**Universidad de las Ciencias Informáticas**

**Facultad 5**

**Título:** Sistema de autenticación para el módulo Seguridad del proyecto Guardián del ALBA

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

**Autor**
Enrique Reyes Bermúdez

**Tutora**
Ing. Iliana Pérez Pupo

**Co-Tutora**
Ing. Dayany Díaz Corona

**Ciudad de la Habana**
**Junio 2009**

## Datos de contacto

Ing. Iliana Pérez Pupo (iperez@uci.cu)

Graduada con el título de Ingeniero en Ciencias Informáticas en el 2007 en la Universidad de las Ciencias Informáticas. Tiene 2 años de experiencia en la especialidad y en el proyecto Guardián del ALBA.

Ing. Dayany Díaz Corona (ddiazc@uci.cu)

Graduada de Ingeniera en Ciencias Informática en la UCI, en el año 2007. Actualmente profesora Instructora de la Universidad de las Ciencias Informáticas, Asesora del Departamento Central de Ingeniería y Gestión de Software. Experiencia en el desempeño del rol de Analista, en proyectos del Polo de Hardware y Automática de la Facultad 5. Se encuentra cursando el Diplomado de Calidad e Ingeniería de Software.

## Dedicatoria

A mi abuela por ser mi ángel guardián a ti te dedico este triunfo.

A mi mama por ser la persona que más quiero en este mundo.

A mi hermano por ser mi segundo padre y haberme presentado la "primera computadora".

## Agradecimientos

A mi familia por apoyarme incondicionalmente en el transcurso de mi vida. A mi tutora Iliana y Dayany por brindarme sus conocimientos en este trabajo. A mis amistades Heidy, Ale, Mariela por soportarme en estos 5 años de mi carrera. A Abdelazis por guiarme en los primeros pasos de la tesis. A Luis Enrique por aportar su lector de huellas dactilares y poder predefender. A Idalmys por ayudarme a aprenderme la tesis. A mi prima Ary y Bencomo por ser parte de mi familia y darme ánimo en los momentos más difíciles de la carrera. A todos los que de una manera u otra me ayudaron en el desarrollo de la tesis, gracias Angel, Ernesto, Angélica, Yusmary.

## Resumen

Producto del convenio entre Cuba y la República Bolivariana de Venezuela y bajo la rectoría de las empresas ALBET-PDVSA surge un proyecto de software nombrado SCADA Nacional, el cual por sus siglas en inglés (Supervisory Control and Data Adquisition) se trata de un sistema de supervisión, control y adquisición de datos, cuyo principal objetivo es lograr primeramente la soberanía tecnológica en esta importante industria y a la vez en el plano técnico, automatizar el proceso productivo de Petróleos de Venezuela. Para el sistema era necesario tener una funcionalidad que se encargara de administrar el acceso de los usuarios a la aplicación. Para dar solución a dicho problema se realiza el siguiente trabajo de diploma que tiene como objetivo principal desarrollar un sistema de autenticación para el módulo Seguridad sobre la plataforma de software libre capaz de brindarles confianza a los usuarios durante su autenticación al sistema.

Para cumplir el objetivo se realizó un estudio de los diferentes métodos de autenticación y cuáles de ellos son utilizados en los SCADAs. Se investigó las principales tecnologías de desarrollo y bibliotecas existentes en el mundo y se hace una propuesta fundamentada de las más importantes. Se presenta además la modelación del sistema a partir de diferentes diagramas y se realiza la implementación del mismo. Por último se muestran las diferentes pruebas realizadas por el cliente logrando finalmente la aceptación del producto.

**Palabras clave**

Autenticación, SCADA, Seguridad, sistema, software libre.

# Índice de contenido

# Índice de figuras

# Índice de tablas

# Introducción

En la Universidad de las Ciencias Informáticas la producción de software y servicios informáticos se basa en la integración de los procesos de formación, investigación y producción en torno a una temática para convertirla en una rama productiva. Este espacio de integración es denominado Polo Productivo. En la facultad 5 se encuentra el polo productivo “Hardware y Automática” del cual el proyecto “Guardián del ALBA” o “SCADA Nacional” forma parte.

Los sistemas de control y adquisición de datos SCADA (Supervisory Control And Data Adquisition) son aplicaciones de software, especialmente diseñadas para funcionar sobre ordenadores en el control de producción, proporcionando comunicación y control de los dispositivos de campo (controladores autónomos, autómatas programables, etc.). Además, envía la información generada en el proceso productivo a diversos usuarios, tanto del mismo nivel como hacia otros supervisores dentro de la empresa, es decir, que permite la participación de otras áreas como por ejemplo: control de calidad, supervisión y mantenimiento.

El proyecto está formado por un número grande de líneas o módulos de los cuales la mayoría han sido desarrollados por los estudiantes y profesionales de la facultad 5. Entre dichos módulos se encuentran Seguridad.

La seguridad en el control industrial es un tema novedoso y en constante evolución en la rama especializada de control automático a nivel mundial, tanto para fabricantes de tecnologías, diseñadores de la ingeniería de sus aplicaciones, como para las organizaciones profesionales que se orientan en el planteamiento de su normativa y regulación.

Estos sistemas industriales, debido a la alta sensibilidad de sus recursos, de los datos que manejan y las operaciones riesgosas que se pueden realizar, requieren de una funcionalidad que les garantice confianza en la autenticación de sus usuarios, y que a su vez sea lo suficientemente segura, incluso en los casos en que la vía de autenticación sea falsa o quien aporte la clave no sea su verdadero dueño, pudiendo proveer de posibles suplantaciones de usuarios, garantizando robustez en la

autenticidad del sistema. Es por ello que surge la necesidad de desarrollar un sistema denominado “Autenticación”.

Debido a que el módulo no cuenta con dicho sistema podemos obtener la siguiente **situación problémica**: El módulo Seguridad del proyecto Guardián del ALBA no cuenta con un sistema de autenticación que administre el acceso de los usuarios a la aplicación, provocando el incremento del nivel de vulnerabilidad que hoy presenta el proyecto.

Ante esta situación se puede definir el siguiente **problema científico**: ¿Cómo desarrollar un sistema dentro del módulo Seguridad del proyecto Guardián del ALBA que garantice la seguridad en la autenticación de los usuarios al sistema?

Para ello el **objeto de estudio** lo constituye: Desarrollo de los sistemas de autenticación. Derivándose que el **campo de acción** que abarca esta investigación sería: Desarrollo del sistema de autenticación para el módulo Seguridad del proyecto Guardián del ALBA.

Esta investigación tiene la siguiente **idea a defender**: Si se desarrolla un sistema de autenticación dentro del módulo Seguridad capaz de administrar el acceso de los usuarios al sistema se garantizará el decremento del nivel de vulnerabilidad que hoy presenta el SCADA.

Para dar respuesta al problema de la investigación se definió que el **objetivo general** consistiera en: Desarrollar un sistema de autenticación para el módulo Seguridad del proyecto Guardián del ALBA bajo los principios del Software Libre.

Conforme a este planteamiento se derivan las siguientes tareas:

- Realización de un estudio del estado del arte de los sistemas SCADAs y los diferentes tipos de autenticación usuales para estos sistemas.
- Investigación de las formas de autenticación segura en sistemas confidenciales y robustos.
- Estudio de las bibliotecas libres existentes diseñadas para realizar autenticaciones biométricas, por tarjetas de código de barra y por contraseñas.

- Estudio de los dispositivos creados para el reconocimiento de patrones biométricos.
- Realización de un estudio de aplicaciones de código abierto.
- Obtención de la documentación correspondiente al análisis, diseño y desarrollo del sistema de autenticación.
- Estudio de los patrones de diseño y su aplicación a las posibles mejoras de la arquitectura del sistema para obtener diseños flexibles y reutilizables.
- Búsqueda, investigación y prueba de las metodologías de análisis y diseño existentes.
- Búsqueda, investigación y prueba de las bibliotecas de código abierto relacionadas con el desarrollo de sistemas de Seguridad.
- Realización de la modelación arquitectónica correspondiente.
- Obtención del código que cumpla a la funcionalidad de autenticación con los requerimientos especificados.
- Realización de pruebas al software con el objetico de garantizar su buen funcionamiento.

La investigación estará estructurada de la siguiente forma:

- Capítulo 1: Describe el estado del arte actual. Se hará referencia a la seguridad y a los diferentes tipos de autenticación existentes en el mundo y cuáles de ellas son utilizadas en los sistemas SCADA.
- Capítulo 2: Se explican las metodologías y tecnologías actuales a considerar para realizar la selección de aquellas que se van a utilizar para el desarrollo de la aplicación teniendo en cuenta también los requisitos del usuario.
- Capítulo 3: Se describe el desarrollo de la solución propuesta a partir de la modelación de diagramas de casos de uso, diagramas de objetos, diagramas de interacción y diagramas de clases persistentes. Se planteará el modelo de datos y se especificarán los diferentes patrones utilizados para enriquecer la arquitectura del sistema.
- Capítulo 4: Se realizará la implementación de la aplicación y se aplicará distintos tipos de pruebas a los casos de usos críticos.

# Capítulo 1: Fundamentación teórica

## Justificación y objetivos

Existen constantes intentos por parte de los usuarios de leer o tomar el control de los datos mostrados en una aplicación de control de sistemas industriales. Todo dato mostrado o modificado a un usuario no identificado debe constituir un riesgo a la seguridad del sistema.

A lo largo de este capítulo se mostrarán cuáles son los diferentes tipos de autenticación existentes a nivel el mundial así como sus ventajas y desventajas. Además se abordará cómo está organizada la seguridad en algunos sistemas SCADA y cuáles de ellos cuentan con un sistema de autenticación capaz de evitar el acceso no autorizado al sistema. Mediante esta información se podrá realizar un análisis para seleccionar el o los tipos de autenticación necesarios en el SCADA Guardián del ALBA.

## SCADA

Los sistemas SCADA dado los procesos que comprenden son de especial atención en la actualidad, muchas veces la economía de un país depende de elementos naturales o artificiales que son llevadas a las industrias para su procesamiento y que necesitan de un sistema que controle y supervise dichos procesos de forma tal que los resultados sean los óptimos. De allí la importancia irrefutable de estos sistemas.

## Funciones principales

Un SCADA debe cumplir tres funciones principales:

- Adquisición de datos para recoger, procesar y almacenar la información recibida.
- Supervisión, para observar desde el monitor la evolución de las variables del proceso.
- Control para modificar la evolución del proceso, actuando bien sobre los reguladores autónomos básicos (consignas, alarmas, menús, etc.), bien directamente sobre el proceso mediante las salidas conectadas. [1]

**Requisitos**

Un SCADA debe cumplir varios objetivos para que su instalación sea perfectamente aprovechada:

- Deben ser sistemas de arquitectura abierta, capaces de crecer o adaptarse según las necesidades cambiantes de la empresa.
- Deben comunicarse con total facilidad y de forma transparente al usuario con el equipo de planta y con el resto de la empresa (redes locales y de gestión).
- Deben ser programas sencillos de instalar, sin excesivas exigencias de hardware, y fáciles de utilizar, con interfaces amigables con el usuario. [1]

**SCADAs disponibles en Cuba y el mundo**

A continuación se muestran una lista de algunos software SCADA y sus fabricantes.

- Eros: Es un sistema de supervisión y control de procesos industriales en él se concentran múltiples facilidades para operar y dirigir cualquier proceso productivo. Desarrollado en 1991 por especialistas de la “Unión del níquel” de la provincia de Holguín.
- Movicon: Representa la tercera generación innovadora de las plataformas industriales del software de supervisión y control (Scada/HMI), desarrollado por el grupo italiano Progea. Ofrece la posibilidad de realizar potentes compactos sistemas de visualización HMI. El panel operador se convierte en una pequeña estación SCADA, ofreciendo independencia del hardware, conectividad con los sistemas superiores de información e incrementando la potencia de la máquina localmente.
- WinCC: Es un sistema de supervisión sobre PC ejecutable bajo Microsoft Windows 95 y Windows NT, desarrollado por la empresa Siemens. Está concebido para la visualización y manejo de procesos, líneas de fabricación, máquinas e instalaciones. El volumen de funciones de sistema incluye la emisión de avisos de eventos de forma adecuada para la aplicación industrial, el archivo de valores medios, creación de recetas y el listado de los mismos. [2]

**Seguridad**

La seguridad consiste en asegurar que los recursos del sistema de información (material informático o programas) de una organización sean utilizados de la manera que se decidió y que el acceso a la información allí contenida así como su modificación sólo sea posible a las personas que se encuentren acreditadas y dentro de los límites de su autorización.[3]

A grandes rasgos se entiende que mantener un sistema seguro (fiable) consiste básicamente en garantizar tres aspectos: confidencialidad, integridad y disponibilidad. Algunos estudios integran la seguridad dentro de una propiedad más general de los sistemas, la confiabilidad, entendida como el nivel de calidad del servicio ofrecido. Consideran la disponibilidad como un aspecto al mismo nivel que la seguridad y no como parte de ella, por lo que dividen esta última en sólo las dos facetas restantes, confidencialidad e integridad.

La confidencialidad dice que los objetos de un sistema han de ser accedidos únicamente por elementos autorizados a él, y que esos elementos autorizados no van a convertir esa información en disponible para otras o entidades; la integridad significa que los objetos sólo pueden ser modificados por elementos autorizados, y de una manera controlada, y la disponibilidad indica que los objetos del sistema tienen que permanecer accesibles a elementos autorizados. [3]

**Autenticación**

La autenticación es el procedimiento de comprobación de la identidad de una entidad del sistema (usuario, proceso). 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. [7]

**Tipos de autenticación**

Los métodos de autenticación se suelen dividir en tres grandes categorías en función de lo que utilizan para la verificación de identidad:

- Algo que el usuario conoce.

- Algo que el usuario posee.
- Una característica física del usuario o un acto involuntario del mismo. [3]

Esta última categoría se conoce con el nombre de autenticación biométrica. Es fácil ver ejemplos de cada uno de estos tipos de autenticación: una contraseña o frase 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.

**Sistemas basados en algo conocido: contraseñas**

El modelo de autenticación más básico consiste en decidir si un usuario es quien dice ser simplemente basándonos en una prueba de conocimiento que a priori sólo ese usuario puede superar; y desde Alí Babá y su "Ábrete Sésamo" hasta los más modernos sistemas, esa prueba de conocimiento no es más que una contraseña que en principio es secreta. Evidentemente, esta aproximación es la más vulnerable a todo tipo de ataques, pero también la más barata, por lo que se convierte en la técnica más utilizada en entornos que no precisan de una alta seguridad.

En todos los esquemas de autenticación basados en contraseñas se cumple el mismo protocolo:

Las entidades (generalmente dos) que participan en la autenticación acuerdan una clave, 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.

**Contraseñas aceptables**

Exigir que los usuarios elijan claves aceptables es una de las medidas básicas de

prevención contra ataques.

Métodos inseguros:

- Las contraseñas no deben ser creadas usando la información personal sobre el usuario o su familia. Una contraseña de craqueo incentivo para romper con su contraseña personal utilizará esta información en primer lugar, hacer estas contraseñas menos contraseñas seguras. Ejemplos de malas contraseñas de este tipo son: el nombre, lugar de nacimiento, nombre, apellido, nombres de animales de compañía, dirección, nombres de los padres, los nombres de los hermanos y similares.
- Las contraseñas no deben ser formadas de palabras de diccionario o cualquier libro. Palabras más largas no suelen añadir mucha protección. Uso de palabras conocidas en cualquier idioma, permite a la contraseña de craqueo a tomar atajos en sus sistemas de obtención ilegal de contraseña, lo que le permite adivinar su contraseña en una muy pequeña fracción del tiempo que tomaría otra cosa. Ejemplos de malas contraseñas de este tipo son: dragón, en secreto, el queso, el dios, el amor, el sexo, la vida y palabras similares.
- Las contraseñas no debe ser compuesto de los nombres propios de lugares, ideas o personas. Estas palabras se encuentran comúnmente en bases de datos de craqueo contraseña. Ejemplos son: Jehová, Tylenol, edutenimiento, Coolio, beesknees, transformadores.
- Las contraseñas no deben ser simples variaciones de las palabras. Aunque las contraseñas no aparecen en un libro o diccionario, es un asunto sencillo para generar una lista de palabras sustitución automática. Estas contraseñas son más seguras que los dos ejemplos anteriores, pero no mucho más seguro. Ejemplos de este tipo de contraseñas son drowssap, l0ve, s3cr3t, dr @ genesittah, y palabras similares.
- No volverlas a utilizar de nuevo recientemente. Si es difícil escoger una nueva contraseña, se debe esperar hasta que le cambiaron la contraseña por lo menos 5 veces antes de la reutilización de una vieja, o 12 meses si son comunes los cambios de contraseña.

Métodos seguros:

- Cambiar la contraseña inmediatamente si la contraseña se ha visto comprometida.
- No escribir la contraseña en donde otros puedan encontrarla.
- Es importante cambiar la contraseña en una frecuencia regular, por lo menos cada seis meses. También es de ayuda si de alguna manera se ha comprometido la contraseña de algún otro modo, sin saberlo.
- Seleccionar contraseñas que utilizan una mezcla de letras mayúsculas, números y caracteres especiales. Tener en cuenta sin embargo, algunos sistemas no permiten el uso de algunos o ningún carácter especial.
- Utilizar los números de la sustitución de letras y letras para los números en sus contraseñas. Aunque este no es un método principal de asegurar la contraseña, añadiendo otra capa de seguridad en la parte superior de una buena contraseña, y que impidan la caída accidental de adivinar la contraseña, debido a las circunstancias.
- Cuando no sea posible utilizar los caracteres en la contraseña (a menos de 14), es recomendable crear una contraseña mediante la creación de una contraseña, y la selección de cartas en una posición específica en cada palabra. Un ejemplo de esto es "jJjshnImn2". Es poco probable que un cracker pueda adivinar la contraseña, sin embargo, es fácil de recordar cuando se nota la frase de paso "de John Jacob Jingleheimer Schmidt, su nombre es mi nombre también".
- Las mejores contraseñas son frases completas. Ellos se denominan a veces "frase de contraseña" en la reflexión del presente. Por ejemplo, una buena contraseña puede ser "Yo limpio mi Glock en el lavaplatos". También puede utilizar el número y la letra de sustitución así como en frase de contraseña.
- Para que una contraseña sea aceptable obligatoriamente ha de cumplir el principio KISS, que hablando de contraseña está claro que no puede significa “Keep it simple, stupid!” sino “Keep it SECRET, stupid!”. La contraseña más larga, la más difícil de recordar, la que combina más caracteres no alfabéticos pierde toda su robustez si su propietario la comparte con otras personas. [8]

**Envejecimiento de la contraseña**

El envejecimiento de contraseñas es una técnica utilizada por los administradores de sistemas para defenderse de las malas contraseñas dentro de la organización. El envejecimiento de contraseñas consiste en proteger las contraseñas de los usuarios dándoles un determinado período de vida: una contraseña sólo va a ser válida durante un cierto tiempo, pasado el cual expirará y el usuario deberá cambiarla. La teoría detrás de esto es que si un usuario es forzado a cambiar su contraseña periódicamente, una contraseña que ha sido descifrada por un cracker sólo le es útil por un tiempo determinado. La desventaja del envejecimiento de contraseñas, es que los usuarios tienden a escribir sus contraseñas en lugares no seguros. [3]

**Sistemas basados en algo poseído: tarjetas inteligentes**

Hace más de veinte años un periodista francés llamado Roland Moreno patentaba la integración de un procesador en una tarjeta de plástico; sin duda, no podía imaginar el abanico de aplicaciones de seguridad que ese nuevo dispositivo, denominado chipcard, estaba abriendo. Desde entonces, cientos de millones de esas tarjetas han sido fabricadas, y son utilizadas a diario para fines que varía desde las tarjetas monedero más sencillas hasta el control de accesos a instalaciones militares y agencias de inteligencia de todo el mundo; cuando a las chipcards se les incorporó un procesador inteligente nacieron las smartcards, una gran revolución en el ámbito de la autenticación de usuarios. Desde un punto de vista formal una tarjeta inteligente (o smartcard) es un dispositivo de seguridad del tamaño de una tarjeta de crédito, resistente a la adulteración, que ofrece funciones para un almacenamiento seguro de información.

En la práctica, las tarjetas inteligentes poseen un chip empotrado en la propia tarjeta que puede implementar un sistema de ficheros cifrado y funciones criptográficas, y además puede detectar activamente intentos no válidos de acceso a la información almacenada; este chip inteligente es el que las diferencia de las simples tarjetas de crédito, que solamente incorporan una banda magnética donde va almacenada cierta información del propietario de la tarjeta.

En la figura 1 se muestra una estructura general de una tarjeta inteligente; en ella se puede observar que el acceso a las áreas de memoria solamente es posible a través de la unidad de entrada/salida y de una CPU (típicamente de 8 bits), lo que evidentemente aumenta la seguridad del dispositivo. Existe también un sistema

operativo empotrado en la tarjeta generalmente en ROM. Cuando el usuario poseedor de una smartcard desea autenticarse necesita introducir la tarjeta en un hardware lector; los dos dispositivos se identifican entre sí con un protocolo a dos bandas en el que es necesario que ambos conozcan la misma clave lo que elimina la posibilidad de utilizar tarjetas de terceros para autenticarse ante el lector de una determinada compañía, además esta clave puede utilizarse para asegurar la comunicación entre la tarjeta y el dispositivo lector. Tras identificarse las dos partes, se lee la identificación personal (PID) de la tarjeta, y el usuario teclea su PIN; se inicia entonces un protocolo desafío-respuesta: se envía el PID a la máquina y ésta desafía a la tarjeta, que responde al desafío utilizando una clave personal del usuario (PK, Personal Key). Si la respuesta es correcta, el host ha identificado la tarjeta y el usuario obtiene acceso al recurso pretendido.

