Universidad de las Ciencias Informáticas
Facultad 2

Título:

Componente de identificación en servidores para aplicaciones basadas en Android.

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO INFORMÁTICO

Autor: **Claudia Caridad Alvarez Campos**

Tutor: **Lex Karel Zayas Hernández**

"Cuidad de la Habana, Junio, 2013"

Estoy convencido de que la mitad de lo que separa a los emprendedores exitosos de los que no triunfan es la perseverancia.

Steve Jobs

DECLARACIÓN DE AUTORÍA

Declaro que soy la única autora de este trabajo y autorizo al Centro de Telemática de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Claudia Caridad Alvarez Campos

_____

Lex Karel Zayas Hernández

_____

RESUMEN

Las tecnologías de telefonía celular tienen un amplio alcance en el mercado mundial, con su utilización se hace posible la creación de considerables números de servicios, tal como es el caso del consumo de servicios web, los cuales requieren en su mayoría que el usuario se autentique para acceder a ellos con el objetivo de asegurar la privacidad o establecer un control en su utilización. Teniendo en cuenta lo anteriormente expuesto, se propone el desarrollo de un componente que haga posible el establecimiento de la conexión con los servidores de aplicación, y garantice el proceso de autenticación en dichos servidores a los sistemas implementados sobre la plataforma Android, ofreciendo a los desarrolladores que utilizan esta plataforma para la implementación de sus sistemas, la oportunidad de transferir todas las responsabilidades de conexión e identificación a dicho componente. Con el objetivo de guiar el proceso de desarrollo se utiliza la metodología ágil XP que garantiza la simplicidad del mismo, para la implementación del componente se emplea el lenguaje de programación Java, haciendo uso de un estilo arquitectónico en capas. El resultado es validado a través de la construcción de dos aplicaciones que hacen uso de las funcionalidades del componente obteniéndose resultados satisfactorios al concluir las pruebas.

PALABRAS CLAVE

Telefonía celular, componente, autenticación, servidores de aplicación.

TABLA DE CONTENIDOS

Introducción

Con el vertiginoso desarrollo de las tecnologías informáticas vienen aparejadas nuevas necesidades, entre ellas se encuentra el surgimiento de escenarios que requieren la identificación de los usuarios, proceso que puede definirse como el momento en que el usuario se da a conocer al sistema, ya sea con el objetivo de asignar permisos o establecer un control de acceso al consumo de determinados servicios.

Tradicionalmente, la gran demanda para soluciones de identificación seguras ha provenido de empresas que buscaban satisfacer sus propias necesidades de seguridad al interior de sus organizaciones, así como de organismos gubernamentales en contextos donde se creía que los intereses de la seguridad nacional estaban en juego. (1)

En los últimos años, la demanda por soluciones de identificación ha estado en aumento en otros contextos que afectan directamente la privacidad de los individuos, como el consumo de servicios que requieran establecer un control sobre los usuarios que acceden a ellos. Esto es mayormente provocado por la creciente popularidad de Internet y de las redes de comunicaciones móviles, así como del rápido incremento de dispositivos de información, tales como teléfonos celulares inteligentes con capacidades web. Estos dispositivos con sus altas prestaciones y las nuevas tecnologías que incorporan han extendido su uso más allá de hacer llamadas telefónicas o enviar mensajes de texto, dejando un escenario abierto para el desarrollo de aplicaciones que permitan a los usuarios identificarse en los distintos servicios web a través de ellos.

Atendiendo a lo anteriormente planteado, el Departamento de Telemática de la Universidad de las Ciencias Informáticas se propone implementar un mecanismo que permita la verificación de identidad de manera segura en los servidores de aplicaciones a aquellos sistemas que utilicen Android como plataforma.

Al poner en común todos estos elementos surge el siguiente problema científico: ¿Cómo realizar la identificación de un usuario en un servidor aprovechando las potencialidades que brindan los teléfonos inteligentes que hacen uso del sistema operativo Android?

Definiéndose como objeto de estudio los procesos de autenticación de usuarios a través de sistemas operativos para teléfonos inteligentes. Donde se entiende como autenticación, la verificación que realiza el sistema sobre los datos de identificación del usuario.

Con el propósito de dar solución al problema existente y estando al tanto del área de conocimiento donde está enmarcado, se concreta como objetivo general del trabajo: Desarrollar un componente que permita

identificar a los usuarios ante un servidor a través de los teléfonos inteligentes que utilicen el sistema operativo Android. Especificándose como definición de componente, un elemento de un sistema de software que ofrece un conjunto de servicios, o funcionalidades.

A partir de esto se obtiene como campo de acción del presente trabajo, los procesos de autenticación de usuarios mediante aplicaciones desarrolladas para el sistema operativo Android. Para cumplir con el objetivo general propuesto y dar solución a la situación problémica planteada, se proponen las tareas siguientes:

1. Analizar las actuales tendencias en los componentes para la identificación en servidores web.
2. Analizar las funcionalidades básicas de un componente de identificación, así como los diferentes contextos donde sea factible su utilización para aplicarlos a la solución.
3. Definir los mecanismos de seguridad y aplicarlos en la solución.
4. Realizar análisis y diseño del componente.
5. Implementar el componente.
6. Realizar las pruebas al componente para comprobar su correcto funcionamiento.
7. Implementar aplicaciones para validar el funcionamiento del componente.

Los métodos científicos que serán utilizados en la investigación, serán los siguientes:

Métodos teóricos:

• Histórico - Lógico: Este método se utiliza para indagar sobre el devenir histórico de los sistemas de autenticación, los principales conceptos que giran alrededor del mismo y los sistemas que ya existen relacionados con este proceso.

• Analítico - sintético: Su utilización será necesaria para el estudio de las diferentes herramientas y metodologías a ser utilizadas, además para profundizar en determinados conceptos asociados al problema.

Métodos empíricos:

• Observación: Este método es utilizado con el objetivo observar cómo funcionan los procesos de identificación y los principales problemas que presentan los mismos, con el fin de aplicar las experiencias reunidas al desarrollo del componente.

• Entrevista: Las entrevistas son utilizadas para la obtención de información e ideas, mediante ellas se extraen conocimientos de personas especializadas en el tema, los mismos aportarán criterios de envergadura para tomar decisiones en la realización del trabajo. Las entrevistas utilizadas son del tipo no directivas e informativas.

El trabajo de diploma se divide en 5 capítulos, los cuales estarán estructurados de la siguiente forma:

Capítulo1. “Fundamentación Teórica”: Comprende un análisis de los sistemas que existen en la actualidad y se vinculan con la investigación, y el estudio de las tecnologías y herramientas a utilizar en el desarrollo del componente.

Capítulo 2. “Características del componente”: Aborda temas como la planificación del proyecto y su estructura. Contiene datos que contribuyen a la identificación de los objetivos del mismo, preparando el camino para su realización en tiempo según los cronogramas trazados, logrando identificar y minimizar posibles riesgos.

Capítulo 3. “Análisis y Diseño del componente”: Se describen las tarjetas: Tarjetas de Cargo o Clase, Responsabilidad y Colaboración (CRC), así como los patrones de diseño.

Capítulo 4. “Implementación”: Se expone todo lo relacionado a los procesos de implementación componente, destacándose la identificación de las tareas de ingeniería y la planificación de cada iteración con respecto a cada una de estas tareas.

Capítulo 5. “Prueba”: Se describen las aplicaciones que validan el correcto funcionamiento del componente de identificación y además se detallan las pruebas a realizar.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1. Introducción

En el presente capítulo se hace referencia a los elementos teóricos que fundamentan la propuesta y el desarrollo del componente sugerido anteriormente. También completa un estudio del arte de las posibles herramientas y tecnologías que serán utilizadas en el desarrollo del componente.

1.1 Conceptos fundamentales relacionados con el problema.

1.1.1 Identificación y autenticación

El término identificación se usa para designar al acto de identificar, reconocer o establecer los datos e información sobre una persona. En el contexto de la informática, para que los usuarios puedan obtener seguridad, acceso al sistema y administración de recursos deberán identificarse, es decir, darse a conocer al sistema. La acción que se realiza sobre dicha identificación, no es más que el proceso de intento de verificar la identidad digital del remitente de una comunicación y es denominado autenticación. (2)

1.1.2 Sistemas de identificación

Los sistemas de identificación se emplean para el manejo de información relativa a las personas y a los objetos. Generalmente, estos sistemas requieren de dos componentes fundamentales: un elemento codificado que contiene la información y un elemento con capacidad de reconocerla. Posteriormente, el receptor realiza diversos procesos; en primer lugar, los datos son decodificados, a continuación la información es verificada, comparada y aceptada para luego realizar alguna decisión lógica.

De manera cotidiana los sistemas de identificación de usuarios pueden ser diversos: para el acceso a una cuenta en un banco, a información restringida, a un servicio, entre otros. Gracias a que los sistemas modernos son automáticos, los procesos se agilizan, se cometen menos errores y en consecuencia se incrementa la confiabilidad y la eficiencia. (3)

1.1.3 Protocolo de comunicación HTTP[1].

El Protocolo de transferencia de hipertexto fue desarrollado por la World Wide Web Consortium y la Internet Engineering Task Forces, a través de este los clientes y los servidores determinan de forma dinámica el formato de los documentos, lo que permite que utilicen formato de datos no estándar para el intercambio de datos.

Las cabeceras de HTTP pueden contener información acerca de los objetos que se transmiten a través de la web. Con la información de las cabeceras, las aplicaciones Cliente-Servidor negocian formatos que pueden utilizar para transferir los objetos. Además el protocolo está basado en texto por lo cual es legible y no necesita decodificación.

HTTP define la sintaxis y la semántica que utilizan los elementos de software de la arquitectura web (clientes, servidores, proxies) para comunicarse; está orientado a transacciones y sigue el esquema petición-respuesta entre un cliente y un servidor. A la información transmitida se la llama recurso y se le identifica mediante un localizador uniforme de recursos (URL). Los recursos pueden ser archivos, el resultado de la ejecución de un programa, una consulta a una base de datos, la traducción automática de un documento, etc.

Entre los métodos de petición que define se encuentran:

- GET (obtener información del servidor): Traer datos que están en el servidor, ya sea en un archivo o base de datos, al cliente. Independientemente de que para eso deba enviarse (request) algún dato que será procesado para luego devolver la respuesta (response) que se espera.

- POST: Enviar información desde el cliente para que sea procesada y actualice o agregue información en el servidor, como sería la carga o actualización en sí de algún dato. Cuando se envían (request) datos a través de un formulario, estos son procesados y luego a través de una redirección por ejemplo se devuelve (response) alguna página con información. (4)

[1] **HTTP**: Hypertext Transfer Protocol o Protocolo de Transferencia de Hipertexto.

