizadas individualmente o combinadas, las mismas son: algo que solamente el individuo conoce (por ejemplo las contraseñas), algo que la persona posee (por ejemplo tarjetas magnéticas), algo que el individuo es o lo identifica unívocamente (por ejemplo huellas digitales) y algo que solo el individuo es capaz de hacer (por ejemplo patrones de escritura).

Paso 3. Autorización: es el proceso de determinar si una cuenta identificada y verificada tiene los permisos para el acceso a los recursos. (9)

El desarrollo de la presente investigación se enmarca en el proceso de autenticación de las aplicaciones de salud desarrolladas en el Centro de Informática Médica.

Debido a la gran cantidad de sistemas en los que un determinado usuario tiene que autenticarse, debe recordar múltiples nombres de usuario y sus respectivas contraseñas lo cual es muy engorroso. Para solucionar este problema de manejar diversas credenciales, existe el procedimiento *Single Sing On* lo cual permite que el proceso de autenticación sea único.

Sistema centralizado de autenticación y autorización (*Single Sign On* (SSO), por sus siglas en inglés): es un procedimiento de autenticación que permite el acceso de los usuarios a varios sistemas heterogéneos a través de una única instancia de identificación. El mismo aumenta la seguridad, brinda facilidad de acceso a recursos y eficiencia del tiempo pues una vez que un usuario inicia sesión, no es necesario introducir nuevamente sus credenciales para acceder a otras aplicaciones que desee utilizar. El SSO se transforma en un punto único de fallo, pues una pérdida de disponibilidad puede resultar en la denegación de acceso a todos los sistemas unificados bajo el mismo. (10)

Existen diferentes tipos de SSO y cada uno de ellos posee características apropiadas para distintos tipos de organización:

- **Sistema centralizado de autenticación y autorización para empresas** (ESSO, por sus siglas en inglés), es un sistema diseñado para reducir el número de veces que un usuario debe escribir su ID y contraseña para iniciar sesión en múltiples aplicaciones, el mismo intercepta los requerimientos de *login* presentados por las aplicaciones para completarlos con el usuario y contraseña. Los sistemas ESSO permiten interactuar con sistemas que pueden deshabilitar la presentación de la pantalla de *login*. (11)
- **Sistema centralizado de autenticación y autorización para *web*** (Web-SSO, por sus siglas en inglés), trabaja sólo con aplicaciones y recursos accedidos vía *web*. Los accesos son interceptados por un componente instalado en el servidor *web* destino. Los usuarios no autenticados que tratan de acceder son redirigidos a un servidor de autenticación y regresan solo después de haber logrado un acceso exitoso. (12)

- **Identidad Federada**: permite la gestión de identidad interdependiente entre compañías y que las aplicaciones identifiquen a los usuarios sin necesidad de que la autenticación sea repetida sino una única vez.

  Mediante soluciones de Identidad Federada los individuos pueden emplear la misma identificación personal para identificarse en diferentes redes, aplicaciones o empresas. De este modo distintas aplicaciones pueden compartir información incluso cuando no comparten tecnologías de directorio, seguridad y autenticación. Para su funcionamiento es necesaria la utilización de estándares que definan mecanismos que permiten a las aplicaciones compartir información entre dominios e identificar a un determinado usuario en una comunidad determinada con acceso a servicios específicos. (13)
- **OpenID**: es un proceso de SSO descentralizado que permite que la identificación digital de un usuario a una página *web* se haga a través de una URL que pueda ser verificado por cualquier servidor que lo soporte. En los sitios que soportan OpenID, los usuarios no tienen que crearse una nueva cuenta de usuario para acceder al mismo sino solamente necesitan disponer de un identificador creado en un servidor que verifique OpenID, llamado proveedor de identidad o IdP que se encarga de confirmar la identificación del usuario. (14)

1.2 Estándares y perfiles

Se utilizan varios estándares o iniciativa de integración con el objetivo de aumentar la interoperabilidad entre las aplicaciones, que han sido desarrolladas con el fin de satisfacer las necesidades de gestión de la información en las instituciones de salud. Muestra de estos son:

- **HL7**: es un estándar sin fines de lucro, acreditado por ANSI[2], organización de desarrollo dedicada a proporcionar un marco global y normas conexas para el intercambio, integración, cooperación y recuperación de la información electrónica de salud que soporta la práctica clínica y la gestión, ejecución y evaluación de los servicios de salud. (6)

[2] ANSI es el Instituto Nacional Estadounidense de estándares.

- **OASIS**: es un consorcio sin fines de lucro que impulsa el desarrollo, la convergencia y la adopción de estándares abiertos para la sociedad global de la información. Promueve consensos de la industria y produce estándares internacionales para la seguridad, internet, la computación en nube, la energía, gestión de emergencias y otras áreas con el objetivo de potenciar la reducción de los costos, estimular la innovación y el crecimiento de estos en el mercado a nivel mundial. (15)
- **ASTM**: establecida en 1898 originalmente como la Sociedad Americana para Pruebas y Materiales, es una de las mayores organizaciones voluntarias desarrolladoras de estándares. ASTM es una organización sin fines de lucro que proporciona un foro para el desarrollo y la publicación de los estándares internacionales de consenso voluntario para materiales, productos, sistemas y servicios. Miembros de ASTM, representan a los productores, usuarios, consumidores, gobierno y académicos de más de 150 países, desarrollan documentos técnicos que son la base para la fabricación, gestión, adquisición, códigos y reglamentos. (16)
- **IHE**: se creó en 1998 por parte de usuarios y empresas de Estados Unidos con el objetivo de desarrollar especificaciones técnicas para lograr solucionar los problemas de interoperabilidad entre los Sistemas de Información Sanitarios. IHE crea perfiles de integración basados en los estándares más apropiados y define las características esenciales para dar soporte a las tareas clínicas que debe tener un producto que quiera declararse conforme a dicho perfil. Los perfiles IHE especifican la información que debe ser intercambiada entre dos sistemas y las acciones que los sistemas receptores deben realizar al recibir la información. (17)

IHE define diversos perfiles de Integración, pero solo XUA y EUA son los encargados de definir una serie de pasos a seguir para lograr una autenticación segura y simple para el usuario.

***Cross-Enterprise User Assertion Profile* (XUA):** proporciona un medio para comunicar afirmaciones de un nodo autenticado (usuario, aplicación, sistema) en las transacciones entre empresas.

El perfil XUA apoya a las empresas que han optado por tener su propio directorio de usuario con su método único de autenticación, así como otras que hubiesen optado por un tercero para realizar la autenticación. (18)

***Enterprise User Authentication* (EUA):** especifica un medio para establecer un nombre por usuario que puede ser utilizado en todos los dispositivos y *software* que participan en este perfil de integración, lo que facilita enormemente la gestión de la autenticación de usuario centralizada y proporciona a los usuarios la

comodidad y rapidez de un inicio de sesión único. Este perfil aprovecha **Kerberos** y el estándar **HL7 CCOW**. (19)

HL7 CCOW *Context Management Standard* (**CMS**), define un medio para la coordinación y la sincronización automática de aplicaciones sanitarias distintas que comparten el mismo escritorio clínico. Las aplicaciones que utilizan el estándar CMS permiten al usuario establecer el contexto clínico para el escritorio, utilizando cualquiera de las aplicaciones habilitadas. Cuando el contexto se ha establecido, todas las aplicaciones habilitadas en el escritorio son automáticamente "sincronizadas" con el mismo **contexto clínico**. (20)

El **contexto clínico** se compone de un conjunto de sujetos. Cada sujeto representa una entidad del mundo real, tal como un paciente en particular, o concepto, tal como un encuentro específico con un paciente. CMS define varios sujetos estándares y permite a los sujetos no estándares (o personalizados) ser definidos de igual modo. (20)

Al compartir contexto, las aplicaciones son capaces de trabajar juntas para seguir las acciones del usuario, ya que interactúan entre ellas, por lo que se dice que están "clínicamente vinculadas". Este comportamiento cooperativo entre las aplicaciones hace que sea mucho más fácil y más seguro para los usuarios introducir y recuperar la información que necesitan para prestar atención a sus pacientes.

1.3 Análisis de las soluciones existentes

Existen diversas aplicaciones que han sido desarrolladas para facilitar el proceso de autenticación en las aplicaciones y así lograr ahorrar tiempo en el acceso a las mismas. A continuación se muestran algunas de las soluciones existentes.

LDAP: es un protocolo de tipo cliente-servidor que trabaja a nivel de aplicación permitiendo el acceso a un servicio de directorio. Este protocolo es utilizado para autenticar entidades y que estas puedan usar el mismo nombre de usuario y contraseña para acceder a diversas aplicaciones de forma unificada.

El empleo de LDAP ofrece diversas ventajas pues es muy rápido en lecturas y escrituras, dispone de un modelo de nombres globales que asegura que todas las entradas sean únicas, funciona sobre TCP[3]/SSL[4], usa un sistema jerárquico para el almacenamiento de la información, además es fácil de instalar, mantener y optimizar. Es considerado una base de datos pero no está diseñada para soportar muchos cambios, sino para realizar lecturas de datos de forma eficiente. Puede ser utilizado en directorios de información, sistemas de autenticación/autorización centralizada y sistemas de correo electrónico. (8)

Kerberos: es un protocolo de autenticación de redes creado por MIT[5] que utiliza criptografía de clave simétrica (misma clave para cifrar y descifrar) y un tercero de confianza llamado Centro de Distribución de Claves (KDC, por sus siglas en inglés) para verificar la identidad del usuario. Este protocolo emplea una arquitectura cliente-servidor (un cliente realiza peticiones y el servidor es quien le da respuesta). Además tiene como objetivos fundamentales centralizar la autenticación de usuarios ya que mantiene una única base de datos para toda la red e impedir que las claves sean enviadas a través de la red y por consiguiente eliminar los ataques de *replay* y *eavesdropping*[6].
En el proceso de autenticación mediante el protocolo Kerberos los usuarios se registran en el servidor Kerberos y reciben un *ticket* que presentan para obtener acceso a las aplicaciones. (21)

SmartSEC: es un sistema desarrollado por Bit4id[7] que tiene como propósito acrecentar el uso de las tarjetas sanitarias como instrumento para la autenticación y el acceso a aplicaciones.

El sistema SmartSEC utiliza como procedimiento para la autenticación un SSO, el cual brinda la posibilidad de que únicamente los usuarios autorizados podrán tener acceso al equipo y posteriormente a todas la aplicaciones que le correspondan, con tan solo introducir su Smart card (tarjeta) en el lector de tarjetas y escribir un pin alfanumérico de ocho dígitos.

---

[3] Protocolo de Control de Transmisión.

[4] Seguridad en la capa de Transporte.

[5] MIT es el instituto tecnológico de *Massachusetts*.

[6] *Eavesdropping* significa escuchar secretamente.

[7] Bit4id proveedor de soluciones tecnológicas para la gestión de la identidad digital.

Uno de los centros que han apostado por esta solución es el complejo hospitalario “Ospedali Riuniti di Bergamo” de Italia. (22)

Sentilliun, Inc: es una industria proveedora de autenticación fuerte, inicio de sesión único, gestión de contexto y soluciones tecnológicas de gestión de la privacidad que permiten a los sistemas de información de salud trabajar juntos, de manera intuitiva y con seguridad. Esta industria es pionera en el aprovechamiento de la norma CCOW, proporcionándoles a los profesionales de la salud el uso eficiente de las aplicaciones sanitarias. La misma cuenta con una suite de productos nombrada Vergence, que están diseñados para satisfacer las necesidades de productividad y privacidad de los proveedores de salud.

Con los productos Vergence a los profesionales de la salud se les proporciona acceso rápido y seguro a la información desde múltiples aplicaciones, aceleración de la integración en el escritorio clínico y el flujo de trabajo. (23)

En la Tabla 1 se muestran los productos de la suite de Vergence y el precio de obtención de la licencia permanente:

Tabla 1 Productos de la suite de Vergence, licencia permanente.

Producto	**Licencia Perpetua**	**Mantenimiento (durante cinco años)**	**Precio Total**
Vergence Sing On Manager	$ 32,845	$ 29,568	$ 62,405
Vergence Context Manager	$ 26,848	$ 24,163	$ 51,011

En la Tabla 2 se muestran los productos de la suite de Vergence y el precio de obtención de la licencia durante cinco años:

Tabla 2 Productos de la suite de Vergence, licencia durante 5 años.

Producto	Suscripción de Licencia (durante cinco años)	Mantenimiento (durante cinco años)	Precio Total
Vergence Sing On Manager	$ 53,261	Incluido	$ 53,261
Vergence Context Manager	$ 43,538	Incluido	$ 43,538

Carefx *Corporation* es una corporación que fue fundada en el 2002 en *Delaware* y actualmente tiene su sede en *Scottsdale*, Arizona, la cual proporciona soluciones de tecnología de información para las organizaciones de salud en América del Norte y Europa. La compañía ofrece una suite denominada Fusionfx la cual cuenta con un grupo de soluciones interoperables que agilizan el flujo de trabajo clínico y de negocios, entre ellos incluye un manejador de contexto. (24)

La suite de productos Vergence que ofrece la corporación Sentilliun, la suite de productos Fusionfx de la corporación Carefx y SmartSEC son soluciones factibles al problema planteado en la investigación, pero no es posible su empleo, por ser elevados los costos por concepto de licencias de software. SmartSEC requiere de un costo adicional al necesitar de una tarjeta electrónica por cada usuario que interactúe con las aplicaciones de la institución.

ccow.contextmanager: es una implementación en Nodejs de algunas interfaces en la especificación CCOW HL7. El estado actual es una aplicación aproximada a un funcional y simple manejador de contexto, pero no es adecuado para su uso en la producción. (25)

A continuación se muestra la Tabla 3 que refleja una comparación de los sistemas analizados anteriormente:

Tabla 3 Comparación entre los sistemas que facilitan el proceso de autenticación de usuarios.

Producto	Autenticación única	Tipo de licencia	Plataforma	Contexto común
SmartSEC	Si	Propietaria	Multiplataforma	No
Suite de productos Vergence y Fusionfx	Si	Propietaria	Multiplataforma	Si
LDAP	No	Libre	Multiplataforma	No
Kerberos	Si	Libre	Multiplataforma	No
ccow.contextmanager	No	Libre	Multiplataforma	Si

1.4 Metodología, Tecnologías, Herramientas y Lenguajes utilizados

Para el desarrollo del componente que facilite el proceso de autenticación, es imprescindible definir cuáles serán las herramientas, tecnologías, lenguajes y metodología de desarrollo que deben ser usadas para obtener un producto de calidad que logre integrarse a los sistemas del Centro de Informática Médica, por tal motivo se hará uso de tecnologías que sean compatibles con la arquitectura de dichos sistemas.

Arquitectura

Servicio *web*. Arquitectura orientada a servicios

Un servicio *web* según W3C[8] es un sistema de *software* diseñado para soportar la interacción máquina a máquina a través de la red y de forma interoperable. Utilizan un conjunto de protocolos y estándares (HTTP, FTP, entre otros) que sirven para intercambiar datos entre aplicaciones. (26)

La arquitectura orientada a servicio (SOA, por sus siglas en inglés) es un concepto de arquitectura de *software* que define la utilización de servicios para dar soporte a los requisitos del negocio. Permite la creación de sistemas altamente escalables que reflejan el negocio de la organización, a su vez brinda una forma estándar de exposición e invocación de servicios lo cual facilita la interacción entre diferentes sistemas propios y terceros. (27)

[8] W3C consorcio internacional que produce recomendaciones para *World Wide Web*.

Basada en componentes

La arquitectura basada en componentes describe una aproximación de ingeniería de *software* al diseño y desarrollo de un sistema, en la cual los componentes son subsistemas que son desarrollados para dar cumplimiento a una funcionalidad especifica del sistema. Esta arquitectura basada en componentes tiene como característica fundamental la reusabilidad de los componentes de *software*, ya que estos deben ser diseñados de tal manera que puedan ser reutilizados en otros sistemas. (28)

1.4.1 Metodología de desarrollo de *software*

Una metodología de desarrollo de *software* se emplea para estructurar, planificar y controlar el proceso de desarrollo de *software*. Es la que define quien debe hacer qué, cuándo y cómo. (29)

Proceso unificado de desarrollo (*Rational Unified Process* (RUP), por sus siglas en inglés)

En la presente investigación se utiliza RUP como metodología de desarrollo ya que la misma provee un enfoque disciplinado en la asignación de tareas y responsabilidades dentro de una organización de desarrollo. Su meta es asegurar la producción de *software* de muy alta calidad que satisfaga las necesidades de los usuarios, dentro de un calendario y presupuesto predecible.

Es un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas de *software* para diferentes áreas de aplicación, tipos de organización y tamaños de proyectos. Utiliza el Lenguaje Unificado de Modelado (*Unified Modeling Language* (UML), por sus siglas en inglés) para preparar todos los esquemas de un sistema de *software*. RUP se caracteriza por ser dirigido por casos de uso, centrado en la arquitectura y por ser iterativo e incremental. (30)

1.4.2 Lenguaje de Modelado

Se selecciona UML como lenguaje de modelado ya que es utilizado para construir, especificar, visualizar y documentar los artefactos de un sistema que involucra una gran cantidad de *software*. UML establece un conjunto de notaciones y diagramas estándar para modelar sistemas orientados a objetos, y describe la semántica esencial de lo que estos diagramas y símbolos significan.