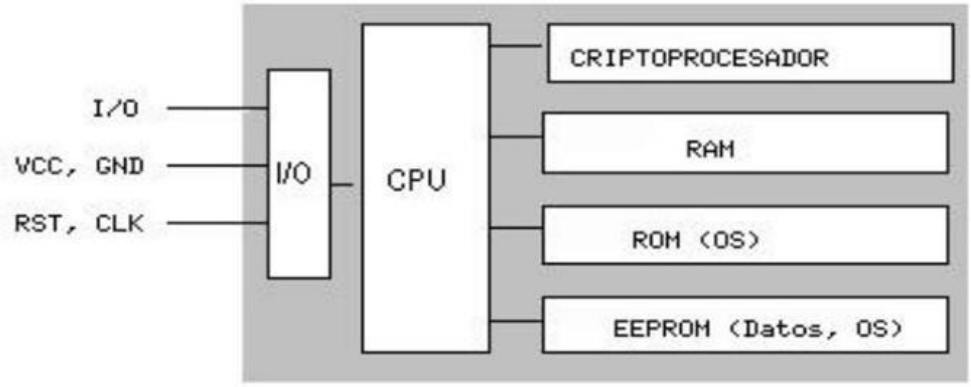


Figura 1: Estructura general de una tarjeta inteligente

Las ventajas de utilizar tarjetas inteligentes como medio para autenticar usuarios son muchas frente a las desventajas; se trata de un modelo ampliamente aceptado entre los usuarios, rápido, y que incorpora hardware de alta seguridad tanto para almacenar datos como para realizar funciones de cifrado. Además, su uso es factible tanto para controles de acceso físico como para controles de acceso lógico a los hosts, y se integra fácilmente con otros mecanismos de autenticación como las contraseñas; y en caso de desear bloquear el acceso de un usuario, no tenemos más que retener su tarjeta cuando la introduzca en el lector o marcarla como inválida en una base de datos (por ejemplo, si se equivoca varias veces al teclear su PIN, igual que sucede con una tarjeta de crédito normal). Como principal inconveniente de las smartcards podemos

citar el coste adicional que supone para una organización el comprar y configurar la infraestructura de dispositivos lectores y las propias tarjetas; aparte, que un usuario pierda su tarjeta es bastante fácil, y durante el tiempo no disponga de ella o no puede acceder al sistema, o hemos de establecer reglas especiales que pueden comprometer nuestra seguridad (y por supuesto se ha de marcar como tarjeta inválida en una base de datos central, para que un potencial atacante no pueda utilizarla). Aparte de los problemas que puede implicar el uso de smartcards en sí contra la lógica de una tarjeta inteligente existen diversos métodos de ataque, como realizar ingeniería inversa-destructiva contra el circuito de silicio (y los contenidos de la ROM), adulterar la información guardada en la tarjeta o determinar por diferentes métodos el contenido de la memoria EEPROM. [3]

**Sistemas de autenticación biométrica**

La biometría es el estudio de métodos automáticos para el reconocimiento único de humanos basados en uno o más rasgos conductuales o físicos intrínsecos. El término se deriva de las palabras griegas "bios" de vida y "metron" de medida.

En las tecnologías de la información (TI), la autenticación biométrica se refiere a las tecnologías para medir y analizar las características físicas y del comportamiento humanas con propósito de autenticación.

Las huellas dactilares, las retinas, el iris, los patrones faciales, de venas de la mano o la geometría de la palma de la mano, representan ejemplos de características físicas, mientras que entre los ejemplos de características del comportamiento se incluye la firma, el paso y el tecleo (dinámicas). [9]

A pesar de la importancia de la criptología en cualquiera de los sistemas de identificación de usuarios vistos, existen otra clase de sistemas en los que no se aplica esta ciencia, o al menos su aplicación es secundaria. Es más, parece que en un futuro no muy lejano estos serán los sistemas que se van a imponer en la mayoría de situaciones en las que se haga necesario autenticar un usuario: son más amigables para el usuario (no va a necesitar recordar contraseña o números de identificación complejos, y, como se suele decir, el usuario puede olvidar una tarjeta de identificación en casa, pero nunca se olvidará de su mano o su ojo) y son mucho más difíciles de

falsificar que una simple contraseña o una tarjeta magnética. Las principales razones por la que no se han impuesto ya en nuestros días es su elevado precio, fuera del alcance de muchas organizaciones, y su dificultad de mantenimiento.

Estos sistemas son los denominados biométricos, basados en características físicas a identificar. El reconocimiento de formas, la inteligencia artificial y el aprendizaje son las ramas de la informática que desempeñan el papel más importante en los sistemas de identificación biométricos; la criptología se limita aquí a un uso secundario, como el cifrado de una base de datos de patrones retinales, o la transmisión de una huella dactilar entre un dispositivo analizador y una base de datos. La autenticación basada en características físicas existentes desde que existe el hombre y, sin darnos cuenta, es la que más utiliza cualquiera de nosotros en su vida cotidiana: a diario se identifican a personas por los rasgos de su cara o por su voz.

Los dispositivos biométricos tienen tres partes principales; por un lado, disponen de un mecanismo automático que lee y captura una imagen digital o analógica de la característica a analizar. Además disponen de una entidad para manejar aspectos como la compresión, almacenamiento o comparación de los datos capturados con los guardados en una base de datos (que son considerados válidos), y también ofrecen una interfaz para las aplicaciones que los utilizan.

El proceso general de autenticación sigue unos pasos comunes a todos los modelos de autenticación biométrica: captura o lectura de los datos que el usuario a validar presenta, extracción de ciertas características de la muestra (por ejemplo, las minucias de una huella dactilar), comparación de tales características con las guardadas en una base de datos, y decisión de si el usuario es válido o no. Es en esta decisión donde principalmente entran en juego las dos características básicas de la fiabilidad de todo sistema biométrico (en general, de todo sistema de autenticación): las tasas de falso rechazo y de falsa aceptación.

Por tasa de falso rechazo (False Rejection Rate, FRR) se entiende la probabilidad de que el sistema de autenticación rechace a un usuario legítimo porque no es capaz de identificarlo correctamente, y por tasa de falsa aceptación (False Acceptance Rate, FAR) la probabilidad de que el sistema autentique correctamente a un usuario ilegítimo; evidentemente, una FRR alta provoca descontento entre los usuarios del sistema, pero

una FAR elevada genera un grave problema de seguridad: estamos proporcionando acceso a un recurso a personal no autorizado a acceder a él. [3]

**Verificación de huellas**

Típicamente la huella dactilar de un individuo ha sido un patrón bastante bueno para determinar su identidad de forma inequívoca, ya que está aceptado que dos dedos nunca poseen huellas similares, ni siquiera entre gemelos o entre dedos de la misma persona. Por tanto, parece obvio que las huellas se convertiría antes o después en un modelo de autenticación biométrico: desde el siglo pasado hasta nuestros días se vienen realizando con éxito clasificaciones sistemáticas de huellas dactilares en entornos policiales, y el uso de estos patrones fue uno de los primeros en establecerse como modelo de autenticación biométrica.

Cuando un usuario desea autenticarse ante el sistema sitúa su dedo en un área determinada (área de lectura, no se necesita en ningún momento una impresión en tinta). Aquí se toma una imagen que posteriormente se normaliza mediante un sistema de finos espejos para corregir ángulos, y es de esta imagen normalizada de la que el sistema extrae las minucias (ciertos arcos, bucles o remolinos de la huella) que va a comparar contra las que tiene en su base de datos; es importante resaltar que lo que el sistema es capaz de analizar no es la huella en sí sino que son estas minucias, concretamente la posición relativa de cada una de ellas. Está demostrado que dos dedos nunca pueden poseer más de ocho minucias comunes, y cada uno tiene al menos 30 o 40 de éstas. Si la comparación de las posiciones relativas de las minucias leídas con las almacenadas en la base de datos es correcta, se permite el acceso al usuario, denegándosele obviamente en caso contrario.

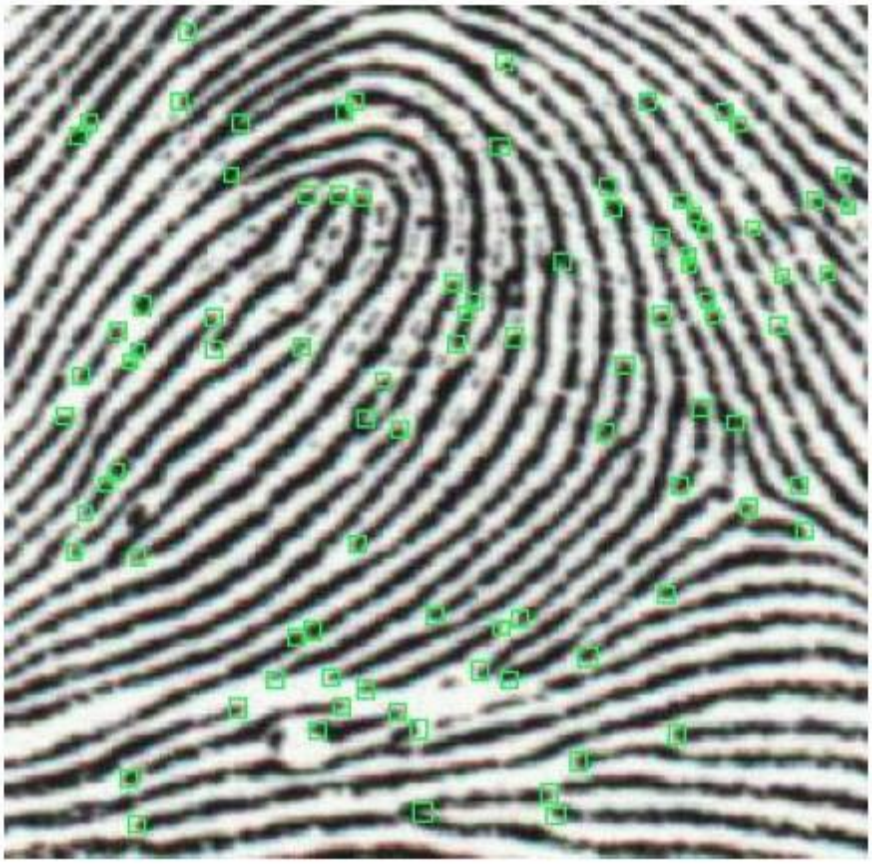

Figura 2: Huella dactilar con sus minucias extraídas

Los sistemas basados en reconocimiento de huellas son relativamente baratos (en comparación con otros biométricos, como los basados en patrones retinales); sin embargo, tienen en su contra la incapacidad temporal de autenticar usuarios que se hayan podido herir en el dedo a reconocer un pequeño corte o una quemadura que afecte a varias minucias pueden hacer inútil al sistema). También elementos como la suciedad del dedo, la presión ejercida sobre el lector o el estado de la piel pueden ocasionar lecturas erróneas. [3]

**Verificación de patrones oculares**

Los modelos de autenticación biométrica basados en patrones oculares se dividen en dos tecnologías diferentes: o bien analizan patrones retinales, o bien analizan el iris. Estos métodos suelen considerar los más efectivos: para una población de 200 millones de potenciales usuarios la probabilidad de coincidencia es casi 0, y además una vez

muerto el individuo los tejidos oculares degeneran rápidamente, lo que dificulta la falsa aceptación de atacantes que puedan robar este órgano de un cadáver.

La principal desventaja de los métodos basados en el análisis de patrones oculares es su escasa aceptación; el hecho de mirar a través de un binocular (o monocular), necesario en ambos modelos, no es cómodo para los usuarios, ni aceptable para muchos de ellos: por un lado, los usuarios no se fían de un haz de rayos analizando su ojo, y por otro un examen de este órgano puede revelar enfermedades o características médicas que a muchas personas les puede interesar mantener en secreto, como el consumo de alcohol o de ciertas drogas. Aunque los fabricantes de dispositivos lectores aseguran que sólo se analiza el ojo para obtener patrones relacionados con la autenticación, y en ningún caso se viola la privacidad de los usuarios, mucha gente no cree esta postura oficial (aparte del hecho de que la información es procesada vía software, lo que facilita introducir modificaciones sobre lo que nos han vendido para que un lector realice otras tareas de forma enmascarada). También se trata de sistemas demasiado caros para la mayoría de organizaciones, y el proceso de autenticación no es todo lo rápido que debiera en poblaciones de usuarios elevadas. De esta forma, su uso se ve reducido casi sólo a la identificación en sistemas de alta seguridad, como el control de acceso a instalaciones militares. [3]

**Retina**

La vasculatura retinal (forma de los vasos sanguíneos de la retina humana) es un elemento característico de cada individuo, por lo que numerosos estudios en el campo de la autenticación de usuarios se basan en el reconocimiento de esta vasculatura. En los sistemas de autenticación basados en patrones retinales el usuario a identificar ha de mirar a través de unos binoculares, ajustar la distancia interocular y el movimiento de la cabeza, mirar a un punto determinado y por último pulsar un botón para indicar al dispositivo que se encuentra listo para el análisis. En ese momento se escanea la retina con una radiación infrarroja de baja intensidad en forma de espiral, detectando los nodos y ramas del área retinal para compararlos con los almacenados en una base de datos; si la muestra coincide con la almacenada para el usuario que el individuo dice ser, se permite el acceso. [3]

**Iris**

El iris humano (el anillo que rodea la pupila, que a simple vista diferencia el color de ojos de cada persona) es igual que la vasculatura retinal una estructura única por individuo que forma un sistema muy complejo de hasta 266 grados de libertad, inalterable durante toda la vida de la persona. El uso por parte de un atacante de órganos replicados o simulados para conseguir una falsa aceptación es casi imposible con análisis infrarrojo, capaz de detectar con una alta probabilidad si el iris es natural o no. La identificación basada en el reconocimiento de iris es más moderna que la basada en patrones retinales; desde hace unos años el iris humano se viene utilizando para la autenticación de usuarios. Para ello, se captura una imagen del iris en blanco y negro, en un entorno correctamente iluminado; esta imagen se somete a deformaciones pupilares (el tamaño de la pupila varía enormemente en función de factores externos, como la luz) y de ella se extraen patrones, que a su vez son sometidos a transformaciones matemáticas hasta obtener una cantidad de datos suficiente para los propósitos de autenticación. Esa muestra, denominada iriscode es comparada con otra tomada con anterioridad y almacenada en la base de datos del sistema, de forma que si ambas coinciden el usuario se considera autenticado con éxito; la probabilidad de una falsa aceptación es la menor de todos los modelos biométricos. [3]

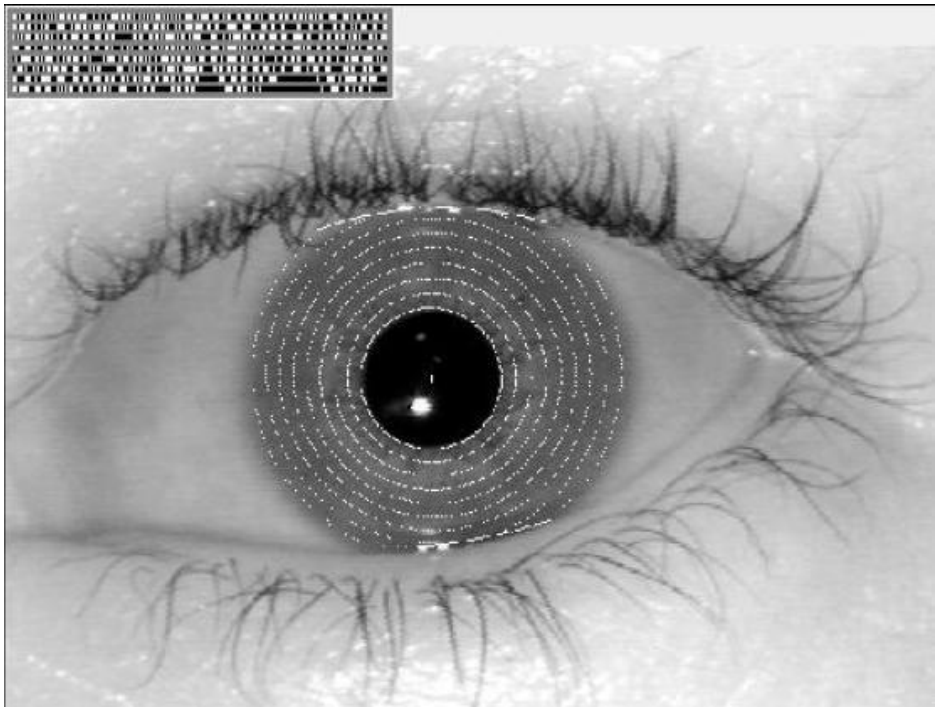

Figura 3: Iris humano con la extracción de su iriscode

**Verificación de la geometría de la mano**

Los sistemas de autenticación basados en el análisis de la geometría de la mano son sin duda los más rápidos dentro de los biométricos: con una probabilidad de error aceptable en la mayor de las ocasiones, en aproximadamente un segundo son capaces de determinar si una persona es quien dice ser. Cuando un usuario necesita ser autenticado sitúa su mano sobre un dispositivo lector con unas guías que marcan la posición correcta para la lectura. Una vez que la mano esté correctamente situada, unas cámaras toman una imagen superior y otra lateral, de las que se extraen ciertos datos (anchura, longitud, área, determinadas distancias) en un formato de tres dimensiones. Transformando estos datos en un modelo matemático que se contrasta contra una base de patrones, el sistema es capaz de permitir o denegar acceso a cada usuario.

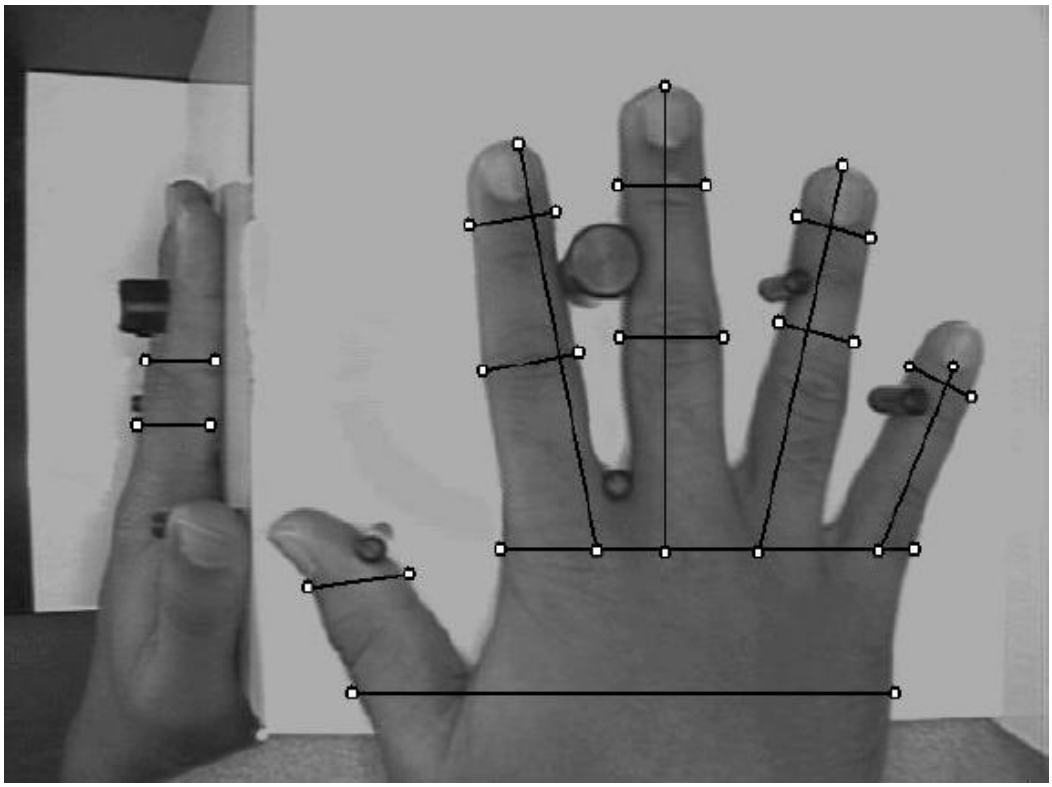

Figura 4: Geometría de la mano con ciertos patrones extraídos

Quizás uno de los elementos más importantes del reconocimiento mediante analizadores de geometría de la mano es que éstos son capaces de aprender: a la vez que autentican a un usuario, actualizan su base de datos con los cambios que se

puedan producir en la muestra (un pequeño crecimiento, adelgazamiento, el proceso de cicatrizado de una herida); de esta forma son capaces de identificar correctamente a un usuario cuya muestra se tomó hace años, pero que ha ido accediendo al sistema con regularidad. Este hecho, junto a su rapidez y su buena aceptación entre los usuarios, hace que los autenticadores basados en la geometría de la mano sean los más extendidos dentro de los biométricos a pesar de que su tasa de falsa aceptación se podrá considerar inaceptable en algunas situaciones: no es normal, pero sí posible, que dos personas tengan la mano lo suficientemente parecida como para que el sistema las confunda. Para minimizar este problema se recurre a la identificación basada en la geometría de uno o dos dedos, que además puede usar dispositivos lectores más baratos y proporciona incluso más rapidez. [3]

**Seguridad y autenticación en los SCADAs**

La seguridad en sistemas SCADA está un poco olvidada y no por dejadez, sino porque la seguridad no es el eje motor de su desarrollo.