1.1.4 Protocolo seguro de comunicación HTTPS[2].

Es un protocolo de aplicación basado en el protocolo HTTP, destinado a la transferencia segura de datos de Hipertexto, es decir, es la versión segura de HTTP. Utiliza un cifrado basado en SSL[3] para crear un canal más apropiado para el tráfico de información sensible. De este modo se consigue que la información sensible (usuario y claves) no pueda ser usada por un atacante que haya conseguido interceptar la transferencia de datos de la conexión, ya que lo único que obtendrá será un flujo de datos cifrados que le resultará imposible de descifrar.
HTTPS no es un protocolo separado, pero refiere el uso del HTTP ordinario sobre una Capa de Conexión Segura. El sistema puede ser usado con el objetivo de limitar el acceso a un servidor web a usuarios autorizados. (5)

1.2 Descripción de los Métodos de Autenticación.

1.2.1 Autenticación básica.

Este sistema de autenticación está diseñado para permitir a un programa cliente, proporcionar credenciales basadas en nombre de usuario y contraseña, que le permitan autenticarse ante un determinado servicio. Su funcionamiento no está pensado para ser utilizado sobre líneas públicas, debido a que las credenciales que se envían desde el cliente al servidor, aunque no se envían directamente en texto plano, se envían únicamente codificadas en base-64, lo que hace que se puedan obtener fácilmente. La combinación de la autenticación básica con el protocolo de capa de conexión segura SSL, permite disponer de una autenticación segura y rápida. (6)

[2] **HTTPS**: Hypertext Transfer Protocol Secure o Protocolo Seguro de Transferencia de Hipertexto.
[3] **SSL**: Secure Sockets Layer o Capa de Conexión Segura.

1.2.2 Autenticación Digest.

La autenticación de acceso Digest es uno de los métodos que un servidor puede utilizar para negociar credenciales con la aplicación cliente, aplica una función Hash (función de truncado) a los datos antes de enviarlos, lo que le atribuye a este sistema mayor seguridad que al de autenticación de acceso básico, es decir, soluciona el problema de la transferencia de contraseñas en claro sin necesidad de usar SSL. Su objetivo no consiste en reemplazar los protocolos de autenticación fuerte, como la autenticación de clave pública. Visto diferente es una función matemática que produce una secuencia de caracteres, normalmente entre 128 y 256 bits de longitud (MD5[4], algoritmo criptográfico), a partir de un archivo de información inicial de cualquier longitud.

Propiedades:

- Cada bit de salida del digest es influenciado por cada bit de entrada.
- Para cualquier bit que se cambie a la entrada, cada bit de salida tiene al menos un 50% de probabilidades de cambiar.
- Dado un fichero (texto) de entrada, debe ser informáticamente inviable encontrar otro texto de entrada que dé el mismo producto en el digest.

Modo de funcionamiento:

1. En lugar de enviar la contraseña, el cliente envía un “digest” de la contraseña, el cual es un entremezclado irreversible de la contraseña.
2. El cliente y el servidor conocen la contraseña secreta, por lo que el servidor puede verificar que el digest proporcionado concuerda con la contraseña.
3. Dado sólo el digest, cualquier otra persona ajena al proceso no tiene manera de determinar la contraseña tal como la puede capturar, a menos que pruebe con todas la contraseñas posibles.

Para evitar ataques por repetición, el servidor le puede pasar al cliente un token especial llamado nonce, el cual cambia frecuentemente (cada un milisegundo o en cada autenticación). El cliente adjunta este token a la contraseña antes de computar el digest. Al combinar el nonce con la

[4] **MD5**: Message Digest Algorithm 5 o Algoritmo de Resumen de Mensaje 5.

contraseña se garantiza que el digest cambie cada vez que cambie el nonce, es decir que el digest de la contraseña almacenado sólo es válido para un valor particular del nonce. (7)

1.3 Sistemas de autenticación del mercado actual.

En el mercado actual existen varios sistemas compatibles con la plataforma Android que son utilizados para autenticarse en servidores de aplicación web a través de teléfonos celulares, a continuación aparecen reflejados dos de ellos con sus principales características.

1.3.1 ESET[5] Secure Authentication.

Consiste en un sistema de autenticación para redes corporativas. La solución proporciona una capa adicional de seguridad en el proceso de autenticación de los usuarios en sus teléfonos móviles, además de la validación habitual con usuario y contraseña, su funcionamiento se basa en un sistema cliente/servidor que define que cuando el usuario necesita validarse, recibe una contraseña desde el servidor, que sirve para un solo uso y que garantiza el acceso al usuario desde su dispositivo móvil con mayor protección de la habitual.

Es compatible con iPhone, Android, Mora, Windows Phone 7 y 8, Windows Mobile y sistemas basados en J2EE. La solución soporta autenticación basada en mensajes de texto, por lo que se puede utilizar incluso en teléfonos antiguos que no puedan instalar la aplicación.

La autenticación se realiza mediante una aplicación que el usuario descarga en su terminal y una vez instalada, le permite generar contraseñas seguras de un solo uso. También se pueden obtener las claves enviando un mensaje SMS en el caso de que el dispositivo del usuario no soporte la instalación de aplicaciones.

Su comercialización es válida para entornos superiores a 5 usuarios con infraestructura cliente/servidor, con un precio de 122 euros por cada licencia. (8)

[5] **ESET**: Compañía de seguridad informática establecida en Bratislava, Eslovaquia. El nombre de la compañía se deriva de la diosa egipcia Isis.

1.3.2 Sistema CAS[6].

CAS es el software utilizado por la Universidad de Murcia para proporcionar acceso único autenticado a las diferentes aplicaciones. Las aplicaciones móviles que quieran hacer uso de la autenticación móvil CAS no tienen que preocuparse del protocolo, ni del intercambio de mensajes con el servidor de autenticación, simplemente distribuirse la librería CasMovil la cual se encuentra disponible para Android y iPhone. Para que una aplicación móvil pueda usar el componente CAS móvil es necesario que la aplicación web de la que hace uso esté dada de alta en el Servidor CAS.

Funcionalidad de la librería.
El CAS es un servicio web que se comunica mediante peticiones HTTP. Se usa para autenticar una parte cliente frente a su parte servidor, como por ejemplo la autenticación aplicación móvil-aplicación web. Para realizar esta autenticación, la parte cliente debe solicitar al CAS un ticket de validación para presentárselo a su parte servidor. La parte servidor debe validar con el CAS este ticket que le presenta el cliente. En este proceso de autenticación la parte cliente y el CAS usan dos tipos de tickets: el ST y el TGC.

Service Ticket (ST): Es una cadena usada por el cliente como credencial para obtener acceso a un servicio concreto, es decir, la aplicación web dada de alta en el CAS. Se obtiene del CAS al presentarle las credenciales del cliente junto al identificador del servicio. Sólo es válido para el servicio que fue indicado y tan solo sirven para una única validación. Una vez se intenta validar, el ticket es inhabilitado.
Ticket-Granting cookie (TGC): Es una cookie HTTP establecida por el CAS que mantiene el estado de autenticación del cliente. Mientras sea válido, el cliente puede presentar este ticket al CAS en lugar de sus credenciales.

Integración en Android.
La librería CAS móvil se distribuye para aplicaciones Android en un fichero .zip que incluye:

- MobileCASClient.jar. Archivo JAR necesario incluir en tu aplicación móvil.
- MobileCASClient_desarrollo.jar. Igual que el anterior pero conecta con el servidor CAS de desarrollo.
- Manual para Android.pdf. Un documento de manual detallado.

[6] **CAS**: Central Authentication Service o Servicio de Autenticación Central.

- Javadoc: Carpeta que contiene la documentación Javadoc de las clases y métodos públicos. (9)

1.4 Métodos, Herramientas y Técnicas a utilizar.

1.4.1 Sistema Operativo Android.

Android en un Sistema Operativo orientado a terminales móviles basado en el núcleo de Linux. Permite controlar dispositivos por medio de bibliotecas desarrolladlas o adaptados por Google mediante el lenguaje de programación Java. Es una plataforma de código abierto, esto quiere decir que cualquier desarrollador puede crear aplicaciones y compilarlas a código nativo de ARM (API de Android). Aunque la mayoría de las aplicaciones están escritas en Java, no hay una máquina virtual Java en la plataforma. El bytecode no es ejecutado, sino que primero se compila en un ejecutable Dalvik que es una máquina virtual diseñada específicamente para este sistema operativo.

Principales características:

- Incluye un emulador de dispositivos, herramientas para depuración de memoria y análisis de rendimiento de software.
- El entorno de desarrollo que utiliza es Eclipse, usando el plugin de herramientas de desarrollo del propio Android.
- El framework de aplicaciones que utiliza permite el reemplazo y la reutilización de los componentes.
- Posee un navegador integrado basado en el motor Open Source Webkit.
- Utiliza una base de datos liviana llamada SQlite, la cual realiza un almacenamiento estructurado que se integra directamente con las aplicaciones.
- Tiene disponible el sistema multitarea real de aplicaciones, es decir, las aplicaciones que no están ejecutándose en primer plano reciben ciclos de reloj.

1.4.2 Entorno de Desarrollo Integrado Eclipse.

Eclipse es un IDE (Entorno de Desarrollo Integrado) de código abierto multiplataforma para desarrollar "Aplicaciones de Cliente Enriquecido" (RCP - Rich Client Platform). Es totalmente libre y no es necesario

comprar ninguna licencia comercial, se puede instalar en sistemas operativos como Windows, Linux, Mac OSX, y esto hace de esta herramienta multiplataforma la ideal para trabajos con Java, SDK, IDE, C/C++.
Entre otras características nos encontramos con un potente editor de texto con la capacidad de resaltar sintaxis, así como también una compilación en tiempo real. Cuenta además con sistemas de pruebas unitarias con JUnit, control de versiones con CVS, asistentes (wizards) para la creación de proyectos, facilidad de refactorización y gestión de plugins.
Es utilizado por muchos desarrolladores de aplicaciones Android ya que el SDK de Google Android SDK está disponible para esta plataforma mediante plugins que permiten depuración de aplicaciones Android así como la instalación en dispositivos y su compilación. Emplea módulos para proporcionar cada una de sus funcionalidades, a diferencia de otros entornos monolíticos donde las funcionalidades están todas incluidas, las necesite el usuario o no.(11)

1.4.3 Lenguaje de Programación JAVA.

Java es un lenguaje de programación que supone un significativo avance en el mundo de los entornos de software, y esto viene avalado por las siguientes características:

- Orientación a objetos: En Java el concepto de objeto resulta sencillo y fácil de ampliar. Además se conservan elementos "no objetos", como números, caracteres y otros tipos de datos simples.
- Riqueza semántica: Aunque cada tarea se puede realizar de un número reducido de formas, posee un gran potencial de expresión e innovación desde el punto de vista del programador.
- Modelo de objeto rico: Existen varias clases que contienen las abstracciones básicas para facilitar a los programas una gran capacidad de representación.
- Completado con utilidades: El paquete de utilidades de Java viene con un conjunto completo de estructuras de datos complejas y sus métodos asociados. Se dispone también de estructuras de datos habituales, como pilas y tablas hash.
- Interactivo y animado: Ofrece la posibilidad de crear programas en red interactivos. Tiene soporte a la utilización de múltiples hilos de programación (multithread).
- Arquitectura neutral: Está diseñado para que un programa escrito en este lenguaje sea ejecutado correctamente independientemente de la plataforma en la que se esté actuando.

