UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 5

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS

Título:

Módulo de Seguridad para el Sistema de Medición Arex

Autor: Enrique Santos Leyva.

Tutores: Ing. Diana Tahirí Galí Ramírez.

Ing. Julio Alberto Leyva Durán.

Ciudad de la Habana, de junio de 2016.

"Año 57 de la Revolución."

El valor de una educación universitaria no es el aprendizaje de muchos datos, sino el entrenamiento de la mente para pensar.

Albert Einstein

DECLARACIÓN DE AUTORÍA

Declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo. Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

_____

Firma del Autor: Enrique Santos Leyva

_____

Firma del Tutor: Ing. Diana Tahiri Galí Ramírez.

_____

Firma del Tutor: Ing. Julio Alberto Leyva Durán.

DATOS DE CONTACTO

Tutor: Ing. Diana Tahirí Galí Ramírez.

Edad: 26 años.

Ciudadanía: cubana.

Institución: Universidad de las Ciencias Informáticas (UCI).

Título: Ingeniero en Ciencias Informática.

Categoría Docente: Instructor.

E-mail: dtgali@uci.cu.

Tutor: Ing. Julio Alberto Leyva Durán.

Edad: 33 años.

Ciudadanía: Cubana.

Institución: Universidad de las Ciencias Informáticas (UCI).

Título: Ingeniero en Ciencias Informática.

Categoría Docente: Instructor.

E-mail: jaleyva@uci.cu

AGRADECIMIENTOS

A todas las personas que me ayudaron y guiaron en la realización de este trabajo de diploma: mis tutores Diana y Julio por estar cada día que los necesitaba, Viltre, Frank y Yordan por molestarlos tanto a causa de la programación, y demás.

A mi familia del proyecto Arex que me han enseñado ser cada día más profesional.

A mi madre por darme este tamaño, apoyarme en cada decisión y guiarme siempre para poder ser quien soy.

A mi padre por llevarme siempre la contraria y al final, confiar en mí y llenarse de orgullo.

A mis hermanos por darme sus votos de confianza.

A mis compañeros de apartamento: Javier, Ernesto, José Carlos, Sadiel, Yanderd, José Manuel, Roberto, Dayron, Felipe, Eliuvis, Leduan, Daylen y Laura por compartir estos 5 años en la UCI.

A mi otra mamá de La Habana Deysi, a sus dos hijos Dayron y Nely por quererme y acogerme como otro miembro de su familia.

A mis amigos que he conocido durante toda mi vida y han mantenido su lealtad.

A mis compañeros del grupo 5502 y vecinos del edificio 66 y 91.

A mí mismo por haber cumplido un sueño de hace 12 años.

DEDICATORIA

A mi madre por estar siempre y apoyarme en todo.

A mi padre por esperar lo mejor de mi.

A mis hermanos Dairo, Yoan, Danyert y Danellis por ser parte de mi vida.

A mis abuelos maternos David y Vilma por ser mis segundos padres y por estar siempre cuando los necesitaba.

A mis abuelos paternos Deysi y Elugerio por apoyarme y darme ánimos cada día.

A mis tios Elugerio, Leandro, Hugo, Leonardo y Eudaldo por confiar en mi.

A mi bisabuela Verena por darme tanto amor.

A mis amigos del 66102 y después 91102, del grupo 5502 por apoyarme en toda la carrera en los buenos y malos momentos.

RESUMEN

Ante el constante crecimiento de los ataques informáticos que existen en la actualidad a nivel mundial, es necesario el desarrollo de aplicaciones con mayores niveles de seguridad para mitigar dichos ataques.

Para que un sistema sea seguro, se debe dotar al mismo de un mecanismo de control de acceso y que garantice que la información transmitida no pueda ser adquirida, alterada o utilizada por alguna entidad ajena al sistema. Con el objetivo de proveer al Sistema de Medición Arex de un mecanismo de autenticación y comunicación segura con las características anteriores, el presente trabajo propone implementar un módulo de seguridad que permita el control de acceso de los usuarios mediante usuarios y contraseñas, que se encargará del acceso al sistema. Además de proponer una mejora para que la información que se intercambia entre los módulos del sistema mediante el protocolo de comunicación sea ilegible.

Para cumplir el objetivo se realizó un estudio de los diferentes métodos de autenticación, de los algoritmos de cifrado y cuáles de ellos son utilizados por sistemas similares. Se investigó las principales tecnologías, herramientas y bibliotecas existentes en el mundo y se hace una propuesta fundamentada de cuales utilizar. Se presenta además la descripción de los requerimientos del sistema y se realiza la implementación de los mismos. Posteriormente las diferentes pruebas realizadas por el cliente, permitió la aceptación del producto.

Palabras clave: ataques informáticos, información, seguridad, sistema, sistema de medición

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE ILUSTRACIÓNES

INTRODUCCIÓN

El desarrollo tecnológico ha dado un gran paso de avance en la automatización de los procesos que se llevan a cabo en las industrias, instituciones u organizaciones; en estas, la gestión de la información es unos de los procesos de mayor envergadura. La misma es obtenida a partir de cualquier elemento que trate con ella (personas o programas que intervienen). La información es un dato o un conjunto de datos que se relacionan entre sí, que arrojan un resultado final para una posterior toma de decisión.

Dicha información puede ser de vital importancia para un negocio, y más si determina el éxito o el fracaso del mismo, en una sociedad que vive en constante envío de datos. Su procesamiento y posterior almacenaje en medios físicos o envío a través de la red, no pueden sufrir modificación, pérdida o espionaje por terceras personas.

Los incidentes de seguridad están disminuyendo cada vez más, al mismo momento que internet crece en todo el mundo (1). La seguridad permite mantener ausencia de amenaza ante cualquier agente que intente traspasar la protección existente. Por tanto, el objetivo fundamental de la seguridad informática es proteger los activos que están asociados directamente con los elementos que integran un sistema informático (datos, archivos o programas), la forma en que se manipula la información, se divulga o se destruye. El control y el monitoreo constante permite mitigar las vulnerabilidades que pueda haber y resguardar los bienes y activos informáticos a partir de la utilización de dispositivos o software que garanticen la seguridad.

La Universidad de las Ciencias Informáticas (UCI) es un proyecto de la Revolución Cubana fundada en el 2002. En esta institución se encuentra el Centro de Informática Industrial (CEDIN) perteneciente a la facultad 5, que tiene como principal objetivo el desarrollo de Sistemas SCADA (del inglés, *Supervisory Control And Data Adquisition*). Los SCADA comprenden las soluciones de aplicación que requieren de la captura de información de un proceso o planta industrial, la cual es utilizada para realizar una serie de análisis o estudios con los que se pueden obtener valiosos indicadores que permiten una realimentación sobre un operador o sobre el propio proceso (2). Este centro también cuenta con el proyecto de pequeña escala llamado Sistema de Medición Arex.

El Sistema de Medición Arex está diseñado para medir procesos de baja y mediana complejidad (procesos que incluyen hasta 100 variables), a partir de la adquisición y procesamiento de datos en tiempo real asociados a variables incidentes dentro del proceso. Está dirigido a servicios de la domótica para aplicar en áreas de hotelería, turismo y pequeñas industrias para detectar el estado de las luces, temperatura, presencia y humedad de determinado lugar. Está compuesto por: interfaz de edición o

diseño, interfaz de tiempo de ejecución y recolección de datos que incluye el manejo de protocolos de comunicación y transferencia de datos por las principales interfaces de red (3). A la información que se intercambia entre los módulos mediante la red no se le garantizan su correcta protección ante algún tipo de ataque informático pudiendo comprometer la validez de cada uno de los datos que se emiten o se reciben. Partiendo de lo analizado anteriormente surge la siguiente **situación problemática**:

- Actualmente la comunicación de sus módulos a través de la red y el intercambio de datos entre las partes se transmite en texto plano sin ser cifrado, lo cual provoca que no exista seguridad en los datos. Siendo esta de vital importancia ya que su funcionamiento se realiza en tiempo real, la alteración del valor de una variable podría provocar que no se notificara la ocurrencia de una alarma crítica.
- No existe además una funcionalidad que posibilite la autenticación de los usuarios de forma segura, con sus privilegios asignados, que permita un control y acceso de modo seguro a la aplicación.

Según lo antes expuesto se plantea como **problema de investigación**: ¿Cómo realizar un intercambio de datos entre los módulos Edición, Recolección y Ejecución, y el control de acceso en el Sistema de Medición Arex, de forma segura?

A partir del problema planteado se delimita el **objeto de estudio:** Los mecanismos de seguridad para la transmisión de datos y sistemas de autenticación.

Estableciendo como **objetivo general:** Desarrollar un módulo que permita el control de acceso a los usuarios y la comunicación cifrada entre los módulos de Edición, Recolección y Ejecución para el Sistema de Medición Arex.

Definiendo como **campo de acción:** Sistemas de cifrado de datos en el intercambio de información entre los módulos y control de acceso de usuarios.
Y para darle cumplimiento al objetivo trazado se proponen como **tareas de investigación:**

- Elaboración del marco teórico de la investigación.
- Selección y análisis de mecanismos de autenticación y control de acceso de usuarios, de los Protocolos Ligero de Acceso a Directorios (LDAP), de registro de usuarios y de algoritmos de cifrados.
- Argumentación de la metodología, herramientas de software y lenguajes de programación para el desarrollo del módulo de seguridad.
- Análisis, diseño e implementación del módulo con el objetivo de lograr el resultado esperado.

- Validación de la calidad del producto final.

Idea a defender:

Con el control de acceso de usuarios, y el cifrado de los datos que se intercambian entre los distintos módulos del Sistema de Medición Arex se obtiene un módulo de seguridad capaz de garantizar el decremento de las vulnerabilidades en el sistema.

Métodos de investigación

Para desarrollar esta investigación y lograr los objetivos planteados se utilizan métodos teóricos y empíricos, mediante los cuales se obtiene una idea más detallada de lo que se quiere lograr.

Métodos Teóricos:

Histórico–Lógico:

En la presente investigación se utilizó este método para realizar el estudio del arte, es decir, para realizar el estudio acerca de los sistemas o soluciones similares, además de los lenguajes, herramientas y metodologías para el desarrollo del sistema que se propone.

Modelación:

Se empleó para realizar las descripciones de cada historia de usuarios necesarios para darle cumplimiento a los requisitos funcionales y no funcionales asociados al sistema. La construcción de diagramas que den un mejor entendimiento de la solución.

Análisis y Síntesis:

Para realizar el análisis y estudio de las bibliografías existentes sobre el tema en cuestión y lograr obtener de manera sintetizada el contenido necesario y suficiente para la realización del presente trabajo.

Métodos Empíricos:

Consulta de la información en todo tipo de fuentes: Permitió la elaboración del marco teórico de la investigación a partir de documentos que contienen datos útiles para luego conocer, distinguir y seleccionar las fuentes de información adecuadas para el trabajo.

Entrevista:

Se realizaron entrevistas con el objetivo de comprender cómo se lleva a cabo el proceso de gestión de la seguridad en la versión en que se encuentra el software y para obtener información acerca de los requerimientos que debe cumplir el software.

Posibles resultados:

- Módulo que permita el control de acceso a los usuarios y la comunicación cifrada entre los módulos de Edición, Recolección y Ejecución del Sistema de Medición Arex.

El documento está estructurado por tres capítulos:

Capítulo 1. Fundamentación Teórica: En este capítulo se presentan los principales conceptos que se deben tener en cuenta para un mejor entendimiento del trabajo y metodología de desarrollo.

Capítulo 2. Propuesta de Solución: En este capítulo se establecerá la arquitectura y los patrones de diseño a utilizar, descripción de los requisitos funcionales y no funcionales para realizar la implementación de la aplicación.

Capítulo 3. Implementación y Prueba: Se describe la implementación de la aplicación, así como la forma en que esta quedará integrada en el Sistema de Medición Arex y las pruebas realizadas a las funcionalidades desarrolladas.

CAPÍTULO 1. Fundamentación Teórica

1.1 Introducción

En el presente capítulo se abordan conceptos relacionados con la seguridad, los mecanismos de autenticación y gestión de sesiones, de los Protocolos Ligero de Acceso a Directorios (LDAP), de registro de usuarios y de algoritmos de cifrados. Además de los diferentes tipos de autenticación existentes a nivel el mundial y cómo está organizada la seguridad de los SCADA mundialmente. Se abordarán las herramientas y metodologías de desarrollo que estarán presentes en el trabajo.

1.2 Seguridad

Un local tiene características específicas y con ella la protección que debe poseer para mantener la integridad de sus activos, bienes o información y así garantizar una seguridad parcial o total. De acuerdo a lo planteado por diversos autores en las bibliografías consultadas se establecen las siguientes definiciones:

Según el Profesor Adjunto Héctor Saint-Pierre de la Universidad Estadual Paulista "Júlio de Mesquita Filho":

"En principio, el término seguridad indica un estado o sensación que produce la percepción de ausencia de amenazas que coloque en riesgo la existencia, la propiedad, los intereses, los valores o el particular modo de ser de quien percibe."

Según la Real Academia de la Lengua Española:

- "Seguridad es cualidad de seguro."
- "Servicio encargado de la seguridad de una persona, de una empresa, de un edificio, etc."
- "Fianza u obligación de indemnidad a favor de alguien."

"La seguridad es una característica de cualquier sistema (informático o no) que indica que ese sistema está libre de todo peligro, daño o riesgo, y que es, en cierta manera, infalible (4)."

Por lo tanto, la seguridad no es más que proteger cualquier elemento ante la presencia de algún riesgo o amenaza que provoque daños o pérdida a corto, mediano o largo plazo en cualquier escenario que se desarrolle. Existen varios tipos de seguridad dentro la cual se encuentra la seguridad activa y la seguridad pasiva en la construcción de automóvil; la seguridad ciudadana, la seguridad jurídica y la seguridad social en la sociedad en la cual se vive; la seguridad informática en las comunicaciones y la seguridad de la información en sistemas informáticos.

1.1. Seguridad Informática

La Seguridad Informática (SI) es el conjunto de métodos y herramientas destinados a proteger los bienes (o activos) informáticos de una institución (5).

La SI es la disciplina que se ocupa de diseñar las normas, procedimientos, métodos y técnicas destinados a conseguir un sistema de información seguro y confiable (6).

El término está estrechamente relacionado con tres aspectos fundamentales de cualquier sistema de computación (7):
-**Confidencialidad:** la información o los activos informáticos son accedidos solo por las personas autorizadas para hacerlo.
-**Integridad:** los activos o la información solo pueden ser modificados por las personas autorizadas y de la forma autorizada.
-**Disponibilidad:** los activos informáticos son accedidos por las personas autorizadas en el momento requerido.

Para analizar la seguridad de un sistema se debe pensar en la forma en que el mismo pudiera sufrir determinada pérdida o daño, para lo cual es necesario identificar las debilidades del sistema (7).
-**Vulnerabilidad:** es una debilidad en el sistema de seguridad.
-**Amenaza:** grupo de circunstancias que tienen el potencial para causar algún daño o pérdida.
-**Control / Mecanismo de defensa:** acción, dispositivo o procedimiento que elimina o reduce una vulnerabilidad.

La seguridad en los sistemas informáticos se puede garantizar mediante la autenticación, la cual se puede realizar de diferentes formas dependiendo de las necesidades requeridas.

1.3 Autenticación

La autenticación es el procedimiento de comprobación de la identidad de una entidad del sistema (usuario). Se basa en la idea de que cada entidad tendrá una información única que la identifique o que la distinga de otras. En el caso de usuarios se realiza normalmente mediante un nombre de usuario y una contraseña (8).

Además, la autenticación es el proceso de intento de verificar la identidad digital del remitente de una comunicación como una petición para conectarse. El remitente siendo autenticado puede ser una persona

que usa un ordenador o un programa del ordenador. Es el acto de establecimiento o confirmación de algo (o alguien) como auténtico, es decir, es verificar que el usuario sea quien dice ser (9).

Una vez analizado los conceptos, se puede establecer que la autenticación es un aspecto fundamental de la seguridad de un sistema, confirma la identidad de cualquier usuario que intenta iniciar la sesión en un dominio, tener acceso a los recursos de la red y a algún sistema informático.

1.3.1 Métodos de Autenticación

Los métodos de autenticación se dividen en tres grandes categorías en función de lo que utilizan para la verificación de identidad (10):

- Sistemas basados en algo conocido.
- Sistemas basados en algo poseído.
- Sistemas basados en una característica física del usuario o un acto involuntario del mismo.

Sistemas basados en algo conocido

El modelo de autenticación más básico, consiste en decidir si un usuario es quien dice ser, simplemente hay que basarse en una prueba de conocimiento que a priori sólo ese usuario puede superar. Evidentemente, esta aproximación es la más vulnerable a los ataques, pero también la más barata. Dentro de este grupo se encuentran las contraseñas y el código de identificación personal.

