Universidad de las Ciencias Informáticas

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

Título:

Componente para la comunicación con el Sistema Multibiométrico a través de ficheros ANSI/NIST por correo electrónico

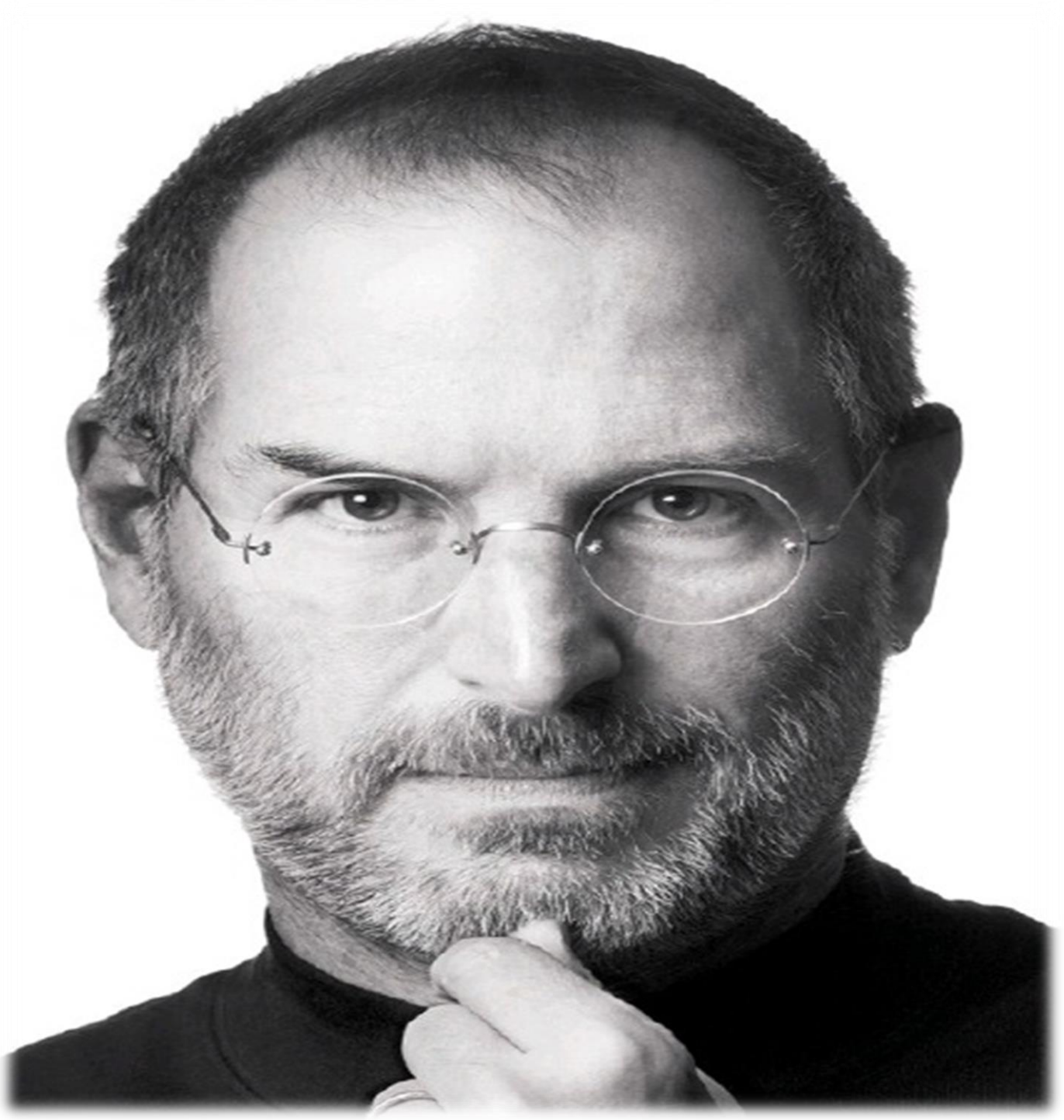
Autores:

- Alexander Lugones Aguiar
- Carlos Iván García Delgado

Tutores:

- MSc. Yeneit Delgado Kios
- Ing. Dannier Sierra Obregón

LA HABANA, JUNIO DE 2014



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

Steve Jobs

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

Autor: Alexander Lugones Aguiar

_____

Firma del Autor

MSc. Yeneit Delgado Kios

_____

Firma del Tutor

Autor: Carlos Iván García Delgado

_____

Firma del Autor

Ing. Dannier Sierra Obregón

_____

Firma del Tutor

DATOS DE CONTACTO

Tutores:

MSc. Yeneit Delgado Kios

Licenciada en Ciencia de la Computación, Universidad de La Habana, con 10 años de experiencia. Máster en Informática Aplicada, Universidad de las Ciencias Informáticas. Trabaja como Profesora Asistente en el Departamento de Programación, Facultad 1, Universidad de las Ciencias Informáticas.

Email: yeneit@uci.cu

Ing. Dannier Sierra Obregón

Ingeniero en Ciencias Informáticas con 4 años de experiencia. Actualmente se encuentra vinculado al Departamento de Componentes del Centro de Identificación y Seguridad Digital de la Universidad de las Ciencias Informáticas.

Email: dobregon@uci.cu

AGRADECIMIENTOS

A mis abuelos por ser ellos los protagonistas de que este sueño se hiciera realidad, apoyándome en todos los momentos de mi vida, gracias por esos sabios consejos, por guiarme en el camino correcto, gracias por ese amor, y dedicación infinita.

A mi tía Ana Ibis (tay), mi tío Águedo y mis primos Adrián y Adriel que estuvieron acompañándome en los buenos y malos momentos de mi carrera, a ellos el infinito agradecimiento.

A mi mamá y mi hermana que siempre pude contar con ellas en cualquier situación y supieron darme la fuerza que en ocasiones necesitaba.

A mi papá, Yuliet, mis hermanos, tío y primo Carlos, abuela Rita y abuelo Tero, por estar siempre al tanto de mis preocupaciones.

A mis tutores Dannier y Yeneit por ayudar en todo lo que hiciera falta y aportar sus granitos de arena para alcanzar este objetivo final.

A mis amigos de La Campana: Migue, Yoenys, Abdier, Luis Javier, Yurién, Rolandito, Pedro Pablo.

A mis compañeros de apartamento en la UCI: Estrada, Robert, Eduardo, Dayán, David, Ismael, el Llopiz, Ángel, Sergio, y Carlos Iván, que juntos siempre fuimos una gran familia. A las amistades del aula que tantas situaciones supimos superar.

Quiero agradecer también a todos aquellos amigos y amigas que conocí a lo largo de estos cinco años y que hicieron que la vida en la universidad fuera más agradable. A Mirita por siempre brindar su ayuda a la hora que se le pidiera. A Alicia, Daynela, Sandra, Rocío, Melissa, Yaima, Yudely, Laritza, Adrián, el Rafa, el Emito y todas las personas que me faltan por mencionar. Gracias por abrirme sus puertas y hacerme saber que siempre podré contar con ustedes.

Alexander Lugones Aguiar

A mis padres que lo han dado todo por mí y a quienes les debo lo que he logrado.

A mi familia por ser ejemplo de unidad y sin los que no hubiera podido estar aquí hoy.

A mis amigos de la UCI y del barrio, por ayudarme siempre y por todos los momentos que compartimos.

A mis compañeros de aula y mis profesores.

A mis tutores por toda la ayuda que me brindaron.

A todos los que de una forma u otra hicieron posible que esté hoy aquí.

Carlos I García Delgado

DEDICATORIA

A mi madre que siempre ha estado ahí para mí.

A mi padre que es mi ejemplo a seguir.

A toda mi familia y mis amistades, muy especial a mi abuelita María Luisa que sé estaría orgullosa.

Carlos I García Delgado

A mis abuelos que han sido ejemplo y guía en toda mi vida.

A mi familia por siempre estar al tanto de mis preocupaciones.

A mis amigos por demostrarme que siempre podré contar con ellos.

Alexander Lugones Aguiar

RESUMEN

Los Sistemas Multibiométricos permiten la identificación de personas a partir de varios rasgos físicos o conductuales. En la actualidad estos sistemas son utilizados para mejorar los niveles de seguridad en múltiples instituciones. El Centro de Identificación y Seguridad Digital (CISED) perteneciente a la Universidad de las Ciencias Informáticas (UCI) se encuentra inmerso en el desarrollo de un Sistema Multibiométrico para la identificación de personas, el cual utiliza RabbitMQ para gestionar las colas de solicitudes, este servicio se encuentra alojado en un servidor que no está disponible para su acceso desde sistemas externos a la universidad, por lo que se hace imposible la comunicación entre los mismos.

El presente trabajo investigativo permitió realizar un estudio de los principales protocolos y estándares utilizados para el intercambio de transacciones biométricas, con el objetivo de diseñar e implementar un servicio de comunicación que permita recibir y responder a solicitudes, enviadas desde un sistema externo al Sistema Multibiométrico de la UCI utilizando para esto el correo electrónico y el estándar ANSI/NIST. Se utilizó como marco de trabajo *Microsoft .NET framework*, el lenguaje de programación C# y *Visual Studio* 2012 como Entorno Integrado de Desarrollo guiado por la metodología *Extreme Programming* (Programación Extrema, XP), garantizando de esta manera el correcto desarrollo del servicio. La realización de pruebas permitió validar el funcionamiento de la solución desarrollada; obteniéndose como resultado un *software* que cumple con los requisitos especificados por el cliente.

PALABRAS CLAVE

ANSI/NIST, correo electrónico, servicio de comunicación, sistemas biométricos, sistema externo, sistemas multibiométricos.

Tabla de Contenidos

Índice de figuras

Índice de tablas

INTRODUCCIÓN

El desarrollo de las formas específicas de identificación de personas ha jugado un papel importante en la organización y regulación de la sociedad. Con el fin de demostrar su singularidad y carácter único, el hombre a lo largo de los años se ha interesado por perfeccionar las distintas maneras de reconocimiento humano.

Una de las estrategias más confiables, seguras y cómodas para la identificación de personas es la biometría; tecnología que permite el reconocimiento de personas mediante el análisis de aquellas características que cada individuo posee y que lo hacen único en comparación con los demás.

La biometría de las palabras griegas "*bios*" de vida y "*metría*" de medida analiza factores personales que pueden ser divididos en dos clases; fisiológicos y de comportamiento. Entre los fisiológicos se pueden encontrar las huellas digitales, los rasgos de la cara, el iris y la retina en los ojos, la morfología de las manos, la voz y los olores corporales. Entre los de comportamiento se pueden mencionar: la forma de caminar, de hablar, de escribir o de teclear. (1)

El estudio de métodos automáticos para el reconocimiento único de personas a través de características conductuales o físicas permitió el surgimiento de los sistemas biométricos. Ejemplo de ellos son los AFIS (Sistema Automatizado de Identificación de Huellas Dactilares, por sus siglas en inglés), sistemas de reconocimiento facial y sistemas de reconocimiento mediante voz, entre otros.

Con el desarrollo de las Tecnologías de la Información y las Comunicaciones (TIC) y la informatización de los principales procesos de la sociedad surgen los sistemas multibiométricos[1], en los que se agrupan varios patrones de verificación automática de la identidad de una persona.

Cuba, debido a las restricciones provocadas por el bloqueo económico hace más de cincuenta años, ha tenido la necesidad de desarrollar *software* propiamente cubanos, que impliquen el uso de la tecnología biométrica. La Universidad de las Ciencias Informáticas (UCI), específicamente el Centro de Identificación y Seguridad Digital (CISED) se encuentra inmerso en el desarrollo de un Sistema Automatizado de Identificación Biométrica (SAIBio) que posee como función principal el reconocimiento biométrico de un individuo teniendo en cuenta varias de las características que lo identifican.

El SAIBio gestiona las solicitudes biométricas mediante el servidor RabbitMQ, este es un gestor de colas de mensajes que se comunica con el sistema multibiométrico a través del Protocolo Avanzado de Colas de Mensajes (AMQP). Por la forma en que el SAIBio utiliza el RabbitMQ solo es posible acceder a él desde un

[1] Un sistema multibiométrico, utiliza más de un identificador biométrico independiente o correlacionado débilmente de un individuo, por ejemplo, la huella dactilar, el reconocimiento facial y la lectura mediante el iris.

ambiente local, lo que indica que los sistemas externos, es decir los sistemas que no se encuentren desplegados en el mismo ambiente que el SAIBio no se puedan comunicar directamente con este servidor para realizar una transacción biométrica[2].