Los niveles de seguridad que presenta son:

- Fuertes restricciones al acceso a memoria, como son la eliminación de punteros aritméticos y de operadores ilegales de transmisión.
- Rutina de verificación de los *códigos de byte* que asegura que no se viole ninguna construcción del lenguaje.
- Verificación del nombre de clase y de restricciones de acceso durante la carga.
- Sistema de seguridad de la interfaz que refuerza las medidas de seguridad en muchos niveles. (12)

1.4.4 Sistema Gestor de Base de Datos SQLite.

SQLite es un sistema de gestión de bases de datos que constituye un proyecto de dominio público.
Principales características:

- A diferencia de los sistemas que utilizan la arquitectura cliente-servidor, el motor de SQLite no es un proceso independiente con el que el programa principal se comunica. El programa utiliza la funcionalidad de SQLite a través de llamadas simples a subrutinas y funciones. Esto reduce la latencia en el acceso a la base de datos.
- El conjunto de la base de datos (definiciones, tablas, índices, y los propios datos), son guardados como un sólo fichero estándar en la máquina host.
- Desde Java se puede acceder mediante el driver de SQLiteJDBC.
- Usa un sistema de tipos inusual, en lugar de asignar un tipo a una columna como en la mayor parte de los sistemas de bases de datos SQL, los tipos se asignan a los valores individuales.
- Debido a su pequeño tamaño, SQLite es muy adecuado para los sistemas integrados como Android. (13)

1.4.5 Servidor web Apache 2.2.22.

El Servidor HTTP Apache es un servidor web HTTP de código abierto para plataformas Unix, Microsoft Windows, Macintosh y otras. Su desarrollo se fundamenta en el trabajo de un reducido grupo de desarrolladores denominado Apache Group.

Apache tiene amplia aceptación en la red, desde 1996 es el servidor HTTP más usado. Alcanzó su máxima cuota de mercado en 2005 siendo el servidor empleado en el 70% de los sitios web en el mundo. La arquitectura que emplea es modular, consta de diversos módulos que aportan mucha de la funcionalidad que podría considerarse básica para un servidor web.

Es usado primariamente para enviar páginas web estáticas y dinámicas en la World Wide Web. Muchas aplicaciones están diseñadas asumiendo como ambiente de implantación a Apache, o que utilizarán características propias de este servidor web. La licencia de software bajo la cual es distribuido es una parte distintiva de su historia y de la comunidad de código abierto, ya que permite la distribución de derivados de código abierto y cerrado a partir de su código fuente original.

Ventajas:

- Modular.
- Código abierto.
- Multi – plataforma.
- Extensible.
- Popular (fácil conseguir ayuda/soporte). (14)

1.3.5 Metodología XP (eXtreme Programming).

La programación extrema o eXtreme Programming (XP) es una metodología de desarrollo de la ingeniería de software. Es el más destacado de los procesos ágiles de desarrollo y se diferencia de las metodologías tradicionales principalmente en que pone más énfasis en la adaptabilidad que en la previsibilidad, esta característica lo convierte en un aproximación más realista que intentar definir todos los requisitos al comienzo del proyecto e invertir esfuerzos después en controlar los cambios de los requisitos. El ciclo de vida ideal de XP consiste en seis fases (Exploración, Planificación de la Entrega, Iteraciones, Producción, Mantenimiento y Muerte del Proyecto).

Principales características:

- Desarrollo iterativo incremental: Pequeñas mejoras.
- Pruebas unitarias continuas: Frecuentemente repetidas y automatizadas incluyendo pruebas de regresión.
- Corrección de errores: Se recomienda corregir los errores antes de añadir una nueva funcionalidad y hacer entregas frecuentes.
- Refactorización del código: Reescribir partes del código para aumentar su legibilidad pero sin modificar su comportamiento. Las pruebas han de garantizar que en la refactorización no se ha introducido ningún fallo.
- Simplicidad en el código: La programación extrema asume que es más sencillo hace algo simple y tener un poco de trabajo extra para cambiarlo si se requiere, que realizar algo complicado y quizás nunca utilizarlo.

Ventajas:

- Es apropiado para entornos volátiles.
- Permite tener una retroalimentación por parte de los usuarios.
- Es más transparente para los clientes ya que conocen la fecha de entrega de las funcionalidades vitales para su negocio.
- Permite definir en cada iteración cuáles son los objetivos de la siguiente.
- La presión está a lo largo de todo el proyecto y no en la entrega final. (15)

Conclusiones

En este capítulo se realizó un estudio acerca de las tecnologías sobre las cuales se desarrollará el componente de identificación y los modos de autenticación que serán implementados, lo que permitió concluir que la solución utilizará como protocolo de comunicación HTTP y su versión combinada con SSL, además ofrecerá la posibilidad de establecer la conexión con el servidor a través de cualquiera de los modos de identificación y métodos de envío descritos anteriormente. También se llevó a cabo un análisis exhaustivo de la metodología y las herramientas de desarrollo de software que serán empleadas durante el ciclo de construcción, definiendo que se utilizará Java como lenguaje de programación haciendo uso del entorno de desarrollo integrado Eclipse y se empleará la metodología XP para guiar el proceso.

CAPÍTULO 2: CARACTERÍSTICAS DEL COMPONENTE

2. Introducción

En este capítulo se hace referencia a las fases de Exploración y Planificación, propias de la metodología de desarrollo utilizada, donde se explican los procesos que serán objeto de automatización a través de la confección de las historias de usuarios. Al mismo tiempo se lleva a cabo la familiarización con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto. Se prueba la tecnología y se exploran las posibilidades de la arquitectura.

2.1 Objeto de estudio.

2.1.1 Objeto de automatización.

Se define como objeto de automatización el proceso de gestión de identificación, el cual tiene lugar en el control de conexión de usuarios a servidores de aplicación a través de dispositivos móviles utilizando el protocolo HTTP y su combinación con SSL.

2.1.2 Propuesta del componente.

Se propone el desarrollo de un mecanismo que ofrezca servicios de autenticación en servidores de aplicaciones web haciendo uso del teléfono celular. De esta manera se decidió la implementación de un componente que pueda integrarse con aplicaciones desarrolladas para el sistema operativo Android, facilitando la comunicación con el servidor a los programadores que utilizan esta plataforma, ver figura 1.
El resultado de la implementación debe permitir al desarrollador que lo utilice escoger el modo de autenticación (Basic / Digest / None) y el método de envío que desea utilizar para establecer la conexión y el intercambio de información con el servidor, además para facilitar el entendimiento de la solución se debe generar una documentación (Javadoc[7]) que suministre información detallada sobre su funcionamiento.

[7] **Javadoc:** Estándar para documentar clases de Java.

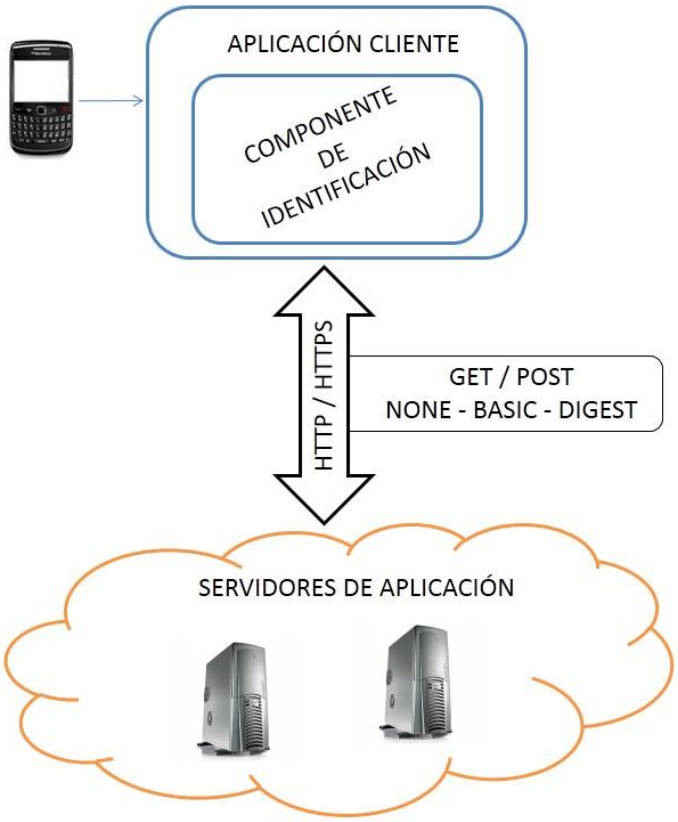

Figura 1: Propuesta del componente.

2.2 Especificación de los requisitos de software.

2.2.1 Características no funcionales del componente.

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable. Los requerimientos no funcionales forman una parte significativa de la especificación. Son importantes para que clientes y usuarios puedan valorar las características no funcionales del producto y de esta forma poder decir si el producto tiene la calidad requerida.

- Requisitos de hardware.

Servidores	Especificaciones
Servidor de aplicaciones web	• RAM: 1 GB o superior.

Tabla 1: Requisitos de hardware (Servidores).

Cliente	Especificaciones
Dispositivo móvil	• Soporte HTTP 1.1

Tabla 2: Requisitos de hardware (Cliente).

- Requisitos de software.

Servidores	Especificaciones
Servidor de aplicaciones web	• Sistema Operativo: Ubuntu 12.04.2 • Servidor web: Apache 2.2

Tabla 3: Requisitos de software (Servidores).

Cliente	Especificaciones
Dispositivo Móvil	• Sistema operativo: Android 2.2

Tabla 4: Requisitos de software (Cliente).

2.3 Fase de Exploración.

La fase de Exploración es la primera fase definida por la metodología XP. Esta fase comienza por la creación de una serie de historias, llamadas Historias de Usuarios (HU), las cuales definen mediante su redacción qué es lo que verdaderamente necesita el cliente. Es aquí donde se define el alcance real, permitiendo una familiarización del equipo de desarrollo con las herramientas y tecnologías a usar en la solución. (16)

2.3.1 Historias de Usuarios.

Las Historias de Usuarios son las técnicas que utiliza la metodología XP para especificar los requisitos del software. Estas son escritas desde la perspectiva del cliente, donde describen de forma breve las características que el componente debe realizar, aunque también los desarrolladores pueden brindar su ayuda en la identificación de las mismas. El contenido de las historias debe ser concreto, sencillo, dinámico y flexible. Cada una de ellas es lo suficientemente comprensible, como para que los programadores puedan estimar con un reducido margen de riesgo, su tiempo de desarrollo. Es el cliente el encargado de asignarle una prioridad a cada Historia de Usuario y es el equipo de desarrollo el encargado de asignarle un costo, este se traduce en las semanas que llevará el desarrollo de las mismas. Por otra parte es bueno destacar que las Historias de Usuarios nuevas pueden describirse en cualquier momento, con esto se comprueba la flexibilidad de la metodología. (17)

2.3.1.1 Clasificación de las Historias de Usuarios.

La prioridad en el negocio:

Alto: Se le otorga a las HU que resultan funcionalidades fundamentales en el desarrollo del componente, a las que el cliente define como principales para el control integral.
Medio: Se le otorga a las HU que resultan para el cliente como funcionalidades a tener en cuenta, sin que estas tengan una afectación sobre el componente que se esté desarrollando.

Bajo: Se le otorga a las HU que constituyen funcionalidades que sirven de ayuda al control de elementos asociados al equipo de desarrollo, a la estructura y no tienen nada que ver con el componente en desarrollo.

El riesgo en su desarrollo:

Alto: Cuando en la implementación de las HU se considera la posible existencia de errores que lleven la inoperatividad del código.
Medio: Cuando pueden aparecer errores en la implementación de la HU que puedan retrasar la entrega de la versión.
Bajo: Cuando pueden aparecer errores que serán tratados con relativa facilidad sin que traigan perjuicios para el desarrollo del proyecto.
Las HU serán representadas mediante tablas divididas por las siguientes secciones:

- Número: Número de la historia de usuario incremental en el tiempo.
- Nombre de Historia de Usuario: El nombre de la historia de usuario sería para identificarlas mejor entre los desarrolladores y el cliente.
- Usuario: Personas involucradas en el desarrollo de la HU.
- Iteración Asignada: Número de la iteración.
- Prioridad en negocio: Alto, Medio o Bajo.
- Riesgo en Desarrollo: Alto, Medio o Bajo.
- Puntos estimados: Tiempo estimado que se demorará el desarrollo de la HU; cada punto es considerado como una semana de trabajo.
- Puntos Reales: Tiempo que se demoró en realidad el desarrollo de la HU.
- Descripción: Breve descripción de la HU.
- Observaciones: Señalamiento o advertencia del componente.
- Prototipo de interfaz: Prototipo de interfaz si aplica.