Las entidades (generalmente dos) que participan en la autenticación acuerdan una clave, que han de mantener en secreto si desean que la autenticación sea fiable. Cuando una de las partes desea autenticarse ante otra se limita a mostrarle su conocimiento de esa clave común, y si ésta es correcta se otorga el acceso a un recurso.

Sistemas basados en algo poseído

Este modelo de autenticación es basado en algo que el usuario lleva con él, o sea, algo que le pertenece, dentro de este tipo de clasificación se encuentran las tarjetas (aunque existen otros dispositivos dentro de este tipo de autenticación), estas pueden ser de banda magnética, tarjetas de código de barra y las tarjetas inteligentes, entre otras.

Tarjetas de banda magnética:

Las primeras tarjetas con banda magnética fueron usadas desde principios de los sesenta en el transporte público. El objetivo de esta tarjeta es identificar a un cliente para acceder a una base de datos remota con la que se establece una conexión. La información que posee la base de datos permite aceptar o rechazar esa transacción.

La utilización de tarjetas de banda magnética resulta interesante ya que la tecnología de lectura/escritura está ampliamente desarrollada y difundida. El coste del equipamiento necesario es relativamente bajo, sobre todo teniendo en cuenta su alta durabilidad. Estas tarjetas poseen múltiples usos, ejemplo: tarjetas de crédito, tarjetas recargables, billetes de tren, avión, metro, etc.

Tarjetas con códigos de barra:

El código de barras es el sistema de identificación electrónica más extendido. Se implementa sobre todo en el comercio para la identificación de mercancía, empleados o clientes. Las tarjetas con código de barras resultan ideales cuando se trata de identificar una gran cantidad de personas gracias a su coste muy económico y rapidez de lectura.

Ilustración 2. Tarjetas con código de barra

Tarjetas Inteligentes:

Una tarjeta inteligente es un dispositivo con características similares a la de una tarjeta de crédito, y que contiene un circuito integrado incrustado, con memoria y capacidades de procesamiento de información que permite ejecutar aplicaciones para el almacenamiento y transferencia de información de forma segura, eficiente y confiable.

Las tarjetas inteligentes surgen ante nuevas necesidades del mercado, las cuales no pueden ser satisfechas por la tarjeta de banda magnética.

Ilustración 3. Tarjetas inteligentes

Sistemas basados en una característica física del usuario o un acto involuntario del mismo

Dentro de este grupo están los conocidos sistemas biométricos, los cuales están basados en las características físicas y conductuales de los individuos. Estos se refieren a las tecnologías para medir y analizar las características físicas y del comportamiento humano con propósito de autenticación.

Las huellas dactilares, las retinas, el iris, los patrones faciales, la geometría de la palma de la mano, representan ejemplos de características físicas, y entre los ejemplos de características del comportamiento se incluye la firma, el paso y el tecleo.

Es fácil ver ejemplos de cada uno de estos tipos de autenticación: una contraseña es algo que el usuario conoce y el resto de personas no, una tarjeta de identidad es algo que el usuario lleva consigo, la huella dactilar es una característica física del usuario, y un acto involuntario podría considerarse que se produce al firmar. Por supuesto, un sistema de autenticación puede (y debe, para incrementar su confiabilidad) combinar mecanismos de diferentes tipos, como en el caso de una tarjeta de crédito junto al PIN (*Personal Identification Number* o Número de Identificación Personal en español) a la hora de utilizar un cajero automático (11).

1.3.2 Protocolos de autenticación

La comunicación mediante la red de redes se realiza de disímiles formas y una de ellas es la utilización de protocolos para garantizar mayor seguridad en la comunicación. Por lo tanto, un protocolo criptográfico tiene el propósito de autenticar entidades que desean comunicarse de forma segura.

Tipos de protocolos de autenticación:

- **PAP:** son las siglas de *Password Authentication* es un protocolo simple de autenticación para autenticar un usuario contra un servidor de acceso remoto o contra un proveedor de servicios de internet.
- **CHAP:** es un protocolo de autenticación por desafío mutuo (CHAP, en inglés *Challenge Handshake Authentication Protocol*). Es un método de autenticación remota o inalámbrica.

- **SPAP:** es el protocolo de autenticación de contraseñas de Shiva (SPAP, Shiva Password Authentication Protocol). Es un mecanismo de cifrado reversible empleado por Shiva.
- **MS-CHAP:** (en inglés Microsoft Challenge Handshake Authentication Protocol), es un protocolo de autenticación de contraseñas de cifrado no reversible.
- **EAP:** Protocolo de autenticación extensible (EAP, Extensible Authentication Protocol), mecanismo de autenticación arbitrario, valida las conexiones de acceso remoto.
- **DIAMETER:** es un protocolo de red para la autenticación de los usuarios que se conectan remotamente a internet a través de la conexión por línea conmutada o Red Telefónica Conmutada (RTC).
- **KERBEROS:** es un protocolo de autenticación de redes de ordenador que permite a dos computadores en una red insegura demostrar su identidad mutuamente de manera segura.
- **NTLM:** es una suite de Microsoft con protocolos de seguridad que proporciona autenticación, integridad y confidencialidad a los usuarios.
- **PEAP:** es un protocolo que encapsula el protocolo de autenticación extensible (EAP) en una conexión cifrada y autenticada *Transport Layer Security* (TLS).
- **RADIUS:** un protocolo de autenticación y autorización para aplicaciones de acceso a la red o movilidad IP.
- **TACACS Y TACACS+:** protocolo de autenticación remota, propietario de cisco, que se usa para comunicarse con un servidor de autenticación comúnmente usado en redes Unix.

Hay varias formas de manejar la autenticación del usuario a la hora de autenticarse con cualquier sistema:

- Autenticación basada en Email.
- Sólo cuentas manuales.
- Sin autenticación.
- Usar un servidor POP3.
- Usar un servidor LDAP.

Cuando un usuario se identifica en un sistema dicha herramienta puede manejar los datos de los usuarios mediante la gestión de sesiones de usuarios para controlar el nivel de seguridad o de permiso poseen cada usuario.

1.4 Gestión de sesiones y usuarios

En informática, en particular en redes informáticas, una sesión es la duración de una conexión empleando una capa de sesión de un protocolo de red, o la duración de una conexión entre un usuario (el agente) y un servidor, generalmente involucrando el intercambio de múltiples paquetes de datos entre la

computadora del usuario y el servidor. Una sesión es típicamente implementada como una capa en un protocolo de red (por ejemplo, telnet y FTP) (12).

Una sesión de cliente / servidor es una serie de comunicaciones relacionados entre un cliente y un servidor que tienen lugar durante un período de tiempo. Con una sesión establecida, el servidor puede identificar al cliente asociado con cada solicitud, y tiene la capacidad de recordar (de entre muchas solicitudes) un cliente específico (13). Sin una sesión establecida, la comunicación entre un cliente y un servidor puede ser renegociado para cada solicitud posterior. La información de estado de la sesión mejora el rendimiento mediante la eliminación de cierre repetido y reapertura de las sesiones de cliente / servidor. El cliente puede iniciar sesión una vez y hacer numerosas peticiones sin realizar una entrada separada para cada solicitud.

En informática, un usuario es un individuo que utiliza una computadora, sistema operativo, servicio o cualquier sistema informático. Por lo general es una única persona (14).

Por tanto, un usuario registrado es aquel que puede identificarse en un sistema utilizando nombre de usuario y contraseña. Al mismo se le puede asociar una única cuenta de usuario la cual puede acceder a los servicios una vez autenticado.

Las sesiones de usuarios tienen un tiempo de vida para que estén activas en el sistema por lo cual cuando este tiempo expire ocurre caducidad de la sesión de usuarios. La caducidad de sesión de usuarios no es más que la desactivación de una sesión que se encontraba inactiva. Una sesión caduca si el usuario está inactivo por un tiempo determinado o la contraseña expiró en el tiempo establecido o ante un fallo de red.

1.5 Registro de usuarios

En los sistemas desarrollados en el mundo la mayoría tienen que interactuar con el usuario y tienen la necesidad de conocer datos específicos de cada uno y se resuelven con los llamados registros de usuarios.

Estos son protocolos y etiquetas ofrecen todo su contenido con libre acceso a todos los usuarios de internet, sin embargo, para poder interactuar o acceder a ciertos servicios, es necesario ser un usuario registrado. Además, se dispone de una política de privacidad que todo usuario deberá leerse y aceptar antes de registrarse en cualquier aplicación (15).

El registro de usuario se puede hacer de varias formas:

- Mediante otros sitios como: Facebook, Twitter, Google, etc.

- Con una dirección de correo electrónico.

En cada registro de usuario, el usuario puede activar su cuenta o desactivarla cuando él desee dependiendo de las funcionalidades que la aplicación ofrezca, si ya se ha registrado y olvidó su contraseña de acceso, puede generar una nueva.

1.6 Protocolo ligero de acceso a directorios (LDAP)

En la actualidad la mayoria de los servicios que brindan acceso a los usuarios desde un lugar o servidor, tienen centralizados los datos de cada usuario que integra una entidad o institucion en una base de datos.

El LDAP (en inglés *Lightweight Directory Access Protocol*) es un conjunto de protocolos abiertos usados para acceder a información guardada centralmente a través de la red. Está basado en el estándar X.500 para compartir directorios, pero es menos complejo e intensivo en el uso de recursos. Por esta razón, a veces se habla de LDAP como "X.500 Lite". El estándar X.500 es un directorio que contiene información de forma jerárquica y categorizada, que puede incluir nombres, directorios y números telefónicos (16).

X.500 es un conjunto de estándares de redes de ordenadores del Sector de Normalización de las Telecomunicaciones de la UIT (ITU-T) sobre servicios de directorio, entendidos estos como bases de datos de direcciones electrónicas (o de otros tipos). El estándar se desarrolló conjuntamente con la Organización Internacional de Normalización (ISO) como parte del modelo de interconexión de sistemas abiertos, para usarlo como soporte del correo electrónico X.400 (16).

1.7 Protocolos de comunicación. Caracterización

En las comunicaciones se utilizan diferentes formas para poder comunicar a un agente y un receptor que se encuentren en la red. Estos dependen de quien sea el fabricante y qué protocolo utilice para su dispositivo.

Un protocolo de comunicación es un conjunto de reglas usadas por computadoras para comunicarse a través de la red. Es una regla o estándar que controla o permite la comunicación en su forma más simple, también puede ser definido como las reglas que denominan la sintaxis, semántica y sincronización de la comunicación. Los protocolos pueden ser implementados por hardware, software, o una combinación de ambos.

1.7.1 Tipos de protocolos de comunicación:

- **TCP/IP:** son las siglas de "*Transfer Control Protocol* / Internet Protocol" es una denominación que permite identificar al grupo de protocolos de red que respaldan a Internet y que hacen posible la transferencia de datos entre redes de ordenadores (17).
- **IPX/SPX** (del inglés *Internetwork Packet Exchange/Sequenced Packet Exchange*), Protocolo Novell o simplemente IPX: es una familia de protocolos de red desarrollados por Novell y utilizados por su sistema operativo de red NetWare. SPX (*Sequenced Packet Exchange*) actúa sobre IPX para asegurar la entrega de los paquetes (15).
- **NetBIOS (***Network Basic Input/Output System*): Es una especificación de interfaz para acceso a servicios de red, es decir, una capa de software desarrollado para enlazar un sistema operativo de red con hardware específico. NetBIOS fue originalmente desarrollado por IBM y Sytek como un API/APIS para el software cliente de recursos de una red local/red de área local (LAN) (18).
- **NetBEUI** (*NetBIOS Extended User Interface*, en español Interfaz extendida de usuario de NetBIOS): es un protocolo de nivel de red sin encaminamiento y bastante sencillo utilizado como una de las capas en las primeras redes de *Microsoft. NetBIOS* sobre NetBEUI es utilizado por muchos sistemas operativos desarrollados en los 1990, como LAN Manager, LAN Server, Windows 3.x, Windows 95 y Windows NT (19).
- **AppleTalk:** Serie de protocolos de comunicaciones diseñados por *Apple Computer*. Actualmente consta de dos fases. La Fase 1, la versión más antigua, soporta una sola red física que puede tener un solo número de red y estar en una sola zona. La Fase 2, la versión más reciente, soporta múltiples redes lógicas en una sola red física y permite que las redes se encuentren en más de una zona (20).
- **ARP** (*Address Resolution Protocol* o protocolo de resolución de direcciones): Es un protocolo de nivel de red responsable de encontrar la dirección hardware (MAC Address) que corresponde a una determinada dirección IP (21).
- **IP** (es la sigla de *Internet Protocol* o, en el idioma español, Protocolo de Internet): Se trata de un estándar que se emplea para el envío y recepción de información mediante una red que reúne paquetes conmutados (19).

En las aplicaciones que intercambian datos mediante la red se utilizan los protocolos de comunicación y según el tipo de fabricante del dispositivo es el protocolo que dicha aplicación debe de utilizar.

1.8 SCADA. Módulos de seguridad

La seguridad en sistemas SCADA estuvo olvidada hace un tiempo. Fueron pensados para ser sistemas aislados y no conectados en red, por lo que carecen de dispositivos de seguridad como cortafuegos,

mecanismos de cifrado o software antivirus. En la actualidad es común encontrarse sistemas SCADA que estén vinculados con las redes de las empresas por lo que es vital garantizar la seguridad de los mismos.

Los principales aspectos de seguridad que se tienen en cuenta en los sistemas SCADA son (22).

- **Características físicas:** la fiabilidad en sus equipos y su disponibilidad es directamente proporcional a su coste. Los precios de un ordenador industrial comparados con su homólogo casero son muy diferentes.
- **Sistemas Operativos:** el tener unos sistemas operativos considerados como estándares (Linux, Windows, principalmente) conlleva adoptar los riegos de los propios sistemas, de dominio público, tales como las conexiones desde el exterior del sistema sin conocimiento del usuario.
- **Comunicaciones:** Al utilizar protocolos abiertos, como ocurre con los sistemas operativos, se añaden más vulnerabilidades a los sistemas de control.
- **Aplicaciones:** la falta de medidas de seguridad (contraseñas, privilegios, limitación de tiempo) hace que los sistemas sean vulnerables. La posibilidad de conexión indiscriminada durante las puestas en marcha o durante el funcionamiento normal abre las puertas a multitud de amenazas, tales como los virus o los piratas informáticos.

Actualmente en el desarrollo de este tipo de sistemas se ha impulsado la creación de mecanismos que garanticen su seguridad ante usuarios maliciosos y las acciones dañinas que estos puedan llevar a cabo, utilizando métodos de identificación, autenticación y control de acceso de los usuarios a los recursos. Pero aún constituye una carencia de estos sistemas el hecho de que en muchos de ellos, no se han implementado medidas de seguridad básicas, tales como cifrado y redundancia; e incluso hay implementaciones cuya pila TCP/IP (en inglés *Transfer Control Protocol / Internet Protocol*) es defectuosa, lo que los hace vulnerables a escaneos de red y a un sinfín de ataques plenamente conocidos (23).

1.8.1 Sistemas SCADA disponibles en Cuba y el mundo

En el mundo existen disimiles sistemas de SCADA que realizan diferentes funciones dependiendo para el ambiente que fueron creados. A continuación, se muestran varios de estos sistemas y como garantizan su seguridad:

SCADA Movicon:

Movicon X representa la innovadora y revolucionaria tercera generación de plataformas de software para supervisión y control industrial (SCADA/HMI), desarrollado por el grupo italiano Progea. Hoy Movicon X

versión 10, renueva el concepto de supervisión, anticipándose al futuro de la automatización con las tecnologías más avanzadas (24).

Se garantiza la máxima seguridad de los datos. Los proyectos pueden ser encriptados con algoritmos de codificación de 128 bits. La contraseña de usuario administrador garantiza el acceso por nivel de seguridad o área. La integración de *Visual Source Safe* garantiza que todo su trabajo se mantiene a salvo (11).

SCADA Wincc:

WinCC, es un sistema de supervisión sobre computadoras, ejecutable bajo Microsoft Windows 95 y Windows NT, desarrollado por la empresa Siemens. Está concebido para la visualización y el manejo de procesos, en líneas de fabricación de máquinas e instalaciones (25).

Para aumentar el grado de seguridad durante la ejecución de un proyecto, la versión WinCC V6.2 ha sido habilitada para usar el cortafuego de Windows. Aparte de *Symantec Antivirus Corporate Edition* (a partir de la versión 8.1), *Trend Micro ServerProtect* (a partir de la versión 5.56) y *Trend Micro OfficeScan NT* (a partir de la versión 5.02), ahora también se admite el escáner antivirus *McAffee*. Para los proyectos WinCC integrados en Step 7[1] existe en el momento una protección contra el acceso indebido (26).