De esta forma existe la posibilidad que un sistema externo con un control de acceso biométrico debidamente desarrollado se quiera comunicar con el sistema multibiométrico con el objetivo de identificar a una persona que no pertenece a su entidad pero si tiene permiso para acceder a ella. En la actualidad el SAIBio no puede procesar solicitudes de sistemas externos.

El análisis de esta situación problemática da paso a la formulación del siguiente **problema a resolver:**

¿Cómo establecer la comunicación de los sistemas externos con el sistema multibiométrico de manera asincrónica para la identificación de personas?

Se especifica como **campo de acción**: Los servicios de comunicación para el intercambio de transacciones biométricas.

Para dar solución al problema se precisa como **objetivo general**:

Desarrollar un componente que permita el intercambio de transacciones biométricas a través de correos electrónicos entre los sistemas externos que requieran la identificación de personas y el sistema multibiométrico.

Se definen como **objetivos específicos:**

1- Realizar un estudio de soluciones existentes que utilizan el estándar ANSI/NIST para el intercambio de transacciones biométricas.

2- Diseñar el componente para el intercambio de transacciones a través de ficheros ANSI/NIST por correo electrónico.

3- Implementar el componente para el intercambio de transacciones a través de ficheros ANSI/NIST por correo electrónico.

4- Realizar pruebas al componente desarrollado.

Para guiar la investigación se detallan las siguientes **tareas.**

1- Caracterización del estándar ANSI/NIST para el intercambio de transacciones biométricas.

---

[2] Una transacción biométrica es una interacción con una estructura de datos biométricos, compuesta por varios procesos que se han de aplicar uno después del otro. La transacción debe realizarse de una sola vez y sin que la estructura a medio manipular pueda ser alcanzada por el resto del sistema hasta que se hayan finalizado todos sus procesos.

2- Análisis del proceso de integración con el sistema multibiométrico de la UCI.
3- Descripción de las herramientas, tecnologías y metodologías de desarrollo seleccionadas para el desarrollo del componente.
4- Definición de la arquitectura sobre la que se desarrollará el componente.
5- Implementación de la biblioteca para la serialización y deserialización en formato ANSI/NIST de transacciones biométricas.
6- Diseño del componente para la confección y lectura del fichero con formato ANSI/NIST según el tipo de transacción biométrica.
7- Realización de pruebas unitarias, de aceptación y de carga.

El desarrollo de la presente investigación exigió el empleo de **métodos teóricos y empíricos** para obtener un acercamiento a los conocimientos relacionados con los servicios de comunicación para el transporte de datos biométricos a través de correos electrónicos.

El método teórico **Analítico-Sintético** permitió realizar un análisis de la bibliografía existente sobre la temática en cuestión, lo que ayudó a la confección del diseño de la investigación.

El empleo del método **Histórico-Lógico** basado en la extracción de lo teóricamente más importante de proyectos similares se utilizó para determinar el devenir histórico de la temática tratada, lo que permitió contar con referencias teóricas sobre el transporte de datos a través de correos electrónicos.

Se utilizó el **método empírico** para definir las relaciones esenciales y las características fundamentales de los servicios de comunicación para el intercambio de transacciones biométricas.

Específicamente, el método empírico **Análisis documental** permitió el estudio de la documentación sobre los protocolos de correos existentes en el mundo, así como los estándares internacionales para el transporte de datos biométricos.

Justificación de la investigación

La creciente necesidad en el mundo del uso de los sistemas multibiométricos como fuente más confiable de autenticación requiere de una alta calidad en sus procesos. Cuba trabaja en la solución de su propio sistema teniendo en cuenta el aseguramiento de los parámetros de calidad requeridos internacionalmente. Para asegurar un mejor funcionamiento del mismo se desarrolla un componente que permitirá la comunicación a través de correos electrónicos de sistemas externos con el propio sistema multibiométrico.

La presente investigación se divide en tres capítulos organizados de la siguiente manera:

Capítulo I: Fundamentación teórica. En este capítulo se muestra una descripción de los principales estándares biométricos y protocolos de correos más utilizados en el mundo. Se realiza el estudio de sistemas similares y se define la metodología, herramientas, tecnologías y lenguajes a utilizar.

Capítulo II: Características de la propuesta de solución. Durante el desarrollo de este capítulo se realiza la descripción de los procesos del negocio de la aplicación. Se capturan los requisitos, de los cuales se derivan las historias de usuarios generadas definiendo a continuación una planificación para su entrega. Posteriormente se diseña el sistema a desarrollar precisando la arquitectura, patrones de diseño, el diagrama de clases y las tarjetas CRC generadas.

Capítulo III: Implementación y pruebas. En este capítulo se abordan aspectos relacionados con la codificación especificando para ello los estándares de codificación utilizados, el diagrama de componentes y el diagrama de despliegue que permitirá tener un mayor conocimiento acerca de la estructura y funcionamiento del sistema. Por último se realizan pruebas que validan el cumplimiento de las funcionalidades definidas.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Introducción

En el presente capítulo se realiza una descripción de conceptos que se contemplan dentro del estado del arte de la investigación desarrollada, así como de los principales estándares, formatos y protocolos de *Internet* que se utilizan para el transporte de datos. Se realiza una búsqueda y análisis de la información correspondiente a soluciones similares existentes en el mundo sobre el intercambio de transacciones biométricas mediante archivos con el formato ANSI/NIST utilizando el correo electrónico. Se estudian las herramientas y metodologías actuales más utilizadas en este campo para posteriormente realizar una descripción de sus características, principales funcionalidades y el aprovechamiento de sus ventajas.

Con el conocimiento de los puntos antes mencionados y otros, se logrará alcanzar elementos necesarios para complementar la investigación e iniciar el desarrollo del nuevo módulo de comunicación.

1.1. Biometría

La biometría proveniente de las palabras *bio* (vida) y *metría* (medida), es una tecnología de seguridad basada en el reconocimiento de una característica física e intransferible de las personas, como por ejemplo la huella digital, el iris, el rostro, entre otros. (2)

1.2. Sistemas biométricos

Los sistemas biométricos se utilizan para la identificación automática de personas mediante el uso de características físicas del individuo o de su comportamiento. (3)

Entre las ventajas que proporcionan estos sistemas se pueden citar:

- Una identificación segura y única del individuo.
- El código de identificación es intransferible. Solamente la persona autorizada es identificada como tal.
- El código biométrico ni se puede perder ni se puede olvidar, pues la persona autorizada siempre lo lleva consigo.

Los sistemas multibiométricos combinan varias características del comportamiento y propiedades fisiológico-biológicas de los individuos, con el objetivo de obtener una mayor fiabilidad, rendimiento y seguridad. (4)

1.3. Estándar biométrico

Un estándar biométrico es un documento que especifica los formatos para el intercambio de datos de un programa de aplicación biométrico. La carencia de estándares biométricos a nivel industrial ha dificultado el desarrollo de algunos tipos de sistemas biométricos y el crecimiento de este sector. Sin embargo la industria biométrica está teniendo un papel muy activo para solucionar el problema de la carencia de estándares generando resultados que empiezan a ser ampliamente aceptados por los industriales y marcando el camino a seguir en un futuro próximo.

Los esfuerzos que se están realizando y los ya realizados han perseguido distintos objetivos que van desde la definición de API (Interfaz de Programación de Aplicaciones), los formatos de los ficheros con la información de parámetros biométricos, la encriptación de la información biométrica, hasta la interacción entre dispositivos biométricos diferentes, etc. (5)

A continuación se exponen las principales características de los estándares biométricos encontrados en la bibliografía.

- **BioAPI**

El consorcio BioAPI nace en Abril de 1998 durante la conferencia CardTech/SecureTech con el apoyo de algunas de las compañías informáticas más importantes a nivel internacional como IBM y Hewlett-Packard. La primera especificación apareció en Septiembre de 2000 y la especificación final en Marzo de 2001. La idea era desarrollar una alternativa a otras iniciativas de estandarización.

BioAPI intenta estandarizar el modo en el que las aplicaciones se comunican con los dispositivos biométricos[3] y la forma en la que los datos son almacenados y utilizados, ofreciendo a los desarrolladores un conjunto común de funciones para interactuar modularmente con los distintos dispositivos, algoritmos, etc. Sin embargo, no pretende estandarizar el modo en el que los datos son generados, ni interponerse en los rasgos distintivos que define la tecnología biométrica de cada fabricante.

[3] Los dispositivos biométricos se encargan de la adquisición análoga o digital de algún indicador biométrico de una persona.

Las funciones de BioAPI cubren aspectos como el entrenamiento, la verificación e identificación de usuarios, la captura de datos, el procesamiento de los mismos, la comparación de patrones y el almacenamiento de la información biométrica. Establece un alto nivel de abstracción que permite a los desarrolladores olvidarse de los detalles particulares de fabricación de los distintos productos y de las tecnologías empleadas por los diferentes fabricantes. (6)

Actualmente muy pocas soluciones en esta área son compatibles con BioAPI aunque es muy importante resaltar que es un estándar ampliamente aceptado por la industria biométrica.

- **BAPI**

BAPI es un nuevo estándar biométrico desarrollado y planeado por un vendedor de soluciones biométricas llamado I/O *Software* en lugar de un consorcio de compañías e instituciones como fue el caso de BioAPI. En Mayo de 2000, Microsoft licenció BAPI, aunque había sido uno de los primeros en apostar por BioAPI, con la intención de incluirlo en las futuras versiones de sus sistemas operativos (*Windows*). La idea de que la tecnología biométrica forme parte de un sistema operativo ha hecho madurar esta área tecnológica, haciendo que deje de ser considerada como una tecnología del futuro y de esta manera integrarse al panorama actual de posibilidades a tener en cuenta a la hora de desarrollar aplicaciones. Otras compañías como Intel, guiadas por la iniciativa de Microsoft han apostado por BAPI, licenciando este estándar para incluirlo en sus computadoras móviles y dotarlas de aspectos de seguridad (6).

- **CBEFF**

El CBEFF (Formato de Ficheros Común para el Intercambio Biométrico, *Common Biometric Exchange File Format*, por sus siglas en inglés) tiene como objetivo definir los formatos de los patrones biométricos para facilitar el acceso y el intercambio de diferentes tipos de datos biométricos a los sistemas que integran esta tecnología o entre diferentes componentes de un mismo sistema.

CBEFF establece un formato para la cabecera de los ficheros definiendo campos obligatorios y opcionales que proporcionan elementos comunes (opciones de seguridad, de integridad de los datos, fecha de creación del fichero, firma, tipo de parámetro biométrico) para el intercambio de información entre los dispositivos biométricos y los sistemas que hacen uso de los mismos, además favorece la interoperabilidad entre las aplicaciones biométricas y los sistemas, simplifica la integración del *software* y el *hardware*, y posibilita la compatibilidad futura frente a los nuevos avances tecnológicos que se vayan produciendo.

CBEFF no busca soluciones de interacción con los dispositivos o con los procesos, sino un método común para manejar los datos biométricos. La definición e implementación de este estándar está siendo considerada para su incorporación en dispositivos como las tarjetas inteligentes bajo los auspicios del grupo de trabajo NIST/BC *Biometric Interoperability*, *Performance and Assurance Working Group*. Actualmente BioAPI y CBEFF se han unido para construir un frente común a la estandarización biométrica. Muchos

fabricantes están adoptando este estándar ofreciendo soluciones compatibles CBEFF, lo que implica que los ficheros que contienen los datos de los patrones biométricos tienen esta cabecera común. (6)

- **ANSI X9.84**

El estándar X9.84 proporciona integridad y privacidad. La privacidad se consigue mediante técnicas criptográficas aplicadas sobre los datos biométricos específicos del usuario, después de aplicar la técnica criptográfica estos datos se concentran en un bloque de datos, el cual se conoce como opaco. La integridad se alcanza aplicando de igual manera técnicas criptográficas como las firmas digitales o un mensaje con un código de autenticación de mensajes (*Message Authentication Code,* MAC por sus siglas en inglés) sobre todos los datos biométricos, es decir, la cabecera y el bloque de datos opaco con la información biométrica específica de ese usuario.

La privacidad y la integridad pueden conseguirse de forma independiente, pero cuando se requieren ambas, primero se consigue la integridad, con la firma y posteriormente la privacidad, ya que se cifran las dos. En particular X9.84 soporta el transporte de claves asimétricas, acuerdos entre claves, etc., tanto para la privacidad como para la integridad.