Se muestran a continuación las Historias de Usuarios de prioridad alta, el resto se describen en el Anexo 1 del presente documento:

Historia de Usuario	
Número: 1	**Nombre de Historia de Usuario:** Autenticar usuario en el servidor mediante el método Basic sobre el protocolo HTTP.
Usuario: Claudia C. Alvarez	**Iteración Asignada:** 1
Prioridad en negocio: Alta	**Puntos estimados:** 1
Riesgo en Desarrollo: Alto	**Puntos Reales:** 1
Descripción: Se envían el usuario y la contraseña en texto plano para que sean verificados por el servidor, en caso de ser positiva la respuesta del mismo se accede al recurso solicitado.	
Observaciones:	
Prototipo de interfaz:	

Tabla 5 HU: Autenticar usuario en el servidor mediante el método Basic sobre el protocolo HTTP.

Historia de Usuario	
Número: 4	**Nombre de Historia de Usuario:** Hacer peticiones al servidor vía HTTP-GET.
Usuario: Claudia C. Alvarez	**Iteración Asignada:** 1
Prioridad en negocio: Alta	**Puntos estimados:** 1
Riesgo en Desarrollo: Alto	**Puntos Reales:** 1

Descripción: El componente a través de comandos Get se conecta al servidor y le realiza una petición para que este le envíe el código HTML del servicio solicitado. Entonces, si los comandos enviados son correctos, el servidor web accede a enviarle el código HTML, el cual es manipulado por el componente para que pueda ser legible para el usuario.
Observaciones:
Prototipo de interfaz:

Tabla 6 HU: Hacer peticiones al servidor vía HTTP-GET.

Historia de Usuario		
Número: 6	**Nombre de Historia de Usuario:** Gestionar datos de la respuesta.	
Usuario: Claudia C. Alvarez		**Iteración Asignada:** 1
Prioridad en negocio: Alta		**Puntos estimados:** 1
Riesgo en Desarrollo: Alto		**Puntos Reales:** 1
Descripción: Los datos que se recogen cuando el cliente se autentica en el servidor deben ser gestionados para su entrega al cliente con la menor inconsistencia posible.		
Observaciones:		
Prototipo de interfaz:		

Tabla 7 HU: Gestionar datos de la respuesta.

2.4 Planificación

En la fase de planificación de la metodología XP es donde se le concede la prioridad a cada Historia de Usuario, describiendo sus características y funcionalidades requeridas para el desarrollo del software. Durante la planeación se realiza una estimación del esfuerzo que costará implementar cada Historia de Usuario, este se expresa utilizando como medida el punto, el cual es considerado como una semana de trabajo, donde el equipo de desarrollo labora a tiempo completo. Esta estimación incluye todo el esfuerzo asociado a la implementación de la Historia de Usuario. (16)

2.4.1 Estimación de esfuerzo por Historia de Usuario.

A continuación se muestra la estimación del esfuerzo por cada Historia de Usuario descrita anteriormente:

Historia de usuario	Estimación
Autenticar usuario en el servidor mediante el método Basic sobre el protocolo HTTP.	1
Autenticar usuario en el servidor mediante el método Digest sobre el protocolo HTTP.	1
Autenticar usuario en el servidor a través del protocolo HTTPS con los métodos Digest o Basic.	1
Hacer peticiones al servidor vía HTTP-GET.	1
Hacer peticiones al servidor vía HTTP-POST.	1
Gestionar datos de la respuesta.	1
Gestionar Certificados.	1
Manejar peticiones asincrónicas mediante el componente.	1

Tabla 8: Estimación de esfuerzo por Historias de Usuarios.

2.4.2 Plan de iteraciones.

Luego de ser identificadas y descritas las Historias de Usuarios, se procede a la planificación de la fase de implementación, donde se realizarán dos iteraciones, las cuales se describen a continuación:

Iteración 1

Esta iteración tiene como objetivo la implementación de las Historias de Usuarios 1, 4 y 6 de prioridad alta. Esta iteración será el inicio de las nuevas funcionalidades para el componente que el cliente necesita.

Iteración 2

En esta iteración se tendrán en cuenta las Historias de Usuarios 2, 3, 5, 7 y 8 que facilitarán un mejor funcionamiento del componente. Concluida esta iteración el producto estará listo para ser evaluado y puesto en práctica en un entorno real.

2.4.3 Plan de duración de iteraciones.

Este plan se encarga de mostrar las Historias de Usuarios en el orden en que se implementarán en cada iteración, así como la duración estimada.

Iteración	Orden de las HU a implementar	Duración total
1	Autenticar usuario en el servidor mediante el método Basic sobre el protocolo HTTP.	3 semanas
	Hacer peticiones al servidor vía HTTP-GET.	
	Gestionar datos de la respuesta.	
2	Autenticar usuario en el servidor mediante el método	5 semanas

	Digest sobre el protocolo HTTP.	
	Autenticar usuario en el servidor a través del protocolo HTTPS con los métodos Digest o Basic.	
	Hacer peticiones al servidor vía HTTP-POST.	
	Gestionar Certificados.	
	Manejar peticiones asincrónicas mediante el componente.	

Tabla 9: Plan de duración de iteraciones.

2.4.4 Plan de entregas.

En este plan se detalla la fecha fin de cada iteración, mostrando una versión desarrollada del producto en ese momento hasta lograr el producto final en la fecha establecida. A continuación se muestra el plan de entrega para cada iteración:

FPA: Funcionalidades con prioridad alta.
FPM: Funcionalidades con prioridad media.

	Fin de la iteración 1 (13 de marzo de 2013)	**Fin de la iteración 2 (8 de mayo de 2013)**
Componente de identificación en servidores de aplicación para aplicaciones basadas en Android.	FPA	FPM

Conclusiones

En este capítulo quedó claramente definida la propuesta del componente la cual puntualiza que el resultado será un mecanismo que ofrezca servicios de autenticación en servidores de aplicaciones web haciendo uso del teléfono celular. Se identificaron las funcionalidades y los requisitos que el mismo debe cumplir, lo que contribuyó a generar una idea clara del producto final permitiendo que el mismo cuente con la calidad requerida por el cliente. Además, se realizó la descripción de las Historias de Usuarios divididas por iteraciones y la planificación del esfuerzo dedicado al desarrollo de cada una de ellas, en el orden en que se les dará cumplimiento ofreciendo una mejor organización del trabajo y permitiendo establecer fechas de culminación para cada iteración.

CAPÍTULO 3: ANÁLISIS Y DISEÑO DEL COMPONENTE

3. Introducción

En este capítulo se describe la fase de diseño del componente propia de la metodología XP. Se define el patrón de arquitectura a seguir, así como los patrones de diseño a utilizar para la implementación del componente y quedarán definidas las tarjetas CRC[8] como técnica de diseño.

1.1 Arquitectura.

La arquitectura de software consiste en el diseño de componentes de una aplicación, generalmente utilizando patrones de arquitectura. Un diseño arquitectónico describe en general cómo se construirá una aplicación de software.

1.1.2 Estilo arquitectónico: Arquitectura en Capas.

Partiendo de la existencia de múltiples estilos arquitectónicos se decidió utilizar el estilo: Arquitectura en Capas (Arquitectura en tres capas) perteneciente a la familia de estilos de Llamada y Retorno, basándose en las características y peculiaridades del componente a desarrollar, ya que se enfoca en la distribución de roles y responsabilidades de forma jerárquica proveyendo una forma muy efectiva de separación de responsabilidades.

Los principales beneficios del estilo de arquitectura son:

- Aislamiento. El estilo de arquitectura de capas permite asilar los cambios en tecnologías a ciertas capas para reducir el impacto en el componente total.
- Rendimiento. Distribuir las capas entre múltiples sistemas (físicos) puede incrementar la escalabilidad, la tolerancia a fallos y el rendimiento.
- Mejoras en Pruebas. La capacidad de realizar pruebas se beneficia de tener una interfaces bien definidas para cada capa así como de la habilidad para cambiar a diferentes implementaciones de las interfaces de cada capa.

[8] **CRC**: Colaborador - Responsabilidad - Clase

- Alta cohesión. Cada capa contiene funcionalidad directamente relacionas con la tarea de dicha capa.
- Reutilizable. Las capas inferiores no tienen ninguna dependencia con las capas superiores, permitiéndoles ser reutilizables en otros escenarios. (18)

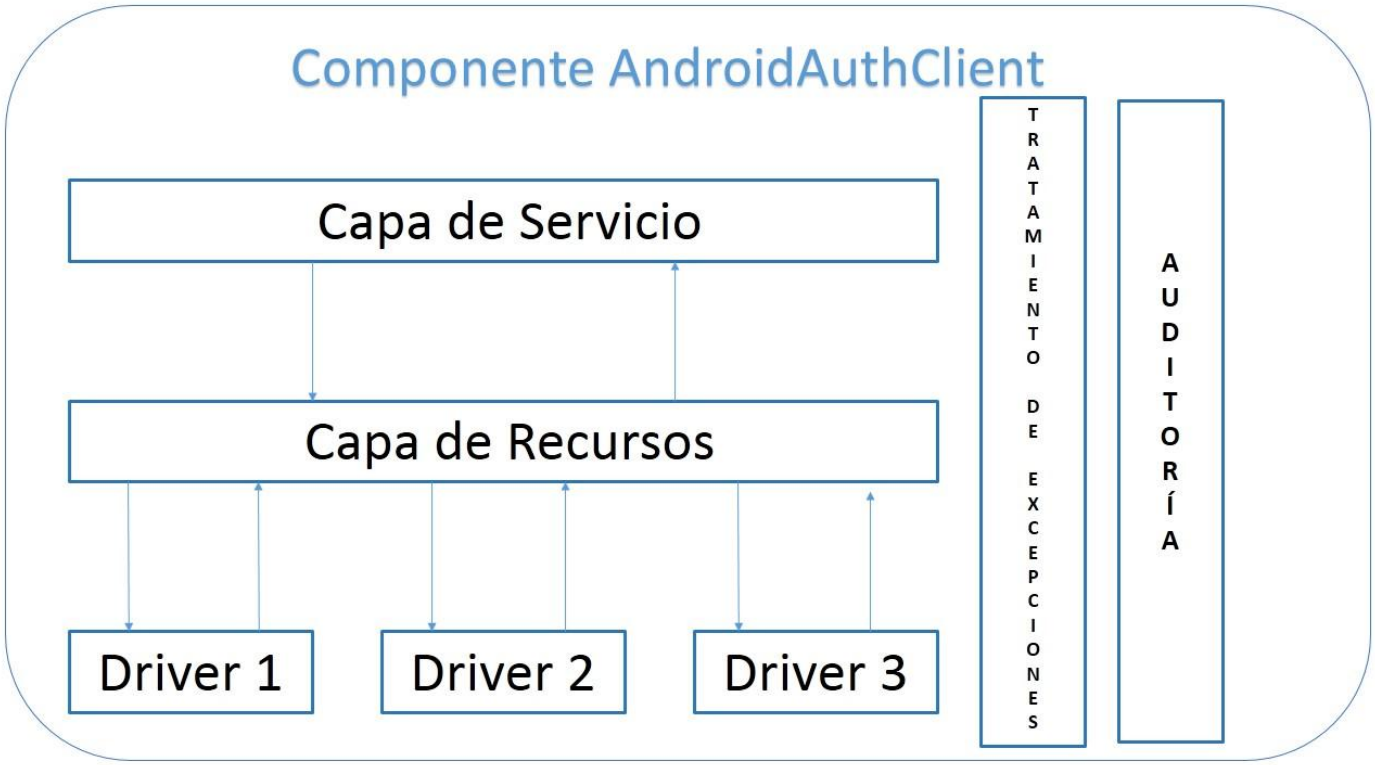

Figura 2: Arquitectura en capas.