Los sistemas de control de procesos fueron diseñados antes del surgimiento de Internet. 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. La capacidad de procesamiento de estos sistemas era limitada y estaban pensados para ejecutar sus tareas, sin capacidad adicional para ejecutar otros programas. Es un segmento donde los ingenieros de software hacen maravillas, pero en el que no existe dominio, por norma general, del desarrollo seguro de aplicaciones.

Hoy en día es bastante común que las redes SCADA estén mezcladas con las redes de las empresas sin ningún tipo de separación o control de acceso. En muchos casos, los ordenadores tienen doble tarjeta de red conectadas a la red de la empresa por un lado y a la red SCADA. Esto crea una puerta de enlace potencial entre ambas, ya que basta comprometer alguna máquina para poder acceder de una red a otra. Los atacantes no necesitan ser expertos en redes de control de procesos. Basta con atacar una máquina de la red local de la empresa que tenga acceso a la red SCADA para poder causar daños. Una carencia de estos sistemas es que no se han implementado medidas de seguridad básicas, tales como cifrado, autenticación o redundancia. [4]

En general no se contemplan políticas de seguridad. Las contraseñas suelen compartirse, incluso existe solamente un usuario en los equipos de red que se comparte entre todos los operadores, lo que limita en gran medida la capacidad de auditoría. Los ficheros de eventos no se recogen ni se analizan. Otro problema usual es la falta de comunicación o fluidez entre departamentos. Muchas veces suelen existir conflictos de intereses entre el departamento de ingeniería y el de informática.

SCADA Movicon:

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. [5]

SCADA Wincc:

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 existe en el momento una protección contra el acceso indebido. [6]

Es importante destacar que los sistemas SCADA se encuentran expuestos al mundo exterior y hay mucha información sobre ellos disponible, incluyendo especificaciones y vulnerabilidades, lo que los hace un blanco fácil para un ataque. Existe un peligro latente por las innumerables fallas de seguridad que presentan en mayor o menor medida todas las plataformas SCADA y debe ser abordado lo antes posible por todas las empresas.

# Capítulo 2: Herramientas y tecnologías actuales

## Introducción

En este capítulo se realiza un análisis del estado del arte de los métodos, técnicas y herramientas empleadas para el modelado del sistema así como de las tecnologías que se emplean en el desarrollo y documentación de la aplicación.

## Sistema operativo: GNU/Linux

Un sistema operativo puede ser contemplado como una colección organizada de extensiones software del hardware, consistente en rutinas de control que hacen funcionar al computador y proporcionan un entorno para la ejecución de programas. Estos programas utilizan las facilidades proporcionadas por el sistema operativo para obtener acceso a recursos del sistema informático como el procesador, archivos y dispositivos de entrada/salida (E/S). De esta forma, el Sistema Operativo constituye la base sobre la cual pueden escribirse los programas de aplicación, los cuales invocarían sus servicios por medio de llamadas al sistema. Por otro lado, los usuarios pueden interactuar directamente con él a través de órdenes concretas. En cualquier caso, actúan como interfaz entre los usuarios/aplicaciones y el hardware de un sistema informático. [20]

## Lenguaje de Modelado: UML

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

Es importante resaltar que UML es un "lenguaje" para especificar y no para describir

métodos o procesos. Se utiliza para definir un sistema de software, para detallar los artefactos en el sistema y para documentar y construir. En otras palabras, es el lenguaje en el que está descrito el modelo. Se puede aplicar en una gran variedad de formas para dar soporte a una metodología de desarrollo de software (tal como el Proceso Unificado Racional), pero no especifica en sí mismo qué metodología o proceso usar. [22]

**Metodología: RUP**

La metodología RUP (Proceso Unificado de Rational, Rational Unified Process en inglés), fue desarrollada en 1998 por Grady Booch, Ivar Jacobson y James Rumbaugh, prominentes metodologistas en la industria de la tecnología y sistemas de información. RUP se caracteriza por ser: dirigido por Casos de Uso, centrado en la arquitectura, iterativo e incremental.

Una metodología de desarrollo de software Orientada a Objeto (OO) consta de los siguientes elementos:

- Conceptos y diagramas.
- Etapas y definición de entrega en cada una de ellas.
- Actividades y recomendaciones.
- Las principales disciplinas de esta metodología son:

| Disciplinas | Descripción |
|---|---|
| Modelado del negocio | Describe los procesos de negocio, identificando quiénes participan y las actividades que requieren automatización. Propone comprender los problemas de la organización e identificar posibles mejoras, evaluar el impacto del cambio introducido con la automatización, asegurar que todos los miembros tienen una misma visión de la organización. |
| Requerimientos | Esta disciplina se encarga de convertir las peticiones de los clientes en un conjunto de requerimientos de software, que definan el alcance y lo que debe hacer el producto a ser construido. Propone mantener un acuerdo a los interesados de lo que el sistema debe hacer, provee a los desarrolladores de una mejor visión de los requerimientos del sistema, |

| | |
|---|---|
| | define la frontera del sistema, crea las bases para la planificación y estimación. |
| Análisis y Diseño | Esta disciplina define como transformar artefactos resultantes del flujo de requerimientos de software en especificaciones del diseño del proyecto a desarrollar, en el se hace evolucionar la arquitectura de modo que se adapte al entorno del usuario final. |
| Implementación | Define como desarrollar, organizar realizar pruebas de unidad e integración de todos los componentes implementados basado en las especificaciones del sistema. |
| Pruebas | Este flujo se centra en encontrar y documentar los defectos en la calidad del software, validar y probar todos los supuestos hechos durante el diseño, verificar que el producto se desempeña como fue diseñado y cubre todos los requisitos pactados durante el flujo de "Captura de Requerimientos". |
| Instalación | Se encarga de garantizar que el producto está disponible para los usuarios en su ambiente, se realizan actividades como empaque, instalación, asistencia a usuarios. |
| Administración del proyecto | Se centra los aspectos esenciales del desarrollo iterativo como son la planificación, control de riesgos, seguimiento del proceso de desarrollo de software y aplicación de métricas. Las restantes serán utilizadas durante el proceso de administración. |
| Administración de configuración y cambios | Se encarga de controlar y sincronizar la evolución de todos los productos generados, introduce el uso de herramientas que faciliten el trabajo colaborativo en el equipo. |
| Ambiente | Contiene actividades que describen los procesos y herramientas que soportarán el equipo de trabajo del proyecto; así como el procedimiento para implementar el proceso en una organización. |

Tabla 1: Principales etapas de RUP

El RUP divide el proceso de desarrollo en ciclos, teniendo un producto final al final de cada ciclo.

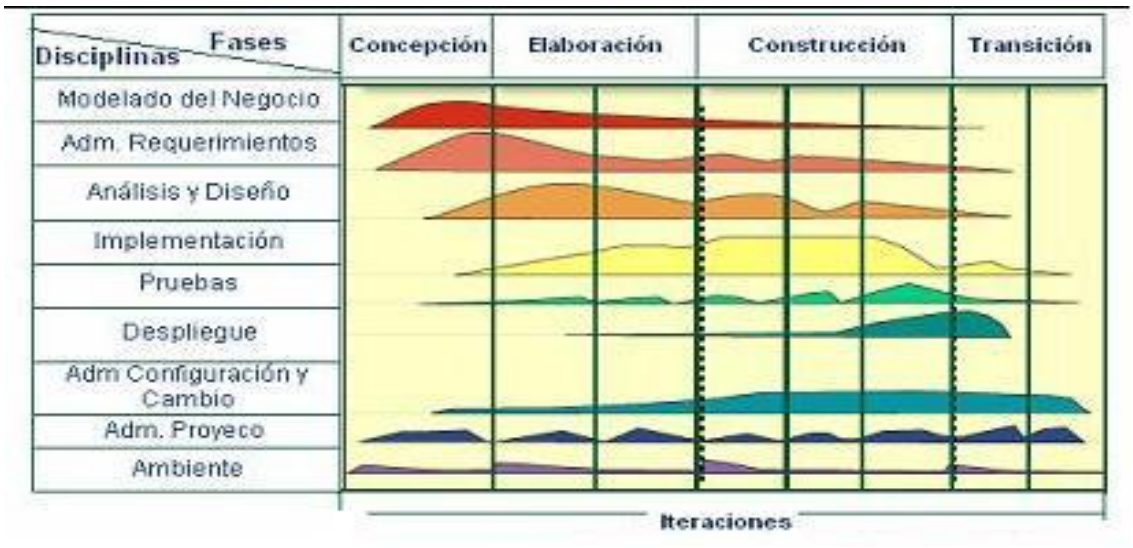


Figura 5: Metodología de desarrollo de software: RUP

Cada ciclo se divide en fases que finalizan con un hito donde se debe tomar una decisión importante:

- Conceptualización (Concepción o Inicio): Se describe el negocio y se delimita el proyecto describiendo sus alcances con la identificación de los casos de uso del sistema.
- Elaboración: Se define la arquitectura del sistema y se obtiene una aplicación ejecutable que responde a los casos de uso que la comprometen. A pesar de que se desarrolla a profundidad una parte del sistema, las decisiones sobre la arquitectura se hacen sobre la base de la comprensión del sistema completo y los requerimientos (funcionales y no funcionales) identificados de acuerdo al alcance definido.
- Construcción: Se obtiene un producto listo para su utilización que está documentado y tiene un manual de usuario. Se obtiene uno o varios entregables del producto que han pasado las pruebas. Se ponen estos entregables a consideración de un subconjunto de usuarios.
- Transición: El entregable ya está listo para su instalación en las condiciones reales. Puede implicar reparación de errores. [27]

**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. [19]

C++ es un lenguaje de programación diseñado a mediados de los años 1980 por Bjarne Stroustrup. La intención de su creación fue el extender al exitoso lenguaje de programación C con mecanismos que permitan la manipulación de objetos. En ese sentido, desde el punto de vista de los lenguajes orientados a objetos, el C++ es un lenguaje híbrido.

Posteriormente se añadieron facilidades programación genérica, que se sumó a los otros dos paradigmas que ya estaban admitidos (programación estructurada y la programación orientada a objetos). Por esto se suele decir que el C++ es un lenguaje multiparadigma.

Actualmente existe un estándar, denominado ISO C++, al que se han adherido la mayoría de los fabricantes de compiladores más modernos. Existen también algunos intérpretes, tales como ROOT.

Una particularidad del C++ es la posibilidad de redefinir los operadores (sobrecarga de operadores), y de poder crear nuevos tipos que se comporten como tipos fundamentales. C++ permite trabajar tanto a alto como a bajo nivel. A partir de dichas razones y por estudios realizados en el proyecto sobre los diferentes lenguajes de programación posibles a utilizar en el proyecto se seleccionó dicho lenguaje.

**Biblioteca de desarrollo de interfaces gráficas de usuario: Gtkmm**

Las interfaces gráficas de usuario son la cara visible de las aplicaciones, de los sistemas operativos y también de otros tipos de dispositivos electrónicos. Ayudan al usuario a interactuar con la máquina, expanden el rango de aplicaciones de una computadora de forma considerable y representan una importante ayuda para el aprendizaje del trabajo con ordenadores. Además han tenido una gran importancia en la popularización de la informática fuera de ámbitos corporativos y científicos, al reducir

la cantidad de conocimiento acerca de las máquinas necesaria para un uso eficaz, práctico y útil de las mismas.

En Ciencias de la Computación, dentro de la disciplina de la Interacción Persona Computador, se define la GUI (Graphics User Interface, Interfaz Gráfica de Usuario), como el medio de interacción entre un usuario y un sistema informático que se realiza mediante el lenguaje visual. Esta interfaz debe proveer al usuario un ambiente agradable y sencillo para el correcto entendimiento y ejecución del programa. [10]

GTK (GIMP Toolkit) es una biblioteca para crear interfaces gráficas de usuario. Su licencia es la LGPL lo que permite desarrollar programas con licencias abiertas, gratuitas, libres, y hasta licencias comerciales no libres sin mayores problemas.

Se llama el GIMP toolkit porque fue escrito para el desarrollo del General Image Manipulation Program (GIMP), pero ahora GTK se utiliza en un gran número de proyectos de programación, incluyendo el proyecto GNU Network Object Model Environment (GNOME). GTK está construido encima de GDK (GIMP Drawing Kit) que básicamente es un recubrimiento de las funciones de bajo nivel que deben haber para acceder al sistema de ventanas sobre el que se programe (Xlib en el caso de X windows). Los principales autores de GTK son:

- Peter Mattis ( petm@xcf.berkeley.edu )
- Spencer Kimball (spencer@xcf.berkeley.edu )
- Josh MacDonald ( jmacd@xcf.berkeley.edu )

GTK es esencialmente una interfaz para la programación de aplicaciones orientadas al objeto (API). Aunque está completamente escrito en C, esta implementado haciendo uso de la idea de clases y de funciones respuesta o de callback (punteros o funciones).

Existe un tercer componente llamado glib, que contiene varias funciones para reemplazar algunas llamadas estándar, así como funciones adicionales para manejar listas enlazadas. Se reemplazan algunas funciones para aumentar la portabilidad de GTK, ya que algunas de las funciones implementadas no están disponibles o no son estándar en otros unixs, como por ejemplo g_strerror(). Algunas otras contienen mejoras a la versión de libc, como g_malloc que mejora las posibilidades de encontrar errores.

Existen recubrimientos GTK para muchos otros lenguajes, incluyendo C++, Guile, Perl, Python, TOM, Ada95, Objective C, Free Pascal, y Eiffel. [11]

**Biblioteca Boost**

Para el desarrollo de la aplicación era necesario trabajar con bibliotecas que permitiera las siguientes funcionalidades:

- Salvar la configuración de la aplicación mediante algún tipo de serialización.
- Trabajo con métodos concurrentes.
- Casteo de diferentes tipos de datos primitivos (string, int, float).
- Utilización de las expresiones regulares como método de validación.

Teniendo en cuenta estudios realizados por el proyecto se decidió utilizar la biblioteca Boost.

Boost es un conjunto de librerías de código abierto. Su licencia permite que sea utilizada en cualquier tipo de proyectos, ya sean comerciales o no. Su diseño e implementación permiten que sea utilizada en un amplio espectro de aplicaciones y plataformas. Abarca desde librerías de propósito general hasta abstracciones del sistema operativo. Con el objetivo de alcanzar el mayor rendimiento y flexibilidad se hace un uso intensivo de plantillas. Boost ha representado una fuente de trabajo e investigación en programación genérica y metaprogramación en C++.

**Biblioteca FPrint**

LibFprint es una biblioteca diseñada para el desarrollo fácil de herramientas que requieran autenticación biométrica por medio de la huella dactilar. Está escrita en C bajo la licencia LGPL. Está desarrollada para Linux pero es posible su portabilidad a partir de la recompilación de su código fuente.

Esta biblioteca ofrece múltiples funcionalidades como:

- Captura de la huella dactilar.
- Eliminación del ruido de la imagen.
- Binarización de la imagen obtenida.

- Obtención de las minucias de las huella.
- Comparación de dos imágenes.
- Persistencia en una imagen la huella obtenida con todas las mejoras antes mencionadas.

Tiene incluido los siguientes drivers:

- aes1610: Authentec AES1610
- aes2501: Authentec AES2501
- aes4000: AuthenTec AES4000
- fdu2000: SecuGen FDU 2000
- upeksonly: UPEK TouchStrip sensor-only
- upektc: UPEK TouchChip
- upekts: UPEK TouchStrip with biometric co-processor
- uru4000: Digital Persona U.are.U 4000/4000B
- vcom5s: Veridicom 5thSense

**Gestor de Base de Datos: PostgreSQL**

Una base de datos es una colección de datos organizados y estructurados según un determinado modelo de información que refleja no sólo los datos en sí mismos, sino también las relaciones que existen entre ellos. Una base de datos se diseña con un propósito específico y debe ser organizada con una lógica coherente. Los datos podrán ser compartidos por distintos usuarios y aplicaciones, pero deben conservar su integridad y seguridad al margen de las interacciones de ambos. La definición y descripción de los datos han de ser únicas para minimizar la redundancia y maximizar la independencia en su utilización.

Los sistemas de gestión de base de datos (SGBD); (en inglés: DataBase Management System, abreviado DBMS) son un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan. [14]

PostgreSQL es un servidor de base de datos orientada a objetos de software libre, liberado bajo la licencia BSD. Como muchos otros proyectos opensource, el desarrollo de PostgreSQL no es manejado por una sola compañía sino que es dirigido por una

comunidad de desarrolladores y organizaciones comerciales las cuales trabajan en su desarrollo. Dicha comunidad es denominada el PGDG (PostgreSQL Global Development Group).

**Herramientas**

Las herramientas son artefactos usados durante todo el ciclo de vida de desarrollo de un software para su realización. Existen varios tipos de ellas según su funcionalidad. Algunos ejemplos de tipos de herramientas son:

- Herramientas para la modelación.
- Herramientas para el desarrollo.
- Herramientas para el desarrollo de interfaz visual
- Gestores de documentación de código.

**IDE: Eclipse**

Un entorno de desarrollo integrado (Integrated Development Environment o IDE) es un programa compuesto por una serie de herramientas que utilizan los programadores para desarrollar código. Esta herramienta puede estar pensada para su utilización con un único lenguaje de programación o bien puede dar cabida a varios de estos.

Las herramientas que normalmente componen un entorno de desarrollo integrado son las siguientes: un editor de texto, un compilador, un intérprete, unas herramientas para la automatización, un depurador, un sistema de ayuda para la construcción de interfaces gráficas de usuario y, opcionalmente, un sistema de control de versiones.

Hoy en día los entornos de desarrollo proporcionan un marco de trabajo para la mayoría de los lenguajes de programación existentes en el mercado (por ejemplo C, C++, C#, Java, Python y Visual Basic entre otros). Además es posible que un mismo entorno de desarrollo tenga la posibilidad de utilizar varios lenguajes de programación, como es el caso de Eclipse. [16]

**CASE: RSA**

Las herramientas CASE (Computer Aided Software Engineering, Ingeniería de Software Asistida por Ordenador) son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de dinero. Estas herramientas nos pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del software en tareas como el proceso de realizar un diseño del proyecto, calculo de costes, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores entre otras.[23]

La herramienta seleccionada por el proyecto a partir de estudios previos realizados fue IBM Rational Software Architect (RSA). RSA es una herramienta de diseño y desarrollo integrados que potencia el desarrollo orientado al modelado con UML para la creación de aplicaciones y servicios con buena arquitectura.

Algunas de las ventajas que nos brinda IBM Rational Software Architect son:

- Unificar todos los aspectos del diseño y desarrollo de software.
- Desarrollar aplicaciones más productivamente.
- Sacar provecho de la tecnología más avanzada en lenguaje de modelado.
- Potenciar con una plataforma de modelado abierta y extensible.
- Simplificar con una solución para diseño y desarrollo en una herramienta.
- Integrar otras facetas del ciclo de vida.

**Generador de interfaz gráfica de usuario: Glade**

En la actualidad la necesidad de utilizar interfaces gráficas en los programas impone una carga extra en los diseñadores y desarrolladores de software, pues deben incorporarlas en sus aplicaciones aún en etapas muy tempranas del desarrollo del mismo. Las interfaces gráficas son muy utilizadas en el desarrollo de prototipos para obtener y validar los requerimientos de las aplicaciones en la etapa de análisis del proceso de desarrollo de software.

Muchos lenguajes de programación no proporcionan librerías estándar para el manejo gráfico y, aún en aquellos en que se proporcionan dichas librerías, el paso de todos los parámetros necesarios para crear y colocar los diversos elementos en la pantalla es muy complicado y requiere un gran esfuerzo por parte de quien implementa dichas interfaces. Para solventar esta necesidad, surgieron una serie de herramientas especializadas de software que permiten crear esas interfaces gráficas en forma sencilla reduciendo el número de parámetros que debe introducir el usuario y proporcionando un alto nivel de abstracción para el diseño de las mismas. Estas herramientas son denominadas creadores de interfaces gráficas de usuario o diseñadores de pantallas.

Ya seleccionado a Gtkmm como lenguaje de programación visual se trabajará con Glade (Glade Interface Desiner, que significa Diseñador de interfaz Glade) como herramienta. Como principal característica se tiene que es totalmente independiente al lenguaje de programación seleccionado y cuenta con la posibilidad de generar un archivo XML que es cargado en tiempo de ejecución de la aplicación.

**Generador de documentación: Doxygen**

El paquete Doxygen contiene un sistema de documentación para C++, C, Java, Objective-C, Corba IDL y, en parte, PHP, C# y D. Es útil para generar documentación HTML y/o un manual de referencia a partir de un grupo de ficheros fuente documentados. También soporta generación de salida en RTF, PostScript, PDF con hiperlinks, HTML comprimido y páginas de manual Unix. La documentación se extrae directamente de las fuentes, lo que hace mucho más fácil mantener consistente la documentación con el código fuente. [18]

# Capítulo 3: Descripción de la solución propuesta

## Introducción

En este capítulo se realiza el proceso de modelación del sistema. Para ellos de especifican los requerimientos funcionales y no funcionales que debe tener la aplicación, se identificarán los actores y se describirán los principales casos de usos y se construirán varios diagramas para su mejor entendimiento.