SCADA GALBA:

El Guardián del ALBA es un software que integra las funcionalidades de alto nivel que permiten la solución de aplicaciones de supervisión y control de procesos, utilizando para ello una arquitectura distribuida de módulos que permite escalar a aplicaciones de gran envergadura (27).

El módulo de seguridad del Guardián del ALBA proporciona las funcionalidades necesarias para garantizar el trabajo autorizado a usuarios y módulos, además brinda las herramientas necesarias para la protección contra ataques maliciosos o involuntarios al sistema por parte de personas o recursos, tales como fallas de energía, problemas de red o servidores, etc. Dentro de las funcionalidades que resuelve este módulo se encuentran (27):

1. Proporciona a los usuarios del sistema un mecanismo de autenticación seguro.
2. Brinda administración de usuarios y grupos de usuarios.
3. Implementa el control de acceso a los recursos basado en privilegios asociados a los usuarios y grupos de usuarios.

[1] STEP 7 es un Software de Programación de PLC (Controladores Lógicos Programables el SIMATIC-S7 de Siemens, es el sucesor de SIMATIC S5 STEP 7.

4. Monitorea eventos anómalos y generar avisos para prevenirlos.

El módulo está subdividido en los siguientes subsistemas (27):

- Subsistema de Autenticación.
- Subsistema de Control de acceso.
- Subsistema de Monitoreo.
- Subsistema de Administración de sesiones.
- Subsistema de Configuración.

1.9 Criptografía

La palabra criptografía proviene en un sentido etimológico del griego Kriptos=ocultar, Graphos=escritura, lo que significaría ocultar la escritura, o en un sentido más amplio sería aplicar alguna técnica para hacer ininteligible un mensaje (28).

En su clasificación dentro de las ciencias, la criptografía proviene de una rama de las matemáticas, que fue iniciada por el matemático Claude Elwood Shannon en 1948, denominada: “Teoría de la Información”. Esta rama de las ciencias se divide en: “Teoría de Códigos” y en “Criptología”. Y a su vez la criptología se divide en Criptoanálisis y Criptografía, como se muestra en la siguiente figura:

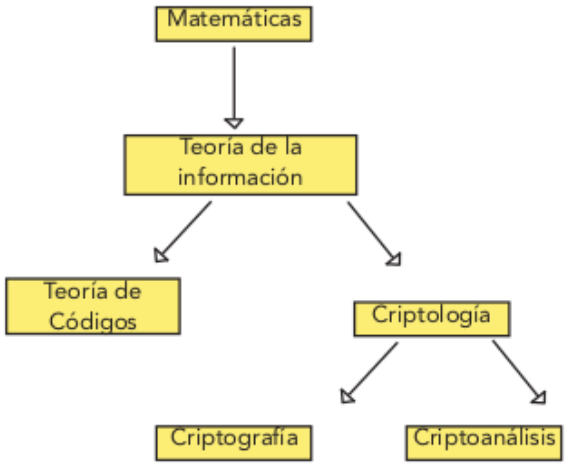

Ilustración 4. Origen de la criptografía

La Criptografía también es la ciencia encargada de diseñar funciones o dispositivos, capaces de transformar mensajes legibles o en claro a mensajes cifrados de tal manera que esta transformación

(cifrar) y su transformación inversa (descifrar) sólo pueden ser factibles con el conocimiento de una o más llaves.

1.9.1 Tipos de criptografía:

La criptografía se divide en clásica y moderna de forma poco formal. Dos métodos engloban lo más relevante de la criptografía clásica:

- ✓ **La sustitución**, consistente en cambiar los caracteres componentes del mensaje original en otros según una regla determinada de posición natural en el alfabeto.
- ✓ **La transposición**, consistente en cambiar los caracteres componentes del mensaje original en otros según una regla determinada de posición en el orden del mensaje.

En la criptografía moderna, las llaves de cifrado constituyen su base fundamental. Estas nuevas técnicas de cifrado de datos cuentan con una serie de elementos que las hacen superior a las técnicas llamadas clásicas. Con la aparición de las computadoras se dispone de una potencia de cálculo muy superior a la presente en los métodos clásicos, lo cual constituye la primera ventaja, mayor velocidad de cálculo, avance de las matemáticas que permitieron encontrar y definir con claridad sistemas criptográficos estables y seguros y las necesidades de seguridad que dieron surgimiento a muchas actividades nuevas que precisaban la ocultación de datos, con lo que la criptografía experimentó un fuerte avance.

A partir de estas bases surgieron nuevos y complejos sistemas criptográficos, que se clasificaron en los dos tipos o familias principales, los de llave simétrica y los de llave pública. Los modernos algoritmos de cifrado simétricos mezclan la trasposición y la permutación, mientras que los de llave pública se basan más en complejas operaciones matemáticas.

1.9.2 Criptografía simétrica

La criptografía simétrica incluye los sistemas clásicos, y se caracteriza porque en ellos se usa la misma clave para cifrar y para descifrar, motivo por el que se denomina simétrica. Toda la seguridad de este sistema está basada en la llave simétrica, por lo que es misión fundamental tanto del emisor como del receptor conocer esta clave y mantenerla en secreto.

Ilustración 5. Algoritmos de cifrados

Para que un algoritmo de este tipo sea considerado fiable debe cumplir varios requisitos básicos:

- ✓ conocido el criptograma (texto cifrado) no se pueden obtener de él ni el texto en claro ni la clave.

- ✓ conocidos el texto en claro y el texto cifrado debe resultar más caro en tiempo o dinero descifrar la clave que el valor posible de la información obtenida por terceros.

Ventajas:

- ✓ Descifran bloques de texto del documento original, y son más sencillos que los sistemas de clave pública.
- ✓ Sus procesos de cifrado y descifrado son más rápidos.
- ✓ No hay que almacenar tantas llaves diferentes.

Desventajas:

- ✓ Las principales desventajas de los métodos simétricos son la distribución de las claves, el peligro de que muchas personas deban conocer una misma clave y la dificultad de almacenar y proteger muchas claves diferentes.
- ✓ Hay dos agentes débiles (emisor y receptor).
- ✓ Como las llaves son iguales cualquiera que intercepte alguna puede descifrar los mensajes que se hayan cifrado con la misma.
- ✓ La llave se envía por un canal inseguro.

Todos los algoritmos de cifrados clásicos se pueden considerar simétricos, pero los principales algoritmos de este tipo en la actualidad son DES, TDES, AES e IDEA. A continuación se muestra las características de cada uno de ellos:

- **DES** (*Data Encryption Standard*): fue desarrollado como un estándar de cifrado de datos en 1997 por IBM, aunque actualmente ya no constituye un estándar. Fue el algoritmo más utilizado en el mundo hasta la década del 90. Utiliza una llave simétrica de 64 bits, de los cuales 56 son usados para el cifrado, mientras que los 8 restantes son de paridad, y se usan para la detección de errores en el proceso.
- **TDES** (*3DES, Triple-DES*): constituye una variante de DES, basado en su uso tres veces, para solventar el problema de la corta longitud de su llave. Normalmente se utiliza una secuencia de cifrado−descifrado−cifrado con tres claves diferentes y no relacionadas entre sí, con lo que se logra longitudes de llave de 192 bits, de los cuales 168 son efectivos.
- **AES** (*Advanced Encryption Standard*): también conocido como Rijndael, es un esquema de cifrado por bloques destinado a remplazar al DES como estándar. AES tiene un tamaño de bloque fijo de 128 bits y tamaños de llave de 128, 192 ó 256 bits. La mayoría de los cálculos de este algoritmo se realizan en un campo finito determinado. Opera en una matriz de 4×4 bytes, llamada state. Este algoritmo es uno de los más populares utilizados en criptografía simétrica.

- **IDEA** (*International Data Encryption Algorithm*): es un cifrador por bloques diseñado por Xuejia Lai y James L. Massey de la Escuela Politécnica Federal de Zúrich y descrito por primera vez en 1991. Cifra bloques de texto de 64 bits, en 8 rondas, operando siempre con números de 16 bits. Utiliza operaciones como XOR, suma y multiplicación de enteros.

1.9.3 Comparación de algoritmos simétricos

Para realizar una comparación entre los principales algoritmos simétricos en la actualidad y poder establecer cual brindaría una óptima seguridad en el cifrado de las comunicaciones del Sistema de Medición Arex, se deben tener en cuenta elementos como: potencialidad de la llave para las operaciones de cifrado y descifrado, velocidad con la que realizan estas operaciones y las debilidades que se han hecho públicas de dichos algoritmos y que pudieran afectar o no la seguridad del sistema. Ver anexo 1.

1.9.4 Criptografía asimétrica o de clave pública:

Se basa en el uso de dos claves diferentes, claves que poseen una propiedad fundamental: una clave puede descifrar lo que la otra ha encriptado.

Ilustración 6. Criptografía asimétrica

Las claves pública y privada tienen características matemáticas especiales, de tal forma que se generan siempre a la vez, por parejas, estando cada una de ellas ligada intrínsecamente a la otra. Los algoritmos asimétricos están basados en funciones matemáticas fáciles de resolver en un sentido, pero muy complicadas de realizar en sentido inverso, salvo que se conozca la clave privada, como la potencia y el logaritmo. Ambas claves, pública y privada, están relacionadas matemáticamente, pero esta relación debe ser lo suficientemente compleja como para que resulte muy difícil obtener una a partir de la otra.

Para que un algoritmo de clave pública sea considerado seguro debe cumplir:

- ✓ conocido el texto cifrado no debe ser posible encontrar el texto en claro ni la clave privada.
- ✓ conocido el texto cifrado (criptograma) y el texto en claro debe resultar más caro en tiempo o dinero, descifrar la clave que el valor posible de la información obtenida por terceros.
- ✓ conocida la clave pública y el texto en claro no se puede generar un criptograma correcto cifrado con la clave privada.
- ✓ dado un texto cifrado con una clave privada sólo existe una pública capaz de desencriptarlo y viceversa.

Ventajas:

- ✓ La llave tiene una longitud mínima de 1024 bits.
- ✓ Confiabilidad de un sistema asimétrico.

Desventajas:

- ✓ Los sistemas de clave pública dificultan la implementación del sistema y son mucho más lentos que los simétricos.
- ✓ Los algoritmos de cifrado son muy complejos computacionalmente por lo que son muy costosos.
- ✓ El espacio de almacenamiento de las llaves tiene que ser mayor.

Los criptosistemas de clave pública, aunque más lentos que los simétricos, resultan adecuados para las funciones de autenticación, distribución de claves y firmas digitales.

El cifrado asimétrico se destaca por ser seguro para el intercambio de las claves aún a través de medios públicos inseguros (como internet). Los principales algoritmos de cifrado asimétrico existentes en la actualidad y que son utilizados para el intercambio de claves son Diffie-Hellman, El Gamal y RSA. A continuación las características de cada uno de ellos:

- **Diffie – Hellman** (DH): Ideado por los matemáticos Whitfield Diffie y Martín Hellmancon y el informático Ralph Merkle a mediados de los 70, este algoritmo ha demostrado sus propiedades y en el tiempo necesario para calcular el valor del logaritmo de un número extremadamente grande y primo. En la práctica sólo es válido para el intercambio de llaves simétricas.
- **RSA** (Rivest, Shamir y Adleman o sistema criptográfico de clave pública): Fue desarrollado en 1977 y en la actualidad es el primer y más utilizado algoritmo de este tipo, es válido tanto para cifrar como para firmar digitalmente. Constituye el algoritmo de cifrado más utilizado en conexiones seguras en línea. La seguridad de este algoritmo radica en el problema de la factorización de números enteros y su funcionamiento se basa en el producto conocido, de dos números primos grandes elegidos al azar y mantenidos en secreto. Por su velocidad suele usarse en sistemas híbridos para cifrar y enviar la llave simétrica que se usa posteriormente en la comunicación cifrada.
- **El Gamal:** Se refiere a un esquema de cifrado basado en problemas matemáticos de logaritmos discretos. Es un algoritmo de criptografía asimétrica basado en la idea de Diffie-Hellman y que funciona de una forma parecida a este algoritmo. Puede ser utilizado tanto para generar firmas digitales como para cifrar o descifrar.

1.9.5 Firma digital

Las mayorías de los documentos oficiales se le garantiza su validez de varias formas: un cuño, un número de serie o una firma. Los documentos digitales también se pueden firmar mediante una firma digital que posea el emisor. La firma digital es una herramienta tecnológica que permite garantizar la autoría e integridad de los documentos digitales, no implica asegurar la confidencialidad del mensaje; un documento firmado digitalmente puede ser visualizado por otras personas.

La firma digital garantiza integridad mientras que el cifrado garantiza confidencialidad, indirectamente la criptografía asimétrica garantiza autenticidad, pero es realmente el certificado digital quien garantiza la autenticidad del emisor ante el receptor.

Ilustración 7. Firma Digital

1.9.6 Criptografía híbrida

La criptografía híbrida es un método criptográfico que usa tanto un cifrado simétrico como un asimétrico. Emplea el cifrado de clave pública para compartir una clave para el cifrado simétrico. El mensaje que se esté enviando en el momento, se cifra usando su propia clave privada, luego el mensaje cifrado se envía al destinatario. Ya que compartir una clave simétrica no es seguro, ésta es diferente para cada sesión.

Los sistemas asimétricos o de clave pública son lentos, pero tienen un intercambio de claves fácil de realizar y además cuentan con la firma digital que constituyen un factor importante para garantizar la autenticidad del usuario que envía el mensaje. Sin embargo, los sistemas simétricos o de llave privada son más rápidos pero el intercambio de la llave constituye su principal inconveniente, así como no cuentan con mecanismos de autenticación como la firma digital.

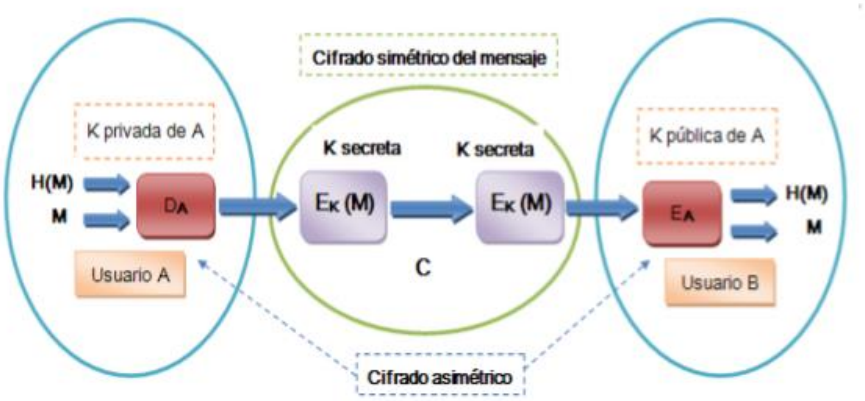

Ilustración 8. Criptografía híbrida

1.10 Sistemas empotrados o embebido

Existen variedades de definiciones para los Sistemas Empotrados (SE), a continuación, se presentan algunas de estas:

Un sistema empotrado es un sistema informático que se encuentra físicamente incluido en un sistema de ingeniería más amplio al que supervisa o controla. Los sistemas empotrados se encuentran en multitud de aplicaciones, desde la electrónica de consumo hasta el control de complejos procesos industriales (29).

De acuerdo a Galeano un SE es “un circuito electrónico computarizado que está diseñado para cumplir una labor especifica de un producto”. Así, la palabra empotrado ha reflejado el hecho de que estos sistemas se incorporan a un sistema de ingeniería más general, en el que se han realizado funciones de control, procesamiento y/o monitorización (30).

Según Natera y Meneses un sistema embebido es un circuito electrónico computarizado que está diseñado para cumplir una labor específica en un producto (31).

De las definiciones anteriores y de la investigación realizada, se define a los SE como:
Un SE es la mezcla entre el software y hardware, con el fin de cumplir una función específica. Están presentes en prácticamente todos los aspectos de la sociedad como, teléfonos móviles, automóviles, control de tráfico, ingenios espaciales, procesos automáticos de fabricación, producción de energía, aeronaves, etc. Además, el auge de los sistemas empotrados está en constante aumento, ya que cada vez más máquinas se fabrican incluyendo un número mayor de sistemas controlados por computador (32).

1.11 Tecnologías y metodologías

El proyecto Sistema de Medición Arex cuenta con un marco tecnológico establecido para su desarrollo. A continuación se presenta detalladamente las características de estas tecnologías y herramientas.

1.11.1 Lenguaje de programación: C++

Un lenguaje de programación es un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. Es utilizado para controlar el comportamiento físico y lógico de una máquina (11).