Aunque X9.84 se ha convertido en un estándar de facto[4], para las compañías que ofrecen servicios financieros, no se puede decir lo mismo para los fabricantes o vendedores de soluciones biométricas especialmente los que se encargan de las funciones de almacenamiento, transmisión, extracción y comparación de los datos, ya que el nuevo estándar les obliga a incorporar un nivel de seguridad y mecanismos criptográficos que no se encuentran implementados en los productos actuales. (6)

- **Estándar ANSI/NIST-ITL 1-2011.**

Esta norma define el contenido, el formato y las unidades de medida para el intercambio de información de huellas digitales, la impresión palmar, plantar, facial, cicatriz, marca y tatuaje, iris y ácido desoxirribonucleico (ADN). La información se compone de una variedad de elementos obligatorios y opcionales. Esta información está dirigida principalmente al intercambio entre las administraciones de justicia penal o las organizaciones que se basan en sistemas de identificación automáticos.

Características del estándar ANSI/NIST-ITL 1-2011.

Especificaciones para la transmisión electrónica de huellas dactilares y plantares, imagen facial, el iris y el ADN son implementaciones del ANSI/NIST-ITL 1-2011 usados por organismos norteamericanos como el

[4] Un estándar de facto es un patrón o norma que se caracteriza por no haber sido consensuada ni legitimada por un organismo de estandarización al efecto. Por el contrario, se trata de una norma generalmente aceptada y ampliamente utilizada por iniciativa propia de un gran número de interesados.

FBI (Oficina Federal de Investigaciones, FBI por sus siglas en inglés), el Departamento de Defensa, el Departamento de Seguridad Nacional, entre otros.

Posee un conjunto de funciones de forense examinador, márgenes que permiten el marcado y el intercambio de un conjunto muy rico de la impresión de huellas dactilares y palmares, información que asegura que los analistas utilicen la misma terminología, referencias y procedimientos para describir detalles tales como los poros y discontinuidades lineales. (7)

Esta norma ofrece la posibilidad de compartir las imágenes de todas las partes del cuerpo y las marcas antropométricas de la cara y la imagen del iris. El estándar define cómo especificar y compartir las coordenadas de geo-posicionamiento en la toma de muestras biométricas. La información relativa a las circunstancias de la recogida de los datos biométricos también puede ser incluida. Esto incluye imágenes de los artículos encontrados alrededor de la escena del crimen y clips de audio y video.

Otras tecnologías también asociadas con ANSI/NIST-ITL 1-2011 son el algoritmo de compresión de imágenes: Cuantización Escalar de Ondeletas (*Wavelet Scalar Quantization*, WSQ por sus siglas en inglés) del FBI y el estándar de compresión y codificación de imágenes del Grupo Conjunto de Expertos en Fotografía (*Join Photographic Expert Group*, JPEG por sus siglas en inglés).

Tipos de registros contenidos en un archivo ANSI/NIST-ITL 1-2011.

Este estándar define la composición de los registros que comprenden una operación que pueda estar transmitida a otro sitio o agencia. El organismo receptor fijará los requisitos para la resolución de escaneado, el número y tipo de registros y otros datos específicos del usuario con el fin de considerar la transacción válida. Todos los registros de una transacción deberán pertenecer a un solo tema. (8)

Según (8) una transacción está compuesta por registros. La norma actualmente soporta diferentes modalidades biométricas, y ha reservado identificadores de registro para la posible adición futura de otras modalidades.

- **Tipo 1**

  El registro de tipo 1 siempre será el primer registro dentro de la transacción. Brindará información que describe el tipo, la finalidad o utilización de la transacción en cuestión, una lista de cada registro incluido en la transacción, el originador o fuente del registro físico, y otros elementos de información útiles y necesarios.

- **Tipo 2**

  Este tipo de registro deberá contener campos de texto definidos por el usuario que proporciona la identificación e información descriptiva asociada con el objeto de la transacción.

- **Tipo 3**

Este tipo de registro está en desuso. No se ajusta a la versión de esta norma.

- **Tipo 4**

Los registros tipo 4 fueron diseñados para transmitir imágenes de huellas dactilares capturadas por un Sistema Automatizado de Identificación de Huellas Dactilares (AFIS, por sus siglas en inglés) u otros dispositivos de captura de imagen que funcionan a una resolución de escaneo nominal de 500 píxeles por pulgada.

- **Tipo 5**

Este tipo de registro está en desuso. No se ajusta a la versión de esta norma.

- **Tipo 6**

Este tipo de registro está en desuso. No se ajusta a la versión de esta norma.

- **Tipo 7**

Fue concebido como una medida temporal para permitir el intercambio de datos de imagen que se definiría por tipos de registros específicos en versiones posteriores de la norma. Dado que algunos sistemas más antiguos siguen utilizando este tipo de registro, se incluye en la norma.

- **Tipo 8**

Este registro deberá contener los datos que representan la firma del sujeto del que la muestra biométrica está siendo recogida y del operador de la captura de datos biométricos.

- **Tipo 9**

El registro tipo 9 debe ser utilizado para el intercambio de minucias.

- **Tipo 10**

El registro tipo 10 deberá contener y ser utilizado para el intercambio de datos de la imagen de la cara, cicatrices, marcas y tatuajes. Nuevo en esta versión de la norma es la extensión del tipo de registro para manejar imágenes de otras partes del cuerpo.

- **Tipo 11**

Registros reservados para uso futuro.

- **Tipo 12**

Registros reservados para uso futuro.

- **Tipo 13**

Contiene los datos de la imagen de huella dactilar latente[5] (de resolución variable). La resolución de las imágenes latentes será como mínimo de 39,37 ppmm[6].

- **Tipo 14**

El registro tipo 14 son impresiones de las diez huellas dactilares. Los datos de imágenes de huellas digitales de resolución variable que contiene este tipo de registro pueden estar en una forma comprimida.

- **Tipo 15**

Este registro es utilizado para el intercambio de datos de la huella de la palma en resolución variable, los cuales pueden estar en una forma comprimida.

- **Tipo 16**

Este registro está diseñado para fines de desarrollo y para el intercambio de imágenes diversas, junto con los campos de la información textual que acompañan la imagen digitalizada.

- **Tipo 17**

El registro tipo 17 contendrá los datos de imagen del iris. Este tipo de registro se ha desarrollado para proporcionar un nivel básico de interoperabilidad entre los sistemas.

- **Tipo 18**

El registro tipo 18 deberá contener y ser usado para el ADN y sus datos relacionados. Con plena consideración a la privacidad, esta norma solo utiliza las regiones no codificantes del ADN. Las regiones del ADN que codifican la información fenotípica se evitan deliberadamente.

- **Tipo 19**

Registro que contendrá los datos de imágenes plantares.

- **Tipo 20**

---

[5] Impresión dactilar que ha quedado estampada sobre una superficie lisa. Tienen la particularidad de que no son fácilmente visibles, de ahí que su revelación se puede realizar mediante el uso de polvos o productos químicos. Este tipo de huellas se origina por el contacto de los dedos de las manos, pies, etc., con cualquier superficie lisa. (9)

[6] Píxeles por milímetro.

El registro tipo 20 deberá contener la fuente de representación de la que se derivaron otros tipos de registros. Algunos de los usos posibles del registro de tipo 20 son:

- A partir de una foto de grupo se almacena en un registro de tipo 20, el rostro de un sujeto que está segmentado y almacenado en un registro de tipo 10.
- A partir de una serie de imágenes de la cara fuera de ángulos, almacenadas en distintos registros de tipo 20, se genera una sola imagen 2D de la cara, usando la fusión que se almacena en un registro de tipo 10.
- **Tipo 21**

El registro de tipo 21 deberá contener una imagen de contexto asociada a un audio / visual u otros datos relacionados.

- **Tipo 98**

El registro de tipo 98 deberá contener la información de seguridad que permite el aseguramiento de la autenticidad e integridad de la transacción, incluyendo información tal como datos binarios, hashes, atributos de auditoría o de identificación de los efectos y las firmas digitales.

- **Tipo 99**

Este registro es utilizado para el intercambio de datos biométricos que no son compatibles con tipos de registros ANSI/NIST-ITL. Esto proporciona un nivel básico de interoperabilidad y la armonización con otros formatos de intercambio biométricos.

Ventajas del estándar ANSI/NIST-ITL 1-2011

La principal ventaja del estándar ANSI/NIST-ITL es la capacidad de contener imágenes múltiples y otros tipos de datos sobre la persona en un solo archivo, obteniendo así una buena interoperabilidad en los sistemas participantes. Un archivo de WSQ contiene solamente una imagen de la huella digital, pero un archivo en formato ANSI/NIST contiene dentro muchas imágenes de la huella digital y otra información. (8)

Específicamente la norma ANSI/NIST-ITL 1-2011, en comparación con su versión anterior el ANSI/NIST-ITL 2-2008, incorpora mejoras sustanciales al implementar nuevos registros que se ajustan a las necesidades actuales de la sociedad, ejemplo de ello es el registro plantar, el ADN, uno de aseguramiento de información de la transacción procesada, además de permitir interoperabilidad con otros estándares de intercambio de información biométrica.

1.4. Comunicación asincrónica

La comunicación asincrónica se refiere al acceso a información entre usuarios de la red de manera no simultánea, es decir los participantes no están conectados en el mismo espacio de tiempo. Cuando se escribe un mensaje por correo electrónico no se tiene una conexión directa con el compañero. Se escribe un texto y se envía; el receptor lo encuentra cuando mira otra vez en su buzón y entonces puede contestarlo. Entre las ventajas que presenta este tipo de comunicación se pueden encontrar:

- Toda la información que se envía queda grabada, de manera que se puede recurrir a ella en cualquier momento.
- La información transmitida llega al instante y a todas las personas que ha sido enviada.
- El contacto establecido, es individual y personalizado.
- Es un medio muy adecuado para fomentar las comunicación y el diálogo.

1.5. Correo electrónico

El correo electrónico es un tipo de comunicación asincrónica que permite que dos o más usuarios se comuniquen entre sí por medio de mensajes que son enviados y recibidos a través de una computadora o dispositivo afín. (10)

Sus principales características son:

- Puede enviar o recibir mucha información, ya que se pueden enviar archivos que contengan libros, revistas, datos.
- Permite trabajar directamente con la información recibida utilizando un procesador de textos, una hoja de cálculo. Cualquier mensaje se puede modificar, reutilizar, imprimir, etc.
- Es multimedia ya que se pueden incorporar imágenes y sonido a los mensajes.
- Permite enviar mensajes a grupos de personas utilizando las listas de correo.
- Puede consultarse en cualquier lugar del mundo.
- Es rápido y económico. (11)

A continuación se muestra una imagen donde se describe el funcionamiento del correo electrónico (Figura 1).

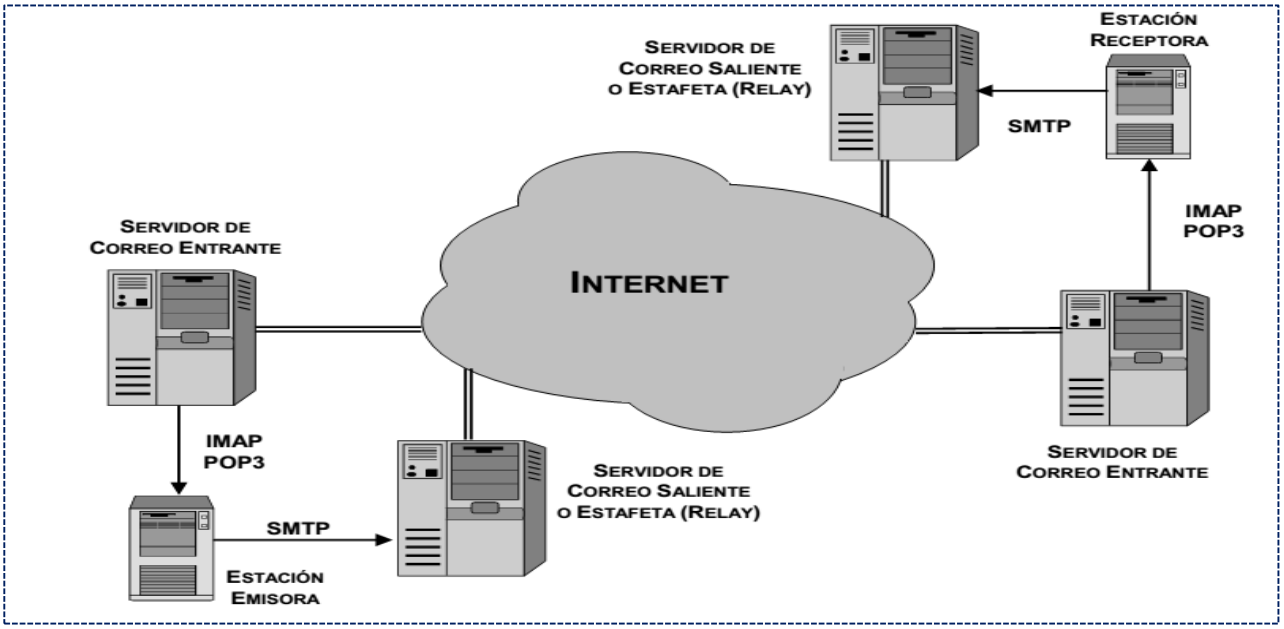


Figura 1: Funcionamiento del correo electrónico (12)

1.5.1. Protocolo de transporte de correo

SMTP

El Protocolo Simple de Transferencia de Correo (SMTP, por sus siglas en inglés) es un protocolo utilizado para el envío de mensajes. Corresponde a un modelo cliente/servidor. Inicialmente el cliente establece una conexión confiable mediante TCP (Protocolo de Control de Transmisión, por sus siglas en inglés) al puerto TCP número 25 del servidor de correo. Luego se intercambian mensajes definidos en el protocolo a través de esa conexión.