A continuación se realiza una breve descripción de cada una de las capas por las que está compuesto el componente de acuerdo a la figura anterior.

La capa de servicio: es la encargada de interactuar con el usuario y se comunica únicamente con la capa de negocio proporcionándole la información necesaria para gestionar la conexión con el servidor.

La capa de recursos: esta capa recibe la petición del usuario a través de la capa de servicio y se encarga de darle curso.

La capa de drivers: define los protocolos de comunicación que pueden utilizarse para el envío de las peticiones y la recepción de la respuesta del servidor.
Tratamiento de excepciones: permite la gestión de excepciones a través de todas las capas del componente.

Auditoría: brinda información a los desarrolladores sobre las acciones que realiza el componente a través del almacenamiento de logs.

1.2 Patrones de diseño.

Un patrón de diseño es una buena práctica documentada, o solución, que se ha aplicado con éxito en ambientes múltiples para solucionar un problema que se repite en los sistemas de dichos ambientes y en situaciones específicas.

3.2.1 Patrones para asignar responsabilidad.

Los patrones GRASP[9] describen los elementos esenciales de la asignación de responsabilidades a objetos, donde son expresados en forma de patrones.
Se emplearon en el componente los siguientes patrones GRASP (19):

Experto
Este patrón se encarga de asignar una responsabilidad al experto en información, o sea, aquella clase que cuenta con la información necesaria para cumplir la responsabilidad. En la solución las clases del negocio realizan las validaciones correspondientes a las peticiones de los usuarios.

Creador
Este patrón es el responsable de asignarle a una clase la responsabilidad de crear una instancia de otra. En la solución las clases de la capa de servicio contienen una instancia de la capa de recursos.

[9] **GRASP**: General Responsibility Assignment Software Patterns o Responsabilidad de la asignación general Patrones de Software.

Controlador

Este patrón asigna la responsabilidad del manejo de los eventos de un sistema a una o varias clases. Cada petición que se genera es manejada por una clase controladora, ésta invoca a las clases de su negocio que necesita para cumplir con la petición, recibe la respuesta y ofrece una respuesta para mostrar finalmente la respuesta al usuario.

Alta cohesión

La cohesión es la medida de la fuerza que une a las responsabilidades de una clase. Una clase con alta cohesión mejora la claridad y la facilidad de su uso, su mantenimiento se simplifica y es fácil de reutilizar. A menudo, se genera un bajo acoplamiento. El componente fue diseñado de manera tal que las clases pertenecientes a las diferentes capas se le asignaran las responsabilidades necesarias y bien delimitadas para garantizar una alta cohesión.

Bajo acoplamiento

Consiste en tener las clases lo menos ligadas entre sí que se pueda. De tal forma que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de clases, potenciando la reutilización y disminuyendo la dependencia entre las clases.

3.2.2 Patrones GOF[10].

Fachada (Facade)

Con este patrón se proporciona una interfaz unificada de alto nivel que, representando a todo un subsistema, facilita su uso. La “fachada” satisface a la mayoría de los clientes, sin ocultar las funciones de menor nivel a aquellos que necesiten acceder a ellas. El patrón fachada es usado en diferentes situaciones, entre ellas cuando se pretende proporcionar una interfaz simple para un subsistema complejo, o cuando es necesario separar al cliente de los componentes de un subsistema, reduciendo así el número de objetos con los que el cliente trata, y facilitando el uso del subsistema (20). En el componente de autenticación se emplea este patrón en la capa de servicio.

[10] GOF: Gang of Four o Grupo de los cuatro (Creadores: Erich Gamma, Richard Helm, Ralph Johnson y John Vlisides).

3.3 Tarjetas Clase – Responsabilidad – Colaborador (CRC).

En la fase de diseño de la metodología XP se propone el modelado de Clase-Responsabilidad-Colaborador (CRC), lo que permite un modelo simple para organizar las clases más relevantes en el componente. El objetivo de este modelado es utilizar tarjetas para la representación organizada de las clases, las cuales tienen los siguientes datos:

- Nombre.
- Responsabilidades.
- Colaboraciones.

Una responsabilidad es lo que sabe o hace la clase. Los colaboradores son aquellas clases que se requieren para que una clase reciba la información necesaria para completar una responsabilidad. Las tarjetas determinan el comportamiento de cada actividad.

Para una primera versión del componente se identificaron 11 tarjetas CRC. A continuación se muestran algunas de estas tarjetas CRC y el resto se especifican en el Anexo 2 del presente documento.

Clase AuthClientException	
Responsabilidad	Colaborador
Gestionar el manejo de excepciones.	

Tabla 10: Clase AuthClientException.

Clase AuthenticatorServices	
Responsabilidad	Colaborador
Interfaz que proporciona la comunicación del componente.	ContentResponse UserRequest

Tabla 11: Clase AuthenticatorServices.

Conclusiones

En este capítulo se realizó un análisis sobre el patrón arquitectónico utilizado para el desarrollo del componente, así como el estudio de los patrones de diseños empleados para obtener la solución, estos pasos permitieron determinar cómo guiar organizadamente el proceso de implementación del software con el objetivo de obtener un producto final consistente y funcional. También quedaron definidos las tarjetas CRC, lo que garantizó la obtención de una representación organizada del componente resultante.

CAPÍTULO 4: IMPLEMENTACIÓN

1. Introducción

La metodología XP plantea que la implementación de un software debe realizarse de forma iterativa, obteniendo al culminar cada iteración un producto funcional que debe ser probado y mostrado al cliente para retroalimentar a los desarrolladores con su opinión. En el presente capítulo se describen las iteraciones realizadas correspondientes a la etapa de implementación del componente, así como las pruebas de aceptación efectuadas sobre el mismo y las tareas de ingeniería generadas en cada una de estas iteraciones.

1.1 Fase de Implementación.

Durante el transcurso de las iteraciones se realiza la implementación de las historias de usuario seleccionadas para cada una de éstas. Al inicio de las mismas, se lleva a cabo una revisión del plan de iteraciones y se modifica de ser necesario. Como parte de este plan, se descomponen las HU en tareas de desarrollo, asignando posteriormente cada una de éstas a un equipo (o una persona) responsable de su implementación. Estas tareas son para el uso de los programadores, pueden escribirse utilizando un lenguaje técnico y no necesariamente deben ser entendibles para el cliente. (21)

1.1.1 Iteraciones.

Ajustándose a la planificación realizada, se llevaron a cabo dos iteraciones de desarrollo sobre el componente, obteniéndose al finalizar, un producto listo para su puesta en funcionamiento. A continuación se detallan cada una de las iteraciones.

1.1.1.1 Iteración 1.

Tareas de Ingeniería.

A continuación se muestran las tareas de ingeniería correspondientes a las historias de usuario de prioridad alta abordadas en esta iteración:

Tareas de ingeniería	
No. Tarea: 1	No. Historia de Usuario: 1
Nombre de la Tarea: Autenticar usuario en el servidor mediante el método Basic sobre el protocolo HTTP.	
Tipo de Tarea: Desarrollo	Puntos de estimación: 1/1
Fecha de Inicio: 13/03/2013	Fecha de Fin: 20/03/2013
Programador Responsable: Claudia C. Alvarez Campos	
Descripción: Se implementarán las funcionalidades necesarias para que el componente envíe las credenciales del usuario con el objetivo de que sean verificadas por el servidor, y en caso de ser positiva la respuesta el sistema accederá al recurso solicitado.	

Tabla 12: Tarea # 1 Autenticar usuario en el servidor mediante el método Basic sobre el protocolo HTTP.

Tareas de ingeniería	
No. Tarea: 2	No. Historia de Usuario: 4
Nombre de la Tarea: Hacer peticiones al servidor vía HTTP-GET.	
Tipo de Tarea: Desarrollo	Puntos de estimación: 1/1
Fecha de Inicio: 21/03/2013	Fecha de Fin: 27/03/2013
Programador Responsable: Claudia C. Alvarez Campos	

Descripción: Se implementarán las funcionalidades necesarias para que el componente se conecte al servidor a través de peticiones Get y solicite el código HTML perteneciente al servicio que se requiera consumir. En caso de recibir una respuesta dicho código el componente deberá transformarlo para que pueda ser legible para el usuario.

Tabla 13: Tarea # 2 Hacer peticiones al servidor vía HTTP-GET.

Tareas de ingeniería	
No. Tarea: 3	No. Historia de Usuario: 6
Nombre de la Tarea: Gestionar datos de la respuesta.	
Tipo de Tarea: Desarrollo	Puntos de estimación: 1/1
Fecha de Inicio: 28/03/2013	Fecha de Fin: 3/04/2013
Programador Responsable: Claudia C. Alvarez Campos	
Descripción: Se implementarán las funcionalidades necesarias para que el componente gestione los datos que se recogen cuando el cliente se autentica en el servidor y puedan ser entregados al cliente con la mayor consistencia posible.	

Tabla 14: Tarea # 3 Gestionar datos de la respuesta.

1.1.1.2 Iteración 2.

Tareas de Ingeniería.

A continuación se muestran dos de las tareas de ingeniería correspondientes a las historias de usuario de prioridad media abordadas en esta iteración, el resto se encuentran expuestas en el Anexo 3 del presente documento.

Tareas de ingeniería	
No. Tarea: 4	No. Historia de Usuario: 2
Nombre de la Tarea: Autenticar usuario en el servidor mediante el método Digest sobre el protocolo HTTP.	
Tipo de Tarea: Desarrollo	Puntos de estimación: 1/1
Fecha de Inicio: 4/04/2013	Fecha de Fin: 10/04/2013
Programador Responsable: Claudia C. Alvarez Campos	
Descripción: Se implementarán las funcionalidades necesarias para que el componente genere un Digest de las credenciales del usuario y lo envíe al servidor a través del protocolo HTTP, en caso de una respuesta positiva el componente debe acceder al servicio solicitado.	

Tabla 15: Tarea # 7 Autenticar usuario en el servidor mediante el método Digest sobre el protocolo HTTP.

Tareas de ingeniería	
No. Tarea: 5	No. Historia de Usuario: 3
Nombre de la Tarea: Autenticar usuario en el servidor a través del protocolo HTTPS con los métodos Digest o Basic.	
Tipo de Tarea: Desarrollo	Puntos de estimación: 1/1
Fecha de Inicio: 11/04/2013	Fecha de Fin: 17/05/2013
Programador Responsable: Claudia C. Alvarez Campos	

Descripción: Se implementarán las funcionalidades necesarias para que el componente pueda establecer comunicación con el servidor haciendo uso de la combinación del protocolo HTTP con la capa de conexión segura SSL, para ello deberá encriptar las credenciales del usuario por el método MD5. Además el componente deberá permitir que la utilización de HTTPS pueda aplicarse a cualquiera de los métodos de autenticación, dígase Basic o Digest.