C++ posee las siguientes características:

- ✓ Es un lenguaje orientado a objetos.
- ✓ Es muy potente y práctico, debido a su robustez, para la creación de sistemas complejos.
- ✓ Es independiente de la plataforma.
- ✓ Es muy utilizado en el mundo por lo que existe abundante información sobre su uso.
- ✓ Existen muchos algoritmos ya desarrollados en C++, de manera que pueden tomarse y amoldarse a la solución deseada.
- ✓ Sirve para programar sistemas empotrados.
- ✓ Es el lenguaje definido por el proyecto para el desarrollo e integración con los demás componentes.

1.11.2 Entorno de desarrollo: Qtcreator

El IDE Qtcreator es una plataforma o soporte, que los desarrolladores de software usan para generar aplicaciones complejas, de forma más sencilla y segura. La plataforma puede constar de varias utilidades, como herramientas, bibliotecas, enlaces multilenguaje, comunicaciones estándar con bases de datos de diferente índole, etc.

Incluye:

- ✓ Un avanzado editor de código C ++.
- ✓ GUI Integrado de diseño y las formas de diseño.
- ✓ Herramientas de gestión de proyectos y construcción.
- ✓ Integrado, sistema de ayuda sensible al contexto.
- ✓ Depurador de Visual.
- ✓ Herramientas de navegación de código rápido.
- ✓ Soporta múltiples plataformas.

1.11.3 Lenguaje de Modelado: UML

Lenguaje Unificado de Modelado (UML) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad.

Se utilizó este lenguaje gráfico para visualizar, especificar, construir y documentar el sistema de software. Además, ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocios y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables.

1.11.4 Bibliotecas de cifrado

En ciencias de la computación, una biblioteca (del inglés *library*) es un conjunto de subprogramas utilizados para desarrollar software. Las bibliotecas contienen código y datos, que proporcionan servicios a programas independientes, es decir, pasan a formar parte de éstos. Esto permite que el código y los datos se compartan y puedan modificarse de forma modular.

La biblioteca a utilizar para la implementación de las funcionalidades de cifrado del componente a desarrollar debe ser una herramienta libre, de código abierto, que permita el uso de los algoritmos seleccionados para realizar el cifrado y descifrado de datos en un sistema híbrido a altas velocidades. A continuación, se muestran algunas bibliotecas de cifrado:

- **Crypto++:** también conocida como CryptoPP, libcrypto++ y libcryptopp. Es una biblioteca para el desarrollo de programas en C++ que requieren algoritmos criptográficos. Permite utilizar algoritmos cifradores por bloque (IDEA, AES, DES, RC5); funciones hash (SHA-1, SHA-256, SHA-512) y sistemas de clave pública (RSA y DSA).
- **SSL:** implementa los protocolos *Secure Sockets Layer* (Capa de Conectores de Red Segura) (SSL v2/v3) y *Transport Layer Security* (Capa de Transporte Segura) (TLS v1). Permite generar claves RSA, DH y DSA, calcular resúmenes de mensajes (SHA-256); cifrar y descifrar con distintos algoritmos de cifrado simétricos (DES, IDEA, AES) y asimétricos (RSA). Con esta biblioteca es posible realizar pruebas cliente/servidor utilizando el protocolo SSL y crear certificados X.50910, CSRs11 y CRLs12.
- **BeeCrypt:** Es una librería criptográfica de código abierto que proporciona una encriptación rápida y potente. Puede ser utilizada en proyectos de código abierto o comercial. Incluye generadores aleatorios; bloques de cifras; función hash (función unilateral inconvertible), y mensaje de

autentificación de códigos. Está optimizado para C e implementación en muchos algoritmos, incluyendo Blowfish, Diffie-Hellman, ElGamal, y SHA-1 (33).

1.11.5 Bibliotecas Boost

Las bibliotecas Boost C++ son una colección de bibliotecas modernas basadas en el estándar de C++. El código fuente se distribuye bajo la licencia de software Boost, que permite a cualquiera utilizar, modificar y distribuir las bibliotecas de forma gratuita. Las bibliotecas son independientes de la plataforma y apoyan los compiladores más populares, así como muchos que son menos conocidas (34).

Boost es un conjunto de bibliotecas de código abierto que permite extender las capacidades que oferta C++. Su licencia posibilita que sea empleada en cualquier tipo de proyecto, ya sea comercial o no. Está diseñada e implementada de manera tal que puede utilizarse en un amplio espectro de aplicaciones y plataformas (35). Este conjunto de bibliotecas brinda funcionalidades, entre las que se encuentran la posibilidad de representar períodos de tiempo y de realizar casteo de diferentes tipos de datos primitivos (string, int, float), que resultan muy útiles para el desarrollo del sistema para la seguridad.

1.11.6 Clase QSslSocket

La clase QSslSocket ofrece una toma de cifrado SSL para los clientes y los servidores. QSslSocket establece una conexión TCP seguro, se puede utilizar para la transmisión de datos cifrados. Puede funcionar tanto en modo cliente y el servidor, y es compatible con los protocolos SSL modernas, como SSLv3 y TLSv1_0. De forma predeterminada, QSslSocket utiliza TLSv1_0, pero se puede cambiar el protocolo SSL llamando la función setProtocol(), siempre y cuando lo haces antes de que comience el apretón de manos.

El cifrado SSL opera en la parte superior de la corriente TCP existente después de la toma de corriente entra en el estado de la conexión (*ConnectedState*). Hay dos formas sencillas para establecer una conexión segura utilizando QSslSocket: Con un protocolo de enlace SSL inmediata, o con un apretón de manos SSL retardada que ocurre después de la conexión se ha establecido en modo no cifrado (34).

Al igual que con un QTcpSocket, QSslSocket entra en el *HostLookupState*, *ConnectingState*, y finalmente el *ConnectedState*, si la conexión es exitosa. El apretón de manos se inicia automáticamente, y si tiene éxito, la señal encriptada se emite para indicar la toma ha entrado en el estado cifrado y está listo para su uso (36).

QSslSocket ofrece una amplia y fácil API para el manejo de los códigos de cifrado, claves privadas y certificados (CA), entre iguales, y entidad de certificación local. También proporciona una API para el manejo de errores que se producen durante la fase de apretón de manos (34).

1.11.7 Base de Datos

La base de datos (BD) son un conjunto de datos interrelacionados entre sí, almacenados con carácter más o menos permanente en la computadora. O sea, que una BD puede considerarse una colección de datos variables en el tiempo. El software que permite la utilización y/o la actualización de los datos almacenados en una (o varias) base(s) de datos por uno o varios usuarios desde diferentes puntos de vista y a la vez, se denomina sistema de gestión de bases de datos (SGBD) (37).

MySQL

Es un sistema gestor de bases de datos (SGBD, *DBMS* por sus siglas en inglés) muy conocido y ampliamente usado por su simplicidad y notable rendimiento. Aunque carece de algunas características avanzadas disponibles en otros SGBD del mercado, es una opción atractiva tanto para aplicaciones comerciales, como de entretenimiento precisamente por su facilidad de uso y tiempo reducido de puesta en marcha. Esto y su libre distribución en internet bajo licencia GPL le otorgan como beneficios adicionales (no menos importantes) contar con un alto grado de estabilidad y un rápido desarrollo.

MySQL está disponible para múltiples plataformas. Sin embargo, las diferencias con cualquier otra plataforma son prácticamente nulas, ya que la herramienta utilizada en este caso es el cliente mysql-client, que permite interactuar con un servidor MySQL (local o remoto) en modo texto. De este modo es posible realizar todos los ejercicios sobre un servidor instalado localmente o, a través de Internet, sobre un servidor remoto.

Ha ganado popularidad por una serie de atractivas características:

- Está desarrollado en C/C++.
- Se distribuyen ejecutables para cerca de diecinueve plataformas diferentes.
- La API se encuentra disponible en C, C++, Eiffel, Java, Perl, PHP, Python, Ruby y TCL.
- Está optimizado para equipos de múltiples procesadores.
- Es muy destacable su velocidad de respuesta.
- Se puede utilizar como cliente-servidor o incrustado en aplicaciones.
- Cuenta con un rico conjunto de tipos de datos.
- Soporta múltiples métodos de almacenamiento de las tablas, con prestaciones y rendimientos diferentes para poder optimizar el SGBD a cada caso concreto.

- Su administración se basa en usuarios y privilegios.
- Se tiene constancia de casos en los que maneja cincuenta millones de registros, sesenta mil tablas y cinco millones de columnas.
- Sus opciones de conectividad abarcan TCP/IP, sockets UNIX y sockets NT, además de soportar completamente ODBC.
- Los mensajes de error pueden estar en español y hacer ordenaciones correctas con palabras acentuadas o con la letra 'ñ'.
- Es altamente confiable en cuanto a estabilidad se refiere.

SQLite

Es un motor de base de datos SQL embebido. A diferencia de la mayoría de otras bases de datos SQL, no tiene un proceso de servidor independiente. SQLite lee y escribe directamente en archivos de disco normal y el formato de archivo de base de datos es multiplataforma (38).

Ventajas de usar SQLite:

- ✓ Tamaño: tiene una pequeña memoria y una única biblioteca es necesaria para acceder a bases de datos, lo que lo hace ideal para aplicaciones de bases de datos incorporadas.
- ✓ Rendimiento de base de datos: SQLite realiza operaciones de manera eficiente y es más rápido que MySQL y PostgreSQL.
- ✓ Portabilidad: se ejecuta en muchas plataformas y sus bases de datos pueden ser fácilmente portadas sin ninguna configuración o administración.
- ✓ Estabilidad: es compatible con ACID, reunión de los cuatro criterios de Atomicidad, Consistencia, Aislamiento y Durabilidad.
- ✓ Interfaces: cuenta con diferentes interfaces del API, las cuales permiten trabajar con C++, PHP, Perl, Python, Ruby, Tcl, Groovy, Qt ofrece el plugin qsqlite, etc.
- ✓ Costo: es de dominio público y, por tanto, es libre de utilizar para cualquier propósito sin costo y se puede redistribuir libremente.

PostgreSQL

Es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD y con su código fuente disponible libremente. Es el sistema de gestión de bases de datos de código abierto más potente del mercado y en sus últimas versiones no tiene nada que envidiarles a otras bases de datos comerciales (39).

PostgreSQL utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando (39).

Las características más importantes y soportadas son:

- Es una base de datos 100% ACID (en español Atomicidad, Consistencia, Aislamiento y Durabilidad).
- Integridad referencial.
- Tablespaces.
- Nested transactions (savepoints).
- Replicación asincrónica/sincrónica / Streaming replication - Hot Standby.
- Two-phase commit.
- PITR - point in time recovery.
- Copias de seguridad en caliente (Online/hot backups).
- Unicode.
- Juegos de caracteres internacionales.
- Regionalización por columna.
- Multi-Version Concurrency Control (MVCC).
- Múltiples métodos de autentificación.
- Acceso encriptado vía SSL.
- Completa documentación.
- Licencia BSD.
- Disponible para Linux y UNIX en todas sus variantes (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64) y Windows 32/64bit.

1.11.8 OpenSsl

Es un proyecto de código abierto, de grado comercial, juego de herramientas y completo para la *Transport Layer Security* (TLS) y los protocolos *Secure Sockets Layer* (SSL). También es una biblioteca de criptografía de propósito general (40).

El conjunto de herramientas OpenSsl está disponible bajo una licencia de tipo Apache, que básicamente significa que es libre de obtener y usar para fines comerciales y no comerciales sujetos a algunas condiciones de la licencia simples.

1.11.9 Metodología de desarrollo

Hoy día en todo el mundo, se proponen diferentes metodologías en dependencia del tiempo de vida y la complejidad del proyecto que se vaya a desarrollar. Una metodología de desarrollo de software es un marco de trabajo usado para estructurar, planificar y controlar el proceso de desarrollo en sistemas de información (41).

Las metodologías, dadas sus características, se enmarcan en dos grandes grupos, las metodologías tradicionales o pesadas y las metodologías ágiles. La diferencia más notable entre estos dos grupos es que mientras las metodologías tradicionales intentan obtener los resultados apoyándose principalmente en la documentación ordenada y persiguen la consecución de un proyecto cuya planificación está bien definida. Las metodologías ágiles tienen como base de sus resultados la comunicación e interacción directa con todos los usuarios involucrados en el proceso además están pensadas para pequeños grupos de personas. Por tanto, por las características del sistema que se desea construir, resulta conveniente seleccionar una metodología ágil. Entre las más destacadas se pueden nombrar:

Scrum

La metodología Scrum es un modelo de referencia que define un conjunto de prácticas y roles que pueden tomarse como punto de partida para el proceso de desarrollo de software. Esta metodología está indicada para proyectos con un alto grado de cambios en los requisitos. Su principal característica es que el desarrollo de software se realiza mediante iteraciones denominadas sprint, donde cada sprint representa un incremento del producto.

Además se realizan reuniones diarias para la coordinación e integración de las actividades a desarrollar. Aunque Scrum se enfoca en la gestión de procesos de desarrollo de software, puede ser utilizado en equipos de mantenimiento de software, o en una aproximación de gestión de programas (42).

Proceso Unificado Ágil (traducido de las siglas en inglés AUP:*Agile Unified Process*)

Proceso Unificado Ágil es una versión simplificada del Proceso Unificado de Desarrollo de Software (traducido de las siglas en inglés RUP: *Rational Unified Process*), describiendo un enfoque simple y fácil de entender. AUP se enfoca principalmente en la gestión de riesgos, haciendo la propuesta de que aquellos elementos con alto riesgo tengan prioridad en el proceso de desarrollo y sean tratados en etapas tempranas del mismo. El AUP aplica técnicas ágiles incluyendo desarrollo orientado a pruebas, modelado y gestión de cambios ágiles y refactorización de bases de datos para mejorar la productividad (43).

Las principales características de la metodología AUP son:

- ✓ Abarca siete flujos de trabajos, cuatro de ingeniería y tres de apoyo: modelado, implementación, prueba, despliegue, gestión de configuración, gestión de proyectos y ambiente.
- ✓ El modelado agrupa los tres primeros flujos de RUP (modelado del negocio, requerimientos y análisis y diseño).
- ✓ Dispone de cuatro fases igual que RUP: creación, elaboración, construcción y transición.

AUP está basada en principios de simplicidad y agilidad lo que implica una considerable ventaja sobre otras metodologías.

Programación Extrema (traducido de las siglas en inglés XP: *Extreme Programming*)

Programación Extrema es una metodología ligera de desarrollo de software basada en la simplicidad, la comunicación y retroalimentación o reutilización del código desarrollado. Esta metodología se centra en fortalecer las relaciones interpersonales para el éxito del desarrollo de software, promueve el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores y propiciando un buen ambiente de trabajo. XP se basa en la interacción continua entre el cliente y el equipo de desarrollo caracterizándose por poseer soluciones implementadas simples y la capacidad de afrontar los cambios. Debido a estas particularidades, XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, donde existe un alto riesgo técnico (44).

Variación Agile Unified Process UCI (AUP-UCI)

La Variación AUP-UCI es una variación de la metodología AUP, de forma tal que se adapte al ciclo de vida definido para la actividad productiva de la UCI. Una metodología de desarrollo de software tiene entre sus objetivos aumentar la calidad del software que se produce, de ahí la importancia de aplicar buenas prácticas, para ello se apoya en el Modelo CMMI-DEV v1.3. El cual constituye una guía para aplicar las mejores prácticas en una entidad desarrolladora. Estas prácticas se centran en el desarrollo de productos y servicios de calidad (45).

El Proceso Unificado Ágil de Scott Ambler o *Agile Unified Process* (AUP) en inglés es una versión simplificada del *Rational Unified Process* (RUP). Este describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de software usando técnicas ágiles y conceptos y aun así se mantiene fiel a las RUP (46).

AUP-UCI es flexible y propone los mismos roles y artefactos que RUP, solo que no hay necesidad de generar toda la documentación que se requiere en cada flujo de trabajo.

Ventajas:

- ✓ Simplicidad: toda la documentación se describe de manera concisa en el proyecto.
- ✓ Agilidad: el proceso unificado ágil se ajusta a los valores y principios de la alianza ágil. Se centra en actividades de alto valor: La atención se centra en actividades que en realidad cuentan en un proyecto, obviando las que no son necesarias para el desarrollo del mismo.

1.11.10 Herramienta CASE:

Una herramienta CASE es un conjunto de métodos, utilidades y técnicas que facilitan la automatización del ciclo de vida del desarrollo de sistemas de información, completamente o en algunas de sus fases (47).

IBM Rational Rose

Proporciona un conjunto de prestaciones controladas por modelo para desarrollar muchas aplicaciones de software, incluidas aplicaciones Ada, ANSI C++, CORBA, Java, Java EE, Visual C++ y Visual Basic. El software permite acelerar el desarrollo de estas aplicaciones con código generado a partir de modelos visuales mediante el lenguaje UML (*Unified Modeling Language*) (48).