Algunos de los mensajes que envía el cliente son:

- *HELO*, para abrir una sesión con el servidor.
- *MAIL FROM*, para indicar quien envía el mensaje.
- *RCPT TO*, para indicar el destinatario del mensaje.
- *DATA*, para indicar el comienzo del mensaje, este finalizará cuando haya una línea únicamente con un punto.
- *QUIT*, para cerrar la sesión.

El servidor emite las respuestas con un código de tres dígitos. Las mismas se dividen por categorías:

- 2XX, para una respuesta satisfactoria.
- 3XX, para una respuesta temporal afirmativa.
- 4XX, para una respuesta de error, pero se espera a que se repita la instrucción.

- 5XX, para una respuesta de error. (13)

1.5.2. Protocolos de acceso a correo

POP3 (Protocolo de oficina de correos)

El protocolo POP3 permite recibir los mensajes electrónicos alojados en un servidor remoto y descargarlos en un ordenador para posteriormente revisarlos. Al igual que el SMTP se basa en el modelo cliente/servidor donde el cliente envía una serie de comandos de textos al servidor POP y este emite como respuesta un número acompañado de un mensaje descriptivo.

A continuación se presentan los comandos más utilizados por el cliente:

- *USER* <nombre>, envía identificación de usuario.
- *PASS* <contraseña>, envía la clave del servidor.
- *STAT*, devuelve el número de mensajes no borrados en el buzón y su longitud total.
- *LIST*, muestra todos los mensajes no borrados con su longitud.
- *RETR* <número>, solicita el envío del mensaje especificando el número.
- *TOP* <número> <líneas>, muestra la cabecera y el número de líneas requerido del mensaje especificando el número.
- *DELE* <número>, borra el mensaje especificando el número.
- *RSET*, recupera los mensajes borrados (en la conexión actual).
- *UIDL* <número>, devuelve una cadena identificadora del mensaje, persistente a través de las sesiones.
- *QUIT*, cierra la sesión actual. (13)

IMAP

IMAP es el acrónimo de *Internet Message Access Protocol* (Protocolo de Acceso de Mensajes de Internet). Este protocolo permite a un cliente de correo electrónico acceder a los mensajes almacenados en un servidor. Es más moderno y cuenta con todas las funciones del antiguo protocolo POP. El protocolo IMAP soporta tanto actividad con conexión como sin ella. Por lo tanto, los mensajes se pueden almacenar tanto en el equipo local como en el servidor de correo electrónico. Entre las ventajas del protocolo IMAP se encuentran:

- Ofrece una comunicación bidireccional entre el servidor de correo *web* y el cliente de correo electrónico.
- Los correos están en todo momento en el servidor, por lo que se puede acceder a ellos desde cualquier lugar, teniendo un dispositivo con acceso a *internet*.

- Si la computadora se daña, se extravía o la roban los correos están seguros en el servidor
- Acceso y gestión fácil desde un dispositivo móvil. (14)

1.6. Estado del arte

La biometría no se puso en práctica en las culturas occidentales hasta finales del siglo XIX, pero era utilizada en China desde al menos el siglo XIV. Un explorador y escritor que respondía al nombre de Joao de Barros escribió que los comerciantes chinos estampaban las impresiones y las huellas de la palma de las manos de los niños en papel con tinta. Los comerciantes hacían esto como método para distinguir entre los niños jóvenes.

En occidente, la identificación confiaba simplemente en la memoria fotográfica hasta que Alphonse Bertillon, jefe del departamento fotográfico de la Policía de París, desarrolló el sistema antropométrico (también conocido más tarde como Bertillonage) en 1883. Este era el primer sistema preciso, ampliamente utilizado científicamente para identificar a criminales y convirtió a la biometría en un campo de estudio. Funcionaba midiendo de forma precisa ciertas longitudes y anchuras de la cabeza y del cuerpo, así como registrando marcas individuales como tatuajes y cicatrices.

El sistema de Bertillon fue adoptado extensamente en occidente hasta que aparecieron defectos en el sistema principalmente problemas con métodos distintos de medidas y cambios de medida. Después de esto, las fuerzas policiales occidentales comenzaron a usar la huella dactilar, esencialmente el mismo sistema visto en China cientos de años antes. (15)

En estos últimos años la biometría ha crecido desde usar simplemente la huella dactilar, a emplear muchos métodos distintos teniendo en cuenta varias características físicas y de comportamiento. Las aplicaciones de la biometría también han aumentado desde solo identificación hasta sistemas de seguridad y más.

Para permitir la integridad e interoperabilidad entre los sistemas surgen los estándares biométricos, garantizando que las soluciones sean más escalables, óptimas y robustas. Además de reducir el costo de desarrollo y mantenimiento, son neutrales y no favorecen a ningún vendedor o modalidad en particular.

Específicamente el estándar ANSI/NIST-ITL, es una norma muy usada en el mundo. Su primera versión lleva por nombre ANSI/NBS-ICST 1-1986 y fue publicado por el Instituto Nacional de Normas y Tecnología (*National Institute of Standards and Technology,* NIST por sus siglas en inglés,) en 1986. Era un estándar basado en minucias que requiere una cantidad mínima de memoria para el intercambio y el almacenamiento de información de huellas dactilares.

En 1993 surgió una versión actualizada del Formato de datos para el intercambio de información de huellas dactilares, estándar ANSI/NIST-CSL 1-1993. Aunque se mantiene la disposición de los datos de las

minucias, el estándar se centró en los formatos para el intercambio de imágenes de huellas dactilares en lugar de datos de minucias procesados. En 1997 se aprobó una adición al prever el intercambio de datos de imágenes faciales y datos de imágenes capturadas de las cicatrices, marcas y tatuajes. La adición llevó el nombre de ANSI/NIST-ITL 1a-1997.

Un taller convocado en 1998 para revisar la norma y su adición dio lugar a una nueva revisión en que se fusionaron los dos documentos, se hizo hincapié en el registro de campo identificado, se introdujeron nuevos tipos de registro para el intercambio de huellas dactilares registradas y la impresión de imágenes de la palma de la mano. La revisión se titularía “Formato de Datos para el Intercambio de Información de Huellas Dactilares, la Cara, Cicatrices y Tatuajes” y llevó la designación de ANSI/NIST-ITL 1-2000. Como resultado de los talleres convocados en el año 2005, la norma se actualizó y amplió en dos partes.

La parte uno ANSI/NIST-ITL 1-2007, versión convencional en formato de campo identificado y se aprobó en abril de 2007. Las principales mejoras en la revisión de la parte uno son la calidad y el apoyo de segmentación, un nuevo bloque de campos minucias, un nuevo tipo de registro para el intercambio de información de iris, y un nuevo tipo de registro que contiene información biométrica que no se describe en la norma, pero conformes con otras normas de formato de datos biométricos registrados.

Luego la parte 2 nombrada ANSI/NIST-ITL 2-2008 llevaría como objetivo describir una correspondencia uno a uno de los elementos XML[7] en los campos convencionales numéricamente etiquetados descritos en la parte 1, así como definir una representación XML que se ajustara al Modelo de Intercambio de Información Nacional para permitir el intercambio de información entre múltiples agencias gubernamentales. En noviembre de 2011 el NIST publicó una norma biométrica que amplía en gran medida el tipo y la cantidad de información que se puede intercambiar entre diferentes sistemas, lleva por nombre “Formato de Datos para el Intercambio de Huella Dactilar, Facial y Otra Información Biométrica” y es la versión más actualizada de esta norma. (16)

En el presente trabajo se realiza un estudio de sistemas que utilizan para el intercambio de información biométrica la norma ANSI/NIST, con el objetivo de seleccionar la versión que más se ajusta al desarrollo de la investigación.

1.6.1. Nivel internacional

- **NISTPack**

[7] Lenguaje de Etiquetado Extensible (*eXtensible Markup Language*).

NISTPack es un SDK (*Kit* de Desarrollo de *Software,* por sus siglas en inglés) que habilita aplicaciones para leer, escribir, ver, editar y validar transacciones de datos biométricos en cumplimiento de las normas ANSI/NIST-ITL 1-2000, -2007, 2-2008 y 2011 como un borrador. El uso de NISTPack asegura que las imágenes biométricas se compriman correctamente, que los datos demográficos se incluyan en el formato correcto y que el objeto resultante se construya debidamente para el intercambio de datos entre normas basadas en sistemas biométricos. NISTPack ofrece una API común en C# o Java para crear y validar transacciones biométricas que cumplen con la codificación binaria tradicional o XML de la norma. La imagen biométrica y los datos biográficos sin procesar se pueden ingresar y el diseño de la API facilita la salida en cualquiera de los dos formatos. Para crear los formatos se usan las mismas funciones de la API. Asimismo, NISTPack admite dos vías de conversión entre los datos con codificación binaria y codificación XML en cumplimiento con la norma. (17)

Funciones principales del sistema:

Lectura de la información de un archivo de transacción. Consiste en leer el archivo en un formato interno y permitir el acceso a la imagen y datos de texto de la transacción. Esto podría ser hecho por usuarios que necesiten mostrar la información contenida en un archivo de transacción.

Creación de un archivo de transacción. Genera un archivo válido, una transacción a partir de información en formato de imagen y de texto. Puede ser hecha por varios usuarios que deban realizar el envío de un archivo que contenga las diez huellas dactilares de una persona o cualquier otro tipo de transacción.

Edición de archivos de transacción. Permite realizar cambios en los archivos existentes o corregir elementos y asegurar que el nuevo archivo cumpla con las normas.

Verificación de un archivo de transacción. Verifica si el archivo de transacción cumple con el formato de transacción general según la especificación ANSI/NIST o de una implementación más específica como la del FBI o una implementación específica estatal. El usuario podría determinar si la transacción se ajusta a las especificaciones y, en caso contrario, generar una lista de errores.

NISTPack admite la implementación genérica de la norma ANSI/NIST mediante un archivo de reglas basadas en texto denominado Archivo de validación NISTPack, además admite las funciones de lectura, escritura y validación de la mayoría de las implementaciones específicas de agencias globales como la Interpol, FBI, Policía Federal Alemana, entre otras. (17)

- **AFIS Civil del Servicio Administrativo de Identificación, Migración y Extranjería (SAIME)**

El Ministerio del Interior y Justicia (MIJ) de la República Bolivariana de Venezuela utiliza un sistema biométrico AFIS para garantizar la integridad del proceso de solicitud y emisión de tarjetas de identidad y

de pasaportes, integrándose al sistema del mismo en forma transparente. La función principal del AFIS Civil es incrementar el nivel de seguridad y confiabilidad del proceso en general, al detectar en forma automatizada los intentos de usurpación de identidad y de múltiple registro de una misma persona bajo diferentes identidades. El sistema AFIS recibe, procesa y responde a solicitudes de identificación y autenticación provenientes del sistema del MIJ o de otros sistemas a través de una interfaz única. El AFIS Civil se integra en este sistema como un bloque "motor de identificación" que responde a solicitudes de identificación, autenticación, o recuperación de datos, por el intermedio de datos en formato ANSI/NIST enviados por el protocolo SMTP.