Este lenguaje de modelado permite mayor rigor en las especificaciones, realizar una verificación y validación del modelo desarrollado, automatizar determinados procesos y generar código a partir de los modelos y viceversa. (31)

1.4.3 Herramientas

Visual Paradigm para UML: es una herramienta CASE[9] que utiliza UML como lenguaje de modelado. Soporta el ciclo de vida completo del desarrollo de *software*: análisis y diseño orientados a objetos, construcción, despliegues y pruebas. El *software* de modelado UML ayuda a una rápida construcción de mejores aplicaciones con alta calidad y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. (32)

PostgreSQL: es una herramienta libre de gestión de bases de datos relacionales orientadas a objetos que destaca por su cumplimiento con los estándares SQL, por su robustez, por ser altamente escalable en cuanto al gran número de usuarios que puede alojar y la gran cantidad de información que puede manejar. Cuenta con versiones para una amplia gama de sistemas operativos, entre ellos: Linux, Windows, Mac OSX, Solaris entre otros. (33)

PgAdmin: es una herramienta de código abierto para la administración de bases de datos PostgreSQL y derivados. Incluye interfaz administrativa gráfica, herramienta de consulta SQL, editor de código procedural y agente de planificación SQL. Está diseñado para responder a las necesidades de todos los usuarios, desde escribir consultas SQL simple hasta desarrollar bases de datos complejas. El interfaz gráfico soporta todas las características de PostreSQL y facilita enormemente la administración. PgAdmin está disponible en varios lenguajes y sistemas operativos como Linux, Mac OSX, Solaris y Microsoft Windows. (34)

***JBoss Developer Studio*:** es una herramienta de desarrollo basada en Eclipse. Proporciona un rendimiento superior para todo el ciclo de vida de desarrollo. Incluye un amplio conjunto de funciones de herramientas y soporte para múltiples modelos y marcos de programación, como *Java Enterprise Edition* 6, *RichFaces*, *Java Server Faces* (JSF), *Enterprise JavaBeans* (EJB), *Java Persistence* API (JPA) e *Hibernate*. (35)

Máquina Virtual de Java (*Java Virtual Machine* (JVM), por sus siglas en inglés): es el entorno en el cual se ejecutan los programas java. Se encarga de interpretar y ejecutar las instrucciones expresadas en

[9] CASE es una herramienta de ingeniería de *software* asistida por una computadora.

código binario (*bytecode*) y convertirlo en lenguaje nativo del sistema, donde su principal característica es garantizar la portabilidad debido a que si una JVM ha sido implementada en una determinada plataforma (Linux, Windows, entre otros), cualquier programa java se puede ejecutar en esa plataforma. (36)

WSO2 *Application Server*: es la herramienta usada para el despliegue de componentes, los cuales son servicios *web* desarrollados en *axis2*, *jax-ws* y *spring-ws*; también sirve para desplegar aplicaciones *web* tradicionales y servicios *restful*. Las aplicaciones *web* desplegadas en esta plataforma y que fueran pensadas para ello pueden acceder a diversas funcionalidades que brinda la herramienta. (37)

JBoss *Application Server* es el servidor de aplicaciones de código abierto más ampliamente desarrollado del mercado. Por ser una plataforma certificada J2EE, soporta todas las funcionalidades J2EE 1.4 e incluye servicios adicionales como *clustering*, *catching* y persistencia. Jboss es ideal para aplicaciones Java y aplicaciones basadas en la *web*. También soporta *Enterprise Java Beans* (EJB) 3.0, lo que hace el desarrollo de las aplicaciones mucho más simple. Además, al ser desarrollado con tecnología Java, es multiplataforma. (38)

1.4.4 Lenguaje de programación

El lenguaje de programación que va a emplear en el desarrollo del componente de autenticación es **java**, el cual es un lenguaje orientado a objetos, concurrente y de propósito general que fue diseñado por *Sun Microsystems*[10] específicamente para presentar pocas dependencias de implementación.

Este lenguaje presenta diversas características que lo hacen ser distintivo de otros, como por ejemplo es compilado e interpretado (todo programa en java es compilado y el código que se genera es interpretado por una máquina virtual), posee un gestor de seguridad que permite restringir el acceso a los recursos del sistema, posee mecanismos para garantizar la seguridad durante la ejecución, define procedimientos para tratar errores. Es indiferente de la arquitectura ya que es compatible con diversos entornos, esto hace que su portabilidad sea muy eficiente ya que sus programas son iguales en cualquier plataforma. Java es

[10] *Sun Microsystems* fue una empresa informática que se dedicaba a vender estaciones de trabajo, servidores, componentes informáticos, *software* y servicios informáticos.

considerado de alto rendimiento por ser rápido a la hora de correr los programas y por ahorrar muchas líneas de código cuando se está programando. (39)

1.4.5 *Framework*, librerías y componentes

Librería: es un fichero que contiene un conjunto de funciones escritas en lenguajes de programación para ser utilizadas por un programa, de forma que si una aplicación necesita usar estas funcionalidades puede acceder a la librería sin tener que reescribir el código.

Las librerías que van a ser usadas para el desarrollo del componente en la presente investigación serán:

1. jboss seam-resteasy, resteasy-jaxrs y jaxrs-api, las cuales se usarán para ejecutar servicios REST sobre el *framework* Jboss Seam.

2. gson-2.2.2 es una biblioteca de Java utilizada para convertir objetos Java en su representación JSON. También se puede utilizar para convertir una cadena JSON[11] a un objeto Java equivalente.

3. jboss-seam es la librería que contiene las clases necesarias para el trabajo con el *framework* JBoss Seam.

***Framework*:** es un marco de trabajo orientado a la reutilización en gran escala de componentes de *software* para el desarrollo rápido de aplicaciones. Define un conjunto de conceptos estandarizados, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia para enfrentar o resolver nuevos problemas similares, facilitando de esta manera el desarrollo del *software*. (40)

Hibernate: es un *framework* de código abierto que realiza el mapeo entre el paradigma orientado a objeto de las aplicaciones y el modelo entidad-relación de las bases de datos. Hibernate no solo realiza esta transformación sino que nos proporciona capacidades para la obtención y almacenamiento de información de la base de datos que nos reducen el tiempo de desarrollo.

Está diseñado para ser flexible en cuanto al esquema de tablas utilizado, para poder adaptarse a su uso sobre una base de datos ya existente. Ofrece un lenguaje de consultas llamado HQL (*Hibernate Query*

[11] Denotación de objetos en JavaScript.

Language).También tiene la funcionalidad de crear bases de datos a partir de la información disponible. (40)

JBoss Seam: es un *framework* que integra y unifica los distintos estándares de la plataforma Java EE 5.0, pudiendo trabajar con todos ellos siguiendo el mismo modelo de programación. Ha sido diseñado intentando simplificar al máximo el desarrollo de aplicaciones. El núcleo principal de Seam está formado por las especificaciones *Enterprise JavaBeans* 3(EJB3) y *Java Server Faces* (JSF) que son tecnologías con más soporte que permiten que el desarrollo de las aplicaciones sea muy ágil, ya que reduce el nivel de configuración necesario para la integración. Además aprovecha al máximo las ventajas de cada una de las tecnologías haciendo el proyecto más estable, legible y predecible. (41)

***Enterprise Java Beans* (EJB3):** es un componente de arquitectura del lado del servidor para la plataforma *Java Enterprise Edition* (Java EE). El estándar EJB3 es desarrollado por *Java Community Process* (JCP). Encapsula la lógica del negocio de una aplicación y permite construir aplicaciones de negocio portables, reutilizables y escalables usando el lenguaje de programación Java. EJB3 puede residir en diferentes servidores y puede ser invocado por un cliente remoto. La lógica del negocio reside en los *Enterprise Java Beans* y no del lado del cliente, permitiendo que el desarrollo del lado del cliente esté desacoplado de la lógica del negocio. (42)

Conclusiones Parciales

Tras culminar el estudio del presente capítulo se arribaron a las siguientes conclusiones:

- No es factible la adopción de las soluciones estudiadas en el análisis de tendencias por lo siguiente:
    - Vergence, Fusionfx y SmartSEC son privativas con altos costos por concepto de licencias de software y adquisición de equipamiento.
    - La utilización de un LDAP permite centralizar la autenticación pero no la hace única.
    - Kerberos como única solución no da respuesta al problema planteado en la investigación, se necesita combinado con otros mecanismos, como SSO y HL7 CCOW.
    - *ccow.contextmanager*, no presenta un grado de terminación y estabilidad apropiados para su adopción en ambiente de despliegue.

- Se decide desarrollar un componente de autenticación que cumpla con las características y estándares necesarios, haciendo uso de las herramientas y tecnologías utilizadas en CESIM para garantizar la integración de los sistemas desarrollados en este centro al componente de autenticación.

CAPÍTULO 2. CARACTERÍSTICAS DEL COMPONENTE PARA FACILITAR EL PROCESO DE AUTENTICACIÓN DE USUARIOS EN APLICACIONES INFORMÁTICAS EN INSTITUCIONES DE SALUD

En el presente capítulo, por la ausencia de una definición de los procesos del negocio, se determina desarrollar un Modelo de Dominio, donde se exponen los principales conceptos asociados a la solución que permitirán comprender el contexto en el que se desarrolla. Se muestran los Requisitos Funcionales y sus respectivos Casos de Uso. Además se especifican los Requisitos No Funcionales así como la propuesta del componente de autenticación a desarrollar.

2.1 Modelo de Dominio

El Modelo de Dominio es un modelo conceptual donde se representan los conceptos (objetos) y se identifica las relaciones existentes entre ellos, en el dominio del problema. La elaboración de un Modelo de Dominio puede ser vista como el punto de partida para el diseño de cualquier sistema, porque cuando se lleva a cabo la programación orientada a objeto (POO) el funcionamiento del *software* debe imitar de cierta forma la realidad, por lo que el mapa de conceptos del Modelo de Dominio sería una primera versión del sistema. (43)

2.1.1 Conceptos fundamentales del dominio

Con el objetivo de un mejor entendimiento del diagrama de Modelo de Dominio, se proporciona una breve descripción de los conceptos asociados al problema.

Usuario: es la persona que interactúa con las aplicaciones, a través de una estación de trabajo introduciendo nombre de usuario y contraseña. Este puede ser cualquier persona que tenga permisos de acceso a los sistemas.

Estación de trabajo: es la computadora con la que interactúa el usuario para acceder a las aplicaciones.

HIS, RIS, CLINICAS, PACS, SIGICEM: son sistemas a los que acceden los usuarios, introduciendo sus credenciales en cada uno de ellos.

2.1.2 Diagrama del Modelo de Dominio

Modelo de Dominio

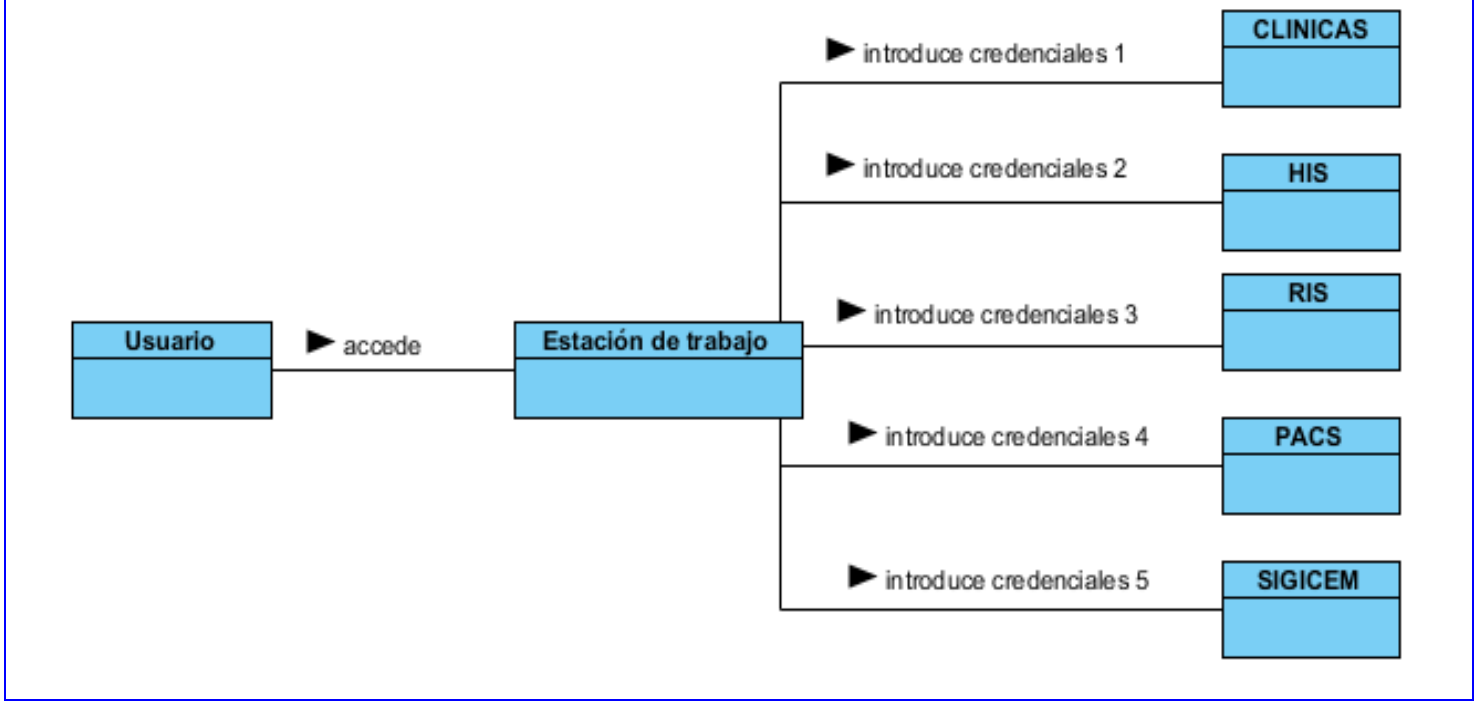

Figura 1 Modelo de Dominio.

El diagrama del Modelo de Dominio explica:

Un determinado usuario, según el rol que desempeñe, puede acceder a una estación de trabajo en el área donde labora. Si este, para el desarrollo de sus funciones necesita hacer uso de varias aplicaciones a la vez, tendrá que autenticarse según la cantidad de sistemas que utilice, introduciendo el par: nombre de usuario y contraseña en cada una de estas. Dicho par puede ser o no similar para cada una de las aplicaciones.

2.2 Especificación de los requisitos del *software*

Los requisitos de *software* son las características que debe tener el *software* instalado en una computadora para soportar y ejecutar una aplicación determinada. Se dividen en funcionales y no funcionales. (29)

Requisitos Funcionales

El requisito funcional es una descripción detallada de la función del sistema, sus entradas y salidas a través de la especificación de las acciones que el sistema debe realizar. (29)

A continuación se muestran los requisitos funcionales definidos para el desarrollo del componente de autenticación:

RF1 Autenticar usuario en Kerberos

- Permite obtener y enviar la información que es usada para autenticar al usuario.

RF2 Registrar bitácora de sucesos

- Permite llevar un registro de sucesos en el proceso de autenticación.

RF3 Unir a un contexto

- Permite hacer una petición de unión al contexto para obtener la información de autenticación almacenada en este.

RF4 Renovar TGT

- Permite renovar un TGT cuando el mismo está a punto de vencerse.

RF5 Cambiar contexto

- Permite realizar un cambio en el contexto para actualizar la información de autenticación que contiene.

RF6 Abandonar contexto

- Permite hacer una petición de abandonar el contexto en el caso de que cualquier aplicación desee abandonarlo y posteriormente se les informa a las demás aplicaciones que comparten el mismo contexto para que todas lo abandonen.

2.2.1 Requisitos No Funcionales

Un requisito no funcional es una cualidad o propiedad que el producto debe tener. Son requerimientos que no se refieren específicamente a las funcionalidades que debe proporcionar el sistema sino a propiedades emergentes como tiempo de respuesta, la seguridad, la capacidad de almacenamiento y fiabilidad del sistema. A continuación se muestran los Requisitos No Funcionales definidos: (29)

RNF 1 *Software*

Contar con un servidor *Network Time Protocol* (NTP, por sus siglas en inglés). Tener sincronizados todos los clientes y el servidor con el NTP de la entidad. Esto se requiere como parte de la implementación del perfil EUA.

RNF 1.1 *Software*

Tener instalado un servidor de Kerberos configurado y totalmente funcional.

RNF 2 Portabilidad

La aplicación podrá ser desplegada sobre sistemas operativos Linux (Ubuntu 14.04 LTS (*Long Term Support*) *Desktop Edition*, Debian 6, OpenSuse 13.2) y Windows. Ambos sistemas operativos con la instalación de máquina virtual de java 6 (JRE 1.6.0_24) o superior.

RNF 3 Seguridad

Denegará el acceso a las personas, que en el proceso de autenticación entren datos incorrectos (personas no autorizadas) y se archivarán todas las trazas de sucesos en el proceso de autenticación. Dichas trazas se registrarán utilizando el formato del registro de eventos de usuario autenticado especificado en el perfil de autenticación ATNA.

RNF 4 Servidor

Cada sistema informático que decida implementar el perfil de autenticación EUA debe reevaluar los requisitos de *hardware* y *software* de sus aplicaciones incluyendo en estos las necesidades que la implementación de ese perfil aporta.

La implementación del manejador de contexto requiere de la instalación de WSO2 AS, para el cual se proponen las siguientes características:

RNF 4.1 Servidor de aplicaciones: memoria de 4 GB como mínimo, JDK versión 1.6 o superior y espacio suficiente en disco para la instalación y despliegue del manejador de contexto y sus dependencias.

2.3 Propuesta del componente para facilitar el proceso de autenticación de usuario

El componente que se propone tiene como fin que el proceso de autenticación de usuario sea centralizado y único. Para lograr este objetivo se implementará la sección básica del perfil EUA perteneciente al dominio de infraestructura IT de IHE. El perfil EUA requiere del uso del servidor de credenciales Kerberos y del estándar HL7 CCOW.

El componente de autenticación está compuesto por dos partes fundamentales:

- Los sistemas desarrollados en el CESIM: actúan como principales actores (Agente de autenticación del cliente y Usuario participante del contexto) en el proceso de autenticación de los usuarios. El sistema que se escoge para ejercer los roles mencionados es el Sistema de Información Hospitalaria.
- Para la autenticación de las aplicaciones se contará con un manejador de contextos el cual tendrá como objetivo almacenar la información de autenticación de los usuarios y estará físicamente ubicado en un servidor de aplicaciones.