Tabla 16: Tarea # 8 Autenticar usuario en el servidor a través del protocolo HTTPS con los métodos Digest o Basic.

Conclusiones

En este capítulo se implementaron las funcionalidades de acuerdo a las historias de usuario anteriormente definidas, obteniéndose resultados ya visibles para los clientes y gratificantes para los desarrolladores, ya que queda implementado el componente con las principales características que se definieron para cada iteración.

CAPÍTULO 5: PRUEBA

Introducción.

El proceso de pruebas es fundamental en el ciclo de vida de un software. En este los desarrolladores comprueban constantemente y tanto como sea posible el código, con el objetivo de reducir al mínimo el número de errores no detectados. Lo que contribuye a elevar la calidad de los productos desarrollados.
XP divide las pruebas en dos grupos: pruebas unitarias, desarrolladas por los programadores y encargadas de verificar el código de forma automática y las pruebas de aceptación, destinadas a evaluar si al final de una iteración se consiguió la funcionalidad requerida además de comprobar que dicha funcionalidad sea la esperada por el cliente. (21)

1.2 Validación del componente.

El proceso de validación se refiere a un conjunto diferente de actividades que aseguran que el software construido se ajusta a los requisitos del cliente. Debe ser suficientemente rígido para promover un seguimiento razonable de la planificación y la gestión a medida que progresa el proyecto (22). Para gestionar dicho proceso en el componente de autenticación en servidores web para aplicaciones basadas en Android, se llevó a cabo la implementación de 2 aplicaciones que contribuyan a probar el correcto funcionamiento del mismo; a continuación se refleja una descripción de cada una de estas aplicaciones.

Arquitectura.
Para el desarrollo de las aplicaciones de validación se decidió utilizar una arquitectura en capas distribuida como se muestra en la figura 4:

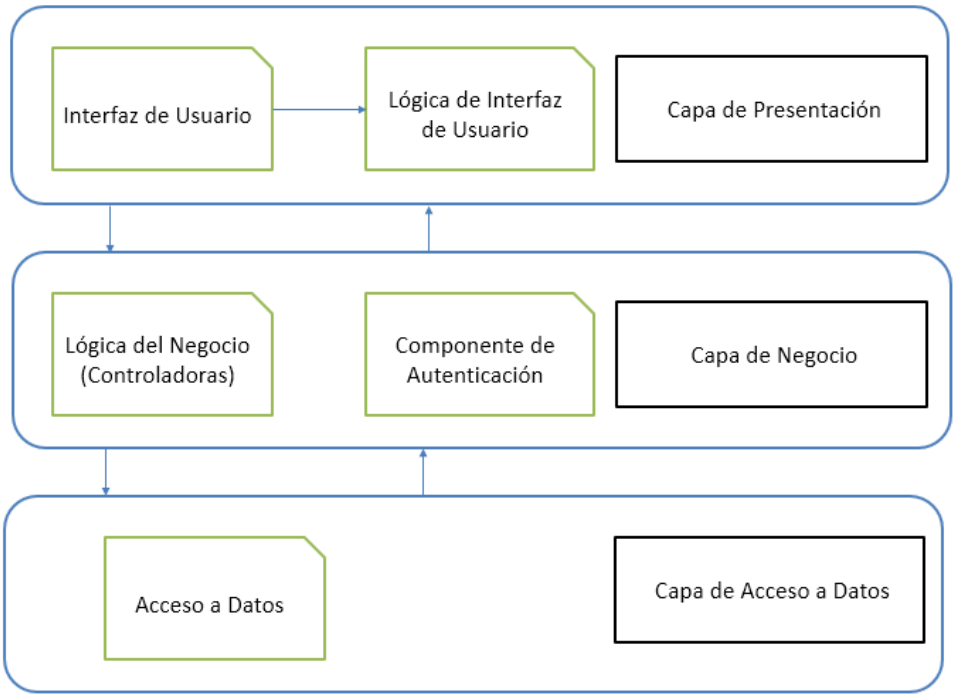

Figura 3: Arquitectura en capas del Sistema de Autenticación para validar el componente.

A continuación se realiza una breve descripción de cada una de las capas por las que está compuesto el sistema de acuerdo a la figura anterior.

La capa de presentación: es la encargada de interactuar con el usuario y se corresponde con lo que tradicionalmente se conoce como interfaz de usuario, esta capa se comunica únicamente con la capa de negocio.

En la capa de negocio o aplicación o intermedia: es donde se localiza la lógica del negocio. Esta capa recibe la petición del usuario a través de la capa de presentación y se encarga de darle curso, dicha capa es donde se implementan las reglas del negocio, las validaciones y cálculos.

La capa de acceso a datos: es la encargada de acceder a los repositorios de información, ejemplo de estas bases de datos.

1.2.1 Aplicación para validar el componente: Sistema de Autenticación.

Propuesta del sistema.

Se propone la puesta en marcha de un sistema que en conjunto con el componente desarrollado ofrezca servicios de autenticación de usuarios en servidores web haciendo uso de su teléfono celular. De esta manera se decidió la implementación de una aplicación para el sistema operativo Android que permita a los usuarios autenticarse a través de dispositivos móviles, con este fin el sistema ofrecerá la posibilidad de seleccionar los parámetros con los cuales se establecerá la conexión con el servidor de aplicaciones, Ver figura 3:

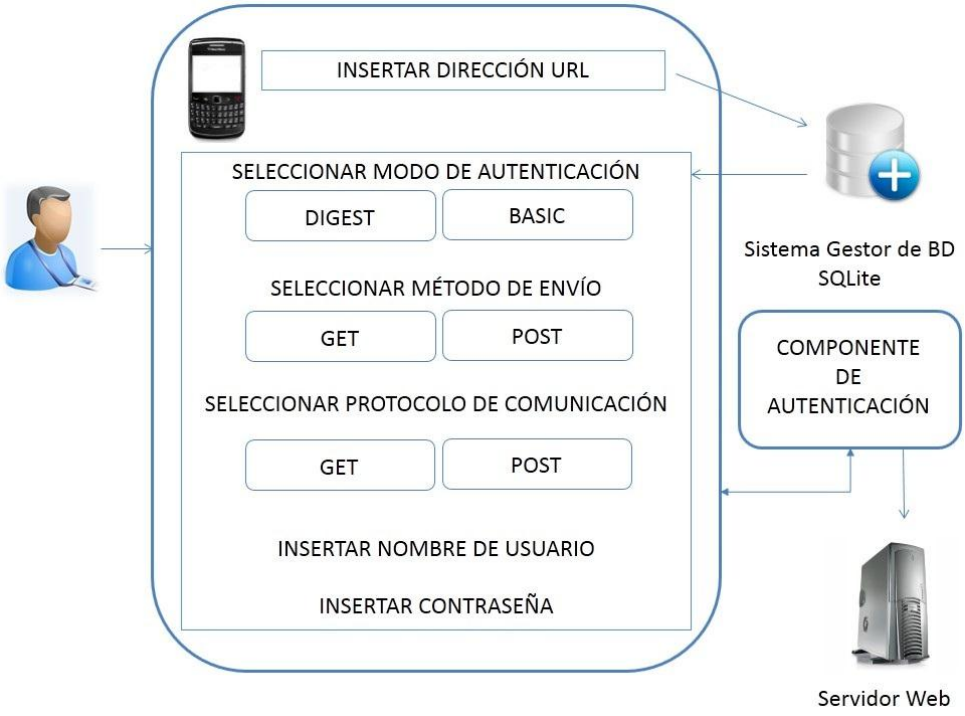

Figura 4: Propuesta del Sistema de Identificación para la validación del componente.

1.2.1.1 Aplicación para validar el componente: Generador de reportes.

Propuesta del sistema.

Se propone la elaboración de una aplicación que complementada con el componente de identificación, permita generar reportes desde el teléfono celular sobre los equipos pertenecientes a la residencia de la Universidad de las Ciencias Informáticas, además el usuario podrá verificar si la situación reflejada en el reporte ha sido solucionada. Para comunicarse con el servidor la aplicación utiliza como modo de autenticación Basic y HTTPS como protocolo de comunicación para garantizar la protección de la información que viaja a través de la red, Ver figura 4.

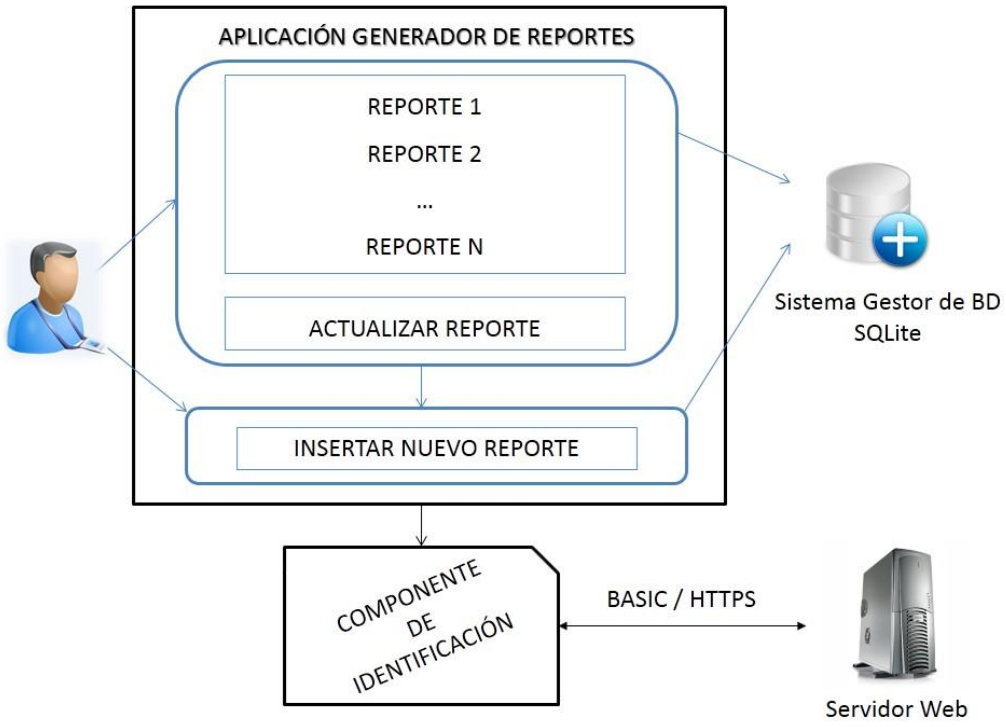

Figura 5: Propuesta del sistema Generador de Reportes.

1.2.2 Pruebas unitarias.

Las pruebas unitarias son una actividad fundamental en la metodología XP, pues asegura que un determinado módulo cumpla con un comportamiento esperado en forma aislada antes de ser integrado al sistema. Estas pruebas brindan al programador una inmediata retroalimentación de cómo está realizando su trabajo, permitiéndole realizar cambios de forma segura respaldado por efectivos casos de prueba.
Las pruebas unitarias fueron empleadas durante todo el proceso de desarrollo del componente, en cada una de las funcionalidades implementadas, probando el código con casos de prueba creados por el equipo de desarrollo y verificando si los resultados brindados eran los esperados para cada funcionalidad, a continuación se muestra una imagen que refleja los resultados algunas de las pruebas realizadas.