## Requerimientos

De las muchas definiciones que existen para requerimiento, a continuación se presenta la definición que aparece en el glosario de la IEEE.

1. Una condición o necesidad de un usuario para resolver un problema o alcanzar un objetivo.
2. Una condición o capacidad que debe estar presente en un sistema o componentes de sistema para satisfacer un contrato, estándar, especificación u otro documento formal.
3. Una representación documentada de una condición o capacidad como en (1) o (2).

Los requerimientos pueden dividirse en requerimientos funcionales y no funcionales.

## Requerimientos funcionales

Los requerimientos funcionales son capacidades o condiciones que el sistema debe cumplir. El sistema de autenticación debe permitir:

1. Autenticar.
   1.1. Autenticación por contraseña.
   1.2. Autenticación biométrica.
2. Cambiar contraseña.
3. Cambiar parámetro biométrico.
4. Administrar usuario.

4.1. Agregar un nuevo usuario.
4.2. Consultar datos de usuarios existentes.
4.3. Modificar datos de los usuarios.
4.4. Eliminar usuarios.

5. Búsqueda avanzada de usuarios.
6. Confirmar contraseña.

**Requerimientos no funcionales**

Los requerimientos 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.

- Requerimiento de Rendimiento y Disponibilidad del Sistema.
  - El sistema debe garantizar su restablecimiento tras alguna falla del servidor o de la red, mostrándole al usuario las vistas de los diferentes tipos de autenticación para darle la posibilidad de que se autentique nuevamente.
- Restricciones en el Diseño y la Implementación.
  - El sistema debe contar con un conjunto de patrones que permita la reutilización del código así como facilidad en la actualización del mismo.
  - El código debe cumplir con los estándares de codificación establecidos por el cliente.
- Requerimientos de Apariencia o Interfaz Externa.
  - Se debe garantizar que los colores de la aplicación sean los mas claros posibles.
  - La interfaz debe ser de fácil comprensión en su funcionamiento permitiendo la utilización del sistema sin mucho adiestramiento.
  - Mostrar el nombre de la aplicación en la parte superior mediante una barra de título así como el nombre de cada una de las ventanas en dependencia de la funcionalidad que está presente.
  - Mostrar opciones funcionales (Mostrar, Eliminar, Bloquear) pudiendo acceder a través de las barras de herramientas o mediante clic derecho sobre los componentes.
- Requerimientos de Seguridad.

- El sistema identificar al usuario a partir de los diferentes tipos de autenticación permitiéndole realizar cualquier acción si su autenticación es satisfactoria.
- El sistema le preguntará nuevamente la identificación del usuario al tratar de realizar operaciones críticas como es el cambio de la contraseña o del parámetro biométrico (huella dactilar).

- Requerimientos de Usabilidad.
  - El sistema podrá ser usado por cualquier tipo de persona que posea conocimientos básicos en el manejo de la computadora y el sistema operativo Linux.
- Requerimientos de Soporte.
  - Se debe ofrecer servicios de mantenimiento y actualización.
- Requerimiento de Portabilidad.
  - El sistema debe funcionar en el sistema operativo GNU/Linux con la distribución Debian.
- Requerimiento de Confiabilidad.
  - Se debe garantizar que el sistema esté disponible las 24 horas del día.
  - Las actividades de mantenimiento, supervisión y edición del sistema no deben interferir en el correcto desempeño del resto del sistema, ni afectar la ejecución de la aplicación.
- Requerimiento de Documentación de Usuario.
  - Se debe garantizar que el sistema cuente con una correcta documentación.
- Requerimiento asociado al Licenciamiento.
  - Se debe garantizar que el sistema de desarrolle bajo los principios del software libre y por tanto cualquier componente de software que se utilice también lo debe ser.

**Modelación del sistema**

El modelado, o modelización, es una técnica cognitiva que consiste en crear una representación ideal de un objeto real mediante un conjunto de simplificaciones y abstracciones, cuya validez se pretende constatar. La validación del modelo se lleva a cabo comparando las implicaciones predichas por el mismo con observaciones. En otras palabras, se trata usar un modelo irreal o ideal, y reflejarlo sobre un objeto.

Un modelo es una simplificación de la realidad, se recogen aquellos aspectos de gran importancia y se omiten los que no tienen relevancia para el nivel de abstracción dado. Se modela para comprender mejor un sistema. Los sistemas complejos no se pueden comprender en toda su completitud. [25]

**Actores**

Los actores representan personas o sistemas externos que interactúan con la aplicación. En el SCADA podemos encontrar 3 perfiles primarios.

- Usuario: Podrá realizar operaciones con los privilegio mínimos requeridos. Estas funcionalidades son:
  - Autenticarse a través de su usuario y contraseña.
  - Autenticarse a través de su huella dactilar.
  - Cambiar su contraseña.
  - Cambiar su parámetro biométrico.
- Administrador del SCADA: Podrá realizar operaciones de administración y tendrá todos los privilegios de los Usuarios. Las funcionalidades adicionales son:
  - Administrar usuarios.
- Administrador de Seguridad: Podrá realizar operaciones de administración distintas de las del Administrador del SCADA y tendrá todos los privilegios de los Usuarios. Las funcionalidades adicionales son:
  - Búsqueda avanzada de usuarios.

**Casos de uso**

A continuación se realizará la descripción de los principales casos de usos.

- Autenticación por contraseña.

Desarrollo SCADA Nacional

Caso de Uso Autenticación por contraseña

Versión 1.0

Histórico de revisión

| Fecha | Versión | Descripción | Autor |
|---|---|---|---|
| 05/04/2009 | 1.0 | Creación del documento | Enrique Reyes |

1. Nombre: Autenticación por contraseña.
2. Breve descripción:

Este caso de uso le permite al usuario autenticarse mediante su usuario y contraseña.

3. Actores: Usuario
4. Flujo básico.
   4.1. El caso de uso comienza cuando el sistema solicita al usuario su autenticación.
   4.2. Introducir el usuario.

   El sistema solicita al Usuario su nombre de usuario. El usuario suministra su nombre de usuario.

   4.3. Introducir la contraseña.

   El sistema solicita al Usuario su contraseña. El usuario suministra su contraseña.

   4.4. Actualizar IP y puerto del servidor.

   El sistema le da la posibilidad al usuario de actualizar el IP y el puerto del servidor.

   4.5. Aceptar la autenticación.

   El sistema realiza la autenticación con los datos suministrados por el usuario.

5. Flujo alterno.
   5.1. Paso 4.5 autenticación fallida.

   Si la autenticación no fue satisfactoria el sistema el sistema muestra información al usuario sobre las posibles causas. El sistema retorna al paso 4.2.

- Autenticación biométrica.

Desarrollo SCADA Nacional

Caso de Uso Autenticación biométrica

Versión 1.0

Histórico de revisión

| Fecha | Versión | Descripción | Autor |
|---|---|---|---|
| 05/04/2009 | 1.0 | Creación del documento | Enrique Reyes |

1. Nombre: Autenticación biométrica.
2. Breve descripción:

Este caso de uso le permite al usuario autenticarse mediante su huella dactilar.

3. Actores: Usuario
4. Flujo básico.
   4.1. El caso de uso comienza cuando el sistema solicita al usuario su autenticación.
   4.2. Introducir la hulla dactilar.

   El sistema solicita al Usuario su huella dactilar. El usuario suministra su huella al pasar su dedo por el lector.

   4.3. Actualizar IP y puerto del servidor.

   El sistema le da la posibilidad al usuario de actualizar el IP y el puerto del servidor.

   4.4. Aceptar la autenticación.

   El sistema realiza la autenticación con los datos suministrados por el usuario.

5. Flujo alterno.
   5.1. Paso 4.5 autenticación fallida.

Si la autenticación no fue satisfactoria el sistema el sistema muestra información al usuario sobre las posibles causas. El sistema retorna al paso 4.2.

- Cambiar contraseña.

Desarrollo SCADA Nacional

Caso de Uso Cambiar contraseña

Versión 1.0

Histórico de revisión

| Fecha | Versión | Descripción | Autor |
|---|---|---|---|
| 05/04/2009 | 1.0 | Creación del documento | Enrique Reyes |

1. Nombre: Cambiar contraseña.
2. Breve descripción:

Este caso de uso le permite al usuario cambiar su contraseña.

3. Actores: Usuario
4. Flujo básico.

   4.1. El caso de uso comienza cuando el usuario selecciona Cambiar contraseña.

   4.2. Verificar usuario.

   El sistema solicita al Usuario su nombre contraseña. El usuario suministra su contraseña. El sistema realiza la verificación del usuario.

   4.3. Introducir la contraseña y su confirmación.

   El sistema solicita al Usuario su contraseña y la confirmación de la misma. El usuario suministra la nueva contraseña.

   4.4. Actualización de la contraseña.

El sistema realiza la actualización de la contraseña.

5. Flujo alterno.

   5.1. Paso 4.2 verificar usuario.

   Si la verificación resulta fallida el sistema retorna al paso 4.2.

   5.2. Paso 4.4. Actualización de la contraseña.

   Si la actualización de la contraseña resulta fallida el sistema retorna al paso 4.3.

- Cambiar parámetro biométrico.

Desarrollo SCADA Nacional

Caso de Uso Cambiar parámetro biométrico

Versión 1.0

Histórico de revisión

| Fecha | Versión | Descripción | Autor |
|---|---|---|---|
| 05/04/2009 | 1.0 | Creación del documento | Enrique Reyes |

1. Nombre: Cambiar parámetro biométrico.
2. Breve descripción:

Este caso de uso le permite al usuario cambiar su huella dactilar.

3. Actores: Usuario
4. Flujo básico.

   4.1. El caso de uso comienza cuando el usuario selecciona Cambiar huella.

   4.2. Verificar usuario.

   El sistema solicita al Usuario su nombre contraseña. El usuario suministra su contraseña. El sistema realiza la verificación del usuario.

4.3. Introducir la huella dactilar y su confirmación.

El sistema solicita al Usuario su huella dactilar y la confirmación de la misma. El usuario suministra la nueva huella dactilar.

4.4. Actualización de la huella dactilar.

El sistema realiza la actualización de la huella dactilar.

5. Flujo alterno.
   5.1. Paso 4.2 verificar usuario.

   Si la verificación resulta fallida el sistema retorna al paso 4.2.

   5.2. Paso 4.4. Actualización de la huella dactilar.

   Si la actualización de la huella dactilar resulta fallida el sistema retorna al paso 4.3.

**Diagramas de casos de uso**

Luego de haber definido los actores y requerimientos funcionales del sistema, se modela la relación que existe entre los actores Usuario, Administrador de Seguridad y Administrador del SCADA con los casos de usos definidos a partir de la captura de los requerimientos.

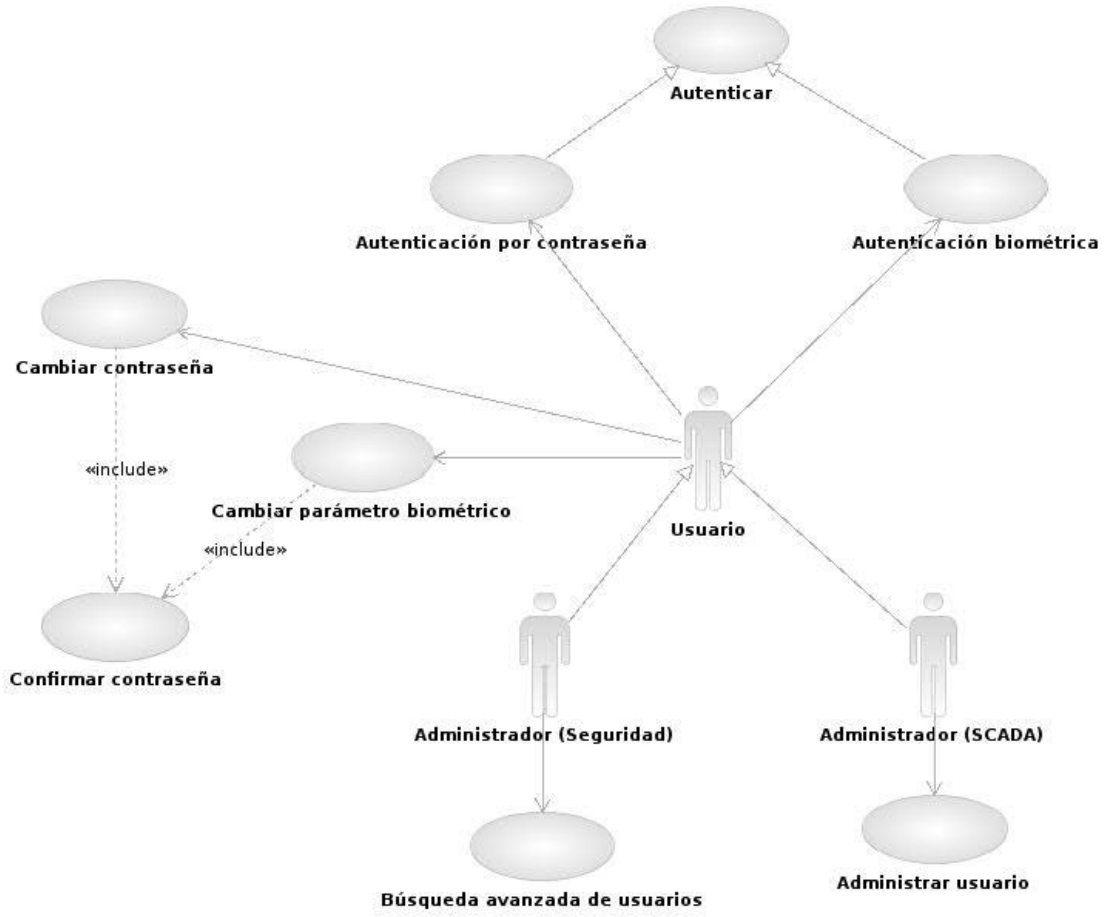


Figura 6: Diagrama de Casos de Uso

**Diagrama de actividades**

El Diagrama de Actividad es un diagrama de flujo del proceso multi-propósito que se utiliza para modelar el comportamiento del sistema. Los diagramas de actividad se pueden usar para modelar un caso de uso, una clase, o un método complicado. A continuación se muestran algunos diagramas de actividad para determinados casos de usos.