2.1.1. Descripción general del componente para facilitar el proceso de autenticación de usuario

Las aplicaciones pertenecientes al CESIM tienen dentro de su flujo de trabajo como primera actividad el proceso de autenticación, este flujo cambia al implementar el perfil EUA, pasando a ser la primera

actividad de cada una de estas aplicaciones el chequeo de la existencia de un contexto con información válida de autenticación de usuario para la estación de trabajo correspondiente.

A continuación se muestran algunos conceptos que se utilizarán en la descripción de los flujos de eventos del componente a desarrollar:

Contexto: es un entorno donde un conjunto de aplicaciones comparten información del usuario.

***Ticket Granting Ticket* (TGT):** credenciales iniciales que verifican si un usuario ha sido autenticado. Es usado para evitar los eventos repetidos de autenticación de usuario y como un *token* para requerir el acceso a un servicio específico.

Servidor Kerberos: servidor encargado de proporcionar la información de autenticación del usuario al agente de autenticación y asignarle un TGT.

Agente de autenticación del cliente: es un componente que permite la autenticación mediante el servidor Kerberos e interactúa con el manejador de contexto en el cual puede compartir la información de autenticación del usuario. La implementación del agente de autenticación del cliente es responsabilidad de la aplicación que desea usar el perfil EUA. Además esta es responsable de la gestión del TGT.

Breve descripción del flujo de eventos

La primera actividad de una aplicación es chequear la existencia de un contexto válido para la estación de trabajo en el manejador del contexto. De encontrarse con la ausencia de este, pasa entonces al proceso de autenticación del usuario. Al usuario introducir sus credenciales (nombre de usuario y contraseña) en la aplicación a la que quiere acceder, esta envía las credenciales al agente de autenticación del cliente, el cual las obtiene y verifica contra el servidor de autenticación Kerberos. De ser correctas estas credenciales, Kerberos le devuelve un TGT el cual será utilizado para la autenticación y posterior acceso a la aplicación. Cuando el agente de autenticación del cliente tiene el TGT que le proporcionó el servidor Kerberos, entonces interactúa con el manejador del contexto, actualizando la información del usuario correspondiente a dicha estación de trabajo. Esta aplicación pasa a ser un usuario participante del contexto.

Después de ocurrir los eventos anteriormente mencionados, cuando el usuario desee acceder a otra aplicación ya no debe introducir nuevamente sus credenciales. Dicha aplicación, al implementar el perfil EUA como primera actividad chequea la existencia de un contexto válido para la estación de trabajo

encontrándose con la existencia de este, pasando entonces al proceso de autorización del usuario, el cual accede directamente a la aplicación con los permisos correspondientes. Esta aplicación pasa a ejercer el rol de usuario participante del contexto. Ver Figura 2.

Las aplicaciones deben registrar los eventos de autenticación satisfactorios, autenticación fallida y cierre de sesión en la bitácora de sucesos de acuerdo a la especificación del perfil de integración ATNA del dominio de la infraestructura IT de IHE. Cada uno de estos eventos posee un código que lo identifica y además tienen un indicador de resultado que puede ser satisfactorio o fallido. En el Anexo 1 se muestra una tabla de cómo se guarda esta información en la bitácora de sucesos.

A continuación se describen los pasos en el flujo del componente de autenticación propuesto:

En el caso de que un usuario acceda a una determinada aplicación y esta sea la primera se seguirán los siguientes pasos:

Paso 1: la aplicación hace una petición de unión a un contexto al manejador de contextos. El manejador de contexto busca si existe algún contexto creado y en caso de que no sea así, crea un contexto vacío y envía los datos del contexto creado a la aplicación.

Paso 2: la aplicación al recibir los datos del contexto vacío muestra el *login*, el usuario introduce sus credenciales, las cuales son enviadas al servidor de autenticación Kerberos.

Paso 3: el servidor de autenticación Kerberos obtiene y verifica si las credenciales que fueron enviadas por la aplicación son correctas.

Paso 4: el servidor de autenticación envía un TGT como confirmación que la autenticación fue válida.

Paso 5: la aplicación registra este evento en la bitácora de sucesos.

Paso 6: la aplicación obtiene el TGT y con el mismo hace un cambio en el contexto que le corresponde en el manejador de contexto.

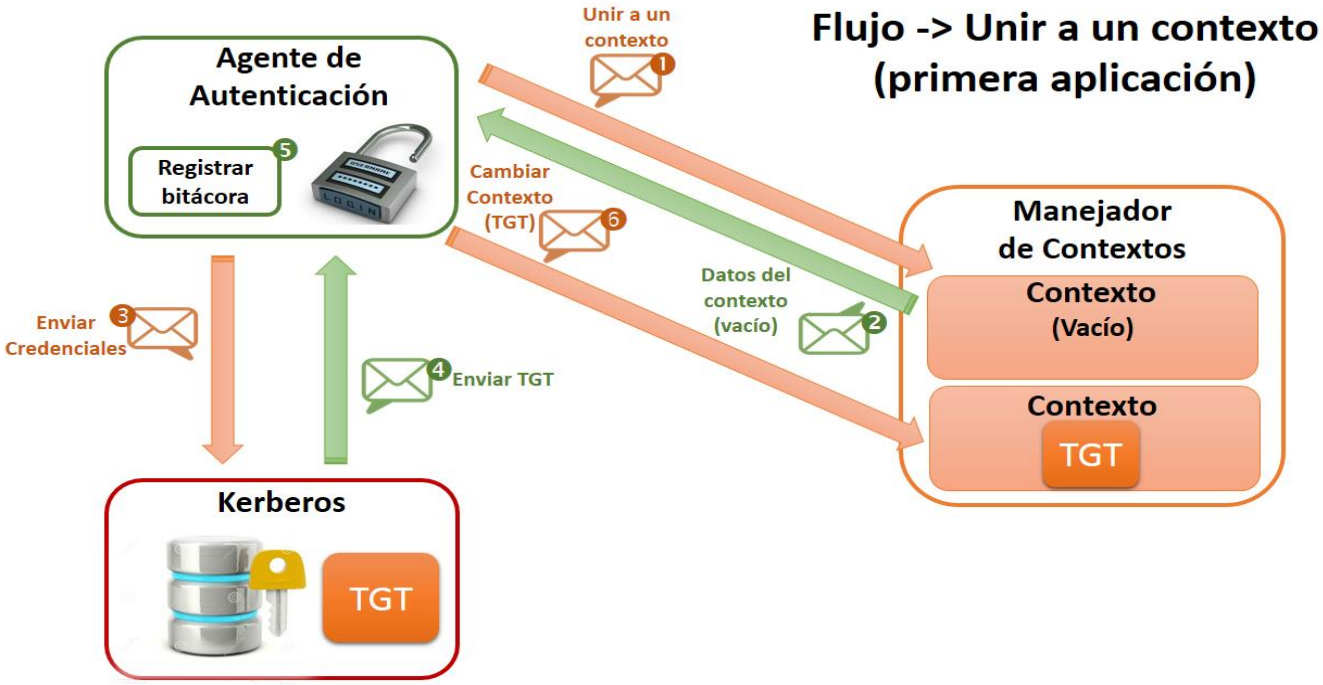

Figura 2 Flujo unir al contexto, primera aplicación.

En el caso de que un usuario desee entrar en una aplicación pero inicialmente ya había accedido a otra aplicación (se había unido a un contexto) se siguen los siguientes pasos:

Paso 1: la aplicación hace una petición de unión al contexto al manejador de contextos.

Paso 2: como ya el contexto se había creado anteriormente con la primera aplicación, el manejador de contextos envía los datos del contexto (TGT) a la aplicación para que esta pueda unirse directamente al contexto sin necesidad de ingresar las credenciales.

Paso 3: cuando la aplicación se logró unir al contexto, la misma registra el evento satisfactorio en la bitácora de sucesos.

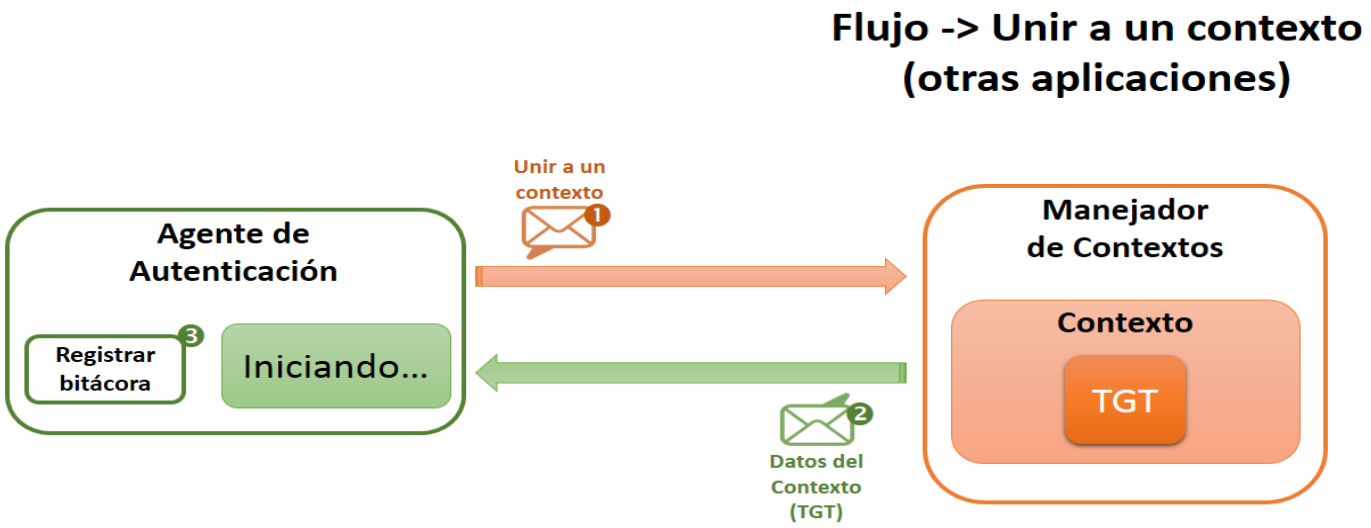

Figura 3 Flujo unir al contexto, otras aplicaciones.

La aplicación efectúa un cambio de contexto en el caso de una unión o renovación del TGT, pues en ambos casos la información contenida en el TGT cambia. El TGT hace falta cambiarlo cuando el tiempo de vida de la sesión exceda el tiempo de vida del TGT, es decir mientras la sesión esté activa y el tiempo de renovación del TGT sea válido.

Paso 1: cuando la aplicación se autentica satisfactoriamente o renueva el TGT, se efectúa un cambio de contexto, actualizando la información del TGT en el manejador de contextos.

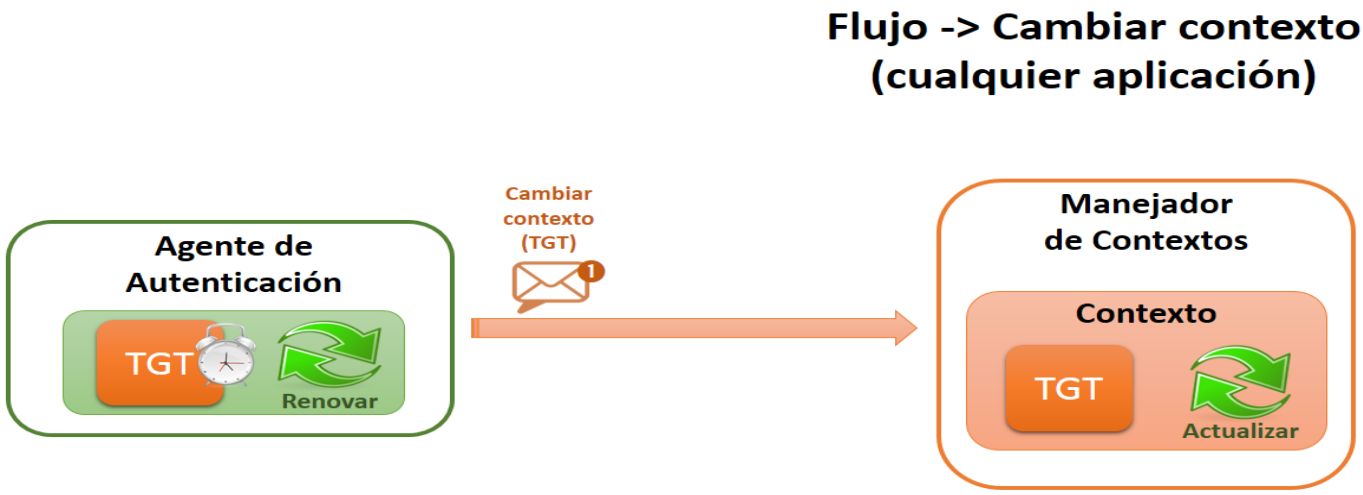

Figura 4 Flujo cambiar contexto.

En el caso de que determinada aplicación desee abandonar el contexto, los pasos serían los siguientes:

Paso 1: la aplicación hace una petición de abandono de contexto al manejador de contextos, el cual la recibe y envía una notificación a todas las demás aplicaciones que estén unidas al mismo contexto para que también abandonen el contexto. Posteriormente se elimina el contexto y el TGT.

Paso 2: una vez que la aplicación haya abandonado el contexto satisfactoriamente, la aplicación registra el evento en la bitácora de sucesos.

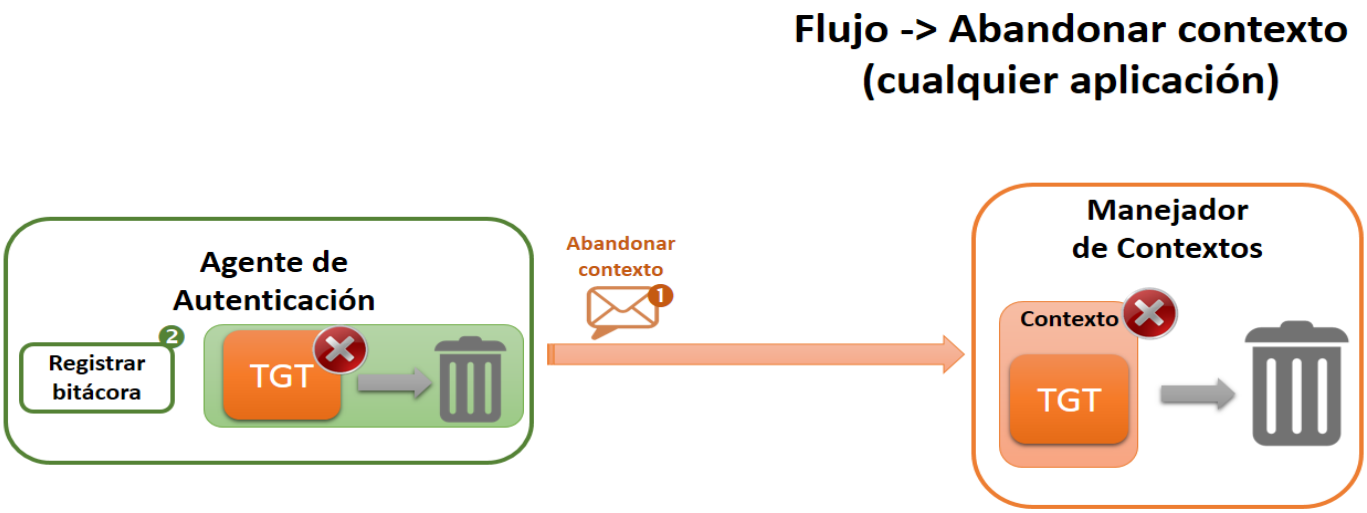

Figura 5 Flujo abandonar contexto.

2.4 Modelo de Casos de Uso del Sistema

Un modelo de caso de uso del sistema describe los requerimientos funcionales del sistema en forma de casos de uso. Este modelo está compuesto por casos de uso, actores y las relaciones entre ellos. (44)

Los actores representan un tipo de usuario del sistema, el cual puede ser una persona, otro sistema informático o unidades organizativas que interactúan con el sistema. Los casos de uso representan el comportamiento del sistema en respuesta a un estímulo de un actor.

En la Tabla 4 se muestran los actores y sus objetivos.

Tabla 4 Definición de los actores del sistema.

Actor	Objetivos
Agente de autenticación del cliente.	Es el encargado de gestionar el contexto (unirse, cambiar y abandonar un contexto). Además de obtener la información de autenticación de usuario para luego enviarla al servidor de autenticación de Kerberos y registrar la bitácora de sucesos.
Timer	Es el encargado de renovar el TGT.