Básicamente el sistema de SAIME se compone de:

Del AFIS Civil mismo, que se encarga de:

- Proporcionar la interfaz con el sistema del MIJ.
- Procesar y responder a solicitudes de identificación y autenticación recibidas del sistema del MIJ o de otros sistemas a través de esta interfaz única.

De módulos biométricos para estaciones de registro. Estos módulos de *hardware* y *software* se encuentran integrados en las estaciones de captura instaladas para la atención al público y permiten efectuar en ellas, además de las funciones normales de captura de datos, las siguientes funciones:

- Captura de las diez huellas dactilares con control de calidad.
- Captura de la fotografía con control de calidad.

El AFIS Civil del SAIME utiliza el Formato de Datos para el Intercambio de Información de Huellas Dactilares, Marcas Faciales, Cicatrices y Tatuajes, ANSI/NIST-ITL 1-2000. (18)

- **MorphoBis**

MorphoBis es un Sistema innovador de Identificación Biométrica Automatizada que permite identificar sospechosos y criminales en tiempo real. Este poderoso AFIS se basa en la profunda experiencia de Morpho[8] en el tema de biometría. (19)

Entre sus características fundamentales se encuentran:

- Integra completamente huellas dactilares, impresiones de la palma de la mano, imágenes del rostro, firmas, descripciones y otros datos.

[8] Empresa de alta tecnología con excelentes resultados en soluciones de seguridad. (20)

- Posee herramientas fáciles de utilizar para el mejoramiento de imágenes en las estaciones de trabajo.
- Cumple con los estándares internacionales ANSI/NIST-ITL1-2007 y 2-2008; FBI EBTS V9.1 y EBTS IEPD-XML V2.0.

MorphoBis emplea la comparación de huellas decadactilares y latentes, con un tratamiento automatizado. Además, maneja la comparación de huellas palmares y búsquedas de huellas de dos dedos a través de terminales móviles. El tratamiento automatizado permite identificar posibles sospechosos que después son verificados por técnicos.

La tecnología de Morpho es reconocida como la tecnología número uno por el Instituto Nacional de Estándares y Tecnologías de EE.UU por la precisión del tratamiento automatizado de huellas dactilares latentes. Las fuerzas de la policía disponen así de la solución más eficiente y económica para procesar un gran número de búsquedas de latentes, permitiendo que sus expertos se enfoquen en los casos más críticos. (20)

1.6.2. Nivel nacional

- **Sistema de Carga Masiva de la *Suite* BIOMESYS AFIS de DATYS.**

La empresa “DATYS, Tecnologías y Sistemas”, cuenta con una división destinada al desarrollo de *software* biométrico conocida como Biomesys *SUITE*. Desde el año 2006 esta poderosa institución implementa el Biomesys AFIS, diseñado para reducir los tiempos y aumentar la efectividad, en la identificación y autenticación de personas. Los sistemas AFIS modernos se encuentran totalmente automatizados. Con el fin de brindar resultados positivos en los procesos anteriores, es necesario contar, ante todo, con una base de datos de impresiones dactilares de las personas. BIOMESYS AFIS Carga Masiva permite realizar la digitalización de grandes volúmenes de fichas dactilares en formato papel así como su procesamiento para su posterior envío hacia cualquier sistema AFIS.

El Sistema de Carga Masiva de fichas dactilares constituye un producto capaz de procesar más de mil fichas por hora y generar archivos con formato ANSI/NIST que pueden ser enviados hacia cualquier sistema de identidad que utilice un AFIS como herramienta de identificación biométrica.

Este proceso de carga masiva se encuentra dividido en tres sub-procesos (Figura 2):

- Preparación de lotes de fichas dactilares.
- Digitalización de lotes de fichas dactilares.
- Asociación de datos biográficos.

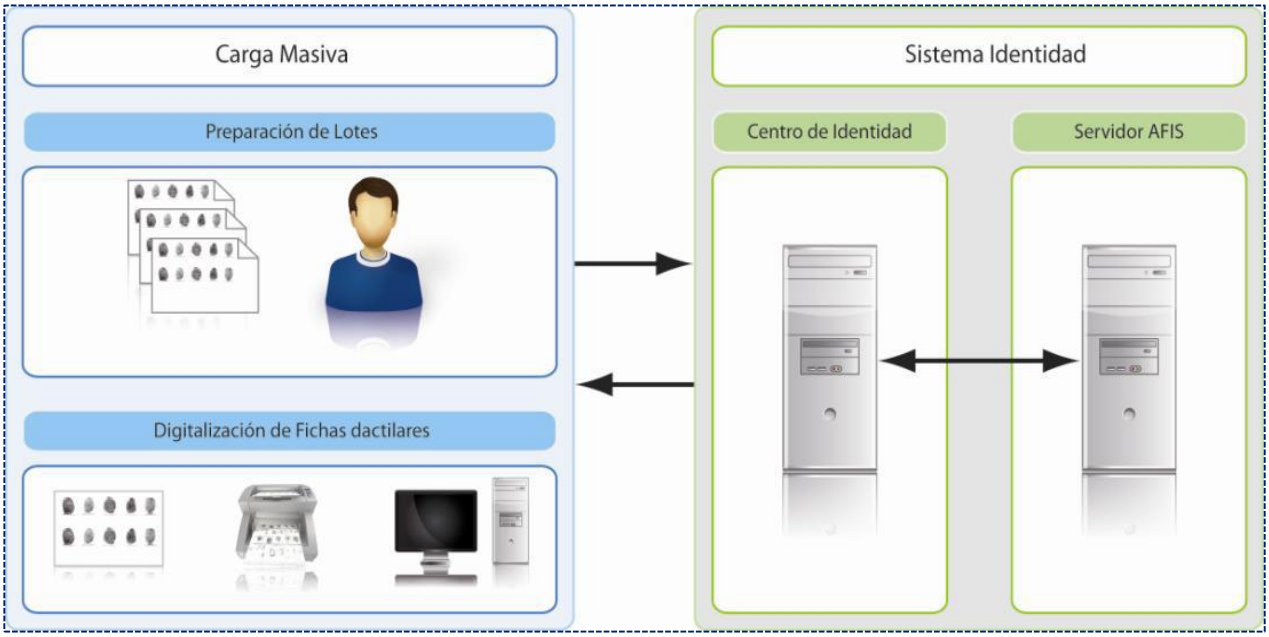


Figura 2: Arquitectura global del Sistema de Carga Masiva (21)

El proceso de “Preparación de Lotes de Fichas Dactilares” consiste en realizar una preparación previa y detallada de toda la información que se pretenda procesar; a cada ficha se le asigna un identificador y se agrupan en lotes a los cuales se les fija, de igual forma, un identificador (código de barras).

La “Digitalización de Lotes de Fichas Dactilares” se fundamenta en el procesamiento digital de una gran cantidad de fichas dactilares en papeles a formato digital de manera automática; este proceso recorre varios estados que van desde la digitalización de la impresión en formato de papel hasta la extracción de la información válida: las impresiones dactilares y los datos biográficos, para su posterior almacenamiento en base de datos.

La “Asociación de datos biográficos” es el proceso mediante el cual se relaciona la información manuscrita en las fichas dactilares a su formato digital. El resultado final consiste en un archivo NIST compatible con la norma ANSI/NIST ITL 1-2000, y con la especificación de la Interpol, versión número 4.22b de octubre de 2008. (21)

1.6.3. Aportes del estado del arte

El estudio de las principales características y funciones de los sistemas analizados arrojó como resultado que:

- El estándar a utilizar será el ANSI/NIST ITL 1-2011 por ser la norma en que más tipos de datos biométricos se pueden transportar.

- El estándar ANSI/NIST es la norma más utilizada por entidades estatales para el transporte de datos biométricos lo que demuestra su confiabilidad y estabilidad en su rendimiento.
- Cuba y específicamente la empresa DATYS posee experiencia en la utilización de este estándar por lo que se obtiene una valiosa fuente de información que sirve de apoyo al desarrollo de esta investigación.

1.7. Metodologías de desarrollo de software

En ingeniería de *software*, una metodología de desarrollo es un marco de trabajo usado para estructurar, planificar y controlar el proceso de desarrollo en sistemas de información, surgida ante la necesidad de utilizar una serie de procedimientos, técnicas, herramientas y soporte documental a la hora de desarrollar un *software*.

1.7.1. Metodologías tradicionales

Las metodologías tradicionales imponen una disciplina de trabajo sobre el proceso de desarrollo del *software*, con el fin de conseguir un producto más eficiente. Para ello, se hace énfasis en la planificación total de todo el trabajo a realizar y una vez que está todo detallado, comienza el ciclo de desarrollo del producto de *software*. Se centran especialmente en el control del proceso, mediante una rigurosa definición de roles, actividades, artefactos, herramientas y notaciones para el modelado y documentación detallada. Las metodologías tradicionales no se adaptan adecuadamente a los cambios, por lo que no son métodos adecuados cuando se trabaja en un entorno donde los requisitos no pueden predecirse o bien pueden variar.

- **RUP**

El Proceso Unificado de *Rational* es un proceso de desarrollo de software creado por la empresa *Rational Software*, que junto con el Lenguaje Unificado de Modelado (UML, por sus siglas en inglés), constituye la metodología estándar más utilizada para el análisis, diseño, implementación y documentación de sistemas orientados a objetos. Se presta especial atención al establecimiento temprano de una buena arquitectura que no se vea fuertemente impactada ante cambios posteriores durante la construcción y el mantenimiento del *software*.

RUP divide el proceso en cuatro fases (Figura 3), dentro de las cuales se realizan varias iteraciones en número variable según el proyecto y en las que se hace un mayor o menor hincapié en las distintas actividades.

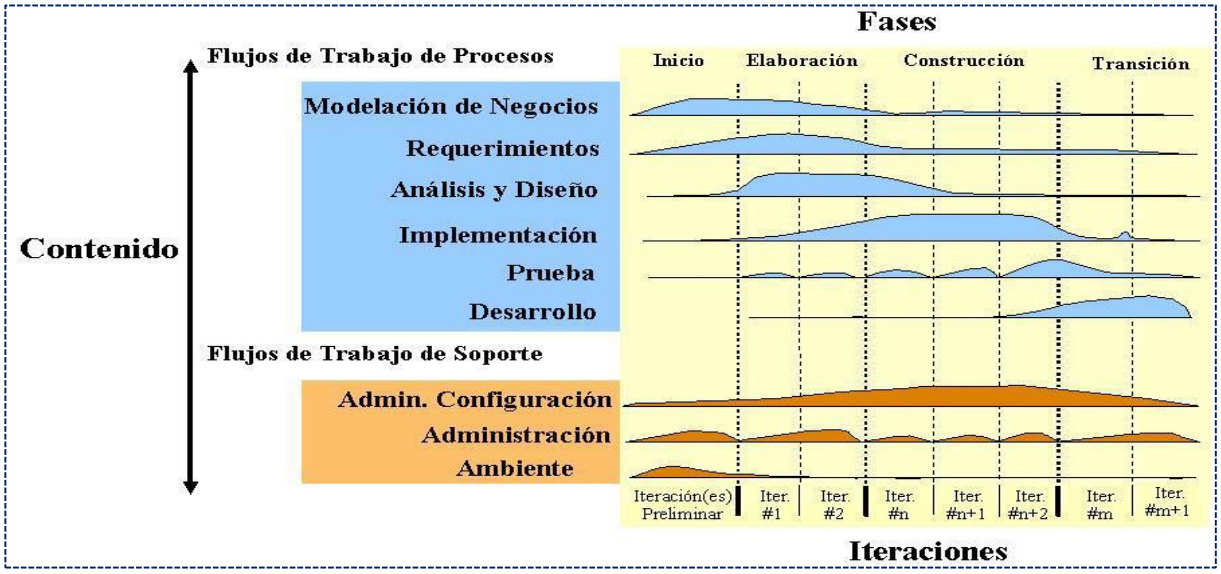


Figura 3: Ciclo de desarrollo propuesto por RUP (22)

Durante la fase de inicio las iteraciones hacen mayor énfasis en actividades de modelado del negocio y de requisitos. En la fase de elaboración, las iteraciones se orientan al desarrollo de la base de la arquitectura, abarcan más los flujos de trabajo de requerimientos, modelo de negocios (refinamiento), análisis, diseño y una parte de implementación orientado a la base de la arquitectura. En la fase de construcción, se lleva a cabo la construcción del producto por medio de una serie de iteraciones. Para cada iteración se selecciona algunos Casos de Uso, se refina su análisis y diseño y se procede a su implementación y pruebas. Se realiza una pequeña cascada para cada ciclo. Se realizan tantas iteraciones hasta que se termine la implementación de la nueva versión del producto. (22)

1.7.2. Metodologías ágiles

Las metodologías ágiles se basan en la planificación adaptativa donde los requisitos y soluciones evolucionan mediante la colaboración de grupos organizados y multidisciplinarios. Existen principios recogidos en el llamado Manifiesto Ágil[9], que justifican su aparición:

- Los individuos y las interacciones entre ellos son más importantes que las herramientas y los procesos empleados.
- Es más importante crear un producto de *software* que funcione, que escribir una documentación exhaustiva.