Rational Rose Enterprise ofrece una herramienta y un lenguaje de modelado común para simplificar el entorno de trabajo y permitir una creación más rápida de software de calidad. Modelado de las aplicaciones más habituales: proporciona prestaciones de modelado visual para desarrollar muchos tipos de aplicaciones de software. Desarrollo de aplicaciones para la web: contiene herramientas web y XML para el modelado de aplicaciones web. Integración del diseño de aplicaciones con el desarrollo: unifica el equipo del proyecto proporcionando una ejecución y una notación de modelos UML comunes (48).

Visual Paradigm

Es un potente, multiplataforma y sin embargo fácil de usar, modelado UML y CASE herramienta visual. Proporciona a los desarrolladores de software de la plataforma de desarrollo de vanguardia para construir aplicaciones de calidad más rápido, mejor y más barato (49). Posee capacidades de ingeniería directa e inversa, puede integrarse a los principales entornos de desarrollo y posee disponibilidad de múltiples versiones para cada necesidad.

Las ventajas que proporciona *Visual Paradimg for UML* son (50):

- Dibujo: facilita el modelado de UML, ya que proporciona herramientas específicas para ello. Esto también permite la estandarización de la documentación, ya que la misma se ajusta al estándar soportado por la herramienta.
- Corrección sintáctica: controla que el modelado con UML sea correcto.

- Coherencia entre diagramas: al disponer de un repositorio común, es posible visualizar el mismo elemento en varios diagramas, evitando duplicidades.
- Integración con otras aplicaciones: permite integra se con otras aplicaciones, como herramientas ofimáticas, lo cual aumenta la productividad.
- Trabajo multiusuario: permite el trabajo en grupo, proporcionando herramientas de compartición de trabajo.
- Reutilización: facilita la reutilización, ya que disponemos de una herramienta centralizada donde se encuentran los modelos utilizados para otros proyectos.
- Generación de código: permite generar código de forma automática, reduciendo los tiempos de desarrollo y evitando errores en la codificación del software.
- Generación de informes: permite generar diversos informes a partir de la información introducida en la herramienta.

Para definir la base de datos y la metodología a usar se realizó un estudio profundo de las ventajas y deficiencias que proporcionan cada uno de estos aspectos de manera que permita seleccionar el ciclo de desarrollo más adecuado para la construcción del software, así como un adecuado funcionamiento del sistema mediante la selección de la base de datos.

Se llega a la conclusión que los módulos que poseen los Sistemas SCADA anteriormente mencionados no se puede utilizar porque las funcionalidades que poseen y su forma de integración no es la misma que el Sistema de Medición Arex. Además se decide que la utilización del LDAP para la autenticación de usuarios en el sistema no es necesario ya que solo se va a tener un grupo de usuarios locales y no centralizado en un servidor de directorios.

Las tecnologías a utilizar en el desarrollo del módulo es el lenguaje de programación C++, como IDE de desarrollo Qt Creator, como lenguaje de modelado UML y herramienta CASE Visual Paradigm, metodología AUP-UCI ya que permite generar la documentación necesaria, se recomienda para proyectos pequeños y como es definida y utilizada en el proyecto para el desarrollo del Sistema de Medición Arex se recomienda mantener dicha metodología. Se seleccionó como base de datos SQLite para el almacenamiento de los datos de los usuarios que tendrán acceso a la aplicación debido a que no tiene necesidad de un servidor, haciendo al sistema más ligero; constituye una herramienta libre y utiliza código SQL para la gestión de la base de datos. Además se selecciona la biblioteca Boost para la verificación de la validez del usuario y contraseña. La biblioteca de cifrado SSL y la clase QSslSocket serán utilizadas en la implementación del sistema para la seguridad de las comunicaciones del Sistema de Medición Arex.

Para la validación del usuario es necesario de cumplir con los siguientes parámetros:

- Comenzar con un carácter alfanumérico.
- Debe ser de al menos 4 caracteres de longitud.
- Puede contener números, pero no comenzar con uno.
- Puede contener guiones bajos, puntos o guiones, pero no al comienzo o al final o tener más de uno junto (ae__, ae_- y ae. _ serán inválidos).

Para la validación de la contraseña de debe cumplir los siguientes parámetros:

- No contener las palabras amor, administrador, user, admin, arex, cedin, editor o contraseña.
- Contraseñas que contengan al menos una letra mayúscula.
- Contraseñas que contengan al menos una letra minúscula.
- Contraseñas que contengan al menos un número o carácter especial.
- Contraseñas cuya longitud sea como mínimo 8 caracteres.
- Contraseñas cuya longitud máxima no debe ser arbitrariamente limitada.

Para generar la llave y el certificado digital se utiliza la herramienta de software libre OpenSsl donde se definen el tipo de algoritmo a utilizar y el tiempo de vida que va a tener el mismo. Por lo tanto, se elige el algoritmo de cifrado asimétrico RSA para generar los mismos ya que constituye el algoritmo más potente y utilizado en conexiones seguras en línea y posteriormente ser utilizado en el protocolo de comunicación SSL socket para cifrar la información que se transmite por cada uno de los módulos.

1.12 Conclusiones parciales

Después de hacer un análisis de las herramientas y tecnologías a utilizar en el desarrollo de la aplicación, los elementos expuestos anteriormente se pueden aplicar en el desarrollo del módulo, estas resultaron poseer las características necesarias y suficientes para desarrollar el módulo conforme a las necesidades existentes.

CAPÍTULO 2. Propuesta de solución

2.1 Introducción

En este capítulo se realiza el proceso de modelación del sistema. Para esto se expone una propuesta del sistema, más adelante se especifican los requerimientos funcionales y no funcionales que debe tener la aplicación, se confeccionan las historias de usuarios, la estimación de esfuerzo, el plan de iteraciones y de entregas. Además, la definición de la arquitectura y de los patrones utilizados.

2.2 Definición de la solución

El sistema para la seguridad el control de acceso de los usuarios y las comunicaciones del Sistema de Medición Arex debe ser diseñado para garantizar el correcto control de acceso de los usuarios y el intercambio seguro de datos entre los diferentes módulos que lo componen, sin afectar la disponibilidad del sistema, así como brindar fiabilidad, confidencialidad y rapidez de las comunicaciones. Aplicando cada una de las herramientas y tecnologías seleccionada en el capítulo anterior para su correcto desarrollo y posterior integración con el sistema actual.

El módulo de seguridad se encargará de controlar el acceso de los usuarios al sistema a través del método de autenticación “basado en algo conocido”, para comprobar si cumple con los parámetros de seguridad para que el usuario y la contraseña sean seguras. El intercambio de datos de forma segura se incluirá en cada uno de los módulos del sistema, incluyendo al de seguridad como una mejora ya que permite que el sistema en general pueda garantizar la integridad y la fiabilidad de la información, si el módulo de seguridad no está activo en algún momento. Una vez definida la propuesta de solución se abordaran las fases inicio, ejecución y cierre pertenecientes a la metodología AUP-UCI que guiará el desarrollo.

2.3 Fase de Inicio

Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo y decidir si se ejecuta o no el proyecto (45).

Se definen los módulos fundamentales a los cuáles es necesario hacer la comunicación cifrada, mediante qué protocolo de comunicación se deben comunicar y los parámetros se deben garantizar para que el sistema no afecte la fiabilidad e integridad de la información que se transmite y la portabilidad.

2.3.1 Recopilación de información para el sistema a desarrollar

Para el levantamiento de información del sistema se realizaron entrevistas al grupo de desarrollo del Sistema de Medición Arex. Fueron entrevistados programadores, directivos y analistas para comprender el funcionamiento del sistema y obtener información.

En la entrevista realizada se recopiló la información necesaria para identificar las funcionalidades a desarrollar. También se definieron los sistemas con los cuales es necesario establecer conexión directa.

2.4 Fase de Ejecución

En esta fase se ejecutan las actividades requeridas para desarrollar el software, se incluye el ajuste de los planes del proyecto teniendo en cuenta los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto (45).

2.4.1 Requerimientos del sistema

Los requerimientos de software son especificaciones que debe cumplir el sistema que se va a desarrollar. Estos requerimientos se clasifican en funcionales y no funcionales. Los funcionales son capacidades o condiciones que el sistema debe cumplir, mientras que los no funcionales son propiedades o cualidades que hacen al producto atractivo, usable, rápido y confiable.

Aquí se describen todas las interacciones que tendrán los usuarios sobre la aplicación. Estos requerimientos contienen las historias de usuario y los requisitos no funcionales, los cuales se muestran a continuación.

Requisitos funcionales

De acuerdo a la metodología seleccionada podemos traducir los requerimientos funcionales como historias de usuarios.

- HU1. Leer los datos de configuración del XML.
- HU3. Guardar los datos del XML correspondiente a la etiqueta *security* en una base de datos.
- HU3. Autenticar usuario.
- HU4. Validar usuario.
- HU5. Validar contraseña.
- HU6. Determinar los permisos de cada usuario.
- HU7. Cifrar y descifrar la información que se intercambia entre los módulos de Arex.
- HU8. Recibir los datos desde el Visualizador.

- HU9. Enviar los datos de respuestas a cada uno de los módulos que lo solicite.

Requisitos no funcionales

Los requisitos no funcionales son propiedades o cualidades que el producto debe cumplir. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido y confiable.

Descripción de requisitos no funcionales

Se especifican los requisitos no funcionales identificados que debe cumplir el sistema. Ver anexo 2.

2.4.2 Descripción de las Historias de Usuarios (HU)

Las historias de usuario son la técnica utilizada en AUP-UCI para especificar los requisitos del software. No son más que tarjetas en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales. Cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla. Es importante destacar, que las HU nuevas pueden describirse en cualquier momento, con esto se comprueba la flexibilidad de la metodología (45).

Las HU se representan mediante tablas las cuales contienen las siguientes secciones:

- Número: Las siglas de HU más un número consecutivo, este permite identificar la historia.
- Nombre del requisito: Nombre que identifica la HU.
- Programador: Persona encargada de desarrollar la HU.
- Prioridad: Esta característica es dada por el cliente con los valores: alta, media o baja en dependencia de la importancia y orden en que desean que sean implementadas.
- Iteración Asignada: Número de la iteración en la cual se desarrollará la HU.
- Tiempo Estimado: Tiempo estimado en semanas que se le asignará.
- Descripción: Breve descripción del proceso que define la historia.
- Riesgo en Desarrollo: Depende de la complejidad que posea la funcionalidad a desarrollar.
- Observaciones: Alguna acotación importante de señalar acerca de la historia.

Se confeccionaron 9 HU, se tomaron para los puntos estimados la unidad como una semana de trabajo. Se definieron 7 HU con prioridad alta y 2 HU con prioridad media. Ver anexo 3.

2.4.3 Estimación de esfuerzo por historia de usuario

Las historias de usuarios seleccionadas para cada entrega son desarrolladas y probadas en un ciclo de iteración, de acuerdo al orden preestablecido. Al comienzo de cada ciclo, se realiza una reunión de planificación de la iteración. Cada historia de usuario se traduce en tareas específicas de programación. A

continuación, en la tabla (Ver anexo 4) se resume la estimación del esfuerzo realizada por parte de los desarrolladores.

2.4.4 Plan de iteraciones

Luego de identificar y definir cada una de las HU y de la estimación del esfuerzo necesario para realizarlas, se debe conformar el plan de iteraciones. Las HU seleccionadas para cada iteración son desarrolladas y probadas de acuerdo al orden de prioridad establecido.

El desarrollo fue dividido en tres iteraciones, para las cuales se desarrollaron partes funcionales de la aplicación. A continuación, se describen cada una de las iteraciones anteriormente mencionadas y la justificación de su selección:

- ✓ **Iteración 1:** Para esta iteración se desarrollan las HU de la 1 a la 3 correspondientes a autenticar usuario, leer los datos de configuración del XML, guardar los datos del XML correspondiente a la etiqueta *security* en una base de datos SQLite. Las HU seleccionadas para esta iteración permiten asegurar la seguridad del sistema mediante la autenticación de usuario, el administrador principal puede realizar acciones sobre cada usuario registrado, además se permite cargar los usuarios, grupos y permisos que cada uno posee a partir de la lectura de un archivo XML que tiene dichos datos y se realizan las pruebas de aceptación. Se realiza una entrega funcional al finalizar la iteración.
- ✓ **Iteración 2:** En esta iteración se desarrollan las HU de la 4 a la 6 correspondientes a validar usuario, validar contraseña y determinar los permisos de cada usuario. Se realizan las pruebas de aceptación y al finalizar la iteración se realizan una entrega funcional.
- ✓ **Iteración 3:** Para la iteración se desarrollan las HU de la 7 a la 9 pertenecientes a cifrar y descifrar la información que se intercambia entre los módulos de Arex, recibir los datos desde el Visualizador y enviar los datos de respuestas a cada uno de los módulos que lo solicite. En esta iteración se desarrollan los elementos asociados de inicio para la comunicación cifrada entre los módulos y la recepción y envió de datos hacia los demás módulos. En la iteración se realizan las pruebas de aceptación y se realiza una entrega funcional.

Para aproximar el tiempo de ejecución de cada iteración, se tomó como medida que cada semana consta de 6 días (lunes, martes, miércoles, jueves, viernes y sábado) en los que se trabajaban 6 horas sin distracciones. En la tabla se muestra el plan de iteraciones, este incorpora el tiempo estimado para cada una de las iteraciones y las HU que se van a desarrollar. La segunda iteración es una poco más larga ya que contiene las HU de mayor complejidad. Ver anexo 5.

2.4.5 Plan de entregas

El plan o cronograma de entregas establece qué HU son agrupadas para conformar una entrega, y el orden de las mismas. Este cronograma es el resultado de una reunión entre todos los actores del proyecto (cliente, desarrolladores, gerentes, etc.). El cronograma de entregas se realiza en base a las estimaciones de tiempos de desarrollo realizadas por los desarrolladores (51).

A partir del plan de iteraciones analizado en el acápite anterior, y en correspondencia con el mismo se realiza el plan de entregas, que se muestra en la tabla siguiente. En el cual se proponen tres versiones funcionales. Ver anexo 6.

2.5 Definición de la Arquitectura

La Arquitectura de Software es la organización fundamental de un sistema la cual está compuesta por un conjunto de patrones que proporcionan una guía para el desarrollo del software durante toda su evolución.

2.5.1 Arquitectura cliente-servidor

La arquitectura cliente-servidor es el procesamiento cooperativo de la información por medio de un conjunto de procesadores, en el cual múltiples clientes, distribuidos geográficamente, solicitan requisitos a uno o más servidores centrales. Este modelo provee usabilidad, interoperabilidad, flexibilidad y escalabilidad en las comunicaciones (52).

En esta arquitectura el cliente envía un mensaje al servidor solicitando un servicio (en el caso del sistema la verificación de los datos de usuarios y la clave para la comunicación) y este envía uno o varios mensajes con la respuesta. La siguiente ilustración muestra el esquema de comunicación cliente-servidor empleado.

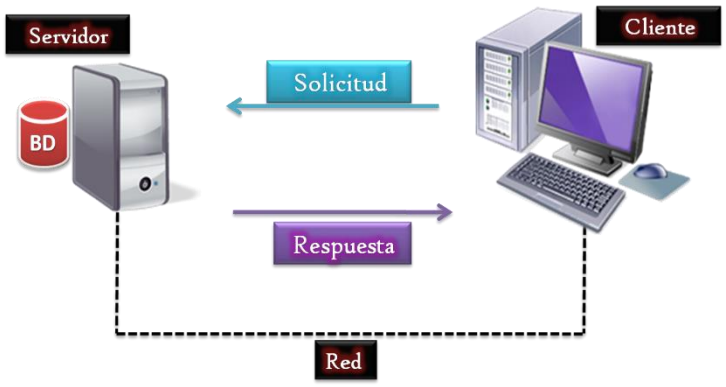

Ilustración 9. Arquitectura cliente-servidor

Los patrones arquitectónicos son patrones de software que ofrecen soluciones a problemas de arquitectura en ingeniería de software. Dan una descripción de los elementos y el tipo de relación que tienen, junto con un conjunto de restricciones sobre cómo pueden ser usados. Un patrón arquitectónico expresa un esquema de organización estructural esencial para un sistema de software, que consta de subsistemas, sus responsabilidades e interrelaciones.

2.6 Descripción del sistema

El sistema para la seguridad de las comunicaciones, gestión de sesiones y control de acceso a usuarios del Sistema de Medición Arex está compuesto por dos subsistemas:

ServerManagement: Subsistema que gestiona que módulo se puede conectar a él, permite verificar la autenticación de cada usuario sea correcta.

Connection: Subsistema que se encarga de iniciar y cerrar la conexión mediante socket de los clientes que intenten conectarse con el módulo. Además, se encarga de recibir y enviar los datos que necesite cada módulo.