2.4.1 Diagrama de Casos de Uso del Sistema

Diagrama de Casos de Uso del Sistema

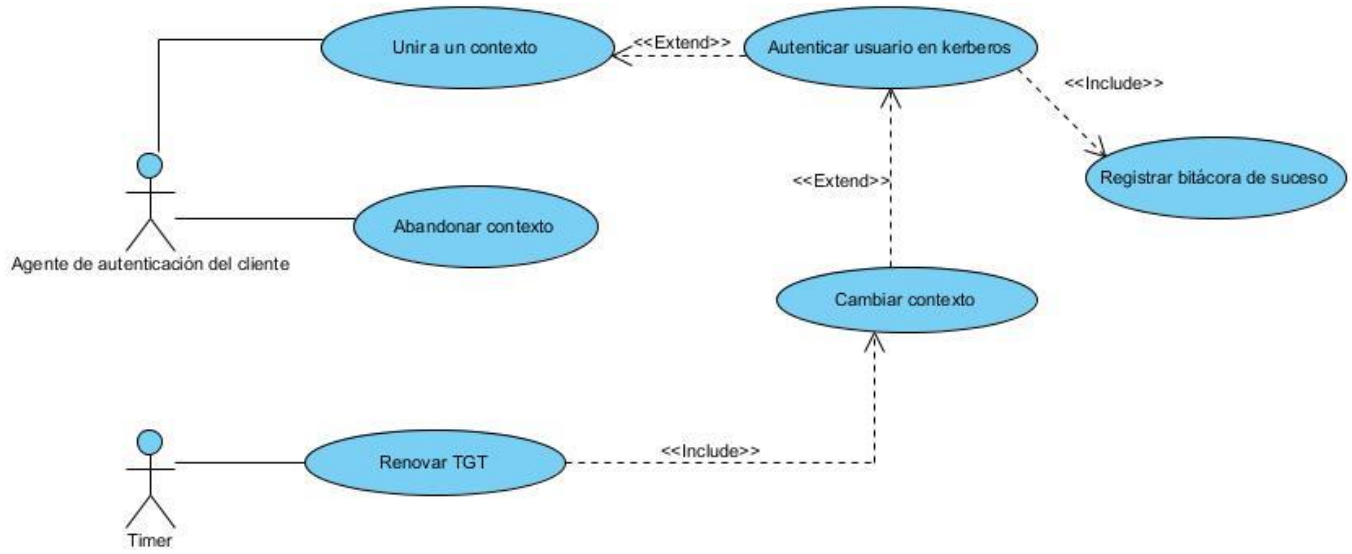

Figura 6 Diagrama de Casos de Uso del Sistema.

2.4.2 Descripción textual de los Casos de Uso

A continuación se describen los casos de uso correspondientes al Diagrama de Casos de Uso del Sistema mostrado anteriormente. La descripción de los restantes casos de uso se encuentra en el artefacto:

"HIS_Componente_Autenticacion_Especificacion_de_casos_de_uso", el cual pertenece a los documentos del proyecto Sistema de Información Hospitalaria.

CU 1. Unir a un contexto

En la Tabla 5 se muestra la descripción del caso de uso unir al contexto.

Tabla 5 Descripción del caso de uso Unir a un contexto.

Objetivo	Permite hacer una petición de unión al contexto.	
Actores	Agente de autenticación del cliente.	
Resumen	El caso de uso inicia cuando el actor hace una petición de unión al contexto y el sistema verifica si existe el mismo, el caso de uso termina.	
Complejidad	Media	
Prioridad	Media	
Precondiciones	No existen	
Postcondiciones	No existen	
Flujo de eventos		
Flujo básico Unir al contexto		
	Actor	**Sistema**

1.	El caso de uso inicia cuando envía una petición de unión al contexto.	
2.		Verifica que existe un contexto creado. En caso de que exista un contexto. Ver **Evento 1** "Existe contexto". En caso de que no exista un contexto. Ver **Evento 2** "No existe contexto".
3.		Termina el caso de uso.
Flujos alternos		
Evento 1 "Existe contexto"		
	Actor	**Sistema**
1.		Devuelve los datos contenidos en el contexto, los cuales serán utilizados para la autenticación.
2.		Termina el caso de uso.
Flujos alternos		
Evento 2 "No existe contexto"		
	Actor	**Sistema**
		Envía un mensaje requiriendo la autenticación del usuario Kerberos.
		Termina el caso de uso.

Relaciones	**CU incluidos**	No existen
	CU extendidos	Autenticar usuario Kerberos. Ver CU Autenticar usuario Kerberos.
Requisitos no funcionales		
Asuntos pendientes	No existen	

CU 2. Autenticar usuario en Kerberos

En la Tabla 6 se muestra la descripción del caso de uso autenticar usuario en Kerberos.

Tabla 6 Descripción del caso de uso Autenticar usuario en Kerberos.

Objetivo	Permite obtener y enviar la información que es usada para autenticar al usuario.
Actores	Agente de autenticación del cliente.
Resumen	El caso de uso inicia cuando el actor obtiene la información que ha sido usada para autenticar a un usuario y la envía al servidor de autenticación Kerberos para verificarla, el caso de uso termina.
Complejidad	Media
Prioridad	Media
Precondiciones	El usuario debe ingresar los datos (nombre de usuario y contraseña) para acceder a la aplicación.
Postcondiciones	No existen

Flujo de eventos		
Flujo básico Autenticar usuario Kerberos		
	Actor	**Sistema**
1.	El caso de uso inicia cuando obtiene la información de autenticación del usuario y la envía como una petición de autenticación al servidor de credenciales Kerberos.	
2.		Verifica que la información de autenticación contenida en la petición sea correcta comprobando que el nombre usuario y contraseña sean válidos. En caso de que no sean válidos los datos de autenticación. Ver **Evento 1** "Datos incorrectos". En caso de que estos datos sean correctos envía un TGT.
3.	Registra el evento en la bitácora de sucesos.	
4.		Termina el caso de uso.
Flujos alternos		
Evento 1"Datos incorrectos"		
	Actor	**Sistema**
1.		Envía un mensaje de error al actor Agente de autenticación del cliente.

2.	Muestra un mensaje de error “Usuario o contraseña inválidos”.	
3.		Termina el caso de uso.

Relaciones	**CU incluidos**	Registrar bitácora de sucesos. Ver CU Registrar bitácora de sucesos.
	CU extendidos	No existen
Requisitos no funcionales		
Asuntos pendientes	No existen.	

CU 3. Registrar bitácora de sucesos

En la Tabla 7 se muestra la descripción del caso de uso registrar bitácora de sucesos.

Tabla 7 Descripción del caso de uso Registrar bitácora de sucesos.

Objetivo	Permite llevar un registro de sucesos en el proceso de autenticación.
Actores	Agente de autenticación del cliente.
Resumen	El caso de uso inicia cuando el actor tiene que llevar un registro detallado de los sucesos en el proceso de autenticación, el caso de uso termina.
Complejidad	Media
Prioridad	Media

Precondiciones	No existen		
Postcondiciones	No existen		
Flujo de eventos			
Flujo básico Registrar bitácora de sucesos			
	Actor		**Sistema**
1.	El caso de uso inicia cuando obtiene toda la información referente al proceso de autenticación.		
2.	Registra toda la información referente al proceso de autenticación.		
3.			Termina el caso de uso.
Relaciones	**CU incluidos**	No existen	
	CU extendidos	No existen	
Requisitos no funcionales			
Asuntos pendientes	No existen		

CU 4. Renovar TGT

En la Tabla 8 se muestra la descripción del caso de uso renovar TGT.

Tabla 8 Descripción del caso de uso Renovar TGT.

Objetivo	Permite renovar el TGT.	
Actores	Timer	
Resumen	El caso de uso inicia cuando el TGT que está usando por el agente de autenticación del cliente está a punto de vencerse, el caso de uso termina.	
Complejidad	Media	
Prioridad	Media	
Precondiciones	El TGT está a punto de expirar y aún el agente de autenticación del cliente permanece en el contexto.	
Postcondiciones	No existen	
Flujo de eventos		
Flujo básico Renovar TGT		
	Actor	**Sistema**
1.	El caso de uso inicia cuando envía una petición de renovación de TGT.	
		Recibe el TGT, verifica que sea válido y envía el nuevo TGT. En caso de que no sea válido el TGT. Ver **Evento 1** "TGT no válido".
2.	Realiza un cambio de contexto. Ver caso de	

	uso: **Cambiar contexto.**	
3.		Termina el caso de uso.
Flujos alternos		
Evento 1 "TGT no válido"		
	Actor	**Sistema**
1.		Envía mensaje de error "No es posible renovar el TGT".
2.	Informa al agente de autenticación del cliente que el TGT no se pudo renovar.	
3.	Hace una petición de abandono del contexto. Ver caso de uso: **Abandonar contexto**.	
4.		Termina el caso de uso.

Relaciones	**CU incluidos**	Cambiar el contexto. Ver CU Cambiar contexto.
	CU extendidos	
Requisitos no funcionales		
Asuntos pendientes	No existen	

Conclusiones Parciales

Una vez realizado el análisis del proceso de autenticación en los sistemas desarrollados en el CESIM se arribaron a las siguientes conclusiones:

- Se identificó como deficiencia más significativa del proceso de autenticación, la introducción repetida de las credenciales por cada usuario, en cada uno de los sistemas que se utilizan en la misma estación de trabajo.
- Se define el perfil EUA como mecanismo estándar para centralizar y hacer única la autenticación en los sistemas que coexisten en una misma estación de trabajo.
- Se identificó con el estudio del perfil EUA, que los flujos de eventos enmarcados en el contexto de la investigación son:
    - Unión al contexto de la primera aplicación ejecutada en la estación de trabajo.
    - Unión al contexto del resto de las aplicaciones que se ejecuten en la estación de trabajo.
    - Cambio de contexto.
    - Abandono de contexto.

CAPÍTULO 3. ANÁLISIS Y DISEÑO DEL COMPONENTE PARA FACILITAR EL PROCESO DE AUTENTICACIÓN DE USUARIOS EN APLICACIONES INFORMÁTICAS EN INSTITUCIONES DE SALUD

En el presente capítulo se presentan los elementos correspondientes al diseño del sistema propuesto. Se explica la arquitectura que se va a emplear en el desarrollo del componente de autenticación. Se realizan los diagramas de clases del diseño, además se brinda una descripción detallada de las clases identificadas para su posterior implementación.

3.1 Descripción de la arquitectura, fundamentación

La arquitectura de *software* es también conocida como arquitectura lógica, la cual es el diseño de más alto nivel de la estructura de un sistema ya que define los principales componentes del mismo y sus relaciones. Una arquitectura de *software* consiste en un conjunto de patrones y abstracciones coherentes que proporcionan el marco de referencia necesario para guiar la construcción del sistema, permitiendo a los programadores, analistas y diseñadores trabajar bajo una línea común que les posibilite la compatibilidad necesaria para alcanzar la meta deseada. El componente propuesto en la investigación presenta una arquitectura híbrida compuesta por la arquitectura orientada a servicio y basada en componentes.

Arquitectura Orientada a Servicio

La arquitectura orientada a servicios es un paradigma de arquitectura para diseñar y desarrollar sistemas distribuidos. Consiste en exponer servicios para su posterior reutilización. Su objetivo es facilitar la interacción entre distintos sistemas desarrollados con diferentes tecnologías en un mismo entorno empresarial y permitir la interoperabilidad de los mismos. La arquitectura SOA establece un marco de diseño para la integración de aplicaciones independientes de manera que desde la red se puede acceder a sus funcionalidades, las cuales se ofrecen como servicio. La forma más habitual de implementarla es mediante servicios *web*, una tecnología basada en estándares e independiente de la plataforma.

Las soluciones SOA han sido creadas para satisfacer los objetivos de negocio las cuales incluyen facilidad y flexibilidad de integración con sistemas legados, alineación directa a los procesos de negocio reduciendo costos de implementación, innovación de servicios de clientes y una adaptación ágil ante cambios.

La arquitectura SOA presenta diversas características entre las que se encuentran: (45)

- La interacción con los servicios es débilmente acoplada, lo que trae consigo beneficios para los sistemas ya que tendrán mayor capacidad para resistir a los cambios que ocurran en la estructura e implementación de cada uno de los servicios.
- Clientes y otros servicios pueden acceder a servicios locales que se ejecutan en el mismo nivel.
- Clientes y otros servicios acceden a servicios remotos sobre una red que los conecta.
- Estos servicios pueden usar un rango de protocolos y formatos de datos para comunicar información.

En el desarrollo de la presente investigación se utilizará la arquitectura SOA ya que permite el empleo de servicios *web*. Específicamente en esta investigación de acuerdo al estándar HL7 CCOW se debe emplear el protocolo HTTP o HTTPS para la transferencia de mensajes entre el manejador de contexto y las aplicaciones que interactúan con él, por lo que se hará uso de servicio *web* basado en REST.

Arquitectura basada en componentes

La arquitectura basada en componentes se centra en la descomposición del *software* en componentes funcionales o lógicos que expongan interfaces de comunicación bien definidas. Tiene como principal objetivo propiciar el desarrollo de sistemas complejos a través del ensamblado de componentes previamente elaborados que puedan ser rehusados en disímiles aplicaciones.

Esta arquitectura consiste en la elaboración de uno o varios componentes desarrollados específicamente para mejorar el funcionamiento de las aplicaciones o dar solución a un determinado problema que la misma presente, con la finalidad de incrementar la flexibilidad y facilidad de mantenimiento de los sistemas.

La arquitectura basada en componentes presenta diversos beneficios entre los que se destacan: (28)

- Facilidad de instalación: cuando una nueva versión de un sistema esté disponible, se podrá reemplazar la versión existente sin impactar los componentes o el sistema como un todo.
- Costos reducidos: el uso de componentes de terceros permite distribuir el costo del desarrollo y mantenimiento.

- Reusable: el uso de componentes reutilizables significa que ellos pueden ser usados para distribuir el desarrollo y mantenimiento entre múltiples aplicaciones.
- Facilidad de desarrollo: los componentes se desarrollan para responder a una determinada funcionalidad del sistema, sin necesidad de afectar otros componentes o sistemas.

Aplicación de estas arquitecturas en la presente investigación

La arquitectura orientada a servicios se aplica en el desarrollo del *Context Participant* que se encuentra en las aplicaciones que participan en el perfil EUA y en el manejador de contexto. Esta arquitectura permite la interoperabilidad entre las aplicaciones, independientemente de la plataforma de *hardware*, sistema operativo o lenguaje de programación en el que el servicio se implementa. Esta se combina con una arquitectura basada en componentes, pues las funcionalidades del agente de autenticación del cliente son expuestas como un conjunto de métodos que pueden ser invocados por las diferentes aplicaciones, lo que permite mayor reusabilidad, así como mantenimiento y garantiza la interoperabilidad de las aplicaciones.

3.2 Estrategias de integración del componente de autenticación

El componente de autenticación de usuario no responde a un área en específico en las instituciones hospitalarias, el mismo puede ser ubicado en las aplicaciones donde se quiera agilizar el proceso de autenticación de usuario en una estación de trabajo. Estas aplicaciones deben cumplir con las especificaciones del perfil EUA y HL7 CCOW. Para lograr una mejor comprensión de lo que se quiere desarrollar, el Sistema de Información Hospitalaria se integrará al componente de autenticación, el cual debe implementar dos componentes para poder unirse al contexto. Debe implementar un *Client Authentication Agent* y un *Context Participant*.

Primeramente debe implementar el *Client Authentication Agent*, el cual debe contener los métodos siguientes:

- Método GetUserAuthentication, el cual recibe como parámetro usuario y contraseña y retorna las credenciales de tipo Kerberos.
- Método GetServiceTicket, el cual se encarga de proveer a la aplicación el TGT válido así como su renovación.

Para que el *Context Participant* se comunique con el manejador de contexto, este debe ser implementado como un servicio REST, el cual debe contar con los siguientes métodos:

- Método Unirse al contexto (joinCommonContext), el cual recibe como parámetro un identificador de la instancia de la aplicación y una referencia al servicio ContextParticipant de la misma. Este método le devuelve a la aplicación su identificador en el contexto.
- Método Abandonar el contexto (leaveCommonContext), el cual recibe como parámetro el identificador de la aplicación en el contexto y le devuelve un mensaje de respuesta a la aplicación informando su abandono satisfactorio.
- Método Cambiar contexto (publishChangesDecision), el cual recibe como parámetro el identificador de la aplicación en el contexto, una lista con los datos que desea cambiar y el identificador del contexto que desea cambiar y retorna un mensaje informando si el cambio fue satisfactorio o no.

Para que el manejador de contexto pueda interactuar con todos sus participantes debe contar con las referencias correctas a cada uno de estos. Las referencias son las url que apuntan al servicio REST implementado por cada una de las aplicaciones participantes en un contexto.

3.3 Modelo de Diseño

Para la elaboración de un sistema es imprescindible realizar un correcto modelado del diseño, el cual será muy útil pues sirve de base para cuando se realice la implementación. Un Modelo de Diseño no es más que una representación de la solución que se quiere construir, el cual proporcionará la estructura y la forma del sistema. Se realiza basándose en los requisitos funcionales, junto a otras restricciones relacionadas con el entorno de implementación y el impacto que tendrán en el sistema.