[9] Documento firmado en 2011 por destacados desarrolladores, escritores y consultores de la industria del *software*. (23)

- La colaboración con el cliente debe prevalecer sobre la negociación de contratos.
- La capacidad de respuesta ante un cambio es más importante que el seguimiento estricto de un plan. (23)

- **SCRUM**

La metodología SCRUM fue desarrollada por Ken Schwaber, Jeff Sutherland y Mike Beedle. Define un marco para la gestión de proyectos, que se ha utilizado con éxito durante los últimos 10 años. Está especialmente indicada para proyectos donde puedan existir cambios en los requisitos. Sus principales características se pueden resumir en dos. El desarrollo de software se realiza mediante iteraciones, con una duración de 30 días. El resultado de cada iteración es un incremento ejecutable que se muestra al cliente. La segunda característica importante son las reuniones a lo largo del proyecto. Estas son las verdaderas protagonistas, especialmente la reunión diaria de 15 minutos del equipo de desarrollo para coordinación e integración. (24)

- **Programación extrema**

Extreme Programming (XP, por sus siglas en inglés) es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de *software*, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en la realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y flexibilidad para enfrentar los cambios. Se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico.

El ciclo de vida ideal de XP consta de seis fases: Exploración, Planificación de la Entrega, Iteraciones, Producción, Mantenimiento y Muerte del Proyecto. (25)

Fase I: Exploración

Los clientes plantean a grandes rasgos las historias de usuario que son de interés para la primera entrega del producto. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto. Se prueba la tecnología y se exploran las posibilidades de la arquitectura del sistema construyendo un prototipo. La fase de exploración toma de pocas semanas a pocos meses, dependiendo del tamaño y familiaridad que tengan los programadores con la tecnología.

Fase II: Planificación de la Entrega

El cliente establece la prioridad de cada historia de usuario, y correspondientemente, los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente.

Fase III: Iteraciones

Esta fase incluye varias iteraciones sobre el sistema antes de ser entregado. El plan de entrega está compuesto por iteraciones de no más de tres semanas. En la primera iteración se puede intentar establecer una arquitectura del sistema que pueda ser utilizada durante el resto del proyecto.

Fase IV: Producción

La fase de producción requiere de pruebas adicionales y revisiones de rendimiento antes de que el sistema sea trasladado al entorno del cliente. Al mismo tiempo, se deben tomar decisiones sobre la inclusión de nuevas características a la versión actual, debido a cambios durante esta fase.

Fase V: Mantenimiento

Mientras la primera versión se encuentra en producción, el proyecto XP debe mantener el sistema en funcionamiento al mismo tiempo que desarrolla nuevas iteraciones. Para realizar esto se requiere de tareas de soporte para el cliente. De esta forma, la velocidad de desarrollo puede bajar después de la puesta del sistema en producción. La fase de mantenimiento puede requerir nuevo personal dentro del equipo y cambios en su estructura.

Fase VI: Muerte del Proyecto

Es cuando el cliente no tiene más historias para ser incluidas en el sistema. Esto requiere que se satisfagan las necesidades del cliente en otros aspectos como rendimiento y confiabilidad del sistema. Se genera la documentación final del sistema y no se realizan más cambios en la arquitectura. La muerte del proyecto también ocurre cuando el sistema no genera los beneficios esperados por el cliente o cuando no hay presupuesto para mantenerlo. (25)

Selección de la metodología de software a utilizar

Luego de realizar un análisis de las principales características, ventajas y desventajas de diferentes metodologías de *software* existentes en el mundo se llega a la conclusión de que la metodología a utilizar será XP por las siguientes razones:

- Las metodologías tradicionales y específicamente RUP está concebida para grandes grupos de proyectos que centran su atención en llevar una documentación exhaustiva de todo el proyecto y tratan de cumplir exactamente con un plan muy bien definido en la fase inicial del desarrollo del *software*, donde no se contemplan cambios durante la realización del mismo, pues implicaría grandes gastos de recursos y tiempo; estas características no son las más que se ajustan al equipo de desarrollo de la investigación, pues se cuenta con un equipo pequeño que no tiene

bien definido el alcance del *software*, lo que pudiera traer consigo cambios en diferentes etapas del proyecto.

- La metodología SCRUM plantea un problema si el desarrollo está restringido por una fecha de entrega y en esta investigación se cuenta con un tiempo determinado para terminar el *software*, además de que se no se le presta una buena atención a la documentación creada.
- Se decide utilizar la metodología XP por estar definida para equipos de desarrollo pequeños donde la prioridad es satisfacer al cliente mediante tempranas y continuas entregas de *software*; que le reporten un valor con el menor intervalo de tiempo posible entre una y otra, enfatizando en la adaptabilidad en lugar de la previsibilidad, por su capacidad de respuesta a los cambios, su conjunto de prácticas que facilitan la finalización de los proyectos y por las técnicas pragmáticas que ofrece. Harán la entrega del componente menos complicada y más satisfactoria tanto para los clientes como para el equipo de entrega.

1.8. Herramientas y tecnologías

1.8.1. Lenguaje de modelado

- **UML 2.3**

El UML es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de *software*. Captura decisiones y conocimiento sobre los sistemas que se deben construir. Se usa para entender, diseñar, hojear, configurar, mantener, y controlar la información sobre tales sistemas.

UML no solo permite realizar el diseño de los sistemas de *software* sino que permite también modelar la arquitectura de *hardware* donde se ejecutarán los mismos. Otra ventaja de este es que es independiente del lenguaje de programación, por lo que los diseños realizados utilizando UML se pueden implementar en la mayoría de estos. (26)

1.8.2. Herramientas para el diseño

- ***Rational Rose Enterprise Edition***

Rational Rose Enterprise Edition es una herramienta de producción y comercialización establecida por *Rational Software Corporation* (actualmente parte de IBM). *Rose* es un instrumento operativo conjunto que utiliza el UML como medio para facilitar la captura de dominio de la semántica, la arquitectura y el diseño. Este *software* tiene la capacidad de: crear, ver, modificar y manipular los componentes de un modelo.

Rational Rose es una herramienta de diseño orientada a objetos, que da soporte al modelado visual, es decir, que permite representar gráficamente el sistema, permitiendo enfatizar en los detalles más importantes, centrándose en los casos de uso y enfocándose hacia un *software* de mayor calidad, empleando un lenguaje estándar común que facilita la comunicación.

Proporciona mecanismos para realizar la ingeniería inversa, es decir, que a partir del código se pueda obtener información sobre su diseño; adicionalmente permite generar código en diferentes lenguajes a partir de un diseño en UML, brinda la posibilidad de que varias personas trabajen a la vez, permitiendo que cada desarrollador opere en un espacio de trabajo privado que contiene el modelo completo y permite que tenga un control exclusivo sobre la propagación de los cambios en ese espacio de trabajo. (27)

- **Visual Paradigm 8.0**

Visual Paradigm para UML es una herramienta CASE (Ingeniería de *Software* Asistida por Computadora, por sus siglas en inglés) que soporta el modelado mediante UML y proporciona asistencia a los analistas, ingenieros de *software* y desarrolladores, durante todos los pasos del ciclo de vida del desarrollo de un proyecto.

Las ventajas que proporciona Visual Paradigm para UML son:

- Dibujo: Facilita el modelado de UML, ya que proporciona herramientas específicas para ello. Esto también permite la estandarización de la documentación, ya que la misma se ajusta al estándar soportado por la herramienta.
- Corrección sintáctica: Controla que el modelado con UML sea correcto.
- Coherencia entre diagramas: Al disponer de un repositorio común, es posible visualizar el mismo elemento en varios diagramas, evitando duplicidades.
- Integración con otras aplicaciones: Permite integrarse con otras aplicaciones, como herramientas ofimáticas, lo cual aumenta la productividad.
- Trabajo multiusuario: Permite el trabajo en grupo, proporcionando herramientas de compartición de trabajo.
- Reutilización: Facilita la reutilización, ya que se dispone de una herramienta centralizada donde se encuentran los modelos utilizados para otros proyectos.
- Generación de código: Permite generar código de forma automática, reduciendo los tiempos de desarrollo y evitando errores en la codificación del *software*.

- Generación de informes: Permite generar diversos informes a partir de la información introducida en la herramienta. (28)

1.8.3. Lenguajes de programación

- ***C-Sharp* (C# 5.0)**

C# es un lenguaje de programación moderno, de alto nivel, de múltiples paradigmas y de uso general para crear aplicaciones con *Visual Studio* y *.NET Framework*. C# se diseñó para que fuera simple, poderoso, con seguridad de tipos y orientado a objetos. Las múltiples innovaciones de C# permiten un desarrollo rápido de aplicaciones con la expresividad y elegancia de los lenguajes al estilo C. (29)

C# es un lenguaje simple, diseñado para escribir aplicaciones empresariales. El lenguaje C# es una evolución de los lenguajes C y C++. Utiliza muchas de las características de C++ en las áreas de instrucciones, expresiones y operadores. A continuación se enumeran las principales características que definen al lenguaje de programación C#:

1. Sencillez de uso.
2. Modernidad.
3. Orientado a objetos.
4. Recolección de basura.
5. Seguridad de tipos
6. Instrucciones seguras
7. Unificación de tipos
8. Extensión de los operadores básicos
9. Eficiente (29)

- **Java**

Java es un lenguaje de programación orientado a objetos desarrollado en 1991 por la Sun Microsystems. Inspirado en C++, Java fue proyectado con la finalidad de obtener un producto de pequeñas dimensiones, simple y portátil sobre diferentes plataformas y sistemas operativos, sea a nivel de código fuente como a nivel de código binario; lo que significa que los programas Java pueden ser ejecutados sobre cualquier computadora en la cual sea instalada la máquina virtual. (30)

Entre las características fundamentales de este lenguaje de programación, se pueden encontrar las siguientes:

1. Es orientado a objetos.
2. Es muy flexible.
3. Funciona en cualquier plataforma.

4. Su uso no acarrea inversiones económicas.
5. Es de fuente abierta. (31)

1.8.4. Entornos integrados de desarrollo

- **NetBeans**

NetBeans (Entorno de Desarrollo Integrado, IDE por sus siglas en inglés) es una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Está escrito en Java pero puede servir para otros lenguajes de programación. Existe además un número importante de módulos para extender el NetBeans IDE. Es un producto libre y gratuito sin restricciones de uso. (32)

Características principales:

- Asistentes para la creación y configuración de distintos proyectos, incluida la elección de algunos *frameworks*[10].
- Buen editor de código, multilenguaje, con el habitual coloreado y sugerencias de código, acceso a clases, control de versiones, localización de ubicación de la clase actual, comprobaciones sintácticas y semánticas, plantillas de código, herramientas de refactorización, entre otras. También existen tecnologías donde se puede usar el pulsar y arrastrar para incluir componentes en el código.
- Optimización de código: ayuda a optimizar las aplicaciones e intentar hacer que se ejecuten más rápido y con el mínimo uso de memoria.
- Simplifica la gestión de grandes proyectos con el uso de diferentes vistas, asistentes de ayuda, y estructura la visualización de manera ordenada, lo que ayuda en el trabajo diario.
- Herramientas para depurado de errores: el *debugger* (en español depurador) que incluye el IDE es útil para encontrar dónde existen fallos. Se pueden definir puntos de ruptura en la línea de código que interese, monitorizar en tiempo real los valores de propiedades y variables, se nos permite ir paso a paso, ejecutar un método completo, o entrar dentro.
- Acceso a base de datos: desde el propio NetBeans, se puede conectar a distintos sistemas gestores de bases de datos, como pueden ser Oracle, MySQL y demás, y ver las tablas, realizar consultas y modificaciones, todo ello integrado en el propio IDE.