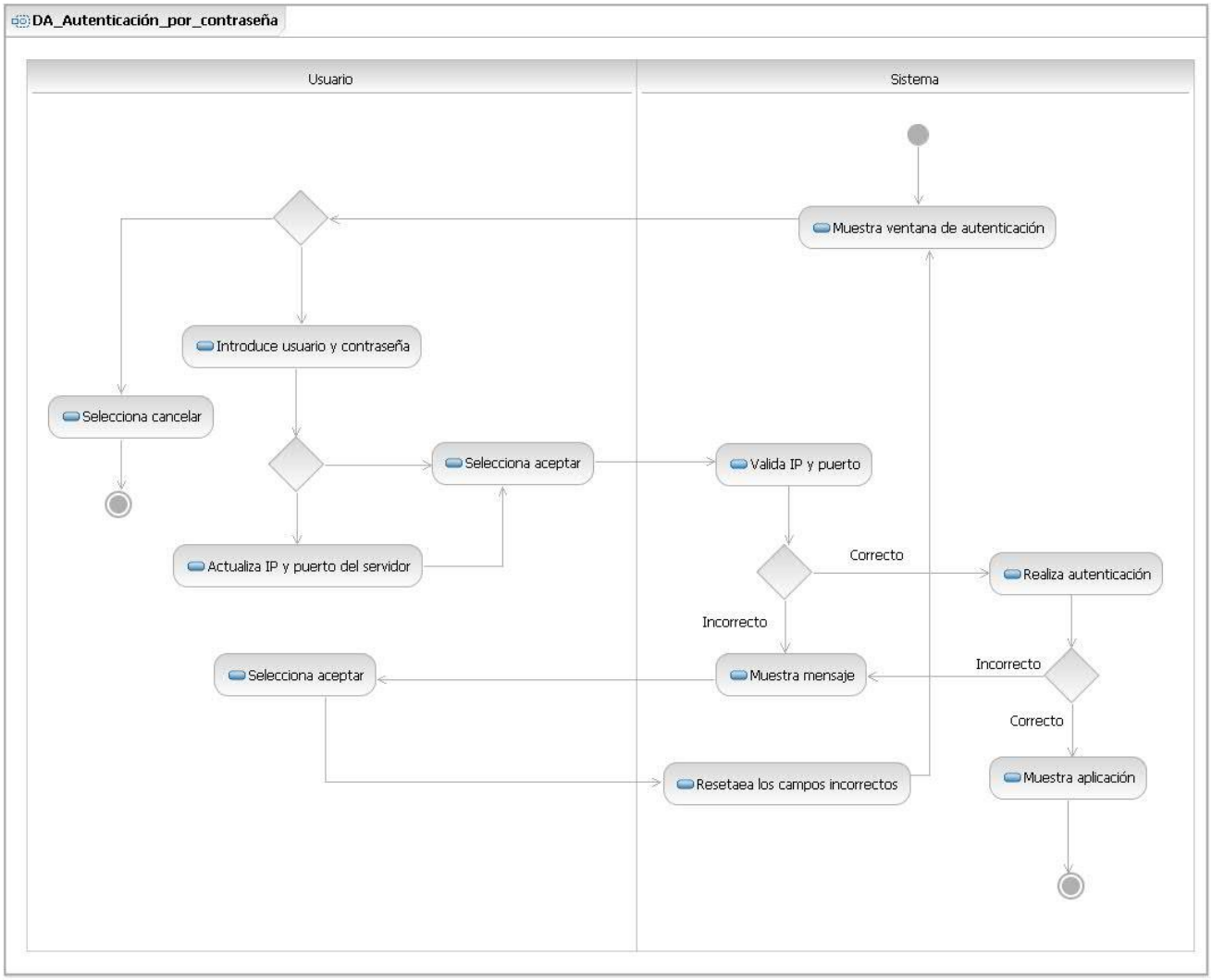


Figura 7: Diagrama de actividad Caso de Uso Autenticación por contraseña

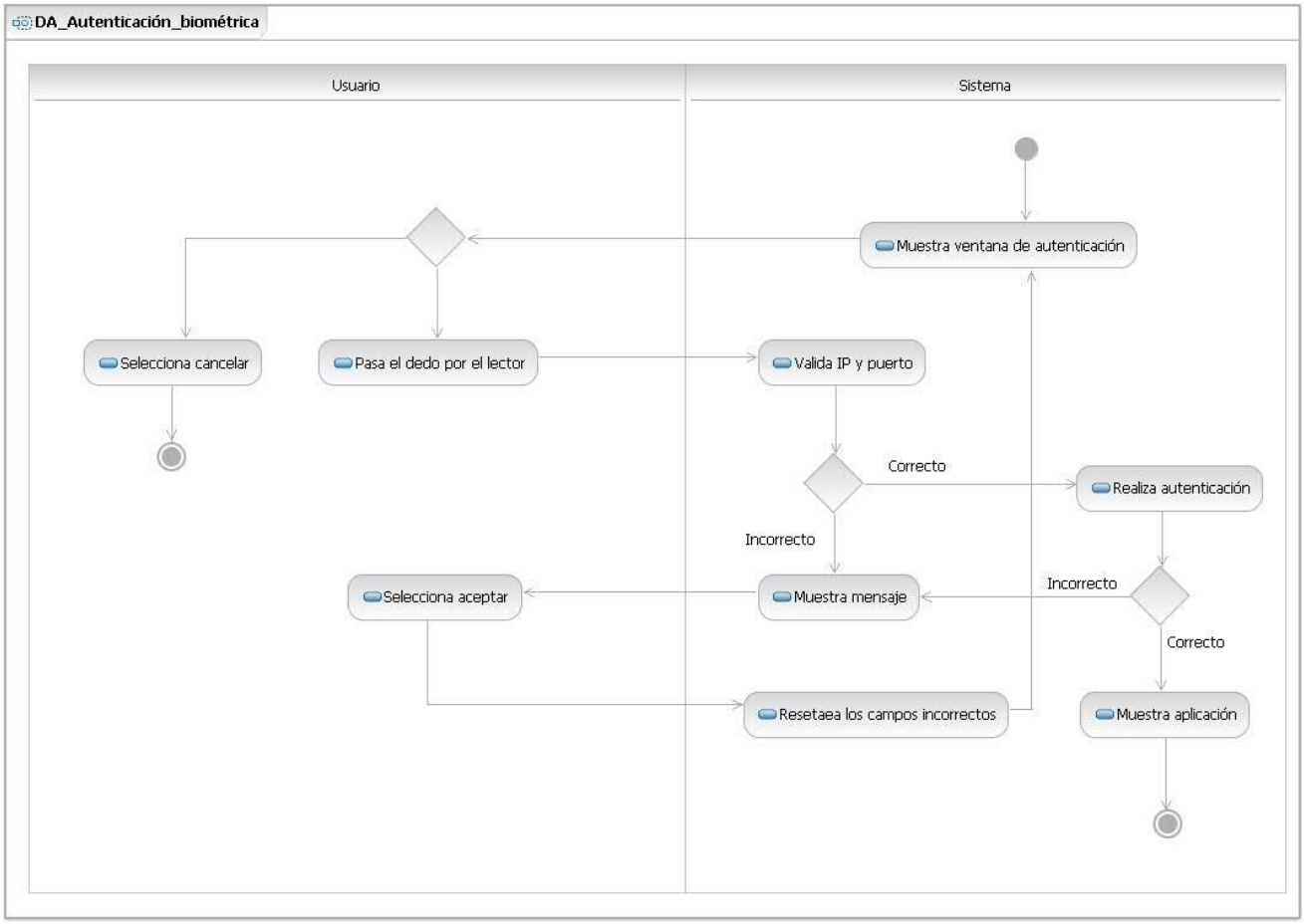


Figura 8: Diagrama de actividad Caso de Uso Autenticación biométrica

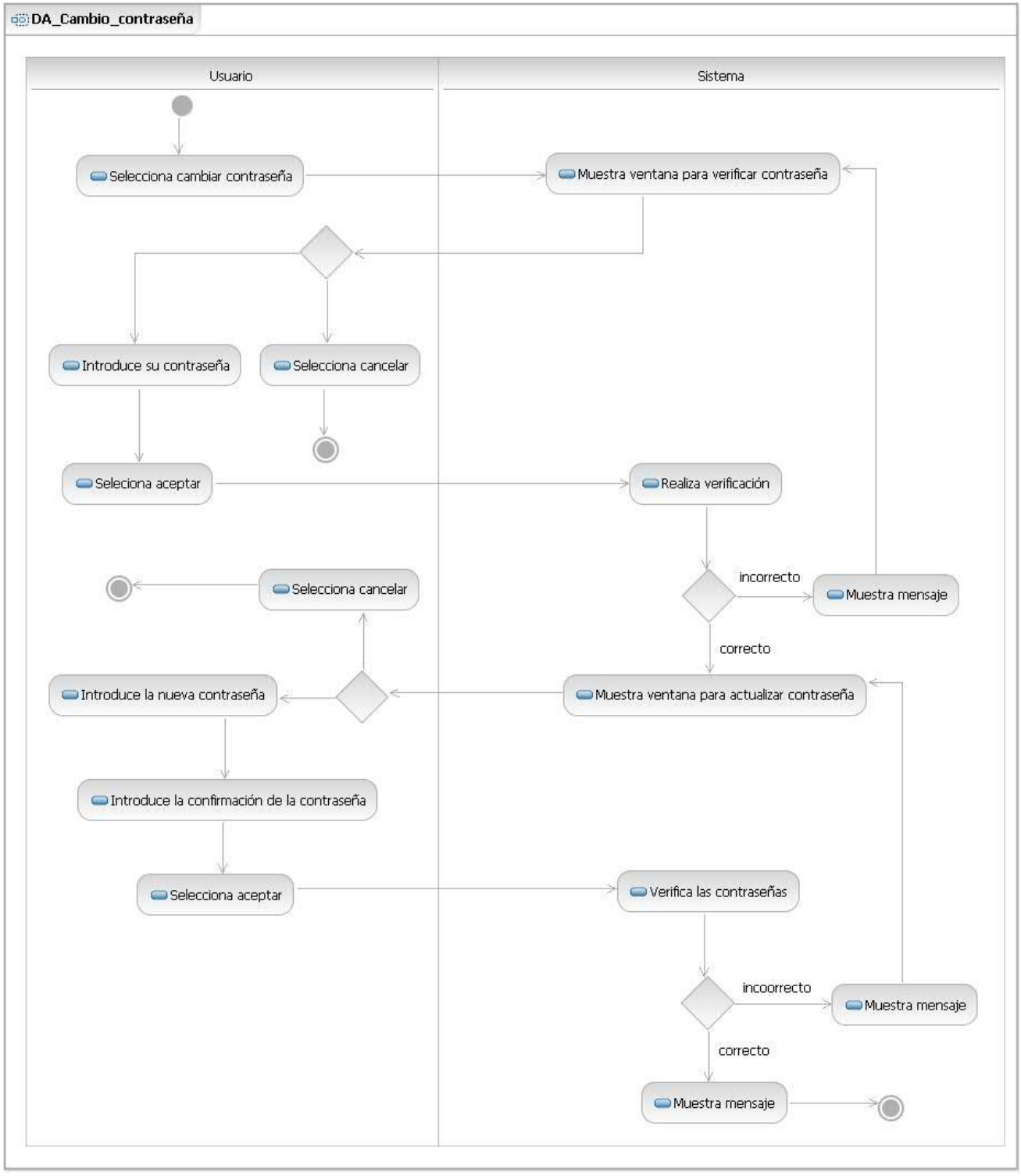


Figura 9: Diagrama de actividad Caso de Uso Cambiar contraseña

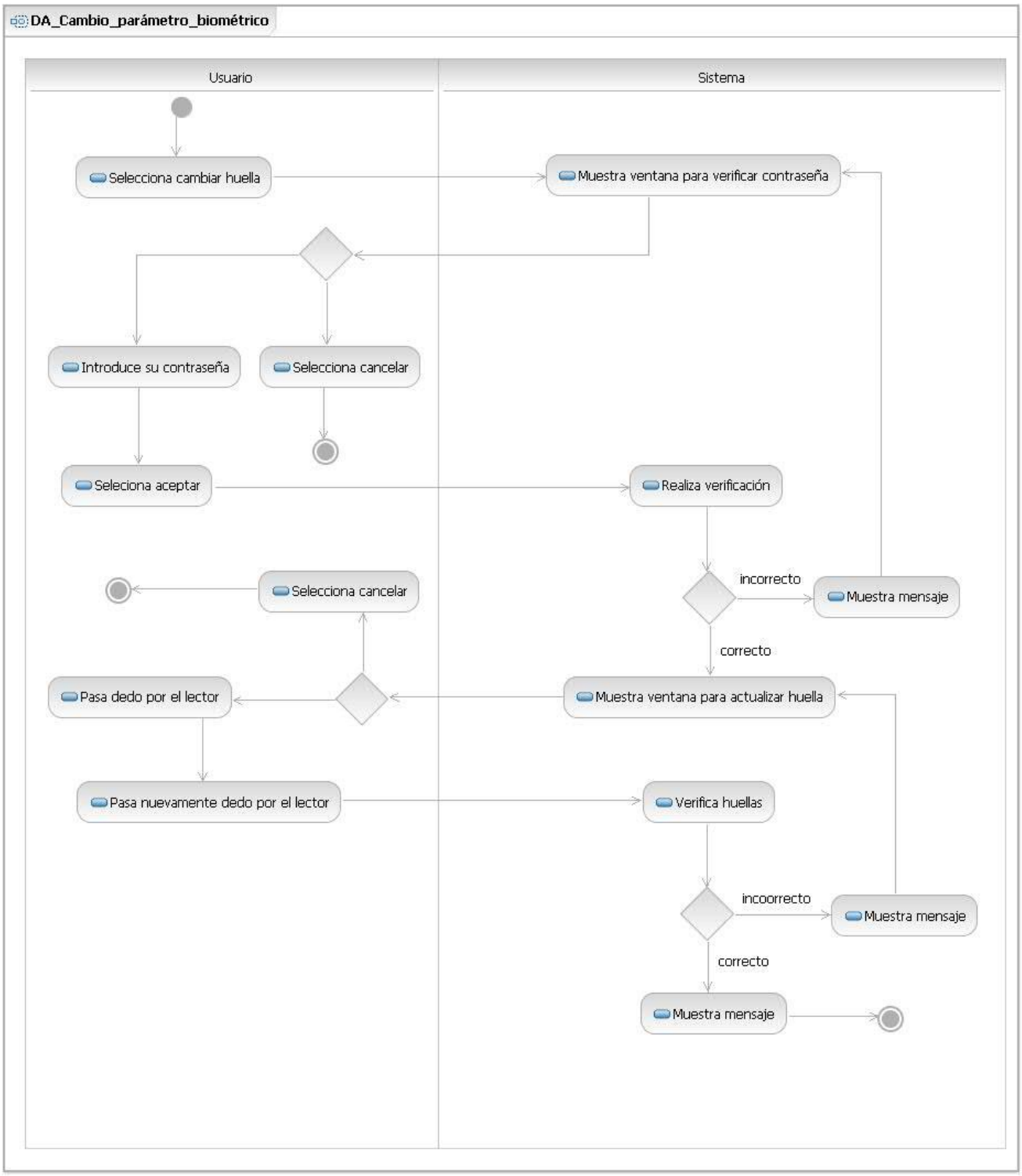


Figura 10: Diagrama de actividad Caso de Uso Cambio del parámetro biométrico

**Análisis y diseño**

A continuación se desarrollará la modelación de las diferentes clases utilizadas en la aplicación así como sus relaciones e interacción.

**Diagrama de clases persistentes**

Para describir la estructura del sistema a través de entidades y sus campos se realizan los diagramas de clases persistentes. Para el desarrollo del sistema se utilizó una sola entidad llamada User con los tipos de datos necesarios.

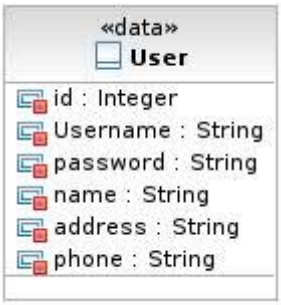


Figura 11: Diagrama de clases persistentes

**Diagramas de clases**

A continuación se mostrarán los diagramas de clases utilizados en la aplicación en dependencia de los diferentes casos de uso.

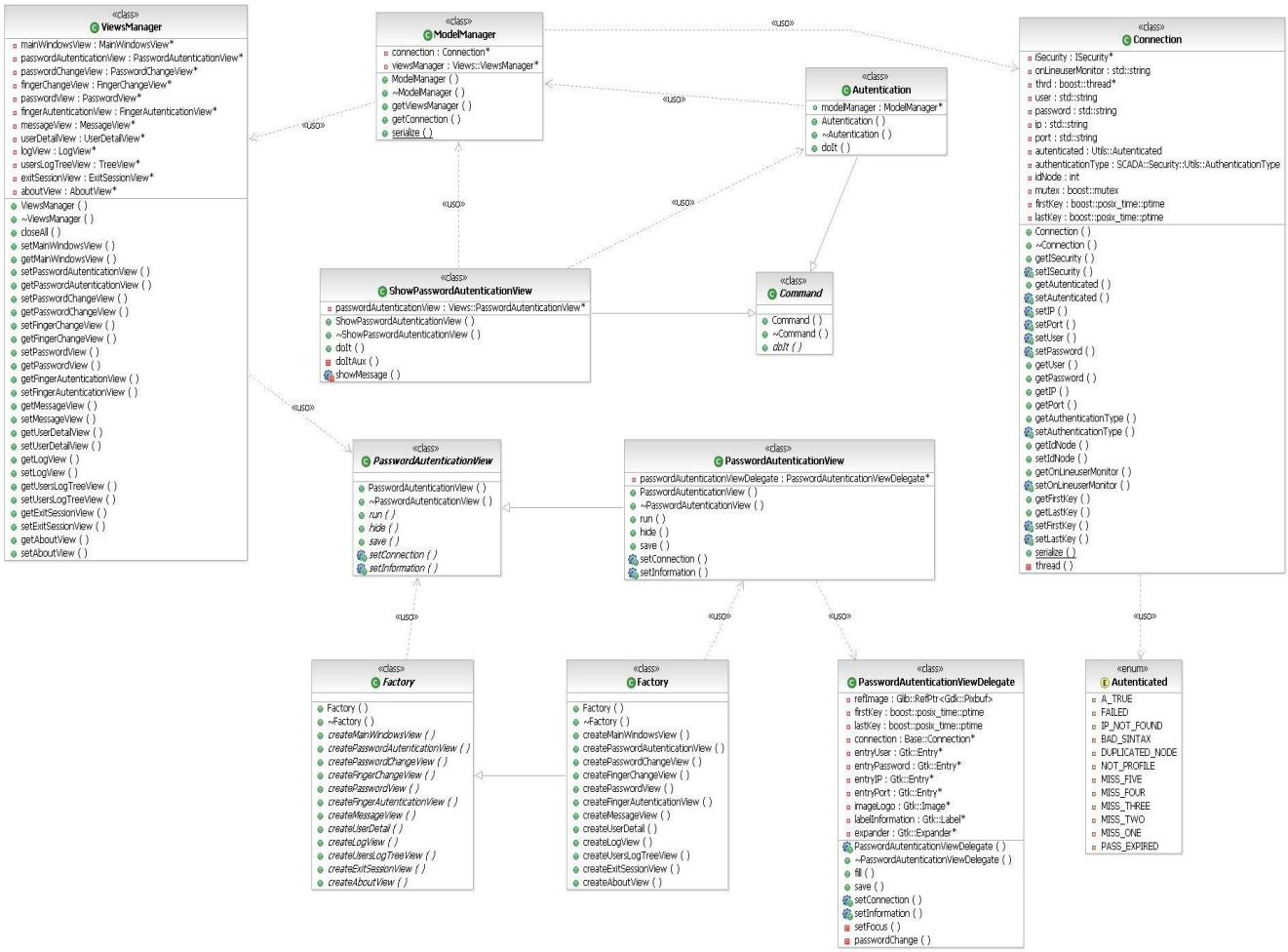


Figura 12: Diagrama de clases Caso de Uso Autenticación por contraseña

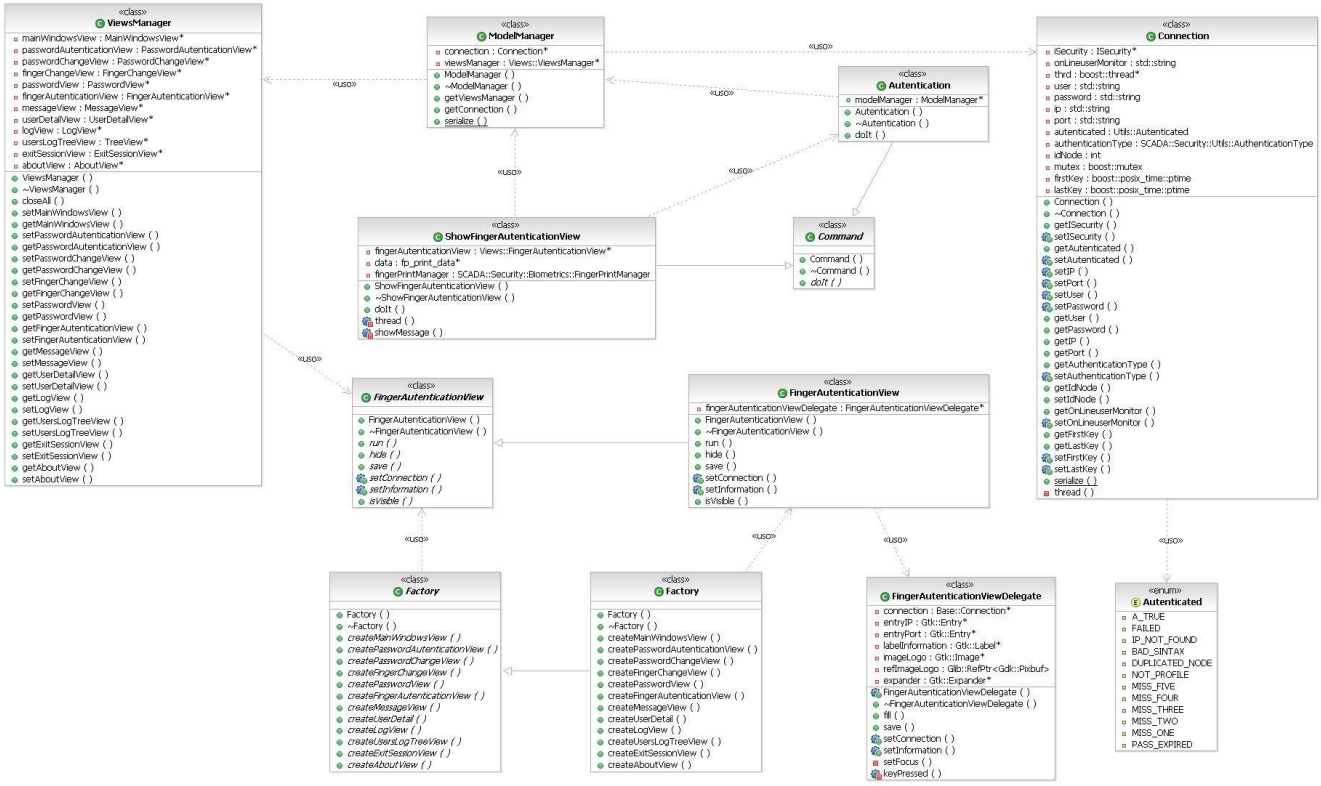


Figura 13: Diagrama de clases Caso de Uso Autenticación biométrica

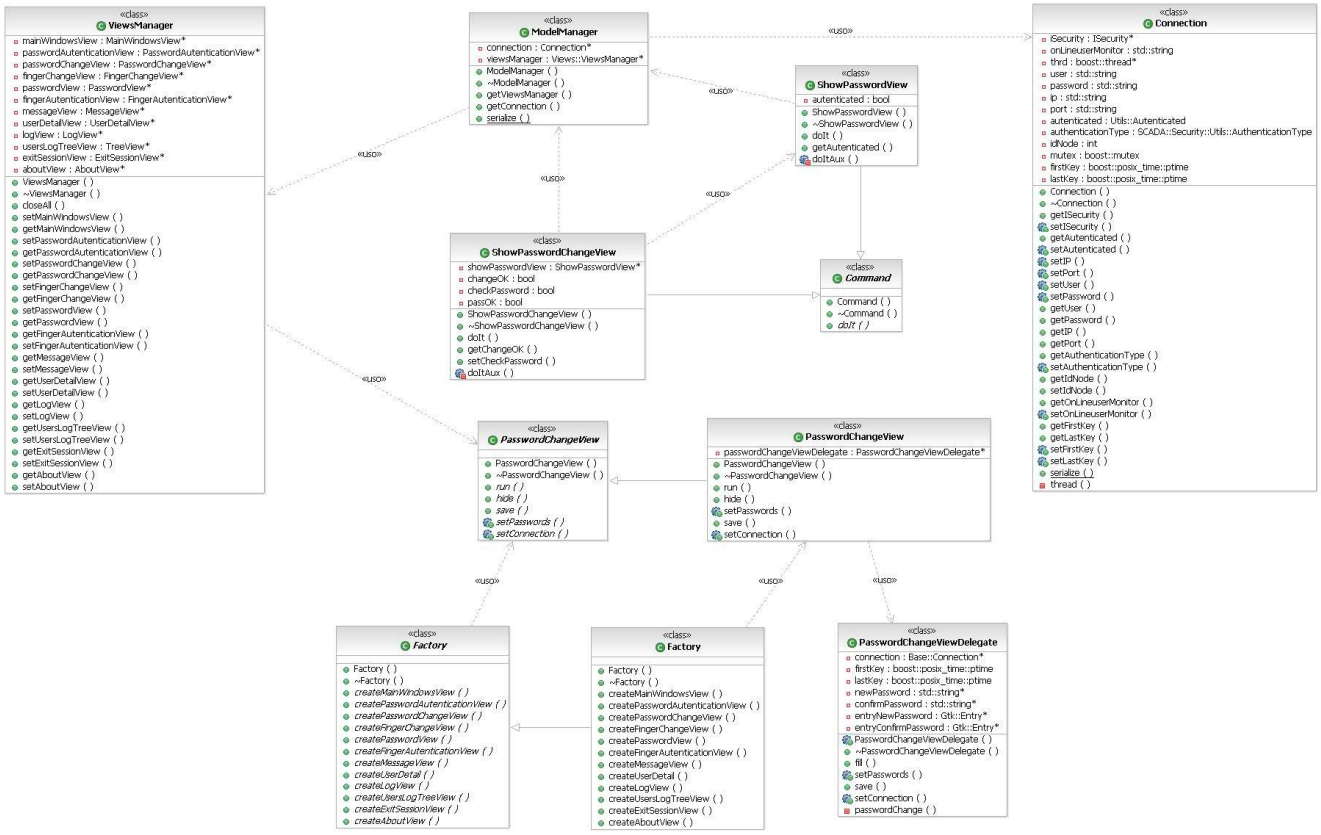


Figura 14: Diagrama de clases Caso de Uso Cambiar contraseña

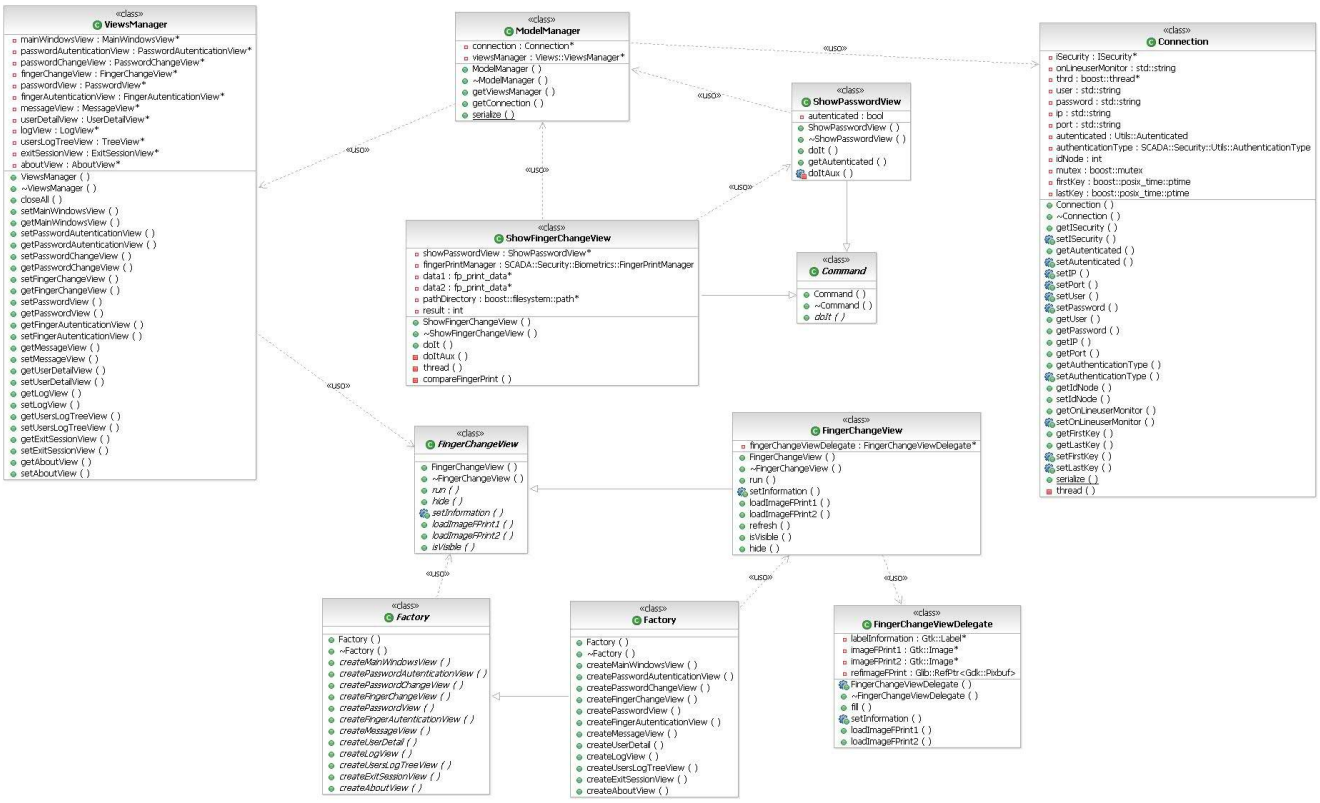


Figura 15: Diagrama de clases Caso de Uso Cambiar parámetro biométrico

**Descripción de las principales clases**

A continuación se realizará la descripción de las principales clases con el objetivo de familiarizarse mejor con sus atributos y métodos.