Para la confección del Modelo del Diseño se crearon un conjunto de clases, las cuales están agrupadas en paquetes lo cual permitirá dividir el sistema en fragmentos manejables para su posterior implementación. A continuación se muestra el diagrama de paquetes utilizado:

Diagrama de Paquetes

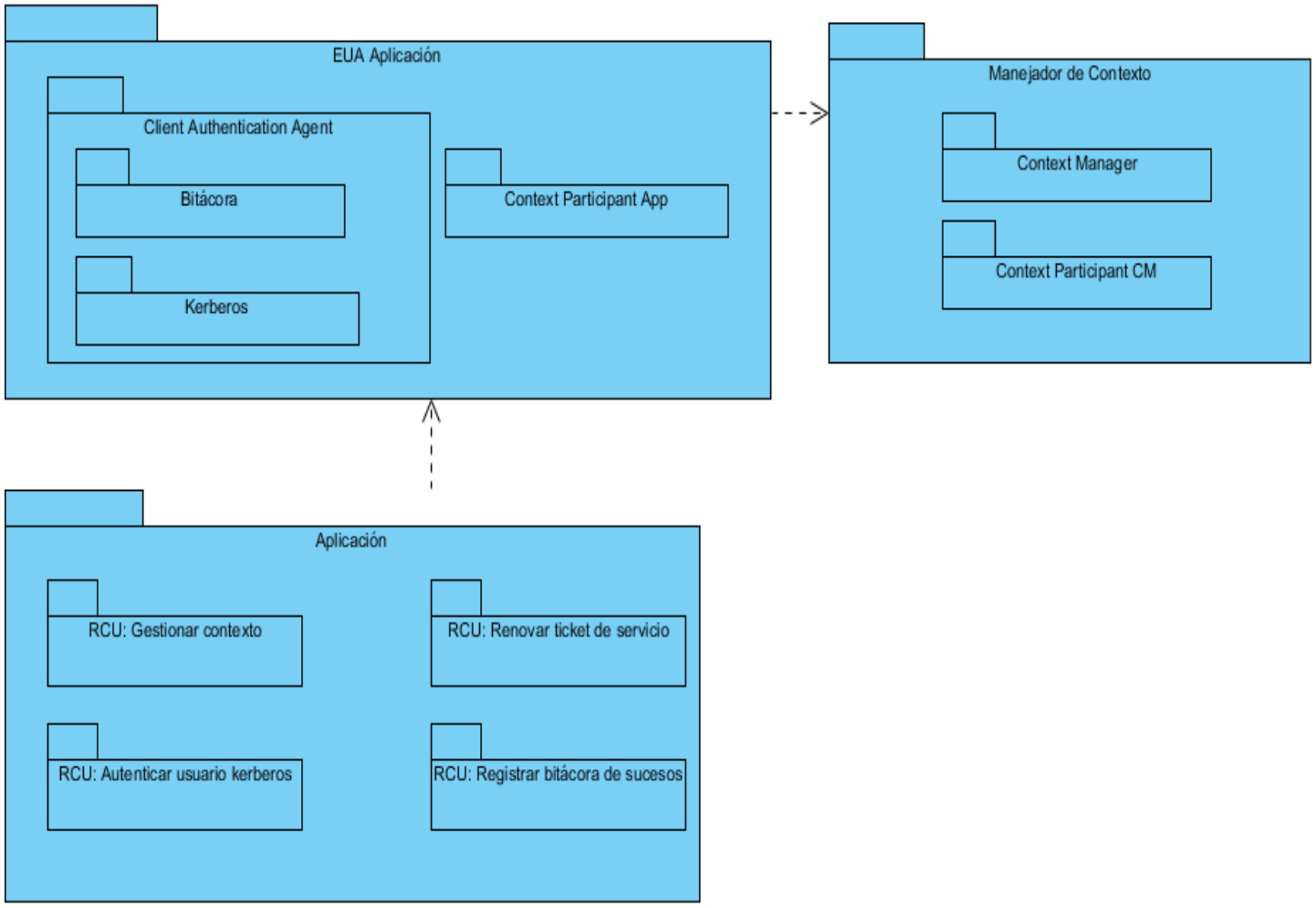

Figura 7 Diagrama de Paquetes.

El paquete EUA Aplicación contiene los siguientes paquetes:

- El paquete *Client Authentication Agent*, contiene los paquetes llamados Bitácora y Kerberos:
  - En el paquete Bitácora se encuentran las clases para el registro de bitácora.
  - En el paquete Kerberos se encuentran las clases de conexión con el servidor de credenciales Kerberos.

- El paquete *Context Participant App*
    - Contiene las clases correspondientes al *Context Participant* de la aplicación.

El paquete EUA Aplicación interactúa con el Manejador de Contexto, el cual contiene los siguientes paquetes:

- El paquete *Context Manager* contiene las clases ContextData, ContextManager, ContextSession, ManagerService, ContextParticipant, SKey que son utilizadas para gestionar un contexto.
- El paquete *Context Participant CM* contiene la clase *Context Participant* del manejador de contexto.

El paquete Aplicación hace uso del paquete EUA Aplicación el cual contiene las realizaciones de los Casos de Uso gestionar contexto, renovar *ticket*, autenticar usuario Kerberos y registrar bitácora de sucesos que debe implementar cada aplicación para usar un contexto.

3.3.1 Diagrama de clases de diseño

Un diagrama de clases del diseño representa, describe gráficamente las especificaciones de las clases e interfaces en una aplicación. Las clases de diseño muestran las definiciones de las clases de *software*. Estos contienen normalmente información de clases, asociaciones, atributos, información sobre los tipos de atributos, navegabilidad, dependencia, interfaces con sus operaciones y constantes. (44)

Los diagramas de clases el diseño, donde se modelan las clases, sus atributos, operaciones y las relaciones que existen entre ellas (clase) se puede consultar en el Anexo 2.

3.3.2 Descripción de las clases de diseño

Paquete Manejador de Contexto

- **Clase ContextManager**

Propósito: es una clase controladora personalizada que permite a un participante[12] unirse o abandonar una sesión de contexto común[13] y llevar a cabo las transacciones de cambio de contexto. Para cumplir con su propósito, es su responsabilidad realizar los siguientes métodos que se describen a continuación:

joinCommonContext(): este método permite a una aplicación unirse a una sesión de contexto común. La aplicación debe proporcionar una referencia a su interfaz ContextParticipant en el valor de entrada contextParticipant.

leaveCommonContext(): permite a una aplicación que es un participante en una sesión de contexto común salir de la misma.

starContextChange(): permite que una aplicación indique que quiere empezar a cambiar el contexto común.

endContextChanges(): permite que la aplicación que originó una transacción de cambio de contexto indique que ha completado sus cambios en el contexto común.

publishChangesDecision(): permite a la aplicación que provocó una transacción de cambio de contexto, informar a las otras aplicaciones en una sesión de contexto acerca de si los cambios se deben aplicar o han sido cancelados.

getContextData(): este método devuelve los todos los datos existentes en el contexto.

geContextDataValuest(): devuelve los datos existentes en el contexto en dependencia de los nombres pasados por parámetros.

deleteDataItems(): elimina los datos existentes en el contexto en dependencia de los nombres pasados por parámetros.

[12] Es la aplicación que interactúa con el manejador de contexto.

[13]Sesión de contexto común es un espacio donde varias aplicaciones comparten un contexto.

Clase ContextManager

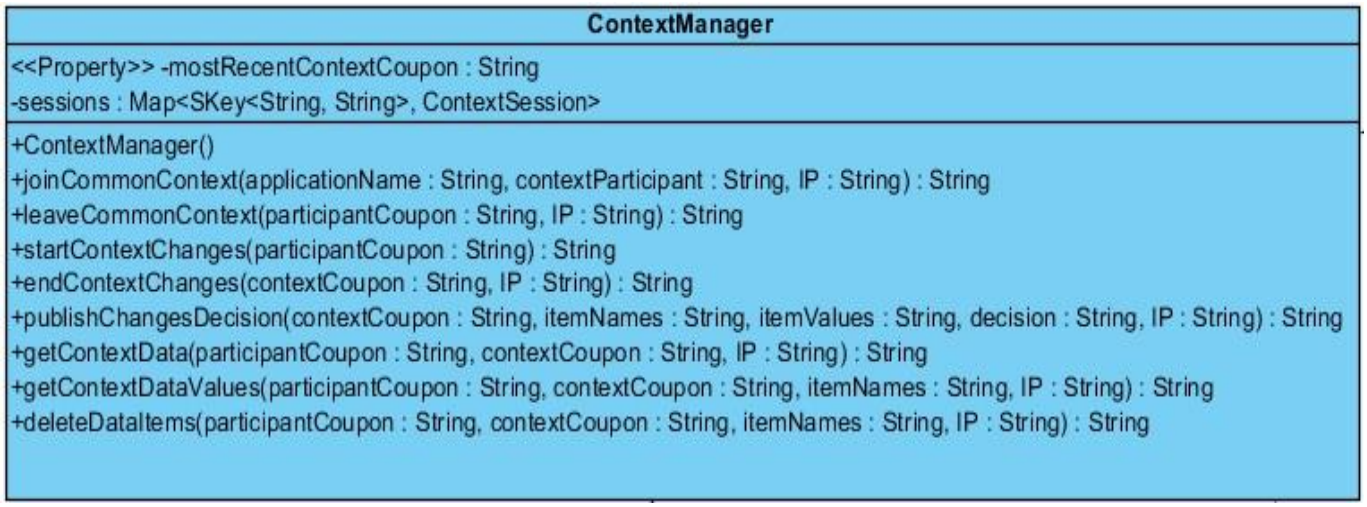

Figura 8 ContextManager.

Clase ContextSession

Propósito: es la clase encargada de representar una sesión de contexto común (es un espacio donde varias aplicaciones comparten un contexto). Para cumplir con su propósito debe realizar los siguientes métodos:

addParticipant(): adiciona un participante a la sesión.

deleteParticipant(): elimina un participante de la sesión.

getParticipantInSession(): obtiene un participante que cumpla con los parámetros que sean requeridos.

setSessionData(): establece los datos de la sesión.

getSessionData(): devuelve los datos de la sesión.

Clase ContextSession

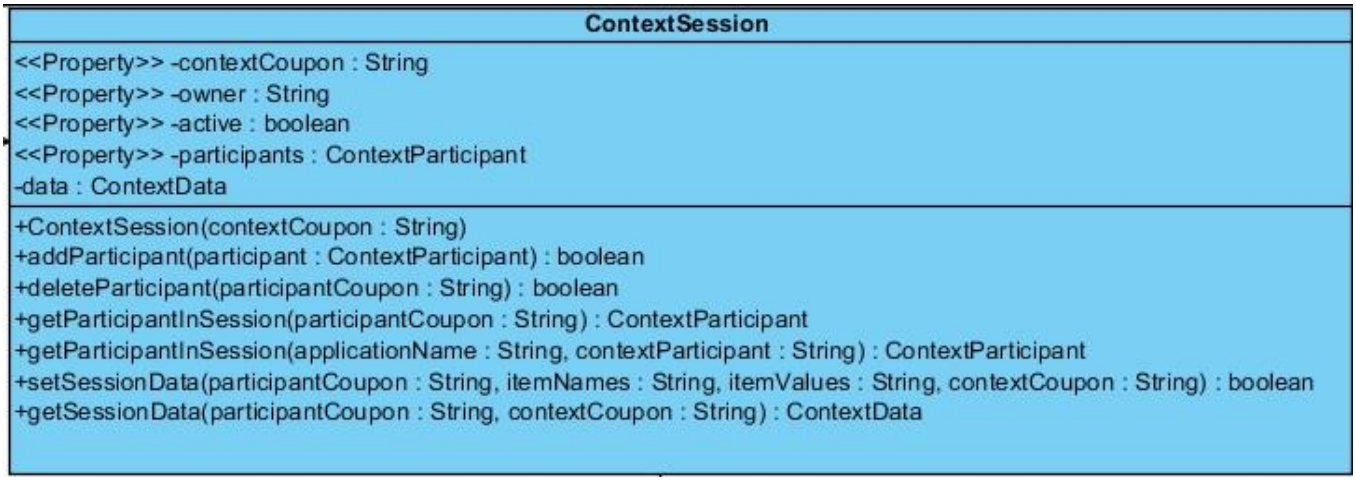

Figura 9 ContextSession.

Paquete Manejador de Contexto

- **Clase ContextData**

Propósito: clase encargada de permitir a un participante del contexto obtener y establecer datos del contexto común. Para cumplir su propósito debe realizar los siguientes métodos:

setItemValues(): permite a una aplicación en una sesión de contexto común establecer el valor de uno o más elementos comunes de contexto.

getItemNames(): permite a un participante en una sesión de contexto común obtener los nombres de los elementos.

getItemValues(): permite a un participante en una sesión de contexto común obtener el valor de uno o más elementos.

deleteItems(): permite a un participante en una sesión de contexto común eliminar el valor de uno o más elementos.

Clase ContextData

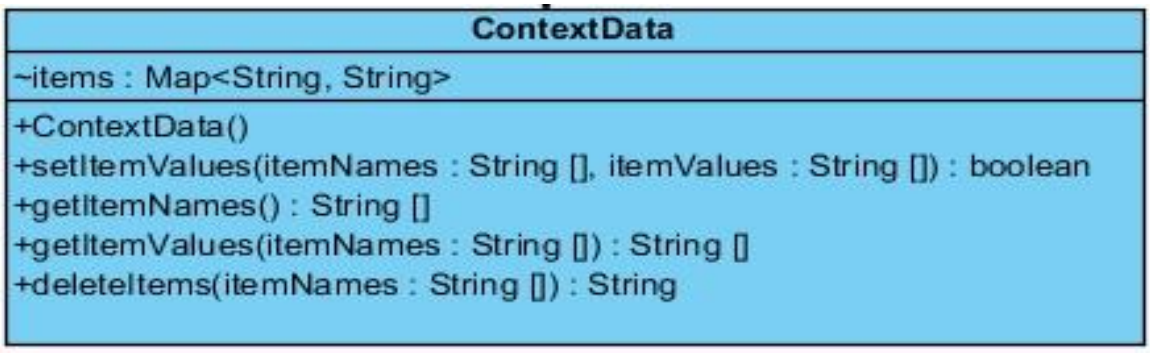

Figura 10 ContextData.

Paquete Manejador de Contexto

- **Clase ContextParticipant**

Propósito: clase encargada de notificar los cambios en un contexto a un participante de mismo, con el objetivo de mantener actualizado a cada uno de sus participantes. Para cumplir con su propósito debe realizar los siguientes métodos:

contextChangesPedding(): informa a un participante en una sesión de contexto común que los cambios en los datos están pendientes.

contextChangeAssepted(): informa a un participante en una sesión de contexto común que el resultado del cambio de contexto más reciente, fue aceptar los cambios y que los datos de contexto común se han establecido.

contextChangeCancelled(): informa a un participante en una sesión de contexto común que una transacción de cambio de contexto se ha cancelado.

commonContextTerminated(): informa a un participante en una sesión de contexto común que una transacción de abandono de contexto ha sido invocada.

getCotentextParticipant(): devuelve una referencia del participante correspondiente.

get(): este método es el encargado de establecer una conexión con el participante correspondiente a través del protocolo HTTP.

Clase ContextParticipant

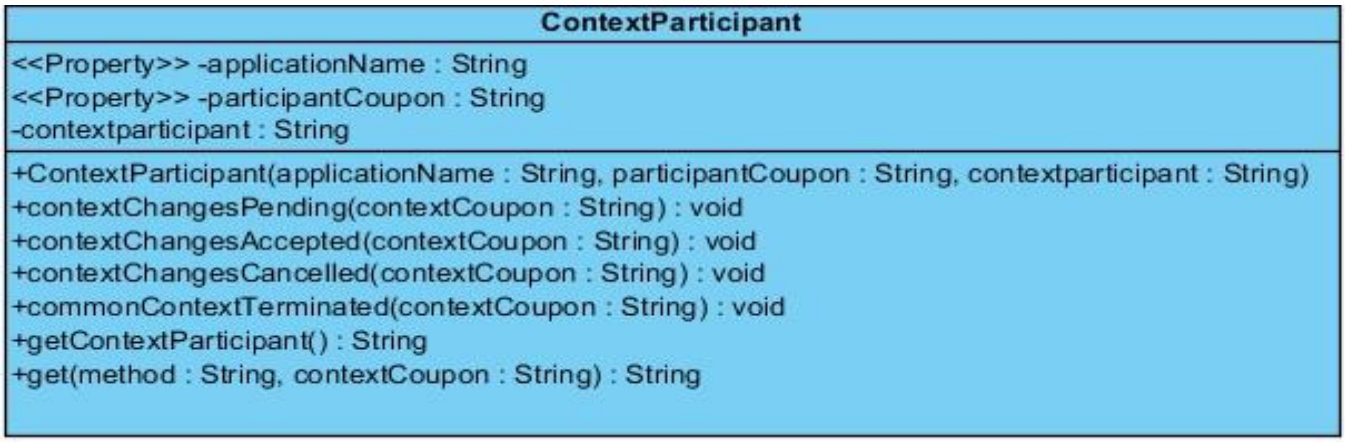

Figura 11 ContextParticipant.

Paquete EUA Aplicación

- **Clase ClientAuthenticationAgent**

Propósito: es una clase controladora encargada de obtener a partir de las credenciales brindadas por la aplicación el TGT del servidor de credenciales Kerberos que son usadas en el proceso de autenticación. Para cumplir con su propósito debe realizar los siguientes métodos:

getInstance(): devuelve el valor de la instancia de la aplicación.

getUserAuthentication(): permite autenticar un usuario con la credenciales proporcionadas por la aplicación en el servidor de credenciales Kerberos.

registerSessionContext(): registra los datos del contexto en la aplicación.

joinContext(): este método permite a la aplicación unirse a una sesión de contexto común en el manejador de contexto.

leaveContext(): permite a la aplicación terminar la sesión de contexto común en el manejador de contexto.

updateInfoContexts(): sincroniza los datos almacenados en el contexto con los datos de la aplicación.

renewCredentials(): renueva las credenciales.

sessionActionRegistre(): registra cuando el proceso de autenticación sea fallido o satisfactorio.

sessionTerminate(): termina una sesión, elimina los datos del contexto contenidos en ella y registra en la bitácora la salida de la aplicación.