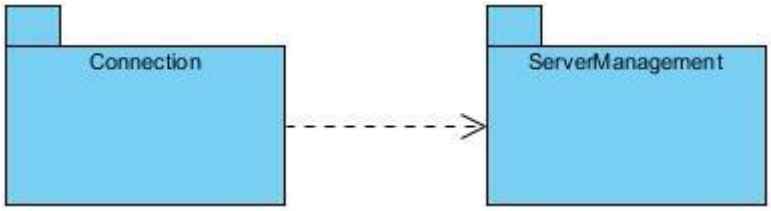

Ilustración 10. Diagrama de subsistemas de la solución propuesta

2.7 Diagrama de clases

El diagrama de clases captura la estructura lógica del sistema y las clases que constituyen el modelo. Es un modelo estático, se describe lo que existe y qué atributos y comportamiento tiene, más que cómo se hace algo. Los diagramas de clases son los más útiles para ilustrar las relaciones entre las clases e interfaces. Las generalizaciones, las agregaciones y las asociaciones son todas valiosas para reflejar la herencia, la composición o el uso y las conexiones respectivamente (53).

Los diagramas de clases son utilizados durante el proceso de análisis y diseño de los sistemas, donde se crea el diseño conceptual de la información que se manejará en el sistema, y los componentes que se encargarán del funcionamiento y la relación entre uno y otro. A continuación, el diagrama de clases realizado para llevar a cabo la aplicación. Ver anexo 7.

2.8 Modelo de datos

En la informática, un modelo de datos es un lenguaje utilizado para la descripción de una base de datos. Por lo general, un modelo de datos permite describir las estructuras de datos de la base (el tipo de los

datos que incluye la base y la forma en que se relacionan), las restricciones de integridad (las condiciones que los datos deben cumplir para reflejar correctamente la realidad deseada) y las operaciones de manipulación de los datos (agregado, borrado, modificación y recuperación de los datos de la base) (54). A continuación, se muestra el modelo de datos. Ver anexo 8.

2.9 Tarjetas CRC

Para poder diseñar el sistema como un equipo, se debe cumplir con tres principios: Cargo o Clase, Responsabilidad y Colaboración (CRC). Las tarjetas CRC permiten desprenderse del método de trabajo basado en procedimientos y trabajar con una metodología basada en objetos. Además permiten que el equipo completo contribuya en la tarea del diseño. Ver anexo 9.

2.10 Patrones de diseño

Los patrones de diseño son aquellos que expresan esquemas para definir estructuras de diseño (o sus relaciones) con las que construir un software. Los patrones de diseño GOF tienen diferentes clasificaciones en dependencia de su funcionalidad (55).

- Patrones de creación: muestran la guía de cómo crear objetos cuando sus creaciones requieren tomar decisiones. estas decisiones serán resueltas dinámicamente decidiendo que clases instanciar o sobre que objeto(s) delegará responsabilidades.
- Patrones estructurales: describen la forma en que diferentes tipos de objetos pueden ser organizados para trabajar unos con otros.
- Patrones de comportamiento. se utilizan para organizar, manejar y combinar comportamientos.

Como patrón creacional se utilizó el Singleton o instancia única para restringir la instanciación del acceso a la clase arexcrypt. Ver anexo 10.

Los patrones de diseño GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones (55).

Patrones básicos de asignación de responsabilidades son experto, creador, alta cohesión, bajo acoplamiento y controlador. En la solución del sistema se aplicaron principalmente los siguientes patrones de asignación de responsabilidades (GRASP) del diseño:

- **Bajo acoplamiento:**

Asignar una responsabilidad para mantener bajo acoplamiento. Este patrón se utilizó con el objetivo de tener las clases menos ligadas entre sí. De tal forma que, en caso de producirse una modificación en

alguna, se tenga la mínima repercusión posible en el resto de clases, potenciando la reutilización y disminuyendo la dependencia entre ellas. Ver anexo 11.

- **Controlador:**

Asignar la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase. Se evidencia en las clases controladoras que se encargan de obtener los datos y enviarlos a las vistas. Ver anexo 12.

2.11 Conclusiones Parciales

Después de la ejecución de las primeras dos fases y parte de la tercera fase propuestas por la metodología AUP-UCI se llegaron las siguientes conclusiones parciales. En la fase de inicio se realizó un correcto levantamiento de requisitos, los elementos de confiabilidad y portabilidad que es necesario suplir y las funcionalidades con las que debe contar el sistema. En la fase de ejecución se realizó la estimación de esfuerzo por HU y a partir de allí se define el plan de iteraciones, con tres iteraciones, se concilio el plan de entregas con tres entregas funcionales. Se analizó el diseño de la aplicación, la aplicación del patrón de arquitectura cliente-servidor y la utilización de patrones de diseño. En la fase de cierre se realizó el análisis de los resultados alcanzado.

CAPÍTULO 3. Implementación y Prueba

En este capítulo se explicará el estándar de codificación utilizando para el desarrollo de la aplicación, es válido señalar que estos son lo que se utilizan en todas las soluciones que pertenecen al proyecto. Además, se explicará mediante un diagrama de despliegue como será incluido el módulo de seguridad en el sistema y se realizarán diferentes pruebas al software.

3.1 Implementación del sistema

En la implementación del sistema se definen los elementos de cómo se estructura el código, los comentarios de cada línea de código y cómo se integrará la solución al actual sistema.

3.1.1 Tareas de ingeniería o programación

La planificación de las tareas de ingeniería o programación se encuentran relacionadas con cada una de las iteraciones, pues las HU se transforman en tareas que son desarrolladas por los programadores del equipo de desarrollo (44). En correspondencia de las HU se realizó la distribución de cada tarea para cada iteración. Ver anexo 13.

La tabla 30 mostrada anteriormente fue escogida, porque a pesar de ser la iteración 1 la más corta de ella dependen en gran medida el desarrollo de las próximas iteraciones, ya que en ella se realizan actividades como leer los elementos de un proyecto desde el archivo XML. Cada tarea de desarrollo fue realizada en un período de cinco a dieciochos días. Para la confección de las tareas se utilizaron tablas que contienen los siguientes campos:

- No. de tarea: numeración continua que identifica a la tarea.
- No. de HU: número de la HU a la cual pertenece.
- Nombre de la tarea: identificación literal de la tarea.
- Tipo de tarea: tipo de tarea, dígase diseño, desarrollo, prueba.
- Puntos estimados: representación en porciento de la cantidad de tiempo estimada de una semana, que se utilizara para su realización.
- Fecha inicio: fecha estimada de inicio de realización.
- Fecha fin: fecha estimada de fin de realización.
- Descripción: se describe en que consiste la tarea y que elementos deben cumplirse para declarar la tarea terminada.

En las tablas desde la 31 hasta la 33 se muestran las tareas asociadas a la HU3 pertenecientes al proceso autenticar usuario. Ver anexo 14.

3.1.2 Diagrama de despliegue

Un diagrama de despliegue muestra cómo y dónde se desplegará el sistema. Las máquinas físicas y los procesadores se representan como nodos, y la construcción interna puede ser representada por nodos o artefactos embebidos. Como los artefactos se ubican en los nodos para modelar el despliegue del sistema, la ubicación es guiada por el uso de las especificaciones de despliegue (53). Ver anexo 15.

3.2 Estándar de codificación

3.2.1 Estilo de codificación

- Se establece un tamaño de indentación[2] estándar de tres espacios, sin tabulaciones.
- Alinear secciones del código.
- Alinear verticalmente llaves de apertura y cierre.
- Usar espacios antes y después de los operadores que el lenguaje de programación permita.
- Emplear líneas en blanco para organizar el código, permitiendo crear párrafos de código para una mejor lectura.
- Evitar colocar más de una sentencia por línea.
- Emplear constantes en sustitución de números o cadenas de caracteres literales.
- Minimizar el alcance de las variables para evitar confusión y facilitar el mantenimiento.
- Emplear cada variable y rutina solo para un propósito.
- Evitar el uso de variables públicas, sustituirlas por variables privadas y métodos que provean el valor de tal variable, para mantener el encapsulamiento.
- Emplear las letras i, j, k, l, m, p, q, r para contadores en ciclos.
- Emplear correctamente los tipos de ciclos: si es al menos una vez usar do-while, si es ninguna o más veces usar while-do, y si se conoce el número exacto de ciclos usar for.
- Emplear líneas en blanco para separar pasos lógicos (Ej.: declaraciones, lazos).

3.3 Nombre de estructuras

- Los nombres de las clases son sustantivos singulares.
- Los nombres de clases y objetos deben reflejar qué hacen y no cómo lo hacen.
- Escoger nombres lo suficientemente largos que expresen correctamente el sentido de lo que se quiere, pero evitando manejar nombres que dificulten la labor de implementación.
- Evitar nombres que permitan una interpretación subjetiva (evitar ambigüedad y asegurar abstracción).
- Evitar redundancia no repitiendo nombres de clases en sus elementos.

[2] Espacio o sangría que se pone a la derecha de cada línea de código.

- Dado que los nombres generalmente son el producto de concatenar varias palabras, se debe emplear la primera palabra en minúscula, mayúscula para denotar la letra de inicio de cada una de las palabras restantes por las que esté formado y minúscula para las letras intermedias en el caso de los nombres de métodos y funciones. Para el caso de los nombres de variables y atributos debe aplicarse la misma convención.
- Los nombres de constantes deben contener solo letras mayúsculas.
- Minimizar el uso de abreviaturas. En caso de ser requeridas, se debe ser consistente en su uso y cada abreviatura debe significar solo una cosa.

3.4 Comentarios de código

Se utilizó la herramienta Doxygen para generar la documentación del código de las distintas clases utilizadas para la implementación de las bibliotecas, los formatos brindados por esta herramienta se explican a continuación.

Estilo de bloques de documentación

Se adopta el estilo de bloques de documentación JavaDoc, el cual consiste en un bloque de comentario de estilo C. Este bloque de comentario comienza con dos asteriscos, los asteriscos que se encuentran en la línea de la mitad son opcionales.

Ejemplo:

```
/**
* Texto
*/
```

Descripción breve con el comando \brief o @brief

Para hacer una descripción breve se adopta el uso del comando \brief o @brief en el bloque de comentarios ya descrito.

La acción de este comando termina al final de un párrafo, de tal manera que la descripción detallada sigue después de una línea vacía, tal como lo muestra el ejemplo:

```
/**
* @brief descripción breve.
* Continuación de la descripción breve.
* La descripción detallada comienza aquí, nótese
* que se debe dejar una línea en blanco para lograr
* tener las dos descripciones (breve y detallada)
```

*/

Nota: Si no se utiliza el comando @brief, Doxygen toma la descripción hecha en el bloque como una descripción detallada y no como una descripción breve.

Descripción de argumentos y métodos

La descripción de argumentos de métodos y funciones se encuentra en línea, es decir, luego de la declaración de cada uno de los argumentos se da una breve descripción de cada uno de ellos. En el siguiente ejemplo se muestra el formato a utilizar:

```
int multFunction (int c, /**<Primer operando de la operación */
int d, /**<Segundo operando de la operación */
int e /**<Tercer operando de la operación */);
```

Documentación de tipos de datos

Para describir la función y los elementos que componen los tipos de datos definidos se pueden utilizar los formatos especificados en los apartados anteriores. A continuación se muestra un ejemplo:

```
/**
* @brief Enumerado de los tipos de datos admitidos
*
* Descripción más detallada de la función de este tipo de dato.
*/ enum DataTypes {INTEGER, /**<Puede ser un valor de tipo entero*/
DOUBLE, /**<Puede ser un valor de tipo double*/
```

Observamos que se genera una descripción breve seguida de una descripción más detallada para la función, luego se explica brevemente el valor de retorno de la misma y la descripción de los parámetros usando el formato de documentación en línea.

Comandos @author y @date

Es importante que se especifique el nombre del autor y la fecha de creación de cualquier estructura en un código, para ello se utilizan los comandos @author y @date, para el nombre del autor y la fecha respectivamente, en el siguiente ejemplo se puede ver la acción de estos comandos:

```
/**@brief Descripción breve
* Aquí comienza la descripción detallada
* @author Enrique Santos Leyva
* @date 21/01/2016
*/
```

Comando @see

Existe otro comando útil que permite hacer referencias a otras clases o métodos cuando esto sea necesario, es importante usar este comando en la documentación a la hora de implantar los métodos o funciones propias de alguna clase, este comando es @see y su uso se muestra en el siguiente ejemplo:

```
/**
* Constructor de la clase
* @see Test:Test()
*/
Test1();
```

Esto genera en la documentación para el constructor de la clase de prueba Test1 una referencia hacia la documentación existente para el constructor de la clase de prueba Test.

Nota: Si se quiere hacer una referencia desde cualquier estructura hacia la documentación completa de una clase, se debe utilizar el comando @see seguido del nombre de la clase de esta forma:

```
/**
* Constructor de la clase
* @see Clase
*/
Test1();
```

3.5 Pruebas

La realización de pruebas es una actividad en la cual un sistema o componente es ejecutado bajo ciertas condiciones o requisitos específicos, cuyos resultados son observados, registrados y evaluados. Estas técnicas ayudan a definir un conjunto de casos de pruebas aplicando un cierto criterio, estos son determinados por los valores a asignar a las entradas en su ejecución, realizando de esta forma un conjunto de pruebas internas a cada uno de los componentes del sistema atendiendo a las funcionalidades a cumplir por este y de aceptación por parte del cliente.

Las pruebas de aceptación son las pruebas finales antes del despliegue del sistema. Su objetivo es verificar que el software está listo y que puede ser usado por usuarios finales para ejecutar aquellas funciones y tareas para las cuales el software fue construido. A continuación las pruebas de aceptación: Ver anexo 16.

En la primera iteración se realizaron dos etapas de pruebas, en la primera etapa se realizaron 4 casos de prueba donde se obtuvieron 2 no conformidades, las cuales se corrigieron y se realizó una segunda etapa

de pruebas donde no se obtuvieron no conformidades, logrando así un 100% de resultados satisfactorios y cumpliendo el criterio de aceptación.

En la segunda iteración se realizaron 6 casos de prueba en la primera etapa de pruebas, las cuales fueron detectadas 3 no conformidades. En la segunda etapa de pruebas se detectaron 2 no conformidades. En la tercera etapa de pruebas no se detectaron no conformidades logrando así el criterio de aceptación.

En la tercera iteración se realizaron 12 casos de prueba en la primera iteración, donde se detectaron 5 no conformidades. En la segunda etapa se realizaron 14 casos de pruebas los cuales se detectaron 4 no conformidades. En tercera etapa de pruebas se obtuvo 100% satisfactorias logando el criterio de aceptación, dando fin a la fase de la iteración. La siguiente imagen muestra las diferentes etapas de pruebas realizadas por iteraciones. Ver anexo 19.

Las pruebas internas verifican el resultado de la implementación probando cada construcción, incluyendo tanto las construcciones internas como intermedias, así como las versiones finales a ser liberadas. Se deben desarrollar artefactos de prueba como: diseños de casos de prueba, listas de chequeo y de ser posibles componentes de pruebas ejecutables para automatizar las pruebas.

Las pruebas internas o de caja blanca constan de tres tipos de pruebas principales las pruebas de camino o ruta básica, pruebas de condiciones y pruebas de bucles. En el caso específico del producto que se está desarrollando se le aplicarán pruebas de camino básico (56).

3.5.1 Pruebas de camino básico

Las pruebas de camino básico se basan en la complejidad del flujo de ejecución desglosado en un conjunto básico de caminos. Los casos en los que se divide la prueba, obtenidos en función del conjunto básico, garantizan que se ejecuta por lo menos una vez cada sentencia del programa.

Gráficos de flujo

El grafico de flujo representa los caminos de ejecución que se pueden seguir en un procedimiento. Es un grafo donde los vértices instrucciones o conjunto de instrucciones que se ejecutan como una unidad como puede ser la asignación de variables, sentencias de control de iteración en bucles y sentencias de comprobación de la iteración en grupos de control. Las aristas representan la posibilidad de que una vez terminada la ejecución de un vértice pase a ejecutarse otro. El gráfico se obtiene al observar el código y asignarle el número de nodo correspondiente (57).

Componentes de la gráfica de flujo:

- **Aristas (E):** caminos a recorrer entre nodos o flujo de control.
- **Nodos (N):** representan una o más sentencias procedimentales.
- **Nodo predicado (P):** nodo del que emanan más dos aristas (if).
- **Región (R):** área que se limitan por aristas y nodos.

Fue seleccionado para realizarle la prueba de camino básico el método *authentication* de la clase *serverManagement*. La siguiente imagen muestra el número de nodo que le corresponde a cada línea de código. Ver anexo 17.

El siguiente grafo muestra la instrucción correspondiente a cada nodo y las regiones en que se subdivide el grafo. Ver anexo 18.