| Nombre: ModelManager | |
|---|---|
| Tipo de clase: Entidad | |
| Atributo | Tipo |
| viewsManager | ViwesManager* |
| connection | Connection* |
| Para cada responsabilidad: | |
| Nombre: | ModelManager() |
| Descripción: | Constructor por defecto |
| Nombre: | ~ModelManager() |
| Descripción: | Destructor |

| | |
|---|---|
| Nombre: | getViewsManager() |
| Descripción: | Retorna el administrador de las vistas |
| Nombre: | getConnection () |
| Descripción: | Retorna el objeto encargado de establecer la conexión con el servidor |
| Nombre: | serialize( Archive & ar, const unsigned int version ) () |
| Descripción: | Realiza la serialización de los atributos de la clase |

Tabla 2: Descripción de la clase ModelManager

| Nombre: Connection | |
|---|---|
| Tipo de clase: Entidad | |
| Atributo | Tipo |
| iSecurity | ISecurity* |
| onLineuserMonitor | std::string |
| thrd | boost::thread* |
| user | std::string |
| password | std::string |
| ip | std::string |
| thrd | boost::thread* |
| user | std::string |
| port | std::string |
| autenticated | Utils::Autenticated |
| authenticationType | SCADA::Security::Utils::AuthenticationType |
| idNode | int |
| mutex | boost::mutex |
| Para cada responsabilidad: | |
| Nombre: | Connection () |
| Descripción: | Constructor por defecto |
| Nombre: | ~ Connection () |
| Descripción: | Destructor |
| Nombre: | getISecurity () |
| Descripción: | Retorna la interfaz de seguridad |
| Nombre: | getAutenticated () |

| | |
|---|---|
| Descripción: | Retorna el estado de la autenticación del cliente |
| Nombre: | setAutenticated( Utils::Autenticated autenticated ) |
| Descripción: | Cambia el estado de la autenticación del cliente |
| Nombre: | setUser( std::string user ) |
| Descripción: | Cambia el usuario |
| Nombre: | setPassword( std::string password ) |
| Descripción: | Cambia la contraseña |
| Nombre: | setIP( std::string ip ) |
| Descripción: | Cambia el IP |
| Nombre: | setPort( std::string port ) |
| Descripción: | Cambia el puerto |
| Nombre: | setAuthenticationType( SCADA::Security::Utils::AuthenticationType authenticationType ) |
| Descripción: | Cambia el tipo de autenticación |
| Nombre: | serialize( Archive & ar, const unsigned int version ) () |
| Descripción: | Realiza la serialización de los atributos de la clase |

Tabla 3: Descripción de la clase Connection

| Nombre: ViewsManager | |
|---|---|
| Tipo de clase: Entidad | |
| Atributo | Tipo |
| mainWindowsView | MainWindowsView* |
| passwordAutenticationView | PasswordAutenticationView |
| passwordChangeView | PasswordChangeView* |
| fingerChangeView | FingerChangeView* |
| passwordView | PasswordView* |
| fingerAutenticationView | FingerAutenticationView* |
| messageView | MessageView* |
| userDetailView | UserDetailView* |
| logView | LogView* |
| usersLogTreeView | TreeView* |
| exitSessionView | ExitSessionView* |

| aboutView | AboutView* |
|---|---|
| Para cada responsabilidad: | |
| Nombre: | ViewsManager () |
| Descripción: | Constructor por defecto |
| Nombre: | ~ ViewsManager () |
| Descripción: | Destructor |
| Nombre: | closeAll() |
| Descripción: | Cierra todas las vistas y bloquea la vista principal |
| Nombre: | getMainWindowsView() |
| Descripción: | Retorna la vista principal |
| Nombre: | getPasswordAutenticationView() |
| Descripción: | Retorna la vista para la autenticación por contraseña |
| Nombre: | getPasswordChangeView() |
| Descripción: | Retorna la vista para el cambio de la contraseña |
| Nombre: | getFingerAutenticationView() |
| Descripción: | Retorna la vista para la autenticación biométrica |
| Nombre: | getFingerChangeView() |
| Descripción: | Retorna la vista para el cambio del parámetro biométrico |
| Nombre: | getPasswordView() |
| Descripción: | Retorna la vista para la confirmación de la contarseña |
| Nombre: | getMessageView() |
| Descripción: | Retorna la vista para mostrar los mensajes |
| Nombre: | getUserDetailView () |
| Descripción: | Retorna la vista para mostrar los datos de los usuarios |
| Nombre: | getLogView () |
| Descripción: | Retorna la vista para realizar la búsqueda avanzada de usuarios |
| Nombre: | getUsersLogTreeView() |
| Descripción: | Retorna la vista donde se mostrará los usuarios encontrados a partir de la búsqueda |
| Nombre: | getExitSessionView () |
| Descripción: | Retorna la vista para mostrar las opciones de cierre de sesión |
| Nombre: | getAboutView () |

| | |
|---|---|
| Descripción: | Retorna la vista para mostrar los datos del proyecto |

Tabla 4: Descripción de la clase ViewsManager

| Nombre: Command | |
|---|---|
| Tipo de clase: Controladora | |
| Atributo | Tipo |
| | |
| Para cada responsabilidad: | |
| Nombre: | Command () |
| Descripción: | Constructor por defecto |
| Nombre: | ~ Command () |
| Descripción: | Destructor |
| Nombre: | doIt() |
| Descripción: | Método principal que ejecuta el objetivo del comando |

Tabla 5: Descripción de la clase Command

| Nombre: PasswordAutenticationViewDelegate | |
|---|---|
| Tipo de clase: Interfaz | |
| Atributo | Tipo |
| userEntry | Gtk::Entry* |
| userPassword | Gtk::Entry* |
| ipEntry | Gtk::Entry* |
| portEntry | Gtk::Entry* |
| logoImage | Gtk::Image* |
| expander | Gtk::Expander* |
| Para cada responsabilidad: | |
| Nombre: | PasswordAutenticationViewDelegate () |
| Descripción: | Constructor por defecto |
| Nombre: | ~ PasswordAutenticationViewDelegate () |
| Descripción: | Destructor |
| Nombre: | fill () |

| | |
|---|---|
| Descripción: | Método encargado de mostrar los datos cargados desde el modelo |
| Nombre: | save () |
| Descripción: | Método encargado de salvar los datos en el modelo |
| Nombre: | setConnection ( Base::Connection* connection ) |
| Descripción: | Método encargado de actualizar en la vista los datos del modelo |

Tabla 6: Descripción de la clase PasswordAutenticationViewDelegate

| Nombre: FingerAutenticationViewDelegate | |
|---|---|
| Tipo de clase: Interfaz | |
| Atributo | Tipo |
| informationLabel | Gtk::Label* |
| ipEntry | Gtk::Entry* |
| portEntry | Gtk::Entry* |
| logoImage | Gtk::Image* |
| expander | Gtk::Expander* |
| Para cada responsabilidad: | |
| Nombre: | FingerAutenticationViewDelegate () |
| Descripción: | Constructor por defecto |
| Nombre: | ~ FingerAutenticationViewDelegate () |
| Descripción: | Destructor |
| Nombre: | fill () |
| Descripción: | Método encargado de mostrar los datos cargados desde el modelo |
| Nombre: | save () |
| Descripción: | Método encargado de salvar los datos en el modelo |
| Nombre: | setConnection ( Base::Connection* connection ) |
| Descripción: | Método encargado de actualizar en la vista los datos del modelo |

Tabla 7: Descripción de la clase FingerAutenticationViewDelegate

**Diagramas de secuencia**

Los diagramas de secuencia muestran el intercambio de mensajes en un momento dado y ponen especial énfasis en el orden y el momento en que se envían los mensajes a los objetos.

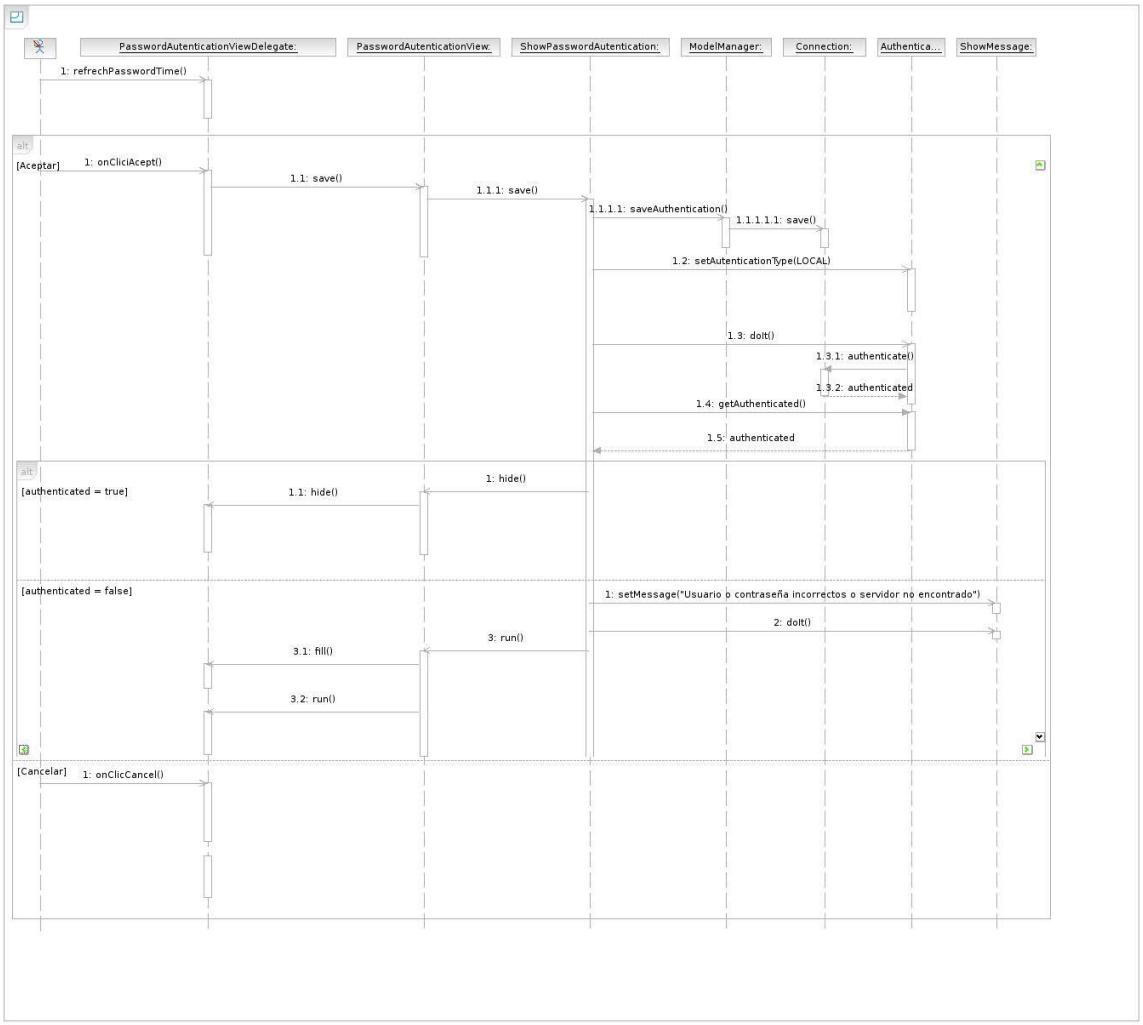


Figura 16: Diagrama de secuencia Caso de Uso Autenticación por contraseña

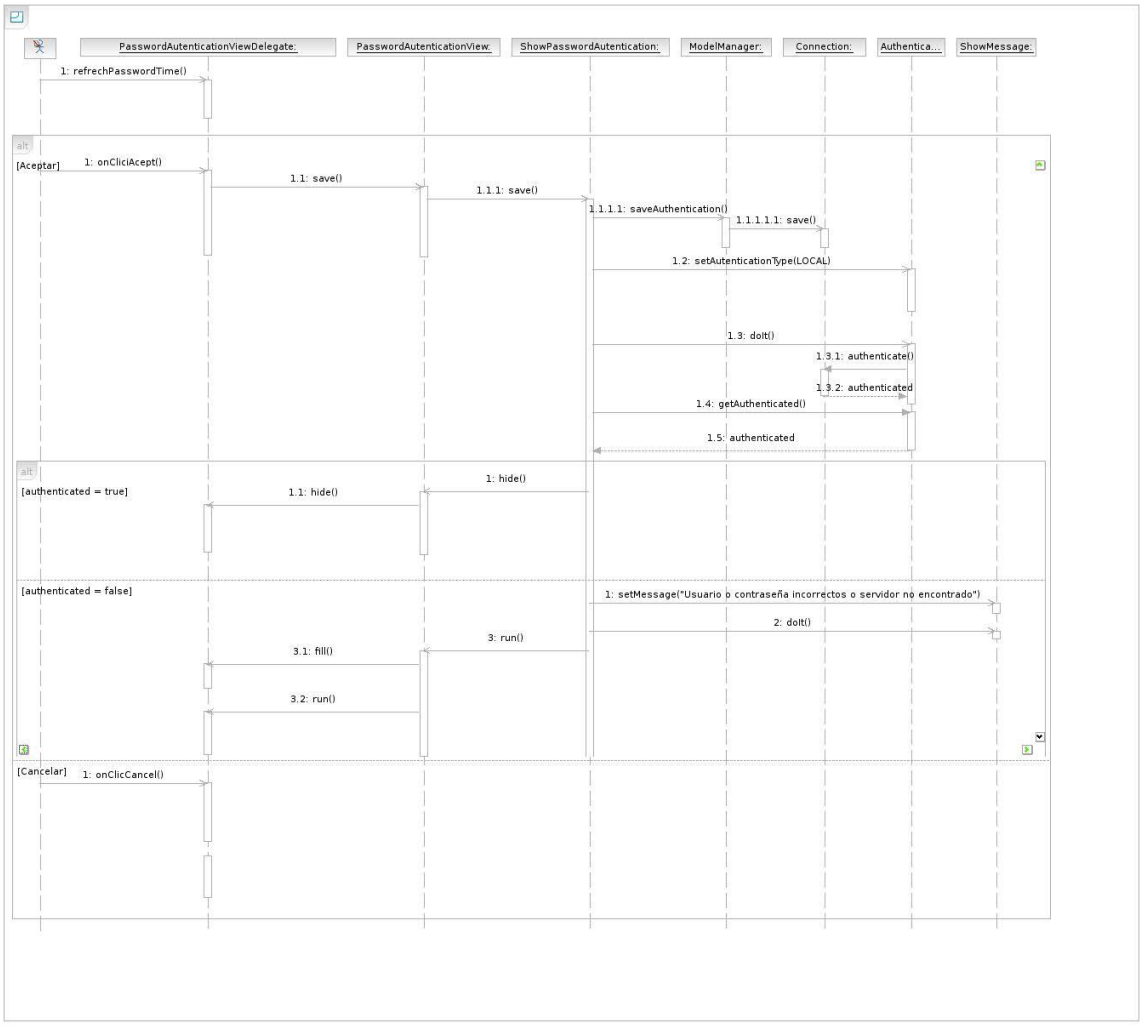


Figura 17: Diagrama de secuencia Caso de Uso Autenticación biométrica

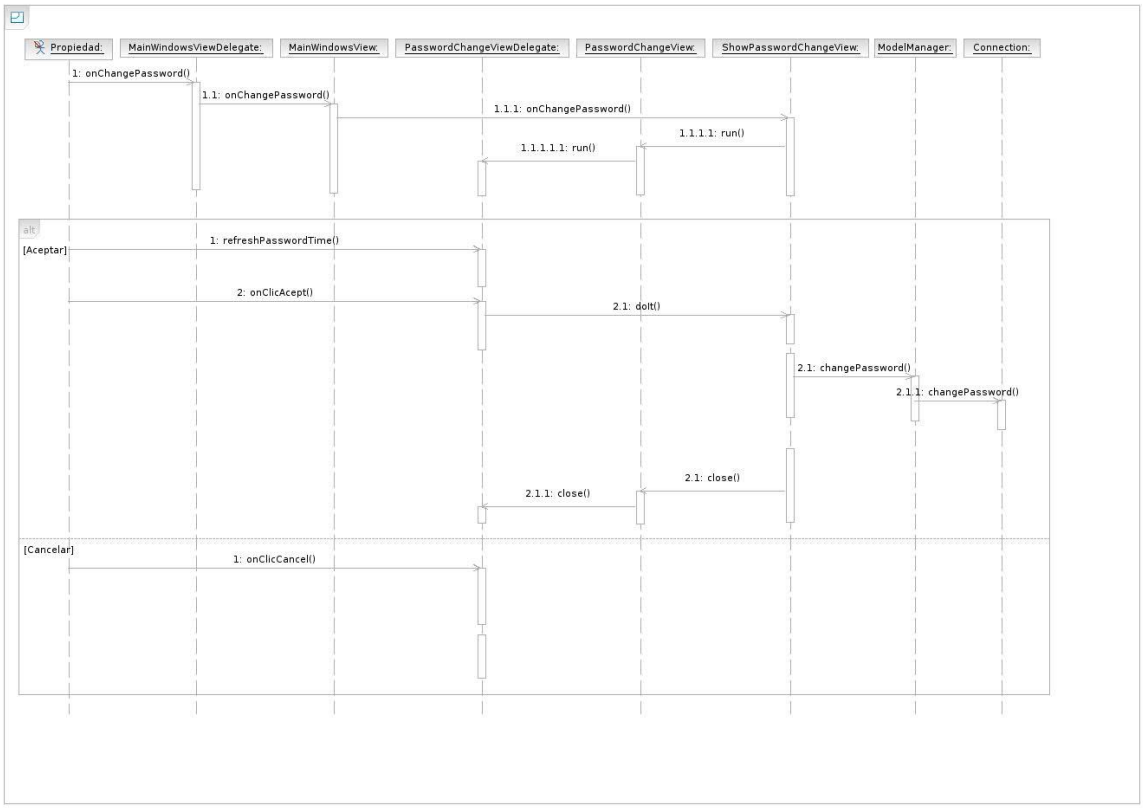


Figura 18: Diagrama de secuencia Caso de Uso Cambiar contraseña

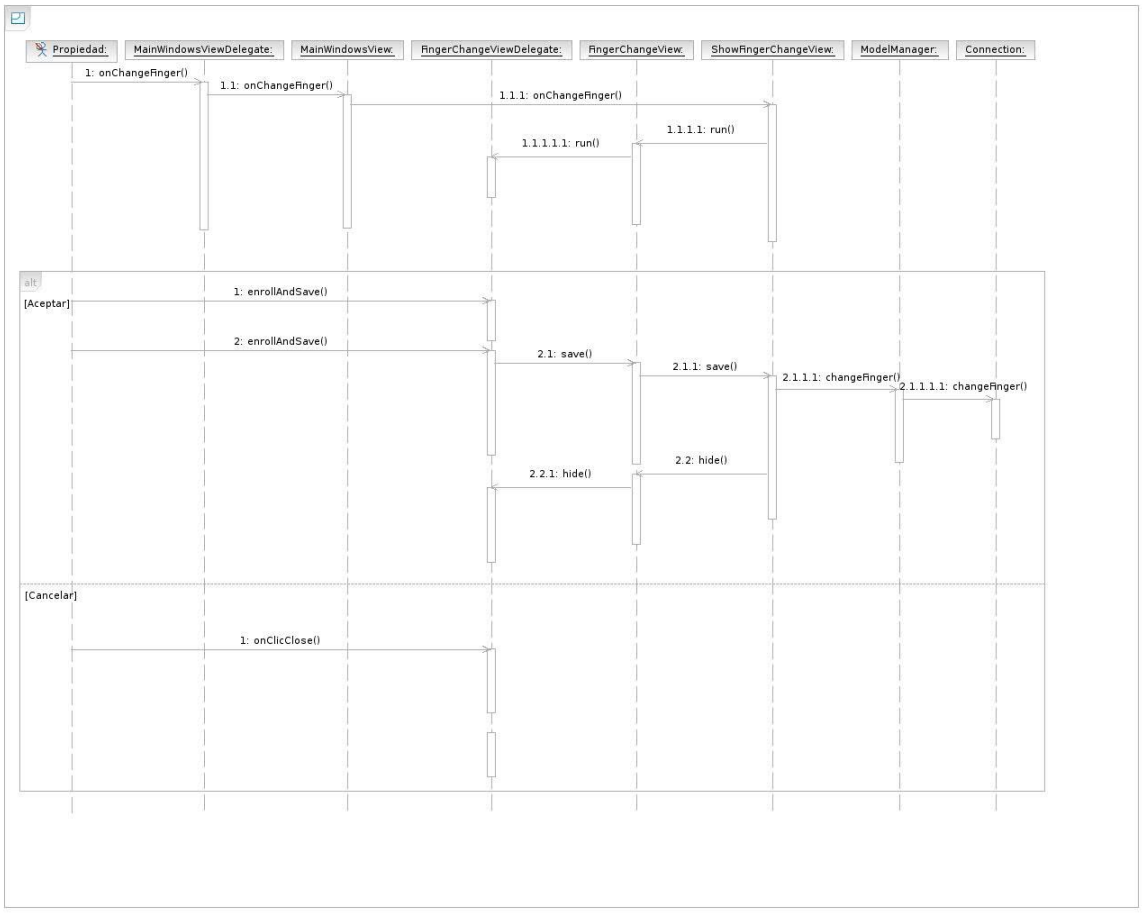


Figura 19: Diagrama de secuencia Caso de Uso Cambiar parámetro biométrico

**Patrón arquitectónico Modelo-Vista-Controlador**

El patrón Modelo-Vista-Controlador es un patrón arquitectónico cuyo objetivo es separar los datos de la aplicación, su interfaz de usuario y la lógica en distintos componentes.