Clase ClientAuthenticationAgent

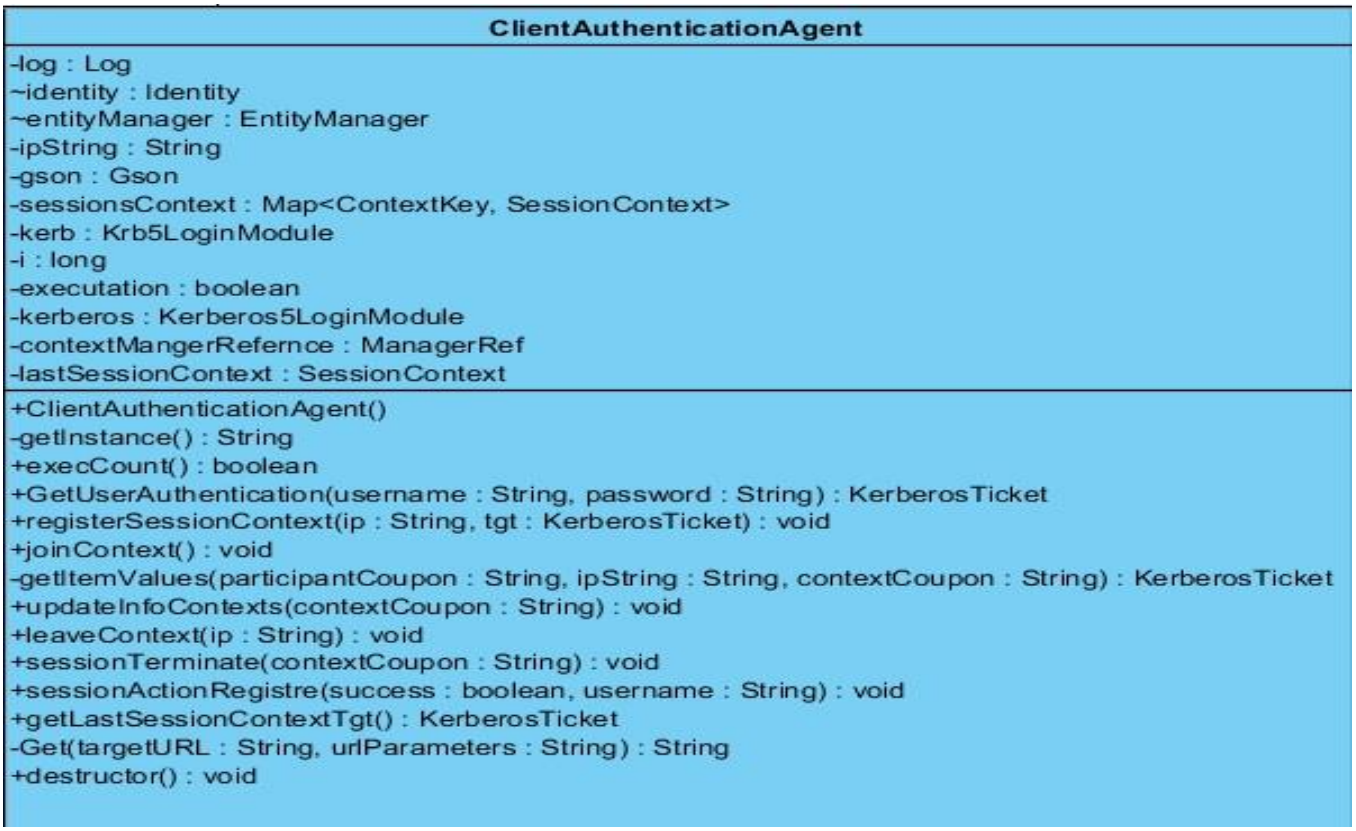

Figura 12 ClientAuthenticationAgent.

Paquete Bitácora

- **Clase BitacoraEUA**

Propósito: es una clase controladora personalizada que permite registrar las trazas de los eventos de autenticación del usuario.

registerLoginSuccess(): registra los eventos de autenticación satisfactorios.

registerLoginFailure(): registra los eventos de autenticación fallidos.

registerLogout(): registra los eventos de cierre de sesión.

register(): método auxiliar que se encarga de recoger los datos y almacenarlos en la base de datos.

Clase Bitácora EUA

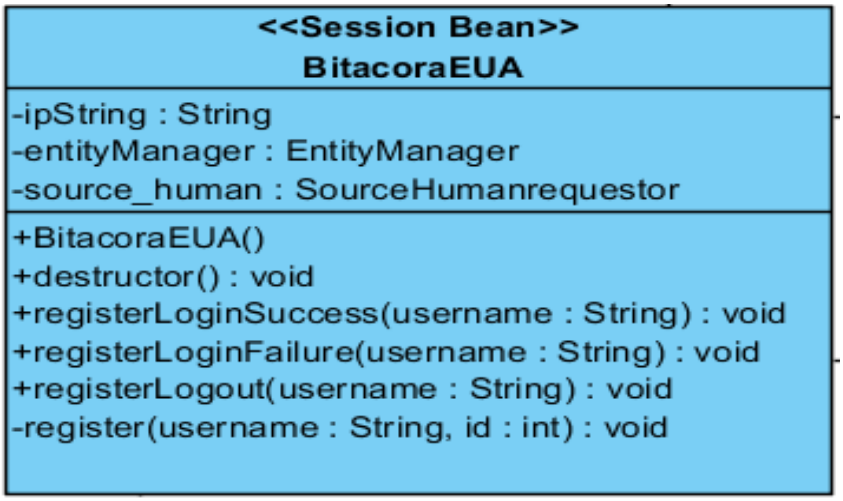

Figura 13 Bitácora EUA.

Conclusiones Parciales

Con la realización de este capítulo se llegaron a las siguientes conclusiones:

- Se determinó que el estilo arquitectónico adecuado para el desarrollo del componente de autenticación es una Arquitectura Basada en Componentes.
- La realización del Diagrama de Clases del Diseño facilitó la comprensión del funcionamiento del componente de autenticación.

CAPÍTULO 4. IMPLEMENTACIÓN DEL COMPONENTE PARA FACILITAR EL PROCESO DE AUTENTICACIÓN DE USUARIOS EN APLICACIONES INFORMÁTICAS EN INSTITUCIONES DE SALUD

En el presente capítulo a partir del diseño elaborado anteriormente, se describe el flujo de trabajo de la implementación. Se muestra el Diagrama de Despliegue y de Componentes los cuales forman parte del modelo de implementación. Además se presentan las estrategias de codificación y la forma en que se tratarán los errores.

4.1 Modelo de implementación

El modelo de implementación está conformado por un conjunto de componentes y subsistemas que constituyen la composición física de la implementación del sistema. Entre sus componentes podemos encontrar datos, archivos, ejecutables, código fuente y los directorios. Fundamentalmente, se describe la relación que existe desde los paquetes y las clases del modelo de diseño a subsistemas y componentes físicos (46). Existen dos tipos de diagramas que se elaboran en el modelo de implementación, llamados Diagrama de Componente y Diagrama de Despliegue, los cuales se muestran a continuación.

4.1.1 Diagrama de Componentes

Este diagrama modela la estructura del *software* y la dependencia entre componentes. Un componente es un grupo de clases que trabajan estrechamente que pueden corresponder a código fuente, binario o ejecutable.

Un Diagrama de Componentes muestra la organización y las dependencias entre un conjunto de componentes, normalmente contienen componentes, interfaces y relaciones entre ellos. Cubren la vista de implementación estática de un sistema. En él se sitúan librerías, tablas, archivos, ejecutables y documentos que formen parte del sistema. (47)

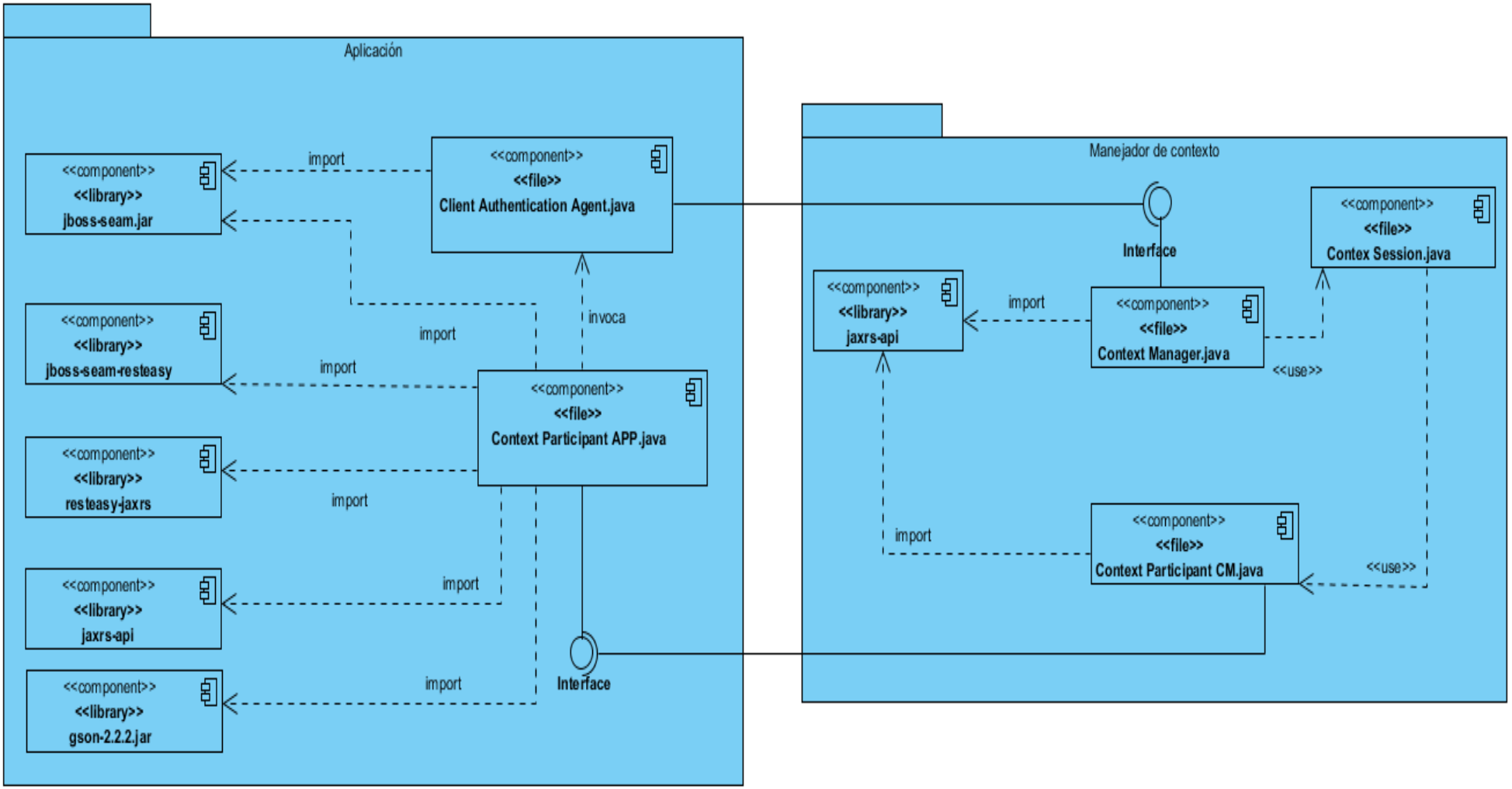

Figura 14 Diagrama de Componentes.

4.1.2 Diagrama de Despliegue

Un Diagrama de Despliegue muestra las relaciones físicas entre los componentes de *hardware* y *software* en el sistema final, es decir, la configuración de los nodos (elementos) de procesamiento en tiempo de ejecución y los componentes *software* (procesos y objetos que se ejecutan en ellos). Su propósito es brindar la vista estática del ambiente de la aplicación. (48)

El Diagrama de Despliegue de la solución propuesta se encuentra estructurado de la siguiente manera: el servidor de aplicaciones JBoss se comunica con el servidor de aplicaciones del componente a través del protocolo HTTP y con el servidor de base de datos a través del protocolo TCP/IP.

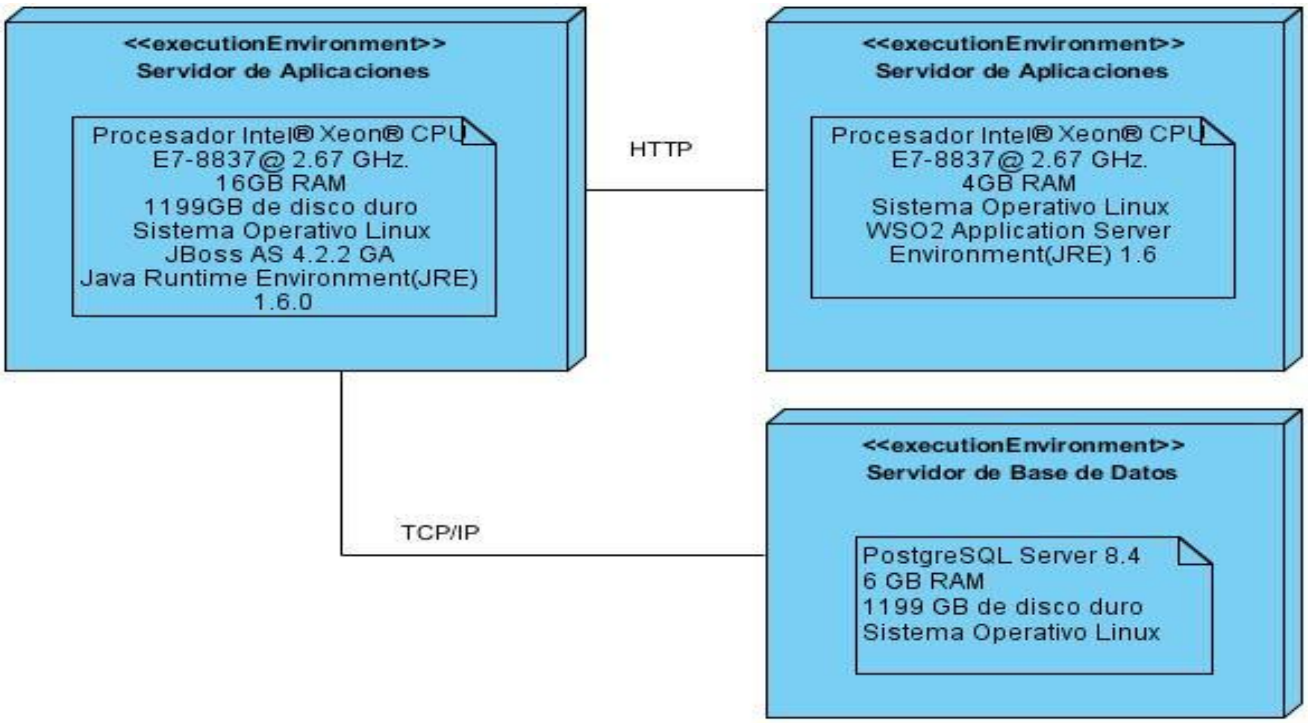

Figura 15 Diagrama de Despliegue.

4.2 Tratamiento de errores

En la programación de un sistema se producen errores más o menos graves, los cuales interrumpen el flujo normal de las sentencias, impidiendo el funcionamiento correcto del mismo, por tal motivo hay que

gestionar y tratar correctamente estos problemas (excepción), para crear un sistema que sea tolerante a fallas y robusto (resistente a errores).

En el componente propuesto se lleva a cabo un tratamiento para los errores a través del control en todos los fragmentos del código donde pueda suceder algún evento inesperado. Esta tarea se realiza mediante una serie de clases las cuales lanzan mensajes sugerentes e informativos acerca de cuál es el error cometido, donde sucedió y como se puede solucionar, en el caso de que el mismo tenga solución. Estas clases se especifican en el documento HL7 *Context Management "CCOW Standard: Technology and Subject Independent Component Architecture*, *Version* 1.6, *February* 2011".

4.3 Estrategias de codificación. Estándares y estilos a utilizar

La forma de escribir el código en la elaboración de un *software* es propia y diferente en cada programador, por tal motivo se definen estándares de codificación donde se especifican un conjunto de reglas para la escritura del código con el fin de garantizar un estilo uniforme en toda la implementación del sistema, haciendo que el mismo sea entendible por otros y posteriormente pueda ser mantenido. Además su uso es de suma importancia en la elaboración de *software*, pues permite elevar la calidad del mismo.

4.3.1 Elementos de los estándares de codificación

Notación Camello (CamelCase)

Para los métodos y variables: la notación que se empleará será específicamente lowerCamelcase, la cual se aplica a palabras compuestas, donde la primera se escribe en minúscula y las restantes palabras internas se escriben en mayúsculas. Si la palabra es simple se escribirá en minúscula.

Ejemplo de notación lowerCamelcase:

clientAuthenticationAgent.

Para denotación de las clases: se emplearán la notación UpperCamelcase, la primea letra de cada palabra en mayúscula.

Ejemplo de notación UpperCamelCase:

public class ClientAuthenticationAgent {…...}

4.3.2 Empleo de márgenes en el código

Para los márgenes se recomienda dejar dos espacios en blanco por bloque de código. Los inicios (“{”) de cada método estarán al final de la declaración de los mismos y los cierre (“}”) de ámbito estarán alineados debajo de la declaración a la que pertenecen. Se dejará un espacio en blanco desde la instrucción anterior para el inicio y fin del bloque, lo mismo sucede para el caso de las instrucciones *while, if, else, for, do while*.

Ejemplo de inicio y cierre de cada método:

```
public int getUserAuthentication() {

}
```

4.3.3 Cometarios

Los cometarios serán ubicados al inicio de cada clase y al final de cada bloque de instrucción. Se recomienda comentar el inicio y fin de cada clase o función especificando el objetivo de la misma así como los parámetros que usa.

Primer ejemplo de comentario:

```
/* Ejemplo de comentario de inicio de clase */

public class ClientAuthenticationAgent {……}
```

Segundo ejemplo de comentario:

```
/* metodo

*@param valor

*/

public void getUserAuthentication (int valor) {……}
```

4.3.4 Declaraciones

Las declaraciones de los métodos y las variables locales en una clase se realizan al principio del mismo y no después de utilizarse la variable, porque traería como consecuencia un error. Además se recomienda una declaración por línea, pues facilita los comentarios.

4.3.5 Líneas y espacios en blanco

Se recomienda emplear espacios en blanco entre los operadores lógicos y aritméticos para obtener mayor legibilidad en el código. Además no se debe dejar espacios en blanco después del corchete abierto y antes del cerrado, ni antes del paréntesis abierto y cerrado, ni después del punto y coma.

4.4 Seguridad

En la elaboración del componente de autenticación en la presente investigación, la seguridad se ve reflejada, pues se hace uso del protocolo Kerberos que elimina la forma insegura de enviar las credenciales del usuario por la red, empleando una clave simétrica para realizar este proceso. Este aspecto se refleja mediante el empleo del TGT, como la confirmación de que un determinado usuario se autenticó satisfactoriamente. El TGT pasado un determinado tiempo (cinco minutos o el tiempo que se desee establecer) se renueva y si es interceptado por un intruso, no le sirve como medio de autenticación al no contener datos claros del usuario.