[10] Estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de software concretos, que puede servir de base para la organización y desarrollo de software.

- ***Visual Studio* 2012**

Visual Studio es una colección completa de herramientas y servicios que permite crear una gran variedad de aplicaciones, tanto para plataformas de Microsoft como para otras. *Visual Basic*, *Visual C++*, *Visual C#* y *Visual J#* utilizan el mismo IDE, que les permite compartir herramientas y facilita la creación de soluciones en varios lenguajes. Asimismo, dichos lenguajes aprovechan las funciones de .*NET Framework*, que ofrece acceso a tecnologías clave para simplificar el desarrollo de aplicaciones y servicios web. (33)

Características principales

- El (IDE) de *Visual Studio* integra todas las tareas de desarrollo en una sola herramienta.
- Es muy personalizable: compatibilidad con varios monitores, diseños continuos entre sesiones y cientos de opciones configurables que permiten realizar una sincronización entre múltiples dispositivos.
- Con un potente depurador integrado, el concentrador de rendimiento y diagnóstico, y las herramientas de generación de perfiles, Visual Studio ofrece un conjunto completo de funciones necesarias para crear y optimizar las aplicaciones
- Ofrece una gran extensibilidad y permite que los desarrolladores y los asociados integren sus propias herramientas y SDK.
- El mapa del código facilita la comprensión incluso del código fuente complejo gracias a la visualización.
- Cuenta con un editor de código/interfaz que soporta los más variados lenguajes (desde Html5 + JQuery, a C++ para dispositivos embebidos, pasando por Phyton con Django, o XMAL para *Windows Phone*).
- Inclusión de pruebas de rendimiento, y del análisis estático del código, redondean un módulo que orienta al desarrollador hacia las mejores prácticas de codificación y de técnicas avanzadas de programación.
- Incluye un completo *framework* de pruebas unitarias y de integración; acompañado por la gestión completa de los planes de prueba por medio del módulo *test manager*.
- También incluye un módulo con entidad propia para realizar el análisis en profundidad de los diagnósticos de prestaciones de cualquier tipo de aplicación desarrollada en .*Net*.
- Módulo orientado a la documentación arquitectónica que permite, por ejemplo, modelar en UML toda la estructura del proyecto, incluso generando código desde los diagramas; navegar por la vista de clases; verificar las referencias circulares, etc.
- Como todos los IDE modernos, permite construir las aplicaciones para todos los dispositivos, plataformas y sistemas operativos soportados; y realizar decenas de operaciones y validaciones

de depuración que permitan encontrar los fallos de manera fácil y sencilla; incluso permite añadirse a un proceso abierto en el equipo de desarrollo (o remotamente) para depurar aplicaciones no soportadas por .NET.

- Permite conectar a una base de datos SQL, comparar los esquemas, comparar los datos, lanzar consultas; crear un GUID (identificador único); ofuscar y analizar código; configurar servicios; obtener la ejecución detallada de procesos; y optimizar y configurar el propio IDE, son algunas de las decenas de herramientas que incluye *Visual Studio*.

Visual Studio 2013 es la última actualización para *Visual Studio* y se ha convertido en un ecosistema de desarrollo que unifica en una sola herramienta servidores de gestión de ciclo de vida, de planes de pruebas, laboratorios de pruebas, sistemas de integración continua, repositorios de código compartido avanzados, etc. Constituye un conjunto de herramientas que comprende todos y cada uno de los aspectos que están relacionados con la mayoría de los escenarios sobre los que puede realizarse programación de aplicaciones informáticas. (34)

Conclusiones parciales

En este capítulo se presentaron los principales conceptos asociados a la investigación que son de vital importancia para un mejor entendimiento de la solución propuesta. Se hizo un estudio y análisis crítico de algunos sistemas existentes en el mundo que utilizan el estándar ANSI/NIST para el intercambio de información biométrica, con el objetivo de seleccionar aquellos que contaran con funcionalidades similares a las que se quiere desarrollar en esta investigación. Este estudio permitió la selección de algunas bibliotecas empleadas en el AFIS Civil del SAIME y que serán reutilizadas en el desarrollo de este trabajo de diploma. Para el cumplimiento de la presente investigación, se decidió utilizar como metodología de desarrollo XP, como herramienta de modelado Visual Paradigm en su versión 8.0, que utiliza UML 2.3 como lenguaje de modelado, C# 5.0 como lenguaje de programación, utilizando el entorno integrado de desarrollo *Visual Studio* 2012.

CAPÍTULO 2: CARACTERÍSTICAS DE LA PROPUESTA DE SOLUCIÓN

Introducción

En el presente capítulo se ofrece una descripción de la solución propuesta con el objetivo de solucionar el problema inicialmente planteado. Se define el modelo conceptual en el que se reflejan los principales conceptos que se deben abarcar para desarrollar el servicio de comunicación. Posteriormente se plantean los requisitos funcionales, no funcionales y las historias de usuario correspondientes.

2.1. Flujo actual de los procesos del negocio

Cuando se desarrolle el sistema multibiométrico del CISED, este no contará con un servicio de comunicación que sea capaz de comunicar un sistema externo con el SAIBio, con el objetivo de identificar a una persona ajena a la entidad y que tiene permisos de acceso a ella, ocasionando pérdida de tiempo en el proceso de verificación de la persona a identificar y no podrá ser autenticado por el sistema. Este proceso se pudiera realizar a través de la vía telefónica pero trae consigo mucha inestabilidad e inseguridad. Existe la otra posibilidad de ir actualizando la base de datos del propio sistema externo pero sería un trabajo muy engorroso sobre todo para empresas muy grandes donde el flujo de personas sea muy activo. Se decide entonces el desarrollo de un componente para la comunicación con el sistema multibiométrico a través de ficheros ANSI/NIST por correo electrónico que posibilite la compatibilidad e interoperabilidad entre los sistemas externos y el sistema multibiométrico.

2.2. Modelo de dominio

Un modelo de dominio captura los tipos más importantes de objetos en el contexto del sistema. Los objetos del dominio representan los conceptos fundamentales y los eventos que suceden en el entorno en el que trabaja el sistema. Muchos de los objetos del dominio o clases pueden obtenerse de una especificación de requisitos o mediante la entrevista con los expertos del dominio. (35)

En el siguiente modelo de dominio (Figura 4) se describen las clases más importantes dentro del contexto del sistema, mostrando los principales conceptos con los que se trabaja. Se definen como conceptos o clases fundamentales:

- ***Sistema Externo:*** Se refiere al sistema con el que se necesita la comunicación para realizar una transacción biométrica.
- ***Servicio de Comunicación:*** Servicio que es capaz de establecer la comunicación de manera asincrónica entre el sistema multibiométrico y el sistema externo. Es el encargado de transportar y manipular los datos biométricos.

- ***Sistema Multibiométrico:*** Sistema automatizado que posee varias formas de reconocimiento biométrico y una base de datos única para la identificación de las personas.
- ***Solicitudes:*** Se refiere a los datos biométricos que se reciben del sistema externo en formato ANSI/NIST, una vez que la persona no pudo ser identificada por el mismo y que se envían al sistema multibiométrico a través del servicio de comunicación para ser procesadas.
- ***Respuestas:*** Mensajes que proporciona el sistema multibiométrico después de procesar una solicitud y que son enviados por el servicio de comunicación al sistema externo con el formato ANSI/NIST.

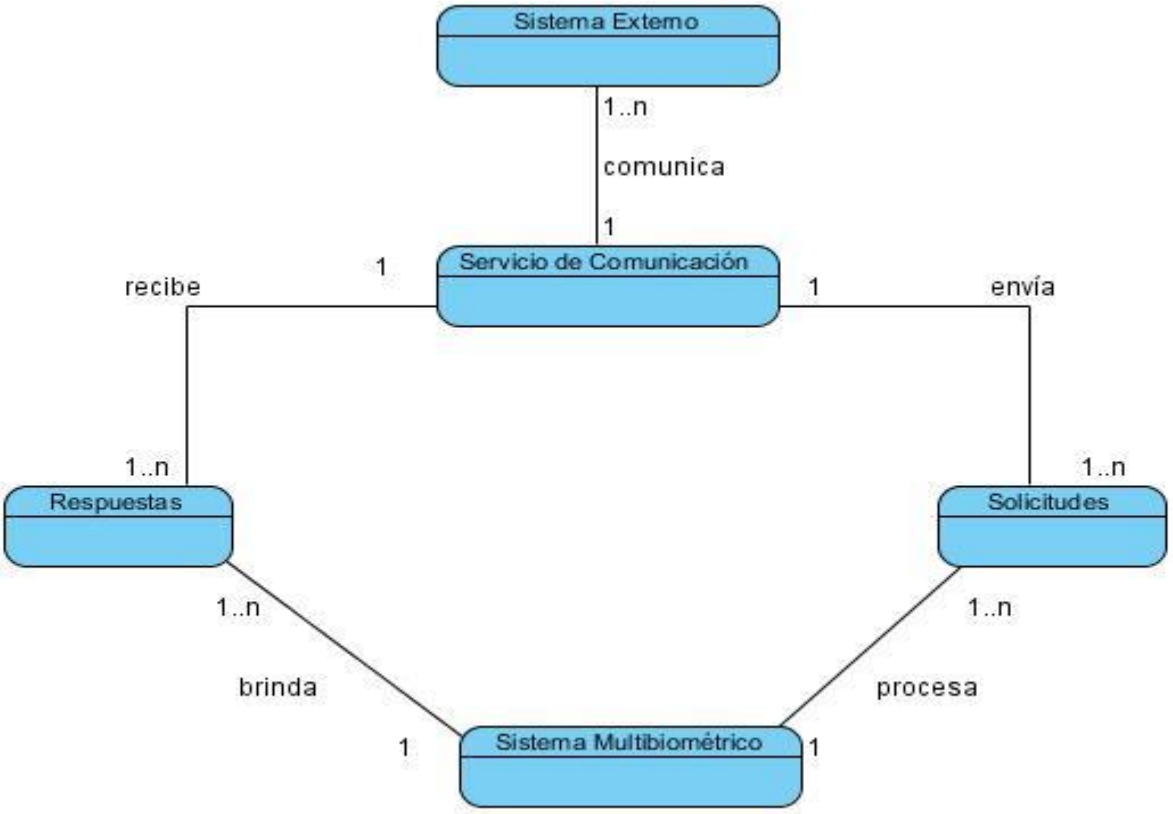


Figura 4: Modelo de dominio

2.3. Propuesta de solución

Para dar solución al problema planteado al inicio de la investigación se propone el desarrollo de un servicio de comunicación que tenga entre sus funciones principales el recibo y procesamiento de solicitudes biométricas desde un sistema externo, siendo capaz de deserializar los datos recibidos, enviarlos al sistema

multibiométrico, serializar los datos que este manda como respuesta y el posterior envío, a través de una comunicación asincrónica, hacia el sistema externo.

Para utilizar los servicios que brinda el sistema multibiométrico se necesita que los sistemas externos capturen los datos biométricos establecidos en el lugar donde se desea identificar y enviar estos en un fichero ANSI/NIST adjunto en un correo electrónico al buzón de correo del sistema multibiométrico, una vez recibida la solicitud se enviará de forma automática una respuesta de confirmación ACK (del inglés *acknowledgement*, acuse de recibo), el servicio de comunicación a desarrollar extraerá el adjunto del correo electrónico y enviará los datos al sistema multibiométrico para que comience a procesar la solicitud, evaluando la correspondencia entre los datos recibidos y los almacenados en la base de datos. Posteriormente dicho sistema generará una respuesta con la cual se conformará un fichero en formato ANSI/NIST que se enviará al sistema externo mediante el correo electrónico.

En la Figura 5 se representa el sistema que se quiere desarrollar.