- Modelo (Model): Encapsula los datos y las funcionalidades. El modelo es independiente de cualquier representación de salida y/o comportamiento de entrada.
- Vista (View): Muestra la información al usuario. Pueden existir múltiples vistas del modelo. Cada vista tiene asociado un componente controlador.
- Controlador (Controler): Reciben las entradas, usualmente como eventos que codifican los movimientos o pulsación de botones del ratón, pulsaciones de teclas, etc. Los eventos son traducidos a solicitudes de servicio (“service

requests") para el modelo o la vista.

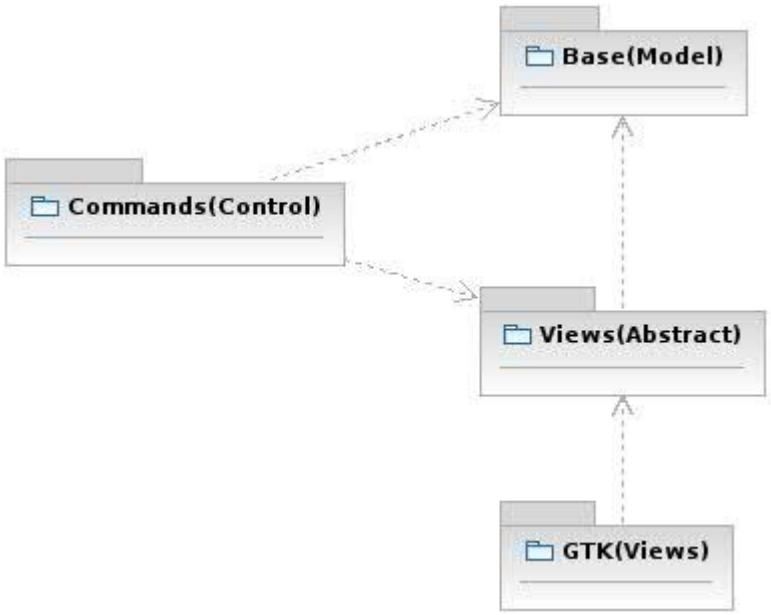


Figura 20: Patrón Modelo-Vista-Controlador

**Patrones de diseño**

Los patrones de diseño tienen diferentes clasificaciones en dependencia de su funcionalidad.

- Patrones de creación: Muestran la guía de cómo crear objetos cuando sus creaciones requieren tomar decisiones. Estas decisiones normalmente serán resueltas dinámicamente decidiendo que clases instanciar o sobre que objetos u objeto 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.

Dentro de los patrones de creación se tuvo en cuenta el Abstract Factory (Fábrica Abstracta) y el Singleton y con el objetivo de facilitar el buen funcionamiento de la aplicación se implementó el patrón de comportamiento Command.

El patrón Abstract Factory es el encargado de crear diferentes familias de objetos, en este caso, la creación de interfaces gráficas. Mediante este patrón se pueden crear los diferentes tipos de objetos para Gtk que es el lenguaje de interfaz gráfica seleccionado y si se necesita emigrar hacia otra biblioteca cambiarían el mínimo de clases posibles. Este patrón contribuye a un mejor funcionamiento para la arquitectura de la aplicación.

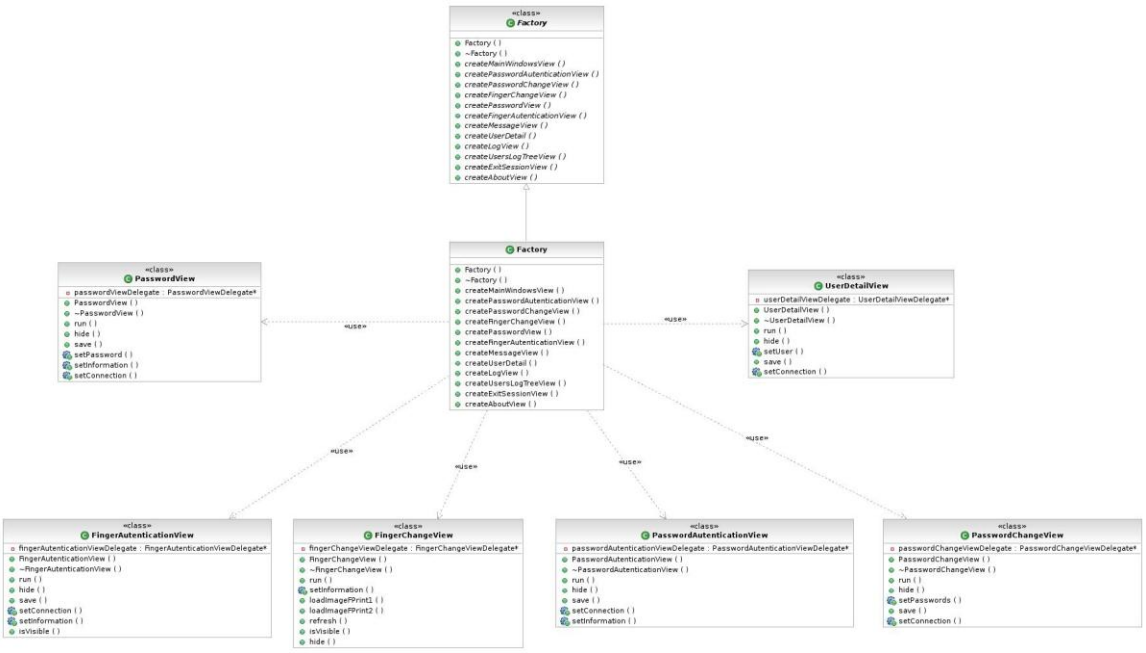


Figura 21: Patrón Fábrica Abstracta

El patrón Singleton restringe la instanciación de una clase o valor de un tipo a un solo objeto. Durante el desarrollo de la aplicación era necesario acceder de forma segura al modelo desde varias clases. Especializar dicha clase fue la solución a dicho problema y de esa forma se aseguraría que solo existiera una sola instancia de dicho modelo.

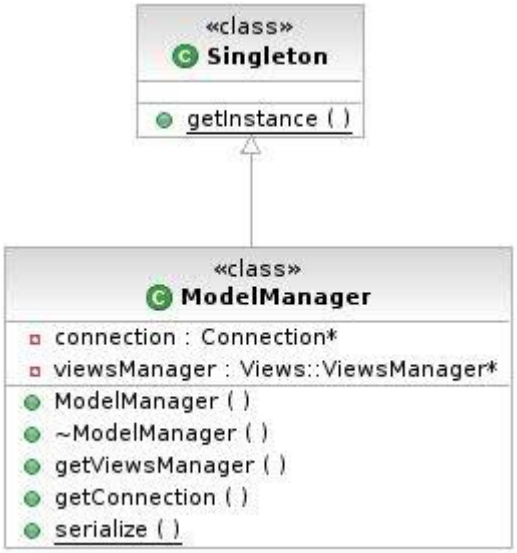


Figura 22: Patrón Singleton

El patrón Command encapsula una petición en un objeto. En la aplicación fue necesario realizar la misma operación en varias ocasiones bajo circunstancias similares. Dicho patrón fue la solución al problema antes mencionado y de esta forma se evita la reimplementación y se garantiza un fácil mantenimiento futuro.

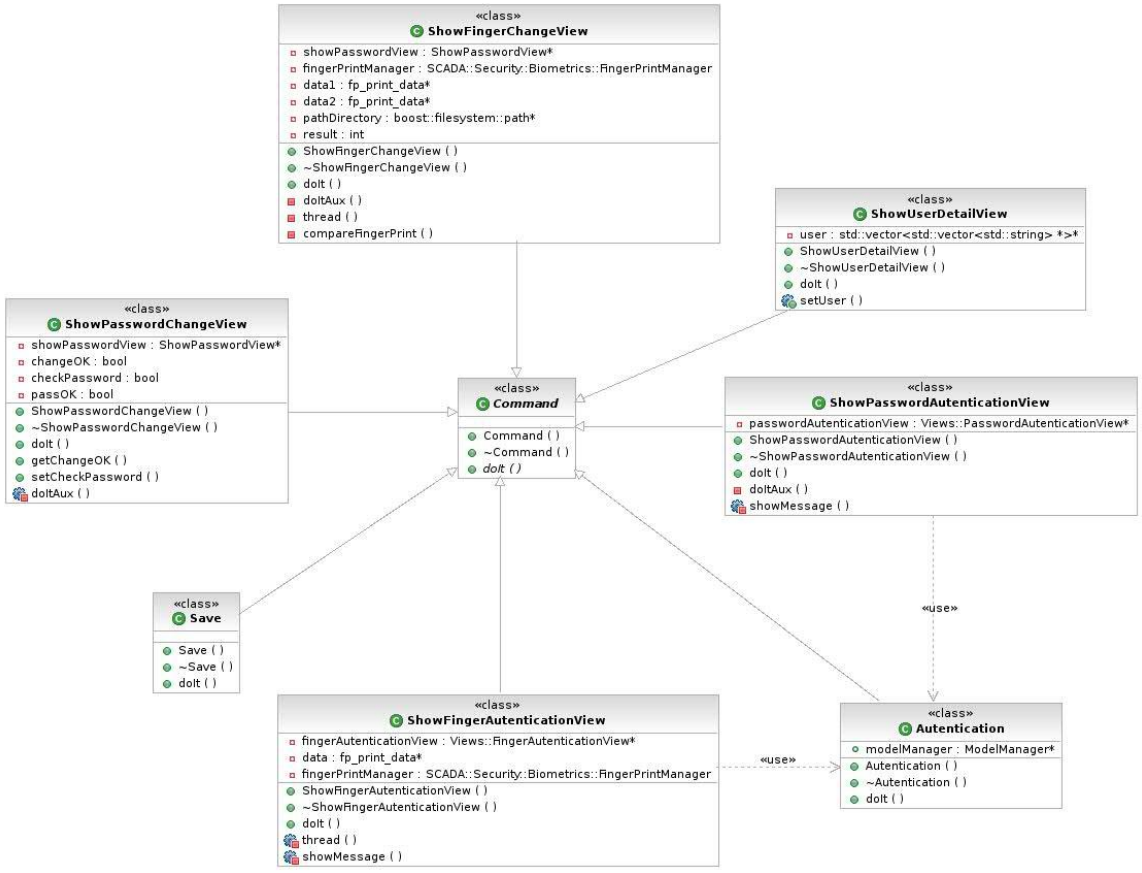


Figura 23: Patrón Command

# Capítulo 4: Desarrollo y pruebas

## Introducción

En este capítulo se explicará el estándar de codificación y documentación utilizado para el desarrollo del código de la aplicación. También se explicara mediante un diagrama de despliegue la distribución física del hardware utilizado y diferentes pruebas de sistema realizadas al software.

## Estándar de codificación

- Nombres
  - Los nombres de las clases son sustantivos singulares.
  - Los nombres deben reflejar el que y no el cómo.
  - Los nombres no deben revelar detalles de implantación.
  - Escoger nombres lo suficientemente largo para ser expresivos, pero evitando manejar nombres que dificulten la labor de implantación.
  - Evitar nombre que permitan una interpretación subjetiva (evitar ambigüedad y asegurar abstracción).
  - Evitar redundancia no repitiendo nombre de clases en sus elementos.
  - Concatenar calificadores de cómputo a las variables que almacenen el producto de tal cómputo (avg, sum, min, max, index).
  - Dado que los nombres generalmente son el producto de concatenar varias palabras, se debe emplear mayúscula para el inicio de cada palabra y minúscula para el resto de las letras para el caso de los nombres de métodos y funciones. Para el caso de los nombres de variables y atributos debe aplicarse la misma convención, con excepción de la primera letra del nombre, la cual debe ser en minúscula.
  - Variables booleanas deben contener is en su nombre.
  - Los nombres de constantes deben contener solo letras mayúsculas.
  - Minimizar el uso de abreviaciones. En caso de ser requeridas, se debe ser consistente en su uso y cada abreviación debe significar solo una cosa. En general agregar a la documentación las abreviaturas.
  - Los nombres de los métodos son frases que incluyen verbos.

- Los nombres de los atributos y parámetros son frases con sustantivos.
- Evitar el reuso de nombres para distintos propósitos.

➢ Manejo de Errores

- Se pueden manejar los errores mediante mecanismos de excepciones o mediante valores de retorno, aunque esto debe ser uniforme dentro de un mismo objeto.
- Es buena práctica emplear herramientas para identificar errores en la codificación en caliente.

➢ Codificación

- Se establece un tamaño de identación estándar de tres (3) 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.
- Minimizar el uso de conversiones de tipo forzadas (castings), cuando se requiera su uso, debe ser comentada la justificación
- Emplear select-case o switch en sustitución de if anidados sobre la misma variables.
- Liberar apuntadores de manera explícita.
- Emplear i, j, k, l, p, q, r para contadores en ciclos.
- Comentar siempre las llaves que cierran.
- Emplear al máximo operadores del tipo: +=, *=, /=, -=, ++, --, etc.

- Mantener la modularidad del código bajo el criterio de la lógica que encierra, no exagerar la modularidad.
- 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.
- Inicializar todas las variables.
- Emplear líneas en blanco para separar pasos lógicos (declaraciones, lazos, etc.).
- Siempre asignar NULL a los apuntadores luego de ser destruidos (solo aplica para C).
- Evitar prácticas que incrementan explosivamente la complejidad, como lo son: objetos y variables globales y altos tipo goto.

**Estándar de documentación**

Se utilizará la herramienta de autodocumentación Doxygen para generar la documentación del código de los distintos módulos involucrados en el proyecto y los formatos brindados por esta herramienta los cuales se explican a continuación.

- Estilo de bloques de documentación JavaDoc.

Se adopta el estilo de bloques de documentación JavaDoc, el cual consiste de un bloque de comentario de estilo C, este bloque de comentario comienza con dos asteriscos de esta manera:

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

En este caso los asteriscos que se encuentran en la línea de la mitad son opcionales, por lo que también es válido que el bloque sea de esta manera:

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

- Descripción Breve con comando \brief ó @brief.

Para hacer una descripción breve se adopta el uso del comando \brief ó @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.

Aquí tenemos un 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 tomará la descripción hecha en el bloque como una descripción detallada y no habrá descripción breve.

- Descripción de Argumentos y Métodos.

A la hora de hacer una descripción de los argumentos de métodos y funciones esta se hará 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 */ );
```

o

```
/**
 * @brief Breve descripción del método
 * @param c Descripción del parámetro c
 * @param d Descripción del parámetro d
 * @param e Descripción del parámetro e
 * @return Retorno del método
 */

int multFunction ( int c , int d , int e );
```

- Documentación de tipos de datos.

Para la documentación de tipos de datos definidos tales como enumerados, uniones o typedefs se pueden utilizar los formatos especificados en los apartados anteriores para describir su función y los elementos que los componen, aquí tenemos 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 */
                STRING /**<Puede ser un valor de tipo string */ };
```

Para este ejemplo Doxygen generará la siguiente salida de documentación:

Error: Macro Image (Imagen1.jpg, 100%) failed
sequence item 0: expected string, NoneType found

- Formato Doxygen

El formato a utilizar a la hora de documentar el código hará que doxygen genere una salida de documentación como se muestra en el siguiente ejemplo:

```
/**
 * @brief Breve descripción de esta función
 *
 * Aquí comienza la descripción detallada de esta función,
 * también se muestra una explicación del valor de retorno de la
 * función si es el caso para esto se utiliza el comando \@return
 * @return Devuelve un valor entero resultado de la operación
 */
int multFunction ( int c , /**<Primer operando de la operación */
                   int d , /**<Segundo operando de la operación */
                   int e /**<Tercer operando de la operación */ );
```

Error: Macro Image (Imagen2.jpg, 100%) failed
sequence item 0: expected string, NoneType found

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.

- Comando @code y @endcode

En algunos casos es necesario mostrar en la documentación una parte del código cuando se hacen comentarios entre líneas o para mostrar algún método en particular, ya que esto puede ahorrar hacer una explicación más extensa; para realizar esto en doxygen se utiliza el comando @code y finaliza la acción del mismo con el comando @endcode, se muestra esto en el siguiente ejemplo:

```
/**
 * @brief Devuelve el contador
 *
 * Se detalla la utilidad de ésta función, el código que se muestra a continuación se incluye haciendo
 * uso del comando \@code y finalizando el bloque de código con el comando \@endcode, de esta
 * manera:
 * @code
 * int getCount()
 * {
 *                  return count;
 * }
 * @endcode
 */
int getCount()
{
    return count;
}
```

Doxygen genera la siguiente salida:

Error: Macro Image(Imagen3.jpg, 100%) failed
sequence item 0: expected string, NoneType found

- 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 Edgar Acosta dragoicaru@hotmail.com
 * @date 08-08-2007
 */
void voidFunction ();
```

Para este caso Doxygen genera lo siguiente:

Error: Macro Image(Imagen5.jpg, 100%) failed
sequence item 0: expected string, NoneType found

- 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()
*/
Test3();
```

Esto generará en la documentación para el constructor de la clase de prueba Test3 una referencia hacia la documentación existente para el constructor de la clase de prueba Test, la salida de Doxygen es la siguiente:

Error: Macro Image(Imagen6.jpg, 100%) failed
sequence item 0: expected string, NoneType found

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

```
/**
* Constructor de la clase
* @see Clase
*/
```

Test3();

La salida generada por Doxygen sería algo como esto:

Error: Macro Image(Imagen7.jpg, 100%) failed
sequence item 0: expected string, NoneType found

**Diagrama de despliegue**

El Diagrama de Despliegue es un tipo de diagrama utilizado para representar el hardware utilizado en la implementación del sistema y sus relaciones.