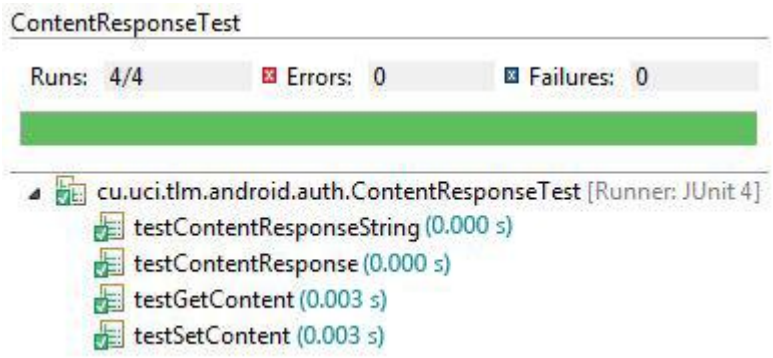

Figura 6: Resultados de las pruebas unitarias realizadas a las funcionalidades de la clase ContentResponse.

1.2.3 Pruebas de aceptación.

Las pruebas de aceptación son pruebas de caja negra que se crean a partir de las historias de usuario. Su principal objetivo es validar que el componente cumple con el funcionamiento esperado y permitir al cliente determinar su aceptación, desde el punto de vista de su funcionalidad y rendimiento. Una historia de usuario no se considera lista hasta que haya pasado todas sus pruebas de aceptación.
Las pruebas de aceptación correspondiente a cada una de las funcionalidades serán representadas mediante tablas divididas por las secciones siguientes:

- Código de la prueba de aceptación.

- Número de la historia de usuario a la que se le realiza la prueba.
- Nombre de la funcionalidad.
- Descripción de la funcionalidad.
- Condiciones de ejecución de la funcionalidad.
- Entrada y pasos de ejecución que realiza el usuario con el objetivo de obtener el resultado esperado.
- Resultado esperado.
- Evaluación de la prueba.

A continuación se muestran tres de los casos de prueba de aceptación correspondientes a las historias de usuario de prioridad alta, el resto se encuentran expuestas en el Anexo 4 del presente documento.

Caso de prueba de aceptación	
Código: HU1-P1	**No. Historia de Usuario:** 1
Nombre: Autenticar usuario en el servidor mediante el método Basic sobre el protocolo HTTP.	
Descripción: Prueba para la funcionalidad Autenticar usuario en el servidor mediante el método Basic sobre el protocolo HTTP.	
Condiciones de ejecución: • El usuario deberá seleccionar como modo de conexión: Basic. • El usuario deberá seleccionar como protocolo de comunicación: HTTP. • El usuario deberá escoger la opción: Conectarse.	
Entrada/Pasos de ejecución: El usuario selecciona entre los parámetros HTTP como protocolo de comunicación y Basic como modo de conexión, el componente conforma la solicitud y la envía al servidor. Se recoge la respuesta y se muestra al usuario el resultado.	
Resultado esperado: El mensaje de respuesta es entregado sin errores.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 17 HU1-P1: Autenticar usuario en el servidor mediante el método Basic sobre el protocolo HTTP.

Caso de prueba de aceptación	
Código: HU4-P4	**No. Historia de Usuario:** 4
Nombre: Hacer peticiones al servidor vía HTTP-GET.	
Descripción: Prueba para la funcionalidad Hacer peticiones al servidor vía HTTP-GET.	
Condiciones de ejecución: • El usuario deberá seleccionar como protocolo de comunicación: HTTP. • El usuario deberá seleccionar como método de envío: GET.	
Entrada/Pasos de ejecución: El usuario selecciona entre los parámetros HTTP como protocolo de comunicación y GET como método de envió, el componente conforma la solicitud y la envía al servidor. Se recoge la respuesta y se muestra al usuario el resultado.	
Resultado esperado: El mensaje de respuesta es entregado sin errores.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 18 HU4-P4: Hacer peticiones al servidor vía HTTP-GET.

Caso de prueba de aceptación	
Código: HU6-P6	**No. Historia de Usuario:** 6
Nombre: Gestionar datos de la respuesta.	
Descripción: Prueba para la funcionalidad Gestionar datos de la respuesta.	
Condiciones de ejecución:	

• El usuario debe introducir los parámetros de conexión y seleccionar la opción: Conectarse.
Entrada/Pasos de ejecución: El usuario introduce los parámetros de comunicación, el componente conforma la solicitud y la envía al servidor. El componente construye una la respuesta consistente a partir de los datos devueltos por el servidor y se muestra al usuario el resultado.
Resultado esperado: El mensaje de respuesta es entregado sin errores.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 19 HU6-P6: Gestionar datos de la respuesta.

Conclusiones

Para lograr una mayor calidad y eficacia, se describieron los principales tipos de prueba utilizadas en el proceso, lo que permitió tener una visión clara de cómo fue controlado todo el proceso de desarrollo del componente con el fin de obtener un producto con la calidad requerida. Además se llevó a cabo la concepción de dos aplicaciones que permitieron la validación del componente obteniendo resultados satisfactorios.

CONCLUSIONES GENERALES

Con el desarrollo de este trabajo se profundizó en el conocimiento de los sistemas de identificación, así como los mecanismos necesarios para su implementación. Se aplicó la metodología XP para guiar el proceso de desarrollo de software, mediante el cual se obtuvo un componente para la identificación en servidores de aplicación de teléfonos celulares que integren el sistema operativo Android. Se puede concluir que se ha cumplido satisfactoriamente el objetivo trazado para este trabajo, enfatizando en los siguientes puntos:

- Fueron definidas las políticas de seguridad para garantizar la transmisión segura de la información en el proceso de intercambio de información durante la conexión con el servidor.

- Fue implementado el componente para realizar la identificación de usuarios en servidores de aplicación utilizando el teléfono celular con sistema operativo Android.

- Se realizaron las pruebas necesarias para comprobar el correcto funcionamiento de las funcionalidades del componente.

- Fueron implementadas dos aplicaciones que permitieron validar el componente haciendo uso de sus funcionalidades.

RECOMENDACIONES

- ✓ Incorporar al componente nuevos protocolos de comunicación para interactuar con los servidores de aplicación.
- ✓ Adicionar nuevos modos de autenticación y métodos de envío de datos para el proceso de identificación en servidores de aplicación.

REFERENCIAS BIBLIOGRÁFICAS

1. Privacy International. [En línea] 2012. https://www.privacyinternational.org/reports/una-guia-de-privacidad-para-hispanohablantes/autenticacion-y-verificacion-de-la-identidad.

2. Definición ABC. [En línea] http://www.definicionabc.com/.

3. César Medina Salgado, Profesor Investigador de la UNIVERSIDAD AUTONOMA METROPOLITANA. Los Sistemas Automáticos de Identificación. [En línea] http://www.azc.uam.mx/publicaciones/enlinea2/num1/1-1.htm.

4. EcuRed. [En línea] http://www.ecured.cu/index.php/Protocolo_de_Transferencia_de_Hipertexto.

5. Seguridad de la información. [En línea] http://seguridadenlainformacionylasredes1.bligoo.com/tag/httpyhttps.

6. Patxi Echarte. esloMas.com. [En línea] http://www.eslomas.com/2006/10/funcionamiento-autenticacion-http-basic/.

7. Universidad de Navarra, servicios informáticos. [En línea] http://www.unav.es/SI/servicios/seguridad/certifica8.html.

8. DiarioTi.com. [En línea] 6 de 06 de 2013. http://diarioti.com/eset-presenta-nuevo-sistema-de-autenticacion-con-doble-factor-y-contrasena-de-un-solo-uso-para-redes-corporativas/63916?lang=es.

9. Scoop.it. [En línea] 28 de 2 de 2012. http://www.scoop.it/t/estandares-de-desarrollo-jee/?tag=Autenticaci%C3%B3n.

10. Vilchez, Angel. Configurar Equipos. *Que es Android: Características y Aplicaciones.* [En línea] 2 de 4 de 2009. http://www.configurarequipos.com/doc1107.html.

11. Aplicaciones.org. *Eclipse: IDE de desarrollo Open Source* . [En línea] 5 de 10 de 2011. http://aplicaciones.org/eclipse-ide-de-desarrollo-open-source/.

12. WebTaller. [En línea] 2011-2013. http://www.webtaller.com/manual-java/caracteristicas-java.php.

13. SQLite. [En línea] http://www.sqlite.org/.

14. EcuRed. [En línea] http://www.ecured.cu/index.php/Servidor_HTTP_Apache.

15. Programación Extrema. [En línea] www.programacionextrema.org/articulos/newMethodology.es.html#tth_sEc5.1.

16. Kent Beck, Cynthia Andres. *Extreme Programming Explained. Embrace Change.* 1999.

17. Pekka Abrahamsson, Outi Salo, Jussi Ronkainen. *Agile Software Development Methods.* 2002.

18. Peláez, Juan. Arquitectura en Capas. [En línea] 2 de 6 de 2009. http://geeks.ms/blogs/jkpelaez/archive/2009/05/29/arquitectura-basada-en-capas.aspx.

19. Andrés Grosso. Práacticas de Software. [En línea] 21 de 3 de 2011. http://www.practicasdesoftware.com.ar/2011/03/patrones-grasp/.

20. EcuRed. [En línea] http://www.ecured.cu/index.php/Facade.

21. Lisa Crispin, Tip House. *Testing Extreme Programming .* 2003.

22. [En línea] http://arco.esi.uclm.es/~david.villa/pensar_en_C++/vol1/ch01s10.html.

ANEXOS

Anexo 1

Historias de Usuario de prioridad media.

Historia de Usuario	
Número: 2	**Nombre de Historia de Usuario:** Autenticar usuario en el servidor mediante el método Digest sobre el protocolo HTTP.
Usuario: Claudia C. Alvarez	**Iteración Asignada:** 2
Prioridad en negocio: Media	**Puntos estimados:** 1
Riesgo en Desarrollo: Alto	**Puntos Reales:** 1
Descripción: El sistema generará a través de una función hash un entremezclado irreversible de las credenciales del usuario, posteriormente lo adjuntará a un token llamado nonce que es proporcionado por el receptor de la comunicación. La combinación de estos elementos conforma el resultado del proceso de encriptación, el cual es enviado al servidor a través del protocolo HTTP y a partir de la respuesta de este accede o no al servicio solicitado.	
Observaciones:	
Prototipo de interfaz:	

Tabla 20 HU: Autenticar usuario en el servidor mediante el método Digest sobre el protocolo HTTP.

Historia de Usuario	
Número: 3	**Nombre de Historia de Usuario:** Autenticar usuario en el servidor a través del protocolo HTTPS con los métodos Digest o Basic.