Cada contexto tiene un identificador único que se genera utilizando un UUID, este se asocia con el ip de la estación de trabajo donde se hace la petición garantizando así que solo las aplicaciones que se ejecutan en esta puedan acceder a la información contenida en el contexto. Cada aplicación que interactúa con un contexto debe proporcionar un nombre de aplicación y la referencia a su *Context Participant*, una vez que haya proporcionado estos datos se le asigna un UUID para identificarla dentro del contexto.

4.5 Validación del componente de autenticación

La validación en una aplicación, es el proceso de revisión para verificar que el sistema de *software* desarrollado cumple con las especificaciones y que logra el objetivo con el que fue creado. Es una parte del proceso de pruebas de *software* de un proyecto. Se trata de una evaluación del sistema o de

componentes, que se efectúa en el transcurso o al final del proceso del desarrollo para determinar si cumple con los requisitos especificados. (49)

Para comprobar las funcionalidades, se modificaron las clases *Authenticator* y *SessionDestroy* pertenecientes al Sistema de Información Hospitalaria. La modificación de clase *Authenticator* permite verificar la información contenida en el manejador de contexto y la autenticación de los usuarios mediante el servidor de credenciales Kerberos. La modificación de clase *SessionDestroy* permite destruir la información de autenticación contenida en el contexto y en la aplicación.

Para el caso de las funcionalidades de cambiar el contexto, renovar TGT y unir al contexto, se verifican observando en la consola del servidor, los mensajes correspondientes a estos eventos. Ejemplo de estos mensajes se muestran a continuación:

```
<Registro de Union al contexto>
WSO2 AS:
...
JoinCommonContext -> Client ip: 10.8.1.97
Sessions: 0
ParticipantCoupon is: 0f7cbe67fb9e4903914a3044f8aee602
Sessions: 1
ParticipantCoupon: 0f7cbe67fb9e4903914a3044f8aee602
GetContextDataValues -> Client ip: 10.8.1.97
```

Figura 16 Unir al contexto en el servidor de aplicaciones WSO2 AS.

```
JBOSS AS:
01:58:17,868 INFO  [ClientAuthenticationAgent] UrlQuery: applicationName=HIS%231&contextParticipant=http%3A%2F%2F10.8.1.97%3A5901%2Fgehos%2Fresource%2F
                                                                                                                    rest&remoteAddr=10.8.1.97
01:58:17,908 INFO  [ClientAuthenticationAgent] response code: 200
01:58:17,924 INFO  [ClientAuthenticationAgent] UrlQuery: participantCoupon=0f7cbe67fb9e4903914a3044f8aee602
01:58:17,929 INFO  [ClientAuthenticationAgent] response code: 200
01:58:18,040 INFO  [ClientAuthenticationAgent] UrlQuery: participantCoupon=0f7cbe67fb9e4903914a3044f8aee602&contextCoupon=e3d2c035dbac4532a0cd8d6607f60f
                                                                                                                      8f&itemNames=krbtgt&remoteAddr=10.8.1.97
01:58:18,047 INFO  [ClientAuthenticationAgent] response code: 200
01:58:18,047 INFO  [STDOUT] Items are:
```

Figura 17 Unir al contexto en el servidor de aplicaciones JBOSS AS.

Estos dos pequeños fragmentos fueron extraídos de la consola del servidor de aplicaciones WSO2 AS y JBOSS AS cuando se ejecutaron las funcionalidades, quiere decir que se estaba haciendo una petición de unión al contexto desde el ip 10.8.1.97, que no había ningún contexto creado anteriormente y que se crea

uno, donde se le asigna un cupón de participante con el siguiente valor 0f7cbe67fb9e4903914a3044f8aee602.

Un cupón de participante es un identificador que se le asigna a cada una de las aplicaciones que participan en un contexto. Es una cadena generada aleatoriamente.

```
<Registro de Cambio y renovacion de la informacion del contexto>
WSO2 AS:
...
EndContextChanges is OK
PublishChangesDecision -> Client ip: 10.8.1.97
...
JBOSS AS:
...
02:00:28,118 INFO  [STDOUT] UrlQuery: contextCoupon=e3d2c035dbac4532a0cd8d6607f60f8f&remoteAddr=10.8.1.97
02:00:28,287 INFO  [STDOUT] response code: 200
02:00:28,435 INFO  [STDOUT] The method is ContextChangesPending
02:00:28,813 INFO  [STDOUT] UrlQuery: contextCoupon=e3d2c035dbac4532a0cd8d6607f60f8f&itemNames=krbtgt&itemValues=%7B%22asn1Encoding%22%3A%5B97%2C-126%2C1%2C
                                                                                    94%2C48%2C-126%2C1%2C90%2C-96 ... decision=accept&remoteAddr=10.8.1.97
02:00:28,825 INFO  [STDOUT] response code: 200
02:00:28,825 INFO  [STDOUT] ContextChanges: OK
```

Figura 18 Cambio y renovación de la información del contexto.

Estos pequeños fragmentos fueron extraídos de la consola del servidor de aplicaciones WSO2 AS y JBOSS AS cuando se ejecutaron las funcionalidades cambio y renovación de la información del contexto, muestra el nuevo valor del TGT y los cambios efectuados satisfactoriamente.

Para el caso de abandonar el contexto, se verifica en la consola del servidor JBoss *Application Server* 4.2 y en la consola del WSO2 *Application Server* 5.2.1, los mensajes correspondientes a este evento.

```
<Registro de Abandono del contexto>
WSO2 AS:
...
LeaveCommonContext is OK
...
JBOSS AS:
...
02:05:59,625 INFO  [ClientAuthenticationAgent] UrlQuery: participantCoupon=0f7cbe67fb9e4903914a3044f8aee602&remoteAddr=10.8.1.97
02:05:59,639 INFO  [ClientAuthenticationAgent] response code: 200
02:05:59,765 INFO  [STDOUT] The method is CommonContextTerminated
```

Figura 19 Abandono del contexto.

Conclusiones Parciales

A partir de los resultados alcanzados en este capítulo se arribaron a las siguientes conclusiones:

- Se implementó el componente de autenticación centralizada y única basado en las especificaciones del perfil EUA.
- La representación del Diagrama de Componentes permitió identificar los principales elementos del componente de autenticación y la interacción entre ellos.
- Se integró el Sistema de Información Hospitalaria del Centro de Informática Médica al Componente de Autenticación.

CONCLUSIONES

Una vez finalizado el proceso de investigación se arriba a la siguiente conclusión:

1. Con la inserción del Componente de Autenticación en un ambiente de despliegue que involucre más de uno de los sistemas desarrollados en el Centro de Informática Médica, se reduce la cantidad de interacciones de los especialistas en el proceso de autenticación.

RECOMENDACIONES

Una vez concluido el trabajo de diploma se recomienda:

- Desarrollar la variante coordinada del perfil EUA del dominio de la infraestructura IT de IHE, para garantizar que no existan pérdidas de información entre las aplicaciones que participan en este perfil.

REFERENCIAS BIBLIOGRÁFICAS

1. *La informática médica en Cuba.* Señor García, Raúl Felipe . 2, La Habana : s.n., 2000, Vol. 6. 1810-4363.

2. Bordils Rovira, Francisco y Chavarría Díaz, Miguel . *Almacenamiento y transmisión de imágenes. PACS.*

3. Open Health. Tecnologías para la salud. [En línea] [Citado el: 20 de Noviembre de 2014.] http://openhealth.com.co/es/ris-sistema-de-informacion-radiologica.

4. López, Antonio. *Manual de salud electrónica para directivos de servicio y sistemas de salud.* 2012.

5. Fernández Puerto, Francisco y Gatica Lara, Florina. *Sistema de Información Hospitalaria.* México : Universidad Nacional Autónoma de México, 2003.

6. Health Level Seven INTERNATIONAL. [En línea] 2007. [Citado el: 25 de Noviembre de 2014.] http://www.hl7.org.

7. *Understanding and using DICOM, the data interchange standard for biomedical imaging.* BIDGOOD, Willian Dean, y otros. s.l. : Journal of the American Medical Informatics Association, 1997, Vol. 4.

8. Howes , Timothy, Smith , Mark y Good, Gordon. *Understanding and deploying LDAP directory services. .* s.l. : Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA , 2003. 0672323168.

9. *Security in computing.* 2006. 978-0-13-239077-4.

10. He, Wenli. Single Sign On. *networks.* 2000, Vol. 33.

11. Baryolo Gómez, Oiner , Rivero Pino, Noel Jesús y López Méndez, Daniel. Sistema de gestión integral de seguridad Acaxia. *Serie Científica.* 7, 2011, Vol. 4.

12. James, Shakir. Web single sign-on systems. *Computer Science Department, WUSTL.* 2007.

13. Cabarcos Arias, Patricia . Dynamic Infrastructure for Federated Identity Management in Open Environments. 2013.

14. Rehman, Rafeeq. *Get ready for OpenID.* s.l. : OpenID Book, 2008. 0972403124.

15. OASIS Advancing open standards for the information society. [En línea] [Citado el: 20 de Noviembre de 2014.] https://www.oasis-open.org.

16. ASTM Internacional. [En línea] [Citado el: 10 de Enero de 2015.] http://www.astm.org.

17. IHE España. Integrating the Healthcare Enterprise . [En línea] [Citado el: 10 de Enero de 2015.] http://www.ihe-e.org.

18. *IHE IT Infrastructure (ITI). p. 111.* 2011.

19. *IHE IT Infrastructure (ITI). p. 33.* 2011.

20. *HL7 Context Management "CCOW" Standard.* 2011.

21. Kohl , John y Clifford, Neuman. *The Kerberos network authentication service (v5).* s.l. : RFC 1510, september, 1993.

22. Bit4id. [En línea] 2013. [Citado el: 23 de Diciembre de 2014.] http://www.bit4id.com/es/index.php?option=com_content&view=article&id=316:windows-8&catid=84:articulos-noticias-anuncios&Itemid=541..

23. Sentillion. [En línea] Sentillion, Inc. [Citado el: 2 de Febrero de 2015.] www.gsaadvantage.gov/ref_text/GS35F0736M/0FP02G.1TCLVF_GS-35F-0736M_SENTILLION2004.HTM.

24. Healthcare IT News. [En línea] [Citado el: 2 de Enero de 2015.] http://www.healthcareitnews.com/directory/carefx.

25. Schlueter, Isaac , Voss, Laurie y Boothby, Rod. NPM Node Package Manager . [En línea] 2014. [Citado el: 4 de Febrero de 2015.] www.npmjs.com.

26. Española, W3C Oficina. Guía Breve de Servicios Web. [En línea] [Citado el: 8 de Febrero de 2015.] http://www.w3c.es/Divulgacion/GuiasBreves/ServiciosWeb.

27. Newcomer, Eric y Lomow, Greg. *Understanding SOA with Web Services (Independent Technology Guides) .* s.l. : Addison-Wesley Professional, 2004. 0321180860 .

28. Blog de Juan Peláez en Geeks.ms. [En línea] 2009. [Citado el: 23 de Febrero de 2015.] http://geeks.ms/blogs/jkpelaez/archive/2009/04/18/arquitectura-basada-en-componentes.aspx.

29. Pressman, Roger . *Ingeniería del Software: Un enfoque práctico.* 1997.

30. Jacobson, Ivar, Booch, Grady y Rumbaugh, James. *El proceso unificado de desarrollo de software. p.4.* s.l. : Addison Wesley Reading, 2000.

31. Orallo Hernández, Enrique . El Lenguaje Unificado de Modelado (UML). Valencia : Departamento de Informática de Sistemas y Computadores. Universidad Politécnica de Valencia, España, 2009.

32. Free Download Manager. [En línea] [Citado el: 23 de Febrero de 2015.] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%5Bcuenta_de_Plataforma_de_Java_14715_p/.

33. Acevedo López, Luis Felipe. Tutorial de PostgreSQL.

34. pgAdmin. PostgreSQL Tools. [En línea] [Citado el: 20 de Febrero de 2015.] http://www.pgadmin.org/.

35. JBossDeveloper. [En línea] [Citado el: 20 de Febrero de 2015.] http://www.jboss.org/products/devstudio/overview/.

36. Gálvez Rojas, Sergio y Ortega Díaz, Lucas. *J2ME (Java 2Micro Edition).* España : Universidad de Málaga. 84-688-4704-6.

37. WSO2. [En línea] [Citado el: 23 de Febrero de 2015.] http://wso2.com/products/application-server/.

38. *SLD237 DESARROLLO DE LA ESPECIALIDAD PSICOLOGÍA DEL MÓDULO CONSULTA EXTERNA DEL SISTEMA ALAS-HIS.* Alvarez Lorenzo, Amaya, y otros. La Habana : s.n., 2013. 978-959-7213-02-04.

39. Belmonte Fernández, Oscar. *Introducción al lenguaje de programación Java. Una guía básica.* 2005.

40. Hibernate. [En línea] [Citado el: 20 de Febrero de 2015.] http://hibernate.org/.

41. SeamFamework.org. [En línea] [Citado el: 20 de Febrero de 2015.] http://seamframework.org/.

42. Oracle. [En línea] [Citado el: 22 de Febrero de 2015.] http://www.oracle.com/technetwork/java/javaee/ejb/index.html.

43. Ivar, Jacobson, Grady, Booch y James, Rumbaugh. *El proceso unificado de desarrollo de software. p.112-115.* s.l. : Addison Wesley Reading, 2000.

44. Larman, Craig. *UML y Patrones.* s.l. : Pearson. Prentice Hall, 1999.

45. *La arquitectura SOA de Microsoft aplicada al mundo real.* s.l. : Microsoft Corporation, 2006.

46. MeRinde. [En línea] Armadillo Integración Tecnológica. [Citado el: 4 de Marzo de 2015.] http://merinde.net/index.php?option=com_content&task=view&id=495&Itemid=29.

47. Letelier Torres, Patricio. *Desarrollo de Software Orientado a Objeto usando UML.* España : Universidad Politécnica de Valencia, 2002.

48. Álvarez Sarmiento, Pablo Ricardo. *Análisis, diseño e implementación de un sistema basado en componentes para automatizar la herramienta Balanced Scorecard en el área administrativa de TELEAMAZONAS.* 2007.

49. Sommerville, Ian. *Ingeniería del software.* s.l. : Pearson Educación, 2005. 8478290745.

BIBLIOGRAFÍA

- Jacobson, Ivar, Booch, Grady y Rumbaugh, James. *El proceso unificado de desarrollo de software.* s.l. : Addison Wesley Reading, 2000.
- Pressman, Roger . *Ingeniería del Software: Un enfoque práctico.* 1997.
- *HL7 Context Management "CCOW" Standard.* 2011.
- *IHE IT Infrastructure (ITI).* 2011.
- Schlueter, Isaac , Voss, Laurie y Boothby, Rod. NPM Node Package Manager.
- Larman, Craig. *UML y Patrones.* s.l. : Pearson. Prentice Hall, 1999.
- Sommerville, Ian. *Ingeniería del software.* s.l. : Pearson Educación, 2005. 8478290745.
- León, Rolando Alfredo Hernández; González, Sayda Coello. *El proceso de investigación científica*. Editorial Universitaria, 2011.
- Steiner, Jennifer G.; Neuman, B. Clifford; Schiller, Jeffrey I. Kerberos: An Authentication Service for Open Network Systems. En *USENIX Winter*. 1988. p. 191-202.

ANEXOS

Anexo 1 Registro en la bitácora de sucesos

Las tres imágenes del Anexo 1 muestran cómo define el perfil de integración ATNA que deben guardarse los registros de sucesos para auditorias.

Registro en la bitácora de sucesos

	Field Name	Opt	Value Constraints
Event AuditMessage/ EventIdentification	EventID	M	EV(110114, DCM, "UserAuthenticated")
	EventActionCode	M	"E" (Execute)
	EventDateTime	*M*	*not specialized*
	EventOutcomeIndicator	*M*	*not specialized*
	EventTypeCode	M	EV(110122, DCM, "Login") EV(110123, DCM, "Logout")
Source (1)			
Human Requestor (1)			
Destination (0)			
Audit Source (Client Authentication Agent) (1)			
Participant Object (0)			

Figura 20 Registro en la bitácora de sucesos.

Registro en la bitácora de sucesos

Audit Source AuditMessage/ AuditSourceIdentification	*AuditSourceID*	*U*	*Not specialized.*
	AuditEnterpriseSiteID	*U*	*not specialized*
	AuditSourceTypeCode	*U*	*not specialized*

Figura 21 Registro en la bitácora de sucesos.

Registro en la bitácora de sucesos

Source AuditMessage/ ActiveParticipant	UserID	M	the process ID as used within the local operating system in the local system logs.
	AlternativeUserID	*U*	*not specialized*
	UserName	*U*	*not specialized*
	UserIsRequestor	M	"true"
	RoleIDCode	M	EV(110150, DCM, "Application")
	NetworkAccessPointTypeCode	M	"1" for machine (DNS) name, "2" for IP address
	NetworkAccessPointID	M	The machine name or IP address, as specified in RFC 3881.
Human Requestor (if known) AuditMessage/ ActiveParticipant	UserID	M	Identity of the human that initiated the transaction.
	AlternativeUserID	*U*	*not specialized*
	UserName	*U*	*not specialized*
	UserIsRequestor	M	"true"
	RoleIDCode	*U*	*not specialized*
	NetworkAccessPointTypeCode	*NA*	
	NetworkAccessPointID	*NA*	

Figura 22 Registro en la bitácora de sucesos.

Anexo 2 Diagramas de Clases del diseño

El primer Diagrama de Clase del Diseño muestra la relación que existe entre las clases ClientAuthenticationAgent, Krb5CallbackHandler, KerberosTGT, KrbRef, Kerberos5LogiModule, para lograr autenticar un usuario en Kerberos.