El cálculo de la complejidad ciclomática (CC (G)) puede realizarse de tres formas distintas:

1. Número de regiones en las que se subdivide el grafo. En este caso sería CC (G) = 2, pues como se puede observar en la imagen anterior el grafo está subdividido en 6 regiones.
2. Otra manera de calcularla es a través de la fórmula CC (G) = Aristas – Vértices + 2. Específicamente en este caso sería CC (G) = 6 – 6 + 2 = 2.
3. La tercera y última forma de calcular la complejidad ciclomática es: CC (G) = P + 1, donde P es el número de vértices desde donde parte más de una arista. Quedaría: CC (G) = 1 + 1 = 2.

Como el cálculo de las tres complejidades ciclomáticas arrojan el mismo resulta se puede concluir que el cálculo fue correctamente realizado, por tanto, el resultado es satisfactorio. A partir del valor de la complejidad ciclomática obtenemos el número de caminos independientes, que dan un valor límite para el número de pruebas a diseñar.

3.6 Conclusiones

Después de haberse realizado la implementación de la aplicación con los estándares establecidos y haber realizado las diferentes pruebas, se obtiene un producto que cumple todos los requisitos para ser utilizado por el Sistema de Medición Arex.

CONCLUSIONES GENERALES

Luego de terminado el proceso de investigación, desarrollo y validación de la aplicación realizada, se arriba a las siguientes conclusiones:

- ✓ Mediante la aplicación de la clase QSslSocket se logra reducir el nivel de vulnerabilidades en las comunicaciones del Sistema de Medición Arex logrando que la información que se intercambia se haga de forma encriptada.
- ✓ El módulo desarrollado incluye la administración de acceso de los usuarios al sistema por autenticación (usuario y contraseña), detención de intentos fallidos y validación de usuarios y contraseñas garantizando el decremento del nivel de vulnerabilidad que hoy presenta el Sistema de Medición Arex.

RECOMENDACIONES

Se recomienda:

- Incorporar al sistema la posibilidad de configurar los permisos para acceder a las variables que se manejan en el sistema por cada usuario.
- Desplegar la solución en la próxima versión del sistema.
- Ampliar los métodos de autenticación mediante los parámetros biométricos (huella dactilar, iris del ojo).

BIBLIOGRAFÍA

1. UNAM-CERT. UNAM-CERT(Equipo de Respuesta a Incidentes de Seguridad). [En línea] [Citado el: 12 de 01 de 2016.] http://www.cert.org.mx/estadisticas.dsc.

2. Introducción a la arquitectura del "Guardián del ALBA". industrial(CEDIN), Colectivo de autores del Centro de Informática. Carretera a San Antonio Km 2 1⁄2 . Torrens. Boyeros. La Habana. Cuba : s.n., 2013.

3. Vargas, Arianna Gómez. Plan de Desarrollo de Software. La Habana,Cuba: s.n.

4. Trabajo de Grados.Ingeniería de Sistemas.Pontificia Universidad Javerina. [En línea] 10 de 2006. [Citado el: 03 de 05 de 2016.] http://pegasus.javeriana.edu.co/.

5. Pfleeger, Charles P. Security in computing(4th Edition). 2006.

6. López, Purificación Aguilera. Seguridad informática. s.l. : Editex, 2010.

7. Conf 2 Principales conceptos de Seguridad Informática.Redes y Seguridad Informatica. Santos, Adriana. Habana : s.n., 2014.

8. Prieto, José-Luis. 1984-2008. Glosario de Terminología Informática. [En línea] [Citado el: 10 de 12 de 15.]

9. Marañón, Gonzalo Álvarez. [En línea] [Citado el: 14 de 03 de 2016.] http://www.iec.csic.es/CRIPTONOMICO.

10. Miltre, Hector B. SlideShare. [En línea] Universidad de Panama, 30 de 04 de 2012. [Citado el: 30 de 10 de 2015.] http://www.slideshare.net/hmitre17/autenticacion.

11. Bermúdez, Enrique Reyes. Sistema de autenticación para el módulo Seguridad. Habana : s.n., 2009.

12. Alegsa.com.ar(DICCIONARIO DE INFORMÁTICA Y TECNOLOGÍA). [En línea] 12 de 05 de 2010. [Citado el: 15 de 10 de 2015.] http://www.alegsa.com.ar/Dic/sesion.php.

13. IBM Corporation. IBM Knowledge Center. [En línea] [Citado el: 13 de 1 de 16.] http://www-01.ibm.com/support/knowledgecenter/SSPREK_6.1.0/com.ibm.itame.doc_6.1/am61_webseal_admin777.htm?lang=es.

14. Alegsa. Alegsa. [En línea] [Citado el: 15 de 03 de 2016.] http://www.alegsa.com.ar/Dic/usuario.php.

15. Protocolo y Etiqueta. [En línea] [Citado el: 13 de 1 de 16.] https://www.protocolo.org/usuarios/registro.html.

16. Red Tauros. [En línea] [Citado el: 23 de 10 de 2015.] http://www.redtauros.com/Clases/Linux/Linux_serv_ldap.pdf.

17. Definición.de. [En línea] [Citado el: 13 de 1 de 16.] http://definicion.de/tcp-ip/.

18. Taringa!-Intelogencia Colectiva. [En línea] 01 de 06 de 2007. [Citado el: 10 de 12 de 2016.] http://www.taringa.net/posts/info/828082/NETBIOS-estas-a-salvo-Una-legitima-puerta-trasera.html.

19. Definición.de. [En línea] [Citado el: 13 de 1 de 16.] http://definicion.de/ip/.

20. Babylon. [En línea] [Citado el: 13 de 1 de 16.] http://diccionario.babylon.com/appletalk/.

21. AJPDsoft. [En línea] [Citado el: 13 de 12 de 2015.] http://www.ajpdsoft.com/modules.php?name=Encyclopedia&op=content&tid=752.

22. Penín, A. R. Sistemas SCADA. Barcelona, España. : Macombo: 2da edición. S.L., 2007.

23. S21sec. Web de Seguridad.Sugerencias para mejorar la seguridad en SCADA. [En línea] [Citado el: 17 de 02 de 2016.] http://www.s21sec.com.

24. Progea. Industrial Automation Software Progea. [En línea] [Citado el: 15 de 13 de 2016.] http://www.progea.com/ .

25. Organización Automatas. Sistemas SCADA. [En línea] 02 de 03 de 2006. [Citado el: 17 de 03 de 2016.] http://www.automatas.org/redes/scada.htm.

26. Siemens. Siemens. [En línea] [Citado el: 13 de 1 de 16.] http://www.siemens.com.ar/sites/internet/legacy/sie-pe/pe/pdf_catalogos/kb_wincc_62_news_es%5B2%5D.pdf.

27. Introducción a la arquitectura del "Guardián del ALBA". (CEDIN), Colectivo de autores del Centro de Informática industrial. Carretera a San Antonio Km 2 1⁄2 . Torrens. Boyeros. La Habana. Cuba: s.n., 2013. Arquitectura y Configuración del Guardián del ALBA. pág. 22.

28. INTRODUCCIÓN A LA CRIPTOGRAFÍA. Paredes, Gibrán Granados. 7, 2007, Vol. 7.

29. Juan Zamorano. Datsi. [En línea] [Citado el: 16 de 12 de 2016.] https://www.datsi.fi.upm.es/docencia/SEUM/.

30. Galeano, Gustavo. Programación de sistemas embebidos en C. s.l. : ALFAOMEGA GRUPO EDITOR, 2009.

31. Carlos Augusto Natera, Manuel Andres Meneses. SISTEMAS EMPOTRADOS . SISTEMAS EMPOTRADOS . Maturin,Venezuela: s.n., 2015. 13.

32. ElectroLinux. [En línea] 05 de 12 de 2014. [Citado el: 13 de 01 de 2016.] http://www.electrolinux.cl/doku.php/embebidos/conceptos01.

33. BeeCrypt. [En línea] [Citado el: 15 de 05 de 2016.] http://beecrypt.sourceforge.net/doxygen/c++/index.html.

34. The Qt Company. Qt Documentation. [En línea] 2016. [Citado el: 12 de 12 de 2015.] http://doc.qt.io/qt-5/qsslsocket.html.

35. Sorio, R. P. Sistema de Vigilancia por Cámaras Cancerbero(SVCC).Trabajo de Diploma. Universidad de las Ciencias Informáticas.Cuba: s.n., 2010.

36. Code Search. [En línea] [Citado el: 16 de 03 de 2016.] http://code.metager.de/source/xref/lib/qt/src/network/ssl/qsslsocket.cpp.

37. García, Lic.Rosa María Mato. Diseño de bases de datos. 1999.

38. SQlite. [En línea] [Citado el: 14 de 1 de 16.] http://www.sqlite.org.

39. PostgreSQL-es. [En línea] 02 de 10 de 2010. [Citado el: 10 de 02 de 2016.] http://www.postgresql.org.es/sobre_postgresql.

40. OpenSsl.Cryptography and SSL/TLS Toolkit. [En línea] 2015. [Citado el: 16 de 03 de 2016.] https://www.openssl.org/.

41. Boehm, Barry W. A Spiral Model of Software Development an Eahancement. 1988.

42. Schwaber, Ken. Advanced Development Methods. SCRUM Development Process Retrieved. . 2010.

43. Sommerville, Ian. Ingeniería de Software. Madrid : PERSON EDUCATION. SA., 2005.

44. Letelier, Canós, José H. & Penadés. Metodologías ágiles en el desarrollo de software: Extreme Programming (XP). 2003.

45. Sánchez, Tamara Rodríguez. Metodología de desarrollo para la Actividad Productiva de la UCI. Habana : s.n., 2014.

46. Scott W. Amber . Ambysoft. [En línea] [Citado el: 14 de 01 de 2016.] www.ambysoft.com/unifiedprocess/agileUP.html.

47. Lobato, Víctor Bello. Scribd. [En línea] 05 de 07 de 2008. [Citado el: 16 de 12 de 2015.] http://www.scribd.com/doc/3062020/Capitulo-I-HERRAMIENTAS-CASE.

48. IBM. [En línea] [Citado el: 15 de 04 de 2016.] http://www-03.ibm.com/software/products/es/enterprise.

49. Visual Paradigm Product Overview. [En línea] [Citado el: 14 de 1 de 16.] http://www.visual-paradigm.com/support/documents/vpuserguide/12/13/5963_visualparadi.html.

50. Guión Visual Paradigm for UML. [En línea] 2013. [Citado el: 15 de 02 de 2016.] http://www.ie.inf.uc3m.es/grupo/docencia/reglada/Is1y2/PracticaVP.pdf.

51. Joskowicz, José. Reglas y Prácticas en eXtreme Programming. Vigo: s.n., 2008.

52. Márquez Avendaño, Bertha Mariel y Zulaica Rugarcía, José Manuel. Colección de Tesis Digitales de la Universidad de las Américas Puebla. [En línea] 2004. [Citado el: 12 de 05 de 2016.] http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/marquez_a_bm/capitulo_5.html.

53. Guía de Usuario de Enterprise Architect. [En línea] 18 de 04 de 2010. [Citado el: 10 de 05 de 2016.] http://www.sparxsystems.com.ar/new/resources/index.php.

54. autores, Colectivo de. Component Source. [En línea] 2010. [Citado el: 06 de 05 de 2016.] http://www.componentsource.com/products/activereports-6/index-es.html.

55. Larman, Craig. UML y Patrones.Introducción al análisis y diseño orientado a objetos . México: s.n., 1999.

56. Expósito, Alicia Barbara. Prueba de caja blanca. 2012.

57. Santana, Alicia Bárbara Expósito. Pruebas de caja blanca. 2012.

58. José H. Canós, Patricio Letelier yMª Carmen Penadés. Métodologías Ágiles en el Desarrollo de Software. Valencia: s.n., 2004.

59. Roberth G. Figueroa, Camilo J. Solís, Armando A. Cabrera. METODOLOGÍAS TRADICIONALES VS. METODOLOGÍAS ÁGILES. 2010.

60. Unidad III Diseño y Programación de Interfaces HMI. Rojas, M.C. Juan Carlos Olivares. Monterry: s.n., 2015. págs. 09-13.

61. Duque, Raúl González. Python para todos. 2013.

62. Andrés Marzal, Isabel Gracía. Introducción a la programación con Python. Castellón : s.n., 2003.

63. Rossum, Guido van. Guía de aprendizaje de Python. s.l.: Python Software Foundation, 2014.

64. Colaboración. Framework - EcuRed. EcuRed. [En línea] http://www.ecured.cu/Framework.

65. Márquez, José E. Briseño. TRANSMISIÓN DE DATOS. Mérida: s.n., 2005.

66. Anónimo. Sistema de adquisición de Datos. Omega Engineering. [En línea] OMEGA, 25 de 04 de 2012. [Citado el: 18 de 01 de 2016.] http://es.omega.com/prodinfo/adquisicion-de-datos.html.

67. Kaplan-Moss, Adrian Holovaty y Jacob. El libro de Django . 2008.

68. MONTERO, SERGIO INFANTE. Maestros del web. 2012.

69. Suaza, Katerine Villamizar. Definición de equivalencias entre historias de usuario y especificaciones en UN-LENCEP para el desarrollo ágil de software. Medellín: s.n., 2013.

70. Estructura de las Aplicaciones Orientadas a Objetos. El patrón Modelo-Vista-Controlador (MVC). Mestras, Juan Pavón. Madrid: s.n., 2009.

71. Sistema SCADA. Yanet Nieto Doce, Ariangna Garces Gilart, Luis Enrique García Hernández, Ariel Chávez Lorenzo y Amado Espinosa Hidalgo. La Habana: s.n., 2013.

72. UNAM-CERT. UNAM-CERT(Equipo de Respuesta a Incidentes de Seguridad en Cómputo). [En línea] [Citado el: 12 de 01 de 2016.] http://www.cert.org.mx/estadisticas.dsc.

73. Schäling, Boris. The Boost C++ Libraries. [En línea] [Citado el: 05 de 12 de 2015.] http://theboostcpplibraries.com/introduction.

74. Boris Schäling. The Boost C++ Libraries. [En línea] [Citado el: 05 de 12 de 2015.] http://theboostcpplibraries.com/introduction.

75. SQlite. [En línea] [Citado el: 14 de 01 de 2019.] http://www.sqlite.orgom/support/documents/vpuserguide/12/13/5963_visualparadi.html.

76. Sistemas Empotrados - Marco Teorico. [En línea] [Citado el: 16 de 12 de 15.] http://sistemasempotrados.wikispaces.com/Marco+Teorico.

77. Huerta, Antonio Villalon. Seguridad en Unix y Redes. 2002.

78. Velezmoro, Karem. Procesos Unificados y AUP. Gestión de Proyectos. Perú: s.n., 2008.

79. Oracle Business. Oracle Business Intelligence Application Architect. [En línea] [Citado el: 15 de 03 de 2016.] http://mkashu.blogspot.com/2013/07/what-is-session-in-informatica.html.

80. Andalucía, Junta de. Marco de Desarrollo de la Junta de Andalucía. [En línea] [Citado el: 15 de 03 de 2016.] http://www.juntadeandalucia.es/servicios/madeja/contenido/subsistemas/desarrollo/gestion-sesiones-y-usuarios.

81. Booch, Grady. El Lenguaje Unificado de Modelado. 1999.

82. Definición de Domótica. Definición ABC. [En línea] 16 de 11 de 2015. [Citado el: 16 de 11 de 2015.] http://www.definicionabc.com/tecnologia/domotica.php.

83. Defensa y seguridad. Atlas Comparativo de la Defensa en América Latina, RESDAL, . SAINT-PIERRE, Hector. Mexico: s.n., 2008.

84. Dai, Wei. Crypto++. [En línea] 04 de 07 de 2007. [Citado el: 15 de 12 de 2015.] http://www.cryptopp.com/.

85. Arex, una solución minimalista para la supervisión y control de procesos. Ing. Antonio Cedeño Pozo, Ing. Karel Delgado Alón, Ing. Alejandro Jiménez López, Ing. Alexander Moreno Limonte, Álvaro Denis Acosta. La Lisa: s.n., 2013.

86. David Wagner. Analysis of the SSL 3.0 Protocol. California: s.n., 1996.

87. J. Kelsey, B. Schneier, D. Wagner, C. Hall. Cryptanalytic Attacks on Pseudorandom Number Generators. 1998.

88. DICCIONARIO DE INFORMÁTICA Y TECNOLOGÍA . ALEGSA. [En línea] ALEGSA, 2010. http://www.desarrolloweb.com/wiki/framework.html.

89. Luis Alberto Casillas Santillán ,Marc Gibert Ginestà, Óscar Pérez Mora. UOC OpenCourseWare. [En línea] [Citado el: 10 de 05 de 2016.] http://ocw.uoc.edu/computer-science-technology-and-multimedia/bases-de-datos/bases-de-datos/P06_M2109_02151.pdf.