Usuario: Claudia C. Alvarez	**Iteración Asignada:** 2
Prioridad en negocio: Media	**Puntos estimados:** 1
Riesgo en Desarrollo: Alto	**Puntos Reales:** 1
Descripción: El componente potencia la opción de combinar el protocolo HTTP con la capa de conexión segura SSL con el fin de crear un canal apropiado para el tráfico de información sensible, los datos del usuario viajaran encriptados por el método RSA[11] el cual basa su funcionamiento en el producto de dos números primos grandes elegidos al azar y mantenidos en secreto. La utilización de HTTPS podrá aplicarse a cualquiera de los métodos de autenticación, dígase Basic o Digest.	
Observaciones:	
Prototipo de interfaz:	

1 Tabla 21 HU: Autenticar usuario en el servidor a través del protocolo HTTPS con los métodos Digest o Basic.

2

Historia de Usuario		
Número: 5	**Nombre de Historia de Usuario:** Hacer peticiones al servidor vía HTTP-POST.	
Usuario: Claudia C. Alvarez		**Iteración Asignada:** 2
Prioridad en negocio: Media		**Puntos estimados:** 1
Riesgo en Desarrollo: Alto		**Puntos Reales:** 1
Descripción: El sistema tendrá la opción de utilizar peticiones de tipo Post en el momento		

[11] **RSA:** Algoritmo descrito en 1977 por Ron Rivest, Adi Shamir y Len Adleman, del Instituto Tecnológico de Massachusetts.

que necesite enviar al servidor datos que se deban ser interpretados por este, es decir está diseñado para solicitar que un servidor web acepte los datos adjuntos en el cuerpo del mensaje de solicitud.
Observaciones:
Prototipo de interfaz:

Tabla 22 HU: Hacer peticiones al servidor vía HTTP-POST.

Historia de Usuario	
Número: 7	**Nombre de Historia de Usuario:** Gestionar Certificados.
Usuario: Claudia C. Alvarez	**Iteración Asignada:** 2
Prioridad en negocio: Media	**Puntos estimados:** 1
Riesgo en Desarrollo: Medio	**Puntos Reales:** 1
Descripción: Gestionar los certificados con que se va a cifrar la información que se transmite al servidor cuando la comunicación se establece a través de HTTPS.	
Observaciones:	
Prototipo de interfaz:	

Tabla 23 HU: Gestionar Certificados.

Historia de Usuario

Número: 8	**Nombre de Historia de Usuario:** Manejar peticiones asincrónicas mediante el componente.
Usuario: Claudia C. Alvarez	**Iteración Asignada:** 2
Prioridad en negocio: Media	**Puntos estimados:** 1
Riesgo en Desarrollo: Alto	**Puntos Reales:** 1
Descripción: El componente permitirá hacer peticiones asíncronas al servidor.	
Observaciones:	
Prototipo de interfaz:	

Tabla 24 HU: Manejar peticiones asincrónicas mediante el componente.

Anexo 2

Tarjetas CRC

Clase AuthenticationMethods	
Responsabilidad	Colaborador
Enumerar los métodos que podrán emplearse en el proceso de autenticación.	

Tabla 25: Clase AuthenticationMethods.

Clase AuthenticatorServicesImpl	
Responsabilidad	Colaborador
Implementación de la interfaz AuthenticatorServices para establecer la comunicación con la capa de recursos.	AuthenticatorServices Connector

	ContentResponse HttpConnector UserRequest

Tabla 26: Clase AuthenticatorServicesImpl.

Clase Connector	
Responsabilidad	Colaborador
Establece los atributos y métodos necesarios que debe implementar cada conector en el momento de establecer una comunicación con un recurso.	AuthClientException ContentResponse UserRequest

Tabla 27: Clase Connector.

Clase ContentResponse	
Responsabilidad	Colaborador
Maneja los datos contenidos en la respuesta.	

Tabla 28: Clase ContentResponse.

Clase EasySSLSocketFactory	
Responsabilidad	Colaborador
Acepta los certificados del servidor para conexiones basadas en SSL.	EasyX509TrustManager

Tabla 29: Clase EasySSLSocketFactory.

Clase EasyX509TrustManager	
Responsabilidad	Colaborador

Acepta certificados con firma personal.	

Tabla 30: Clase EasyX509TrustManager.

Clase HttpConnector	
Responsabilidad	Colaborador
Implementa la comunicación mediante el protocolo HTTP.	Connector EasySSLSocketFactory HttpMethods AuthClientException AuthenticationMethods ContentResponse UserRequest

Tabla 31: Clase HttpConnector.

Clase HttpMethods	
Responsabilidad	Colaborador
Enumerar los métodos de envío de los parámetros de comunicación.	

Tabla 32: Clase HttpMethods.

Clase UserRequest	
Responsabilidad	Colaborador

Maneja los datos de la petición efectuada por el cliente.	HttpMethods AuthenticationMethods

Tabla 33: Clase UserRequest.

Anexo 3

Tareas de Ingeniería: 2da Iteración.

Tareas de ingeniería	
No. Tarea: 6	No. Historia de Usuario: 5
Nombre de la Tarea: Hacer peticiones al servidor vía HTTP-POST.	
Tipo de Tarea: Desarrollo	Puntos de estimación: 1/1
Fecha de Inicio: 18/04/2013	Fecha de Fin: 24/04/2013
Programador Responsable: Claudia C. Alvarez Campos	
Descripción: Se implementarán las funcionalidades necesarias para que el componente pueda hacer peticiones de tipo Post en el momento que necesite enviar al servidor datos que se deban ser interpretados por este.	

Tabla 34: Tarea # 4 Hacer peticiones al servidor vía HTTP-POST.

Tareas de ingeniería	
No. Tarea: 7	No. Historia de Usuario: 7
Nombre de la Tarea: Gestionar Certificados.	

Tipo de Tarea: Desarrollo	Puntos de estimación: 1/1
Fecha de Inicio: 25/04/2013	Fecha de Fin: 1/05/2013
Programador Responsable: Claudia C. Alvarez Campos	
Descripción: Se implementarán las funcionalidades necesarias para que el componente gestione los certificados con que se va a cifrar la información que se transmite al servidor cuando la comunicación se establece a través de HTTPS.	

Tabla 35: Tarea # 5 Gestionar Certificados.

Tareas de ingeniería	
No. Tarea: 8	No. Historia de Usuario: 8
Nombre de la Tarea: Manejar peticiones asincrónicas mediante el componente.	
Tipo de Tarea: Desarrollo	Puntos de estimación: 1/1
Fecha de Inicio: 2/05/2013	Fecha de Fin: 8/05/2013
Programador Responsable: Claudia C. Alvarez Campos	
Descripción: Se implementarán las funcionalidades necesarias para que el componente realice peticiones asíncronas al servidor.	

Tabla 36: Tarea # 6 Manejar peticiones asincrónicas mediante el componente.

Anexo 4

Casos de prueba de aceptación.

Caso de prueba de aceptación

Código: HU2-P2	**No. Historia de Usuario:** 2
Nombre: Autenticar usuario en el servidor mediante el método Digest sobre el protocolo HTTP.	
Descripción: Prueba para la funcionalidad Autenticar usuario en el servidor mediante el método Digest sobre el protocolo HTTP.	
Condiciones de ejecución: • El usuario deberá seleccionar como modo de conexión: DIGEST. • El usuario deberá seleccionar como protocolo de comunicación: HTTP. • El usuario deberá escoger la opción: Conectarse.	
Entrada/Pasos de ejecución: El usuario selecciona entre los parámetros HTTP como protocolo de comunicación y Basic como modo de conexión, el componente genera el digest de las credenciales y conforma la solicitud que posteriormente envía al servidor. Se recoge la respuesta y se muestra al usuario el resultado.	
Resultado esperado: El mensaje de respuesta es entregado sin errores.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 37 HU2-P2: Autenticar usuario en el servidor mediante el método Digest sobre el protocolo HTTP.

Caso de prueba de aceptación	
Código: HU3-P3	**No. Historia de Usuario:** 3
Nombre: Autenticar usuario en el servidor a través del protocolo HTTPS con los métodos Digest o Basic.	
Descripción: Prueba para la funcionalidad Autenticar usuario en el servidor a través del protocolo HTTPS	

con los métodos Digest o Basic.
Condiciones de ejecución: • El usuario deberá seleccionar como modo de conexión: DIGEST o BASIC. • El usuario deberá seleccionar como protocolo de comunicación HTTP en su versión combinada con SSL: HTTPS. • El usuario deberá escoger la opción: Conectarse.
Entrada/Pasos de ejecución: El usuario selecciona entre los parámetros HTTPS como protocolo de comunicación y Basic o Digest como modo de conexión, en caso de que el usuario selecciones el modo Digest el componente genera el digest de las credenciales y conforma la solicitud que posteriormente envía al servidor, y en caso de que el usuario selecciones el modo Basic las credenciales serán enviadas sin encriptar. Se recoge la respuesta y se muestra al usuario el resultado.
Resultado esperado: El mensaje de respuesta es entregado sin errores.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 38 HU3-P3: Autenticar usuario en el servidor a través del protocolo HTTPS con los métodos Digest o Basic.

Caso de prueba de aceptación	
Código: HU5-P5	**No. Historia de Usuario:** 5
Nombre: Hacer peticiones al servidor vía HTTP-POST.	
Descripción: Prueba para la funcionalidad Hacer peticiones al servidor vía HTTP-POST.	
Condiciones de ejecución:	

• El usuario deberá seleccionar como protocolo de comunicación: HTTP. • El usuario deberá seleccionar como método de envío: POST.
Entrada/Pasos de ejecución: El usuario selecciona entre los parámetros HTTP como protocolo de comunicación y POST como método de envió, el componente conforma la solicitud y la envía al servidor. Se recoge la respuesta y se muestra al usuario el resultado.
Resultado esperado: El mensaje de respuesta es entregado sin errores.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 39 HU5-P5: Hacer peticiones al servidor vía HTTP-POST.

Caso de prueba de aceptación	
Código: HU7-P7	**No. Historia de Usuario:** 7
Nombre: Gestionar Certificados.	
Descripción: Prueba para la funcionalidad Gestionar Certificados.	
Condiciones de ejecución: • El usuario deberá seleccionar como protocolo de comunicación HTTP en su versión combinada con SSL: HTTPS. • El usuario deberá escoger la opción: Conectarse.	
Entrada/Pasos de ejecución: El usuario selecciona entre los parámetros HTTPS como protocolo de comunicación. El componente le solicita al servidor el certificado para encriptar los parámetros de conexión.	

Resultado esperado: El mensaje de respuesta es entregado sin errores.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 40 HU7-P7: Gestionar Certificados.

Caso de prueba de aceptación	
Código: HU8-P8	**No. Historia de Usuario:** 8
Nombre: Manejar peticiones asincrónicas mediante el componente.	
Descripción: Prueba para la funcionalidad Manejar peticiones asincrónicas mediante el componente.	
Condiciones de ejecución: • El usuario deberá realizar peticiones asíncronas al componente.	
Entrada/Pasos de ejecución: El usuario realiza varias peticiones al servidor. El componente gestiona cada petición de forma independiente y asíncrona.	
Resultado esperado: El mensaje de respuesta es entregado sin errores.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 41 HU8-P8: Manejar peticiones asincrónicas mediante el componente.

GLOSARIO

HTTP: Hypertext Transfer Protocol o Protocolo de Transferencia de Hipertexto.

HTTPS: Hypertext Transfer Protocol Secure o Protocolo Seguro de Transferencia de Hipertexto

SSL: Secure Sockets Layer o Capa de Conexión Segura.

ESET: Compañía de seguridad informática establecida en Bratislava, Eslovaquia. El nombre de la compañía se deriva de la diosa egipcia Isis.

CAS: Central Authentication Service o Servicio de Autenticación Central.

CRC: Colaborador - Responsabilidad - Clase

GRASP: General Responsibility Assignment Software Patterns o Responsabilidad de la asignación general Patrones de Software.

GOF: Gang of Four o Grupo de los cuatro (Creadores: Erich Gamma, Richard Helm, Ralph Johnson y John Vlisides).

RSA: Algoritmo descrito en 1977 por Ron Rivest, Adi Shamir y Len Adleman, del Instituto Tecnológico de Massachusetts.