DCD_ Autenticar usuario en Kerberos

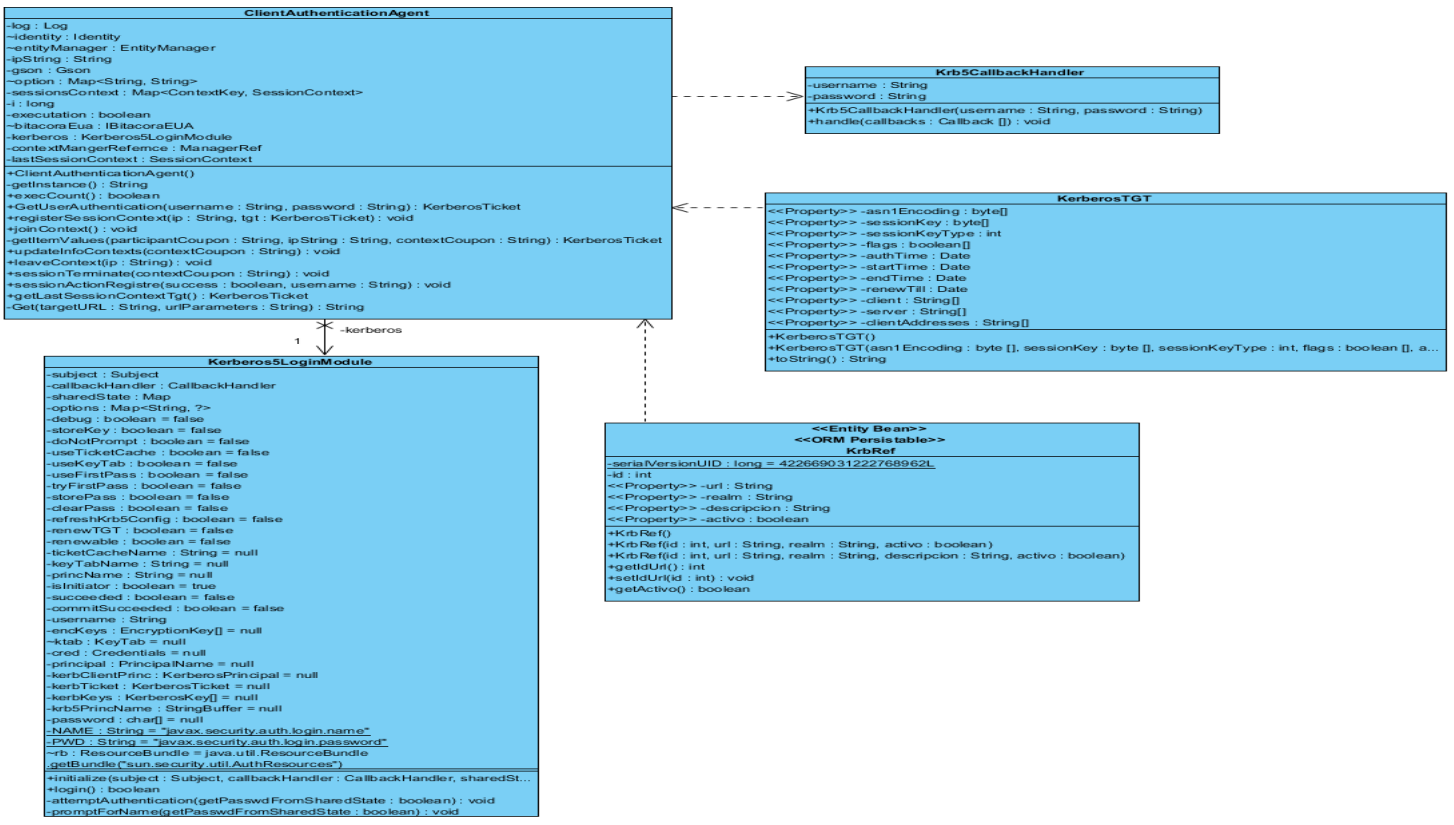

Figura 23 Diagrama de clases del diseño: DCD_Autenticar usuario Kerberos.

El segundo Diagrama de Clase del Diseño muestra la relación que existe entre las clases ContextManager, ManagerService, ContextParticipant, SKey, ContextSession, ContextData para gestionar un contexto.

DCD_Gestionar contexto

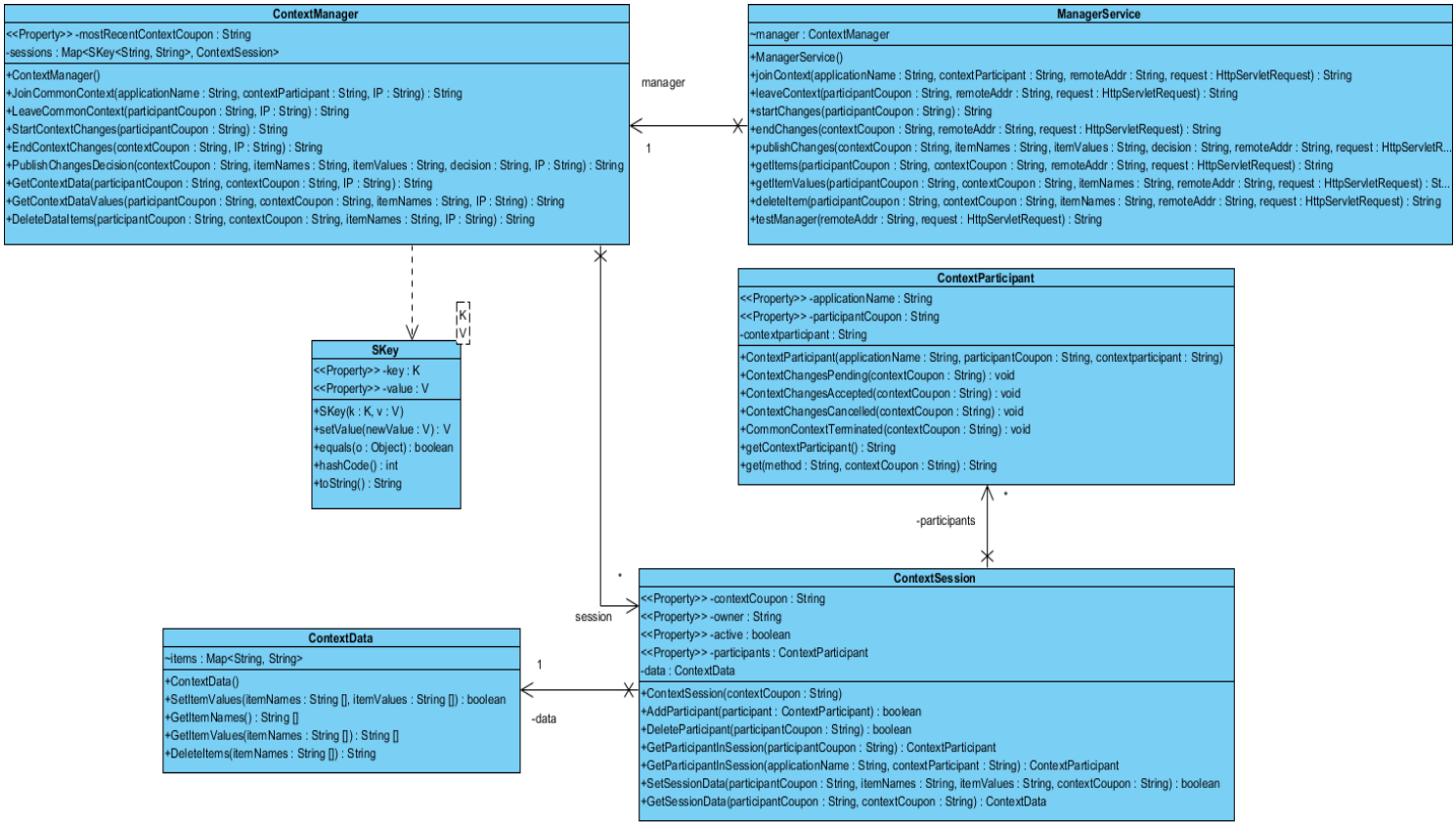

Figura 24 Diagrama de clases del diseño: DCD_Gestionar contexto.

El tercer Diagrama de Clase del Diseño muestra la relación que existe entre las clases ClientAuthenticationAgent, ContextKey, SessionContext, SKey, ContextSession, ReneverTask, Kerberos5LoginModule, ManagerRef, IClientAuthenticationAgent para renovar el TGT.

DCD_Renovar TGT

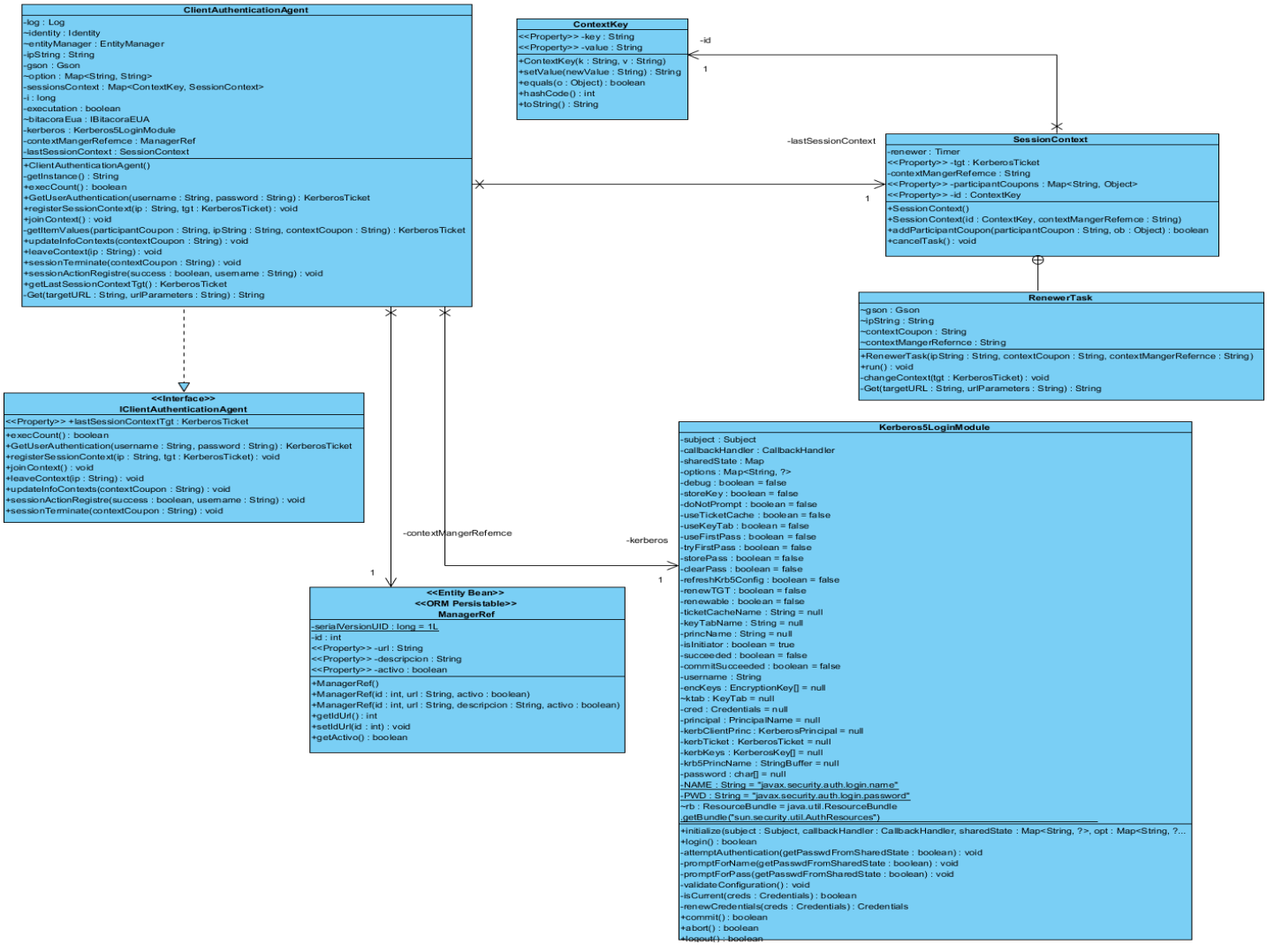

Figura 25 Diagrama de clases del diseño: DCD_Renovar TGT.

El cuarto Diagrama de Clase del Diseño muestra las relación que existe entre las clases AuditSource, AtnaCode, Auditevent, SourceHumanrequestor, BitacoraEUA, IBitacoraEUA, para lograr registrar en la bitácora de sucesos.

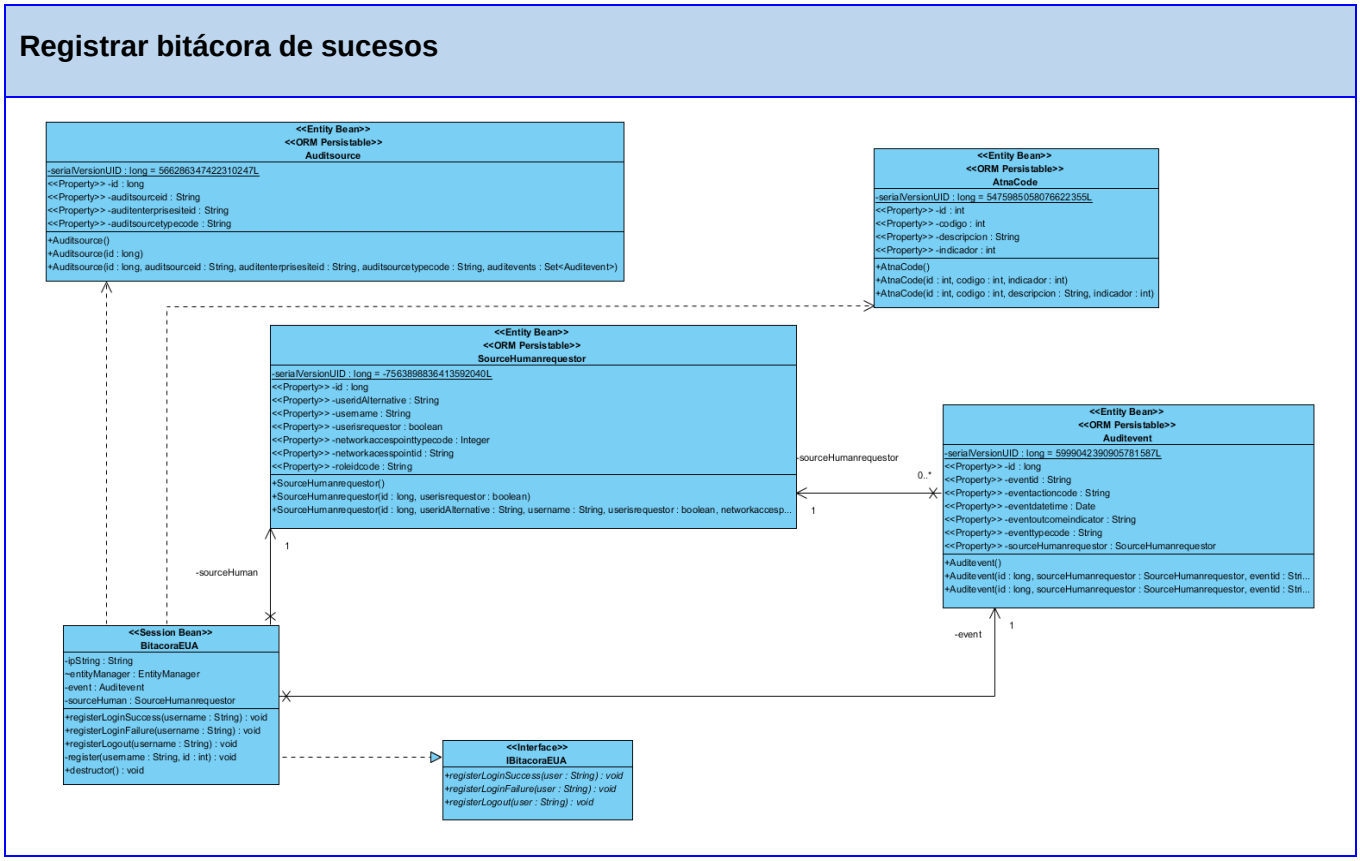

Figura 26 Diagrama de clases del diseño: DCD_Registrar bitácora de sucesos.

GLOSARIO DE TÉRMINOS

- ***Audit Trail and Node Authentication* (ATNA, por sus siglas en inglés):** es un perfil de integración que establece las características de un nodo básico seguro. En él se definen los requisitos básicos de auditoría, el entorno de seguridad (identificación de usuario, autenticación, autorización, control de acceso) y establece las características de la comunicación de mensajes de auditoria.
- **Bitácora:** es un archivo en el cual se lleva un registro de las acciones realizadas en una determinada actividad.
- **HTTP:** protocolo de transferencia de hipertexto.
- **HTTPS:** protocolo de transferencia de hipertexto seguro.
- **REST:** *Representational State Transfer*, es un estilo arquitectónico que se centra en el uso de los estándares XML (Lenguaje de Marcas Extensible) y HTTP para la transmisión de datos.
- **Sesión de contexto común:** es un espacio donde varias aplicaciones comparten un contexto.
- **Nodejs:** entorno de programación en la capa del servidor basado en el lenguaje de programación Javascript, con entrada/salida de datos en una arquitectura orientada a eventos y basado en el motor Javascript V8.
- **Perfil de integración:** describe una necesidad clínica de integración de sistemas y la solución para llevarla a cabo. Define también los componentes funcionales, que se llamarán actores IHE, y especifica con el mayor grado de detalle posible las transacciones que cada actor deberá llevar a cabo, basadas siempre en estándares.
- ***Ticket*:** es un grupo temporal de credenciales electrónicas que verifican la identidad de un cliente para un servicio en particular.
- **UUID:** *Universally Unique Identifier*, es un identificador único global representado como una cadena de 128 bits que se genera aleatoriamente.