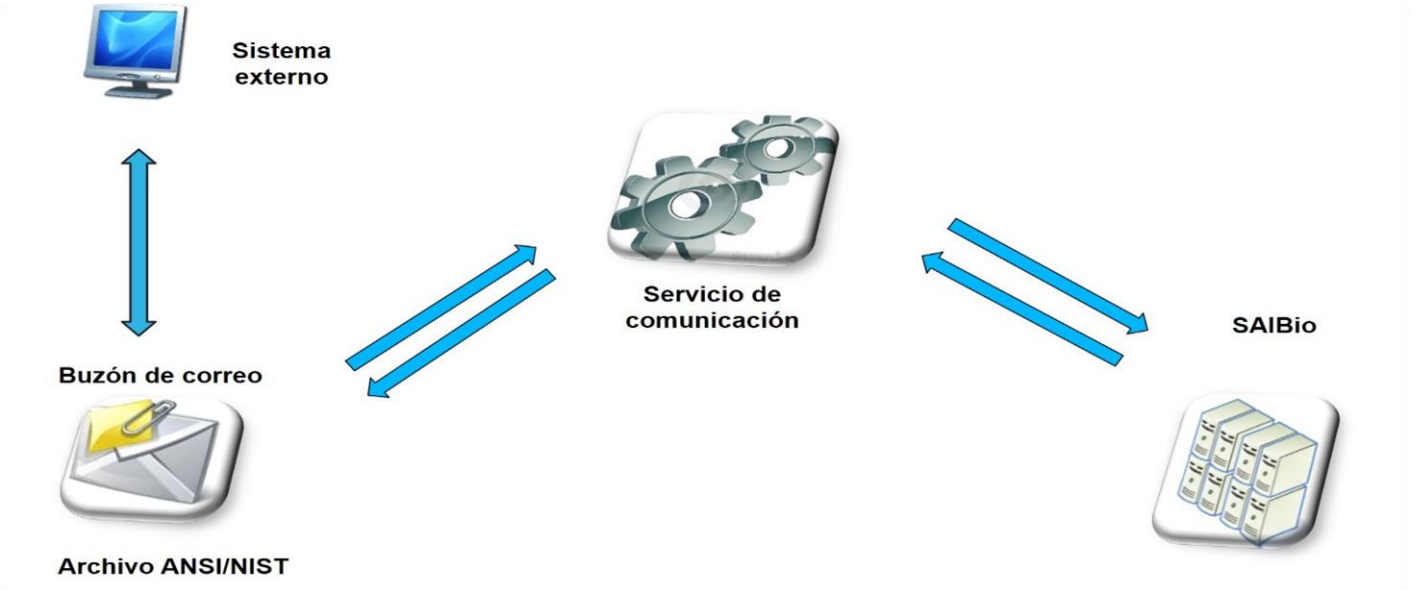


Figura 5: Procesos de la propuesta de solución

2.4. Captura de requisitos

2.4.1. Requisitos funcionales

Los requisitos funcionales definen el comportamiento interno del *software*, cálculos, detalles técnicos, manipulación de datos y otras funcionalidades específicas que muestran cómo se pondrá en práctica el

sistema a implementar. A continuación se muestran los requisitos funcionales que describen las funcionalidades que el componente debe cumplir:

RF-1 Recibir el mensaje del servidor de correos electrónicos.

RF-2 Extraer adjunto del correo electrónico.

RF-3 Interpretar fichero obtenido como un formato ANSI/NIST.

RF-4 Extraer los datos de la solicitud biométrica.

RF-5 Enviar solicitud biométrica al sistema multibiométrico mediante el servidor RabbitMQ.

RF-6 Obtener datos de la respuesta generada por el sistema multibiométrico.

RF-7 Obtener datos nominales de la persona.

RF-8 Elaborar paquete ANSI/NIST con los datos obtenidos.

RF-9 Codificar un mensaje electrónico con el fichero ANSI/NIST como adjunto.

RF-10 Transferir mensaje al servidor de correos electrónicos.

2.4.2. Requisitos no funcionales

Los requisitos no funcionales especifican los criterios que se deben usar para juzgar el funcionamiento de un sistema. En general, los requisitos no funcionales indican cómo un sistema debería ser.

Software

- El sistema operativo sobre el cual funcionará la aplicación es *Windows*.

Hardware

- 512 MB de memoria RAM como mínimo.
- Procesadores *Pentium* IV o superiores.

Requerimientos en el diseño y la implementación.

- Lenguaje de programación C#.
- *Visual Studio* como IDE de desarrollo.
- *Visual Paradigm* como herramienta CASE.

Fiabilidad

- La reparación del sistema en caso de surgir fallas en el mismo debe realizarse en el menor tiempo posible, poniendo todos los esfuerzos en función de que no supere las 24 horas.

Usabilidad

- Se debe lograr un diseño adaptable, con la capacidad de poder soportar funcionalidades adicionales o modificar las funcionalidades existentes.

Soporte

- La instalación del servicio debe ser sencilla y clara.
- Se realizarán pruebas al *software* una vez concluido para comprobar su funcionalidad.
- El servicio de comunicación permitirá implementar cambios, ya sea cualquier corrección, mejora o adaptación del *software*.

2.4.3. Historias de usuario (HU)

Las historias de usuario son la técnica utilizada en XP para especificar los requisitos del software. En ellas se describen brevemente las características que el sistema debe poseer. El tratamiento de las historias de usuario es muy dinámico y flexible. Cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas (25). Se emplean para hacer estimaciones de tiempo y para el plan de lanzamientos, y presiden la creación de las pruebas de aceptación. Deben tener como características fundamentales: ser independientes unas de otras, negociables, valoradas por los clientes o usuarios, estimables, pequeñas y verificables.

A continuación se describen dos historias de usuario generadas en la investigación. Las demás se pueden encontrar en el Anexo 1.

Historia de Usuario	
No. Hist: 1	**Usuario:** Carlos Iván García Delgado.
Nombre de historia: Recibir el mensaje del servidor de correos electrónicos.	
Prioridad en negocio: Muy alta	**Riesgo en desarrollo:** Alto
Puntos estimados: 2	**Iteración asignada:** 1
Programador responsable: Carlos Iván García Delgado.	
Descripción: Se establece la conexión con el servidor de correos y se ejecutan los comandos necesarios para recibir el mensaje electrónico.	
Observaciones: El mensaje debe contener un fichero adjunto con el formato ANSI/NIST.	

Tabla 1: HU_1

Historia de Usuario	
No. Hist: 6	**Usuario:** Alexander Lugones Aguiar.
Nombre de historia: Obtener datos de la respuesta generada por el Sistema Multibiométrico.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alto
Puntos estimados: 1	**Iteración asignada:** 2
Programador responsable: Alexander Lugones Aguiar.	
Descripción: Después de procesar la solicitud biométrica el sistema multibiométrico genera una respuesta positiva en caso de que los datos biométricos de la persona se encuentren en la base de datos o negativa en caso contrario y la coloca en el servidor RabbitMQ.	
Observaciones: El sistema multibiométrico debe garantizar que se coloque una respuesta en el servidor RabbitMQ.	

Tabla 2: HU_6

2.5. Planificación

2.5.1. Plan de entregas

En esta etapa se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente, estableciendo la prioridad de cada historia de usuario. (22)

La entrega del primer lanzamiento permite calcular la velocidad del proyecto que no es más que el número de historias de usuario que se han implementado, optimizando de esta manera las fechas de entrega de las distintas iteraciones (Tabla 3).

Entregable	Fin Iteración 1	Fin Iteración 2	Fin Iteración 3
Componente para la comunicación con el sistema multibiométrico a través de ficheros ANSI/NIST por correo electrónico.	Marzo 2014	Abril 2014	Mayo 2014

Tabla 3: Plan de entregas

2.5.2. Plan de iteraciones

Los elementos que deben tomarse en cuenta durante la elaboración del Plan de la Iteración (Tabla 4) son: historias de usuario no abordadas, velocidad del proyecto, pruebas de aceptación no superadas en la iteración anterior y tareas no terminadas en la iteración anterior. Todo el trabajo de la iteración es expresado en tareas de programación, cada una de ellas es asignada a un programador como responsable, pero llevadas a cabo por parejas de programadores. (22)

Iteración	Historia de Usuario	Semanas Estimadas
1	Recibir el mensaje del servidor de correos electrónicos.	5
	Extraer adjunto del correo electrónico.	
	Interpretar fichero obtenido como un formato ANSI/NIST.	
	Extraer los datos de la solicitud biométrica.	
2	Enviar solicitud biométrica al sistema multibiométrico a través del servidor RabbitMQ.	4
	Obtener datos de la respuesta generada por el sistema multibiométrico.	
	Obtener datos nominales de la persona.	
3	Elaborar paquete ANSI/NIST con los datos obtenidos.	4
	Codificar un mensaje electrónico con el fichero ANSI/NIST como adjunto.	
	Transferir mensaje al servidor de correos electrónicos.	

Tabla 4: Plan de iteraciones

Las Tareas de Ingeniería a realizarse en cada iteración se agrupan en la siguiente tabla (Tabla 5). En el Anexo 2 se especifican los detalles de cada una de las tareas.

Iteración	Historia de Usuario	Tarea de Ingeniería
	Recibir el mensaje del servidor de correos electrónicos.	-Abrir sesión POP3/IMAP con el servidor de correos electrónicos. -Enviar comandos al servidor de correos electrónicos.

1	Extraer adjunto del correo electrónico.	-Decodificar el mensaje electrónico. -Obtener archivo adjunto del mensaje de correos electrónicos.
	Interpretar fichero obtenido como un formato ANSI/NIST.	-Descifrar el adjunto como un fichero ANSI/NIST
	Extraer los datos de la solicitud biométrica.	-Extraer datos biométricos de la solicitud.
	Enviar solicitud biométrica al sistema multibiométrico mediante el servidor RabbitMQ.	-Conformar solicitud a enviar al sistema multibiométrico. -Colocar en el gestor de colas del Sistema Multibiométrico la solicitud a procesar.
2	Obtener datos de la respuesta generada por el sistema multibiométrico.	-Consultar identificador de la solicitud procesada por el sistema multibiométrico. -Obtener datos biométricos.
	Obtener datos nominales de la persona.	Extraer de la base de datos nominal la información de la persona identificada.
3	Elaborar paquete ANSI/NIST con los datos obtenidos.	-Conformar el archivo ANSI/NIST con la respuesta a enviar.
	Codificar un mensaje electrónico con el fichero ANSI/NIST como adjunto.	-Elaborar el mensaje de correo electrónico con el archivo ANSI/NIST adjunto.
	Transferir mensaje al servidor de correos electrónicos.	-Iniciar sesión SMTP. -Enviar comandos al servidor de correos electrónicos.

Tabla 5: Tareas de ingeniería

2.6. Diseño

El diseño del *software* es el proceso de aplicar distintas técnicas y principios con el propósito de definir un producto con los suficientes detalles como para permitir su realización. Constituye la primera etapa técnica del proceso de ingeniería del software y el punto de partida para realizar las actividades de implementación traduciendo los requisitos funcionales en una representación lógica del software a desarrollar.

2.6.1. Descripción de la arquitectura

Según Bass L, Clements P y Kazman R en su libro "*Software Architecture in Practice"* (36) (Arquitectura de *Software* en Práctica)*:* "*la arquitectura de software es la estructura o las estructuras del sistema que incluyen los componentes del software, las propiedades visibles externamente de esos componentes y las relaciones entre ellos."*

Es decir la arquitectura de *software* brinda una visión global del sistema permitiendo analizar su efectividad para cumplir con los requisitos establecidos. Es la encargada de establecer la comunicación entre las partes involucradas y constituye un modelo pequeño pero a la vez comprensible de cómo está estructurado el sistema y como trabajan juntos sus componentes.

Para desarrollar el Servicio de Comunicación se propone el uso de una arquitectura en capas basada en *plugins* (Figura 6)*,* la cual se describe a continuación:

- **Capa de presentación:** Contiene el panel de control que representa una interfaz para la configuración del servicio y los *plugins* de la aplicación, permitiendo además detener, iniciar y reiniciar los procesos de la capa de servicio.
- **Capa de servicio:** Capa que integra el servicio de *Windows* y el *Core* o núcleo de la aplicación, encargado de manejar el funcionamiento de la misma, la integración entre los componentes y el manejo de las dependencias, permitiendo de esta manera la extensibilidad de las funcionalidades.
- **Capa *plugins*:** Está conformada por los *plugins* implementados, que representan la lógica del negocio y se encargan de realizar las acciones necesarias para lograr el correcto funcionamiento de la aplicación y gracias a la cual es posible extender las funcionalidades del sistema sin afectar los demás componentes una vez desplegado.

Entre las principales ventajas de la arquitectura en capas se encuentra que se puede llevar a cabo en varios niveles y, en caso de que sobrevenga algún cambio, solo se modifica el nivel requerido, sin tener que revisar todo el código. Permite distribuir el trabajo de creación de una aplicación por niveles; de este modo, cada grupo de trabajo está totalmente abstraído del resto de los niveles, de forma que basta con conocer la API que existe entre los mismos.