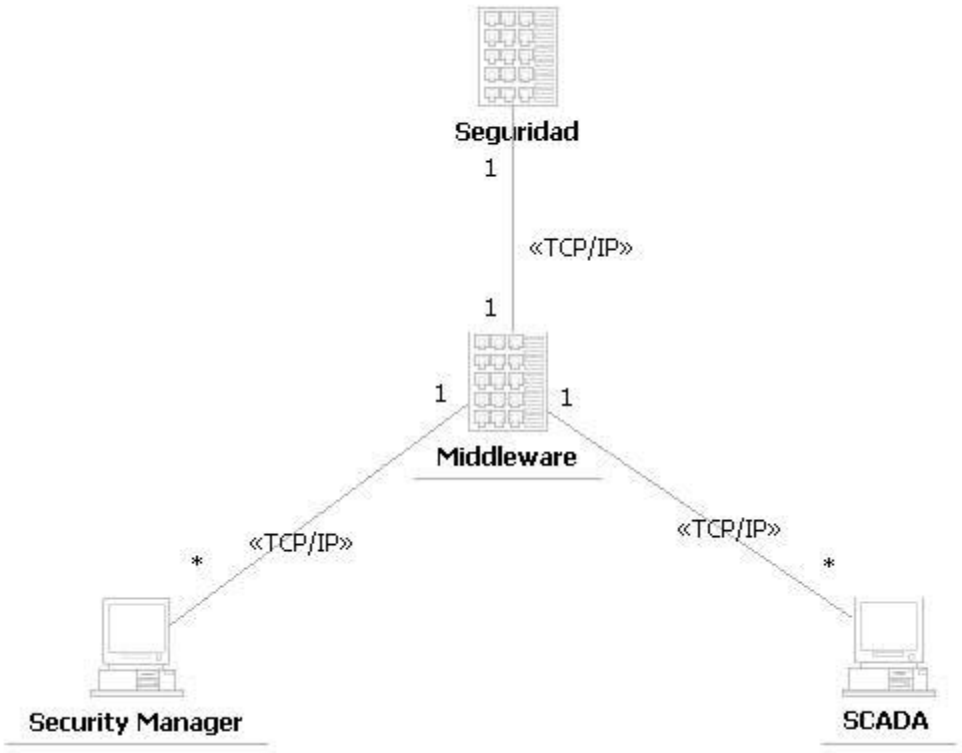


Figura 24: Diagrama de despliegue

**Pruebas**

Las pruebas es una actividad en la cual un sistema o componente es ejecutado bajo unas condiciones o requerimientos específicos, los resultados son observados y registrados, y una evaluación es hecha de algún aspecto del sistema o componente.

Los diferentes niveles de pruebas son:

- Nivel de unidad: Está enfocada al código fuente de los componentes y verifica todos los flujos de control. Este tipo de prueba pasa primero por la revisión del programador.
- Nivel de integración: Prueba los componentes combinados para ejecutar u casos de uso. Tiene como objetivo descubrir errores en las especificaciones de las interfaces de las clases.
- Nivel de sistema: Prueba el software funcionando como un todo. Aceptable cuando el software se encuentra en fase de construcción.
- Nivel de aceptación: Es la prueba final antes del despliegue del sistema. Generalmente las realizan los usuarios finales y es llamada prueba piloto.

A continuación se muestra el desarrollo de algunas pruebas de sistema que fueron realizadas por el cliente.

Caso de Uso Autenticación por contraseña.

Descripción general.

Este caso de uso permite al usuario autenticarse con el sistema a partir de su usuario y su contraseña. El usuario también tiene la posibilidad de insertar el IP y el puerto donde se encuentra ubicado el servidor de Seguridad.

Caso de prueba #1: Autenticación por contraseña

1. Descripción

   Este caso de prueba le permite al usuario autenticarse con el sistema.

2. Flujo central
   2.1. El usuario inserta su usuario y su contraseña. También tiene la posibilidad de actualizar el IP y puerto del servidor de Seguridad.
   2.2. El usuario selecciona la opción Aceptar.
   2.3. El sistema realiza la validación de los datos insertados y lleva a cabo la autenticación.

3. Iteraciones

| Clases Válidas | Clases Inválidas | Resultado Esperado | Resultado de la Prueba | Observaciones |
|---|---|---|---|---|
| Introduce todos los campos correctamente, Ejemplos usuario: ereyes contraseña: t02s0m0sqbanos IP: 192.168.2.20 puerto: 9992 | | El sistema realiza la validación de os datos correctamente y lleva a cabo la autenticación. | | |
| | El usuario deja campos vacios Ejemplo: usuario: ereyes contraseña: | El sistema muestra un mensaje de error alertando que existen campos vacios y enfoca el campo sin información. | | |
| | El usuario introduce valores incorrectos. Ejemplo: usuario: ereyes contraseña: t02s0m0sqbanos IP: 192.168.2.20 puerto: kike | El sistema muestra un mensaje de error informando al usuario que existen errores en los datos introducidos y enfoca el campo con errores. | | |
| | El usuario introduce usuario o contraseña no válidas Ejemplo: usuario: ereyes contraseña: t02s0m0sqbanos | El sistema muestra un mensaje de error informando al usuario que su usuario o su contraseña no son válidos. | | |

| | | | | |
|---|---|---|---|---|
| | El usuario introduce Ip o puertos no válidas Ejemplo: IP: 192.168.2.20 puerto: 9993 | El sistema muestra un mensaje de error informando al usuario que no se encontró el destino del servidor de Seguridad. | | |

Caso de Uso Autenticación biométrica.

Descripción general.

Este caso de uso permite al usuario autenticarse con el sistema a partir de su huella dactilar. El usuario también tiene la posibilidad de insertar el IP y el puerto donde se encuentra ubicado el servidor de Seguridad.

Caso de prueba #2: Autenticación biométrica.

1. Descripción

   Este caso de prueba le permite al usuario autenticarse con el sistema.

2. Flujo central.
   2.1. El usuario pasa su dedo por el lector de huellas dactilares. También tiene la posibilidad de actualizar el IP y puerto del servidor de Seguridad.
   2.2. El sistema realiza la validación de los datos insertados y lleva a cabo la autenticación.
3. Iteraciones

| Clases Válidas | Clases Inválidas | Resultado Esperado | Resultado de la Prueba | Observaciones |
|---|---|---|---|---|
| Introduce todos los campos correctamente y pasa su dedo por el lector. Ejemplos IP: 192.168.2.20 | | El sistema realiza la validación de los datos correctamente y lleva a cabo la autenticación. | | |

| | | | | |
|---|---|---|---|---|
| puerto: 9992 | | | | |
| | El usuario pasa una parte de su dedo que no contiene su huella dactilar. | El sistema muestra un mensaje de error alertando al usuario que hubo un error durante la captura de la huella. | | |
| | El usuario introduce valores incorrectos.<br>Ejemplo:<br>IP: 192.168.2.20<br>puerto: kike | El sistema muestra un mensaje de error informando al usuario que existen errores en los datos introducidos y enfoca el campo con errores. | | |
| | El usuario pasa su dedo por el lector sin existir en la base de datos. | El sistema muestra un mensaje de error informando al usuario que la huella capturada no existe en la base de datos. | | |
| | El usuario introduce Ip o puertos no válidas<br>Ejemplo:<br>IP: 192.168.2.20<br>puerto: 9993 | El sistema muestra un mensaje de error informando al usuario que no se encontró el destino del servidor de Seguridad. | | |

Caso de Uso Cambiar contraseña.

Descripción general.

Este caso de uso permite al usuario cambiar su contraseña.

Caso de prueba #3: Cambiar contraseña

1. Descripción

Este caso de prueba le permite al usuario cambiar su contraseña.

2. Flujo central.
   2.1. El usuario selecciona cambiar contraseña.
   2.2. El sistema solicita la contraseña actual al usuario.
   2.3. El usuario teclea su contraseña.
   2.4. El sistema muestra la ventana para que introduzca su nueva contraseña así como su confirmación.
   2.5. El sistema realiza el cambio de la contraseña.
3. Iteraciones

| Clases Válidas | Clases Inválidas | Resultado Esperado | Resultado de la Prueba | Observaciones |
|---|---|---|---|---|
| El usuario introduce todos los campos correctamente. Ejemplos contraseña actual: t02s0m0sqbanos nueva contraseña: 4ut0m4t1c4 confirmación de la contraseña: 4ut0m4t1c4 | | El sistema realiza la actualización de la contraseña. | | |
| | El usuario introduce una contraseña actual no válida. Ejemplos contraseña actual: t02s0m0sqb; | El sistema muestra un mensaje de error alertando al usuario que la contraseña actual no es válida. | | |
| | El usuario introduce una nueva contraseña y su confirmación diferentes. Ejemplos nueva contraseña: 4ut0m4t1c4 confirmación de la contraseña: | El sistema muestra un mensaje de error informando al usuario que la nueva contraseña y su confirmación no coinciden. | | |

| | | | | |
|---|---|---|---|---|
| | 4ut0m4t1c4; | | | |

Caso de Uso Cambiar parámetro biométrico

Descripción general.

Este caso de uso permite al usuario cambiar su huella dactilar.

Caso de prueba #4: Cambiar parámetro biométrico

1. Descripción

Este caso de prueba le permite al usuario cambiar su huella dactilar.

2. Flujo central.
   2.1. El usuario selecciona cambiar huella.
   2.2. El sistema solicita la contraseña actual al usuario.
   2.3. El usuario teclea su contraseña.
   2.4. El sistema muestra la ventana para que pase su dedo por el lector 2 veces para su comprobación.
   2.5. El sistema realiza el cambio de la huella dactilar.
3. Iteraciones

| Clases Válidas | Clases Inválidas | Resultado Esperado | Resultado de la Prueba | Observaciones |
|---|---|---|---|---|
| El usuario introduce todos los campos correctamente. Ejemplos contraseña actual: t02s0m0sqbanos Pasa su dedo correctamente. | | El sistema realiza la actualización de la huella dactilar. | | |

| | | | | |
|---|---|---|---|---|
| | El usuario introduce una contraseña actual no válida. Ejemplos contraseña actual: t02s0m0sqb; | El sistema muestra un mensaje de error alertando al usuario que la contraseña actual no es válida. | | |
| | El usuario no pasa el mismo dedo por el lector durante la rectificación de la huella dactilar. | El sistema muestra un mensaje de error informando al usuario que la nueva huella y su confirmación no coinciden. | | |

Caso de Uso Cerrar sesión.

Descripción general.

Este caso de uso permite al usuario cerrar la aplicación.

Caso de prueba #5: Cerrar sesión

1. Descripción

   Este caso de prueba le permite al usuario cerrar la aplicación.

2. Flujo central.
   2.1. El usuario selecciona salir.
   2.2. El sistema muestra la confirmación si desea realmente salir.
   2.3. El usuario selecciona la opción Aceptar.
   2.4. El sistema realiza el cierre de sesión y cierra la aplicación.
3. Iteraciones

| Clases Válidas | Clases Inválidas | Resultado Esperado | Resultado de la Prueba | Observaciones |
|---|---|---|---|---|
| El usuario selecciona salir de | | El sistema realiza el cierre de sesión | | |

| | | | | |
|---|---|---|---|---|
| la aplicación. | | correctamente. | | |
| | El usuario selecciona salir de la aplicación y cuando el sistema desea verificar si realmente quiere cerrar la aplicación selecciona la opción Cancelar. | El sistema regresa a su estado anterior y no realiza el cierre de sesión. | | |

## Conclusiones generales

A partir de la investigación realizada para la elaboración de este sistema utilizando RUP como metodología idónea para lograr una mejor comprensión de los requisitos de la aplicación y formalización de los mismos, se arriba a las siguientes conclusiones:

- El sistema desarrollado incluye, la administración del acceso de los usuarios al sistema por autenticación (usuario y contraseña) o biométrica, garantizando el decremento del nivel de vulnerabilidad que hoy presenta el Guardián del Alba.
- La utilización de este sistema contribuye al fortalecimiento del módulo Seguridad evitando el acceso no autorizado de usuarios al sistema.

Con la propuesta y el estudio realizado se materializan los objetivos planteados al inicio de esta investigación: Desarrollar un sistema de autenticación para el módulo Seguridad del proyecto Guardián del Alba, disminuyendo las vulnerabilidades existentes en el sistema.

## Recomendaciones

Se recomienda:

- Continuar con el desarrollo de este trabajo buscando nuevas herramientas con el objetivo de construir un sistema de autenticación más potente.
- Continuar la investigación de nuevos drivers para el desarrollo de la biblioteca FPrint con el objetivo de ampliar la compatibilidad de otros dispositivos lectores de huellas dactilares.
- Ampliar los métodos de autenticación mediante otros parámetros biométricos como por ejemplo (iris del ojo).
- Utilizar el sistema desarrollado en otros proyectos con el objetivo de administrar el acceso al sistema a través de los diferentes tipos de autenticación.

## Referencias bibliográficas


1. *Autómatas programables.* **Balcells, Josep y Romeral, Luis. 1997.** s.l. : Marcombo, 1997.
2. *SISTEMAS SCADA.* **Mendiburu Diaz, Henry.**
3. **Villalón Huerta, Antonio. 2002.** *SEGURIDAD EN UNIX Y REDES.* 2002.
4. *Sugerencias para mejorar la seguridad en SCADA.* **Liberal, Daniel. 2007.** 2007.
5. **Movicon. 2008.** Progea. [En línea] 2008. [Citado el: 10 de marzo de 2009.] http://www.progea.com/software-automation-scada/movicon-11/movicon-11-xml-based.html.
6. *SIMATIC WinCC Versión 6.2.* **Siemens. 2007.** 2007.
7. **Prieto, José-Luis. 1984-2008.** Glosario de Terminología Informática. [En línea] 1984-2008. [Citado el: 2009 de marzo de 10.]
8. **2009.** ¿Cuáles son las contraseñas? [En línea] 2009. [Citado el: 10 de marzo de 2009.] http://es.tech-faq.com/passwords.shtml.
9. GLASARIO DE TERMINOS. [En línea] http://www.maersa.com.mx/glosario.html.
10. *Generadores de Interfaces de Usuario: QT Designer, NetBeans y Windows Forms Designer.* **Alvarez, Alejandro y Valerio, Esteban.**
11. **Gale, Tony y Main, Ian. 1999.** GTK Tutorial v1.2. [En línea] 21 de febrero de 1999. [Citado el: 21 de marzo de 2009.] http://www.linuxlots.com/~barreiro/spanish/gtk/gtk_tut_12/gtk_tut_12.es-1.html.
12. *PyQT Desarrollando Aplicaciones de Escritorio.* **Maldonado, Daniel.**
13. Tecnología: La reutilización de código. [En línea] [Citado el: 21 de marzo de 2009.] http://zintegra.blogspot.com/2007/07/la-reutilizacin-de-cdigo.html.
14. **Microsoft.** BASE DE DATOS. [En línea] [Citado el: 21 de marzo de 2009.] http://gloriaisabelparra.spaces.live.com/blog/cns!409FD0578C87C97D!153.entry.
15. **M, Gabriel. 2008.** Sistema de Gestión de Base de Datos. [En línea] 8 de julio de 2008. [Citado el: 21 de marzo de 2009.] http://unecomputacion.wordpress.com/2008/07/.

16. Entornos de desarrollo Integrado. [En línea] http://petra.euitio.uniovi.es/~i1667065/HD/documentos/Entornos%20de%20Desarrollo%20Integrado.pdf.
17. **NetBeans.** NetBeans. [En línea] [Citado el: 24 de marzo de 2009.] http://www.netbeans.org/index.html.
18. **2002-2005.** Más alla de Linux. [En línea] 2002-2005. [Citado el: 26 de marzo de 2009.] http://www.escomposlinux.org/lfs-es/blfs-es-6.0/general/doxygen.html.
19. Lenguaje de Programación. [En línea] [Citado el: 27 de marzo de 2009.] http://www.scribd.com/doc/13719562/Lenguaje-de-Programacion.
20. **Ruiz Muñzquiz, Pablo. 2003.** *Sistemas operativos.* 2003.
21. **Larman, Craig. 2004.** *UML y Patrones.* La Habana : Félix Varela, 2004.
22. Modelado UML. [En línea] [Citado el: 25 de mayo de 2009.] https://forja.rediris.es/docman/view.php/282/444/uml20.pdf.
23. *Lecturas sobre computadoras digitales.* **2007.** 2007.
24. Requerimientos de un sistema de información. [En línea] [Citado el: 20 de mayo de 2009.] http://speedvirtualextr.googlepages.com/RequerimientosdeunSistemadeInformaci.pdf.
25. **2008.** Técnicas y materiales. [En línea] 7 de diciembre de 2008. [Citado el: 20 de mayo de 2009.] http://jessicahuelva.wordpress.com/2008/12/07/tema-3/.
26. **2000-2009.** Infosec. [En línea] 2000-2009. [Citado el: 22 de mayo de 2009.] http://www.infosecwriters.com/texts.php?op=display&id=521.
27. **S.Pressman, Roger. 2005.** *Ingeniería del Software.* La Habana : Félix Varela, 2005.

## Bibliografía

*Boost*. (2004-2007). Retrieved abril 15, 2009, from http://www.boost.org/

*C++ Library Reference*. (200-2009). Retrieved abril 10, 2009, from http://www.cplusplus.com/reference/

*Eclipse*. (2009). Retrieved abril 12, 2009, from http://www.eclipse.org/

*FPrint*. (2008, noviembre 23). Retrieved from http://reactivated.net/fprint/wiki/Main_Page

*Glade - a User Interface Designer for GTK+ and GNOME*. (n.d.). Retrieved marzo 25, 2009, from http://glade.gnome.org/

Grady, & Rumbaugh. (2004). *El Proceso Unificado de Desarrollo de Software.* La Habana.

*GTK*. (2007-2008). Retrieved abril 10, 2009, from http://www.gtk.org/

*gtkmm 2.4 documentation*. (n.d.). Retrieved abril 10, 2009, from http://www.gtkmm.org/docs/gtkmm-2.4/docs/

Larman. (2004). *UML y Patrones.* La Habana.

*Manual de Doxygen.*

Pressman. (2005). *Ingeniería de Software.* La Habana.

# Anexos

| | Ojo-Iris | Ojo-Retina | Huellas dactilares | Geometría de la mano | Escritura-Firma | Voz |
|---|---|---|---|---|---|---|
| Fiabilidad | Muy alta | Muy alta | Alta | Alta | Alta | Alta |
| Facilidad de uso | Media | Baja | Alta | Alta | Alta | Alta |
| Prevención de ataques | Muy alta | Muy alta | Alta | Alta | Media | Media |
| Aceptación | Media | Media | Media | Alta | Muy alta | Alta |
| Estabilidad | Alta | Alta | Alta | Media | Media | Media |
| Identificación y autenticación | Ambas | Ambas | Ambas | Autenticación | Ambas | Autenticación |
| Estándares | - | - | FBI | - | - | SVAPI |
| Interferencias | Gafas | Irritaciones | Suciedad, heridas | Artritis, reumatismo | Firmas faciales | Ruido, resfriados |

Anexo 1: Comparación de los métodos biométricos

## Glosario de términos

**ALBA:** La Alternativa Bolivariana para los Pueblos de Nuestra América o ALBA es una propuesta de integración enfocada para los países de América Latina y el Caribe que pone énfasis en la lucha contra la pobreza y la exclusión social con base en doctrinas de izquierda.

**BDTR:** Base de Datos en tiempo Real, aplicación usada en el SCADA Nacional Guardián del ALBA.

**GNOME:** GNU Network Object Model Environment es un entorno de escritorio basado en las librerías GTK diseñadas para GIMP para sistemas operativos de tipo Unix bajo tecnología X Window. Al igual que KDE, permite la interacción entre programas. Se encuentra disponible actualmente en más de 35 idiomas. Forma parte oficial del proyecto GNU.

**GUI (Graphical User Interfaces):** Componente de una aplicación informática que el usuario visualiza y a través de la cual opera con ella. Está formada por ventanas, botones, menús e iconos, entre otros elementos.

**HMI:** Interfaz Hombre-Máquina (Human Machine Interface).

**IDS:** Intruder Detection System, sistema encargado de detectar ataques o violaciones.

**LDAP:** Protocolo Ligero de Acceso a Directorios (Lightweight Directory Access Protocol), es un protocolo a nivel de aplicación que permite el acceso a un servicio de directorio ordenado y distribuido para buscar diversa información en un entorno de red. LDAP también es considerado una base de datos (aunque su sistema de almacenamiento puede ser diferente) a la que pueden realizarse consultas.

**Licencia BSD:** es la licencia de software otorgada principalmente para los sistemas BSD (Berkeley Software Distribution).

**Logs:** Registros del sistema, se refieren a acciones, comportamientos realizados por los sistemas o sobre ellos

**Middleware:** Subsistema de Comunicación del SCADA Nacional Guardián del ALBA

**Multiplataforma:** Es un término utilizado frecuentemente en informática para indicar la capacidad o características de poder funcionar o mantener una interoperabilidad de forma similar en diferentes sistemas operativos o plataformas.

**Multiplataforma:** es un término utilizado frecuentemente en informática para indicar la capacidad o características de poder funcionar o mantener una interoperabilidad de forma similar en diferentes sistemas operativos o plataformas.

**PDVSA:** Petróleos de Venezuela, Sociedad Anónima (PDVSA) es una empresa estatal venezolana que se dedica a la explotación, producción, refinación, mercadeo y transporte del petróleo venezolano.

**Sistemas de Control y Adquisición de Datos (SCADA):** Aplicación de software especialmente diseñada para funcionar sobre computadores en el control de producción, proporcionando comunicación con los dispositivos de campo y controlando el proceso de forma automática desde la pantalla del operador, proveyendo toda la información asociada al proceso.

**XML (eXtensible Markup Language, 'lenguaje de marcas extensible'):** Metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C). Permite definir la gramática de lenguajes específicos, por lo tanto XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades. Algunos de estos lenguajes que usan XML para su definición son XHTML, SVG y MathML.