90. Ramió, J. A. Libro electrónico de seguridad informática y criptografía. Sexta edición. . Madrid, España. Universidad Politécnica de Madrid.: s.n., 2006.

GLOSARIO

GUI: La interfaz gráfica de usuario, conocida también como GUI (del inglés *graphical user interface*), es un programa informático que actúa de interfaz de usuario, utilizando un conjunto de imágenes y objetos gráficos para representar la información y acciones disponibles en la interfaz.

STEP 7: Es un Software de Programación de PLC (Controladores Lógicos Programables el SIMATIC-S7 de Siemens, es el sucesor de SIMATIC S5 STEP 7 está ampliamente extendido en toda Alemania.

UIT: La Unión Internacional de Telecomunicaciones (UIT) es el organismo especializado en telecomunicaciones de la Organización de las Naciones Unidas (ONU), encargado de regular las telecomunicaciones a nivel internacional entre las distintas administraciones y empresas operadoras.

IBM: *International Business Machines* es una reconocida empresa multinacional estadounidense de tecnología y consultoría con sede en Armonk, Nueva York. IBM fabrica y comercializa hardware y software para computadoras, y ofrece servicios de infraestructura, alojamiento de Internet, y consultoría en una amplia gama de áreas relacionadas con la informática, desde computadoras hasta nanotecnología.

GNU: La Licencia Pública General de GNU o más conocida por su nombre en inglés *GNU General Public License* (o simplemente sus siglas del inglés GNU GPL) es la licencia más ampliamente usada6 en el mundo del software y garantiza a los usuarios finales (personas, organizaciones, compañías) la libertad de usar, estudiar, compartir (copiar) y modificar el software.

BSD: Es la licencia de software otorgada principalmente para los sistemas BSD (*Berkeley Software Distribution*), un tipo del sistema operativo Unix-like. Es una licencia de software libre permisiva como la licencia de OpenSsl o la MIT License.

ANEXOS

ANEXO 1: Comparación de algoritmos simétricos.

	DES	TDES	AES	IDEA
Potencialidad de la llave	Llave corta y de longitud fija, por lo que no asegura una fortaleza adecuada	Llaves fijas de 128 y 192 bits.	Longitud de llave variable.	Longitud de llave de 128 bits
Velocidad	35 Kbytes/seg	12 Kbytes/seg	55 Kbytes/seg	53 Kbytes/seg
Complejidad de implementación	Fácil de implementar	Fácil de implementar	Fácil de implementar	Fácil y rápido de implementar
Vulnerabilidad del algoritmo	Ha sido roto. Sus claves han sido vulneradas en menos de 24 horas.	Considerado mucho más seguro que el DES al basarse en sus debilidades.	Para ser vulnerado requiere que el atacante pueda ejecutar programas en el mismo sistema.	Su única debilidad conocida es un conjunto de 251 claves débiles, que dentro del universo de llaves no constituyen un peligro.

Ilustración 11. Comparación de algoritmos simétricos

ANEXO 2: Descripción de requisitos no funcionales.

Atributo de Calidad	Usabilidad
Sub-atributos/Sub-características	
Objetivo	El módulo de seguridad debe poder ser usado para el intercambio de datos y el control de sesiones y usuarios de forma segura del Sistema de Medición Arex.
Origen	Externo
Artefacto	Seguridad
Entorno	Ejecución del sistema.
Estímulo	**Respuesta: Flujo de eventos (Escenarios)**
1.a Interacción con el sistema	
	La aplicación inicia a la espera de los clientes.
Medida de respuesta	

Tabla 1. Requisito no funcional 1

Atributo de Calidad	Portabilidad

Sub-atributos/Sub-características	Adaptabilidad
Objetivo	La aplicación debe funcionar en el sistema operativo GNU/Linux.
Origen	Externo
Artefacto	Seguridad
Entorno	Ejecución del sistema.
Estímulo	**Respuesta: Flujo de eventos (Escenarios)**
1. a Ejecución de seguridad.	
	Se inicia la aplicación normalmente
Medida de respuesta	
Independientemente del entorno de escritorio donde se esté ejecutando la aplicación esta debe mostrarse correctamente y permitir acceder a todas las funcionalidades que esta brinda.	

Tabla 2. Requisito no funcional 2

Atributo de Calidad	Disponibilidad
Sub-atributos/Sub-características	
Objetivo	La aplicación debe garantizar su disponibilidad en todo momento para no interferir con la característica de funcionamiento en tiempo real de los sistemas domóticos.
Origen	Interno-Externo
Artefacto	Seguridad
Entorno	Ejecución del sistema.
Estímulo	**Respuesta: Flujo de eventos (Escenarios)**
1. a Conectar servidor-cliente	
	La aplicación debe estar en espera de los clientes.
Medida de respuesta	
La aplicación debe de notificar al cliente que está conectado.	

Tabla 3. Requisito no funcional 3

Atributo de Calidad	Disponibilidad
Sub-atributos/Sub-características	
Objetivo	La aplicación debe garantizar la rapidez de las comunicaciones para la validez de la información

	en tiempo real.
Origen	Interno-Externo
Artefacto	Seguridad
Entorno	Ejecución del sistema.
Estímulo	**Respuesta: Flujo de eventos (Escenarios)**
1. a Conectar por el protocolo TCP/IP	
	La aplicación inicia por el protocolo TCP/IP.
Medida de respuesta	

Tabla 4. Requisito no funcional 4

Atributo de Calidad	N/A
Sub-atributos/Sub-características	Restricciones de diseño
Objetivo	Implementar el sistema haciendo uso del lenguaje de programación C++, el framework QT y el IDE QT Creator.
Origen	Interno
Artefacto	El sistema completo.
Entorno	
Estímulo	**Respuesta: Flujo de eventos (Escenarios)**
Medida de respuesta	

Tabla 5. Requisito no funcional 5

Atributo de Calidad	Implementación
Sub-atributos/Sub-características	
Objetivo	El sistema debe contar con un conjunto de patrones que permita la reutilización del código, así como facilidad en la actualización del mismo.
Origen	Interno
Artefacto	Seguridad
Entorno	
Estímulo	**Respuesta: Flujo de eventos (Escenarios)**
1. a Inicializar el sistema	

Medida de respuesta	

Tabla 6. Requisito no funcional 6

Atributo de Calidad	Implementación
Sub-atributos/Sub-características	
Objetivo	El código debe cumplir con los estándares de codificación establecidos por el cliente.
Origen	Interno
Artefacto	Seguridad
Entorno	Documentación del sistema
Estímulo	**Respuesta: Flujo de eventos (Escenarios)**
1. a Apagar el sistema	
Medida de respuesta	

Tabla 7. Requisito no funcional 7

Atributo de Calidad	Seguridad
Sub-atributos/Sub-características	
Objetivo	Las llaves para el cifrado/descifrado de datos tienen un tiempo de caducidad definido, lo que las hace válidas por un intervalo restringido de tiempo.
Origen	Interno-Externo
Artefacto	Seguridad
Entorno	
Estímulo	**Respuesta: Flujo de eventos (Escenarios)**
1. a Falta de corriente	
Medida de respuesta	

Tabla 8. Requisito no funcional 8

Atributo de Calidad	Confiabilidad
Sub-atributos/Sub-características	

Objetivo	Se debe garantizar que el sistema no afecte la fiabilidad e integridad de la información que se transmite.
Origen	Interno-Externo
Artefacto	Seguridad
Entorno	Ejecución del sistema
Estímulo	**Respuesta: Flujo de eventos (Escenarios)**
1. a	
Medida de respuesta	

Tabla 9. Requisito no funcional 9

Atributo de Calidad	Confiabilidad
Sub-atributos/Sub-características	
Objetivo	Se debe garantizar que el sistema no afecte la disponibilidad de los módulos del Sistema de Medición Arex.
Origen	Interno-Externo
Artefacto	Seguridad
Entorno	Ejecución del sistema
Estímulo	**Respuesta: Flujo de eventos (Escenarios)**
1. a Desconexión del módulo seguridad	
	El sistema debe seguir funcionando normalmente.
Medida de respuesta	
El sistema informa que el módulo de seguridad ha dejado de funcionar.	

Tabla 10. Requisito no funcional 10

Atributo de Calidad	
Sub-atributos/Sub-características	Documentación
Objetivo	Se debe garantizar que el sistema cuente con una correcta documentación.
Origen	Interno
Artefacto	Documentación del código.
Entorno	
Estímulo	**Respuesta: Flujo de eventos (Escenarios)**

1. a	
Medida de respuesta	

Tabla 11. Requisito no funcional 11

Atributo de Calidad	
Sub-atributos/Sub-características	Licenciamiento
Objetivo	Se debe garantizar que el sistema se desarrolle bajo los principios del software libre y por tanto cualquier componente de software que se utilice también debe cumplir con esta premisa.
Origen	Externo
Artefacto	Seguridad
Entorno	
Estímulo	**Respuesta: Flujo de eventos (Escenarios)**
1. a	
Medida de respuesta	

Tabla 12. Requisito no funcional 12

Atributo de Calidad	N/A
Sub-atributos/Sub-características	Soporte
Objetivo	Se debe ofrecer servicios de mantenimiento y actualización.
Origen	Interno
Artefacto	Documentación del código
Entorno	Sistema Operativo
Estímulo	**Respuesta: Flujo de eventos (Escenarios)**
1. a Consultar documentación.	
	Se muestra la documentación de la aplicación.
Medida de respuesta	

Tabla 13. Requisito no funcional 12

ANEXO 3: Historias de usuarios.

Número:1	**Nombre del requisito:** Leer los datos de configuración del XML.
Programador: Enrique Santos Leyva	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 1.5
Riesgo en Desarrollo: Media	
Descripción: El sistema debe cargar un archivo en formato XML donde está guardado la configuración del sistema y ser capaz de leer cada elemento que posee el archivo.	
Observaciones: 1. El collector especifica el host y el port que se está empleando para la conexión y la información referente a cada uno de los dispositivos. Los dispositivos agrupan las variables a leer por el recolector, y las variables contiene las alarmas que se le fueron configuradas. 2. En view se encuentran cada los HMI con la información correspondiente a cada uno de los despliegues. 3. El nodo bdh especifica el puerto TCP que se emplea para las conexiones con el módulo Base de Datos Histórico, BDH-Server. 4. En security se encuentran los grupos de usuario los cuales tienen un nombre y un conjunto de usurarios asignados de los cuales se tiene el nombre y la contraseña encriptada.	
Prototipo de interfaz:	

Tabla 14. Leer los datos de configuración del XML

Número:2	**Nombre del requisito:** Guardar los datos del XML correspondiente a la etiqueta security y la clave del archivo XML en una base de datos SQLite.
Programador: Enrique Santos Leyva	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 1.5
Riesgo en Desarrollo: Alta	
Descripción: Se deben guardan los siguientes datos del XML en la base de datos: 1. La clave del archivo (key). 2. En la etiqueta security:	

I. Los datos de la etiqueta usergroups: ✓ VisiblesHmis. ✓ Name. II. Los datos de la etiqueta user: ✓ Password. ✓ User.
Observaciones:
Prototipo de interfaz:

Tabla 15. Guardar los datos del XML correspondiente a la etiqueta security y la clave del archivo XML en una base de datos SQLite

Número:3	**Nombre del requisito:** Autenticar usuario
Programador: Enrique Santos Leyva	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 1.0
Riesgo en Desarrollo: Alta	
Descripción: Para acceder a algunas opciones del sistema es necesario estar autenticado. Se insertan los datos y se presiona Aceptar. Los datos que debe especificar son los siguientes: 1. Nombre de Usuario (Obligatorio). 2. Contraseña (Obligatorio).	
Observaciones: Para poder autenticarse debe estar registrado en el sistema.	
Prototipo de interfaz:	

Tabla 16. Autenticar usuario

Número:4	**Nombre del requisito:** Validar usuario
Programador: Enrique Santos Leyva	**Iteración Asignada:** 2
Prioridad: Alta	**Tiempo Estimado:** 1.0
Riesgo en Desarrollo: Alta	
Descripción: El sistema debe comprobar que el nombre de usuario debe cumplir los siguientes parámetros de seguridad: ✓ Debe comenzar con un carácter alfanumérico.	

✓ Ser de al menos 4 caracteres de longitud. ✓ Puede contener números, pero no comenzar con uno. ✓ Puede contener guiones bajos, puntos o guiones, pero no al comienzo o al final o tener más de uno junto (ae__, ae_- y ae _ serán inválidos).
Observaciones: El sistema notificará si el usuario no cumple con los parámetros.
Prototipo de interfaz:

Tabla 17. Validar usuario

Número:5	**Nombre del requisito:** Validar contraseña
Programador: Enrique Santos Leyva	**Iteración Asignada:** 2
Prioridad: Alto	**Tiempo Estimado:** 1.0
Riesgo en Desarrollo: Alta	
Descripción: El sistema debe comprobar que la contraseña debe cumplir los siguientes parámetros de seguridad: ✓ Contraseñas que contengan al menos una letra mayúscula. ✓ Contraseñas que contengan al menos una letra minúscula. ✓ Contraseñas que contengan al menos un número o carácter especial. ✓ Contraseñas cuya longitud sea como mínimo 8 caracteres. ✓ Contraseñas cuya longitud máxima no debe ser arbitrariamente limitada.	
Observaciones: El sistema notificará si el usuario no cumple con las condiciones.	
Prototipo de interfaz:	

Tabla 18. Validar contraseña

Número:6	**Nombre del requisito:** Determinar los permisos de cada usuario.
Programador: Enrique Santos Leyva	**Iteración Asignada:** 2
Prioridad: Alta	**Tiempo Estimado:** 2.0
Riesgo en Desarrollo: Alta	
Descripción: El sistema debe verificar cada uno de los permisos del usuario autenticado posee y devolverlo en una lista.	
Observaciones: Un usuario puede tener más de un permiso en el sistema.	

Prototipo de interfaz:

Tabla 19. Determinar los permisos de cada usuario

Número:7	**Nombre del requisito**: Cifrar y descifrar la información que se intercambia entre los módulos de Arex.
Programador: Enrique Santos Leyva	**Iteración Asignada:** 3
Prioridad: Alta	**Tiempo Estimado:** 2.0
Riesgo en Desarrollo: Alta	
Descripción: El sistema debe ser capaz de transmitir la información hacia cada módulo de forma encriptada a través del protocolo de comunicación TCP/IP. 1. Se deben especificar el puerto por donde la aplicación recibirá las conexiones entrantes para recibir y enviar los datos y el protocolo de seguridad que se utilizara. ✓ Puerto válido: 5500. ✓ Puerto Inválido: Cualquiera distinto a 5500.	
Observaciones: Cualquier dirección ip es válida.	
Prototipo de interfaz:	

Tabla 20. Cifrar y descifrar la información que se intercambia entre los módulos de Arex

Número:8	**Nombre del requisito**: Recibir los datos desde el Visualizador.
Programador: Enrique Santos Leyva	**Iteración Asignada:** 3
Prioridad: Media	**Tiempo Estimado:** 2.0
Riesgo en Desarrollo: Baja	
Descripción: El sistema debe ser capaz de recibir cada uno de los datos que el visualizador envía. Los datos son: ✓ Clave del archivo XML. ✓ Usuario. ✓ Contraseña.	
Observaciones:	
Prototipo de interfaz:	

Tabla 21. Recibir los datos desde el Visualizador

Número:9	**Nombre del requisito**: Enviar los datos de respuestas a cada uno de los módulos que lo solicite.
Programador: Enrique Santos Leyva	**Iteración Asignada:** 3
Prioridad: Media	**Tiempo Estimado:** 2.0
Riesgo en Desarrollo: Baja	
Descripción: El sistema debe enviar el nombre del usuario autenticado con sus respectivos permisos a cada módulo de Arex que lo requiera para su funcionamiento.	
Observaciones: El módulo de ejecución debe recibir el identificador del usuario, el tiempo de sesión del mismo, nombre del usuario y los permisos.	
Prototipo de interfaz:	

Tabla 22. Enviar los datos de respuestas a cada uno de los módulos que lo solicite

ANEXO 4: Estimación de esfuerzo por HU.

No.	Historias de usuario	Puntos de estimación
1	Leer los datos de configuración del XML.	1.5
2	Guardar los datos del XML correspondiente a la etiqueta security en una base de datos. SQLite.	1.5
3	Autenticar usuario.	1.0
4	Validar usuario.	1.0
5	Validar contraseña.	1.0
6	Determinar los permisos de cada usuario.	2.0
7	Cifrar y descifrar la información que se intercambia entre los módulos de Arex.	2.0
8	Recibir los datos desde el Visualizador.	2.0
9	Enviar los datos de respuestas a cada uno de los módulos que lo solicite.	2.0

Tabla 24. Estimación de esfuerzo por historia de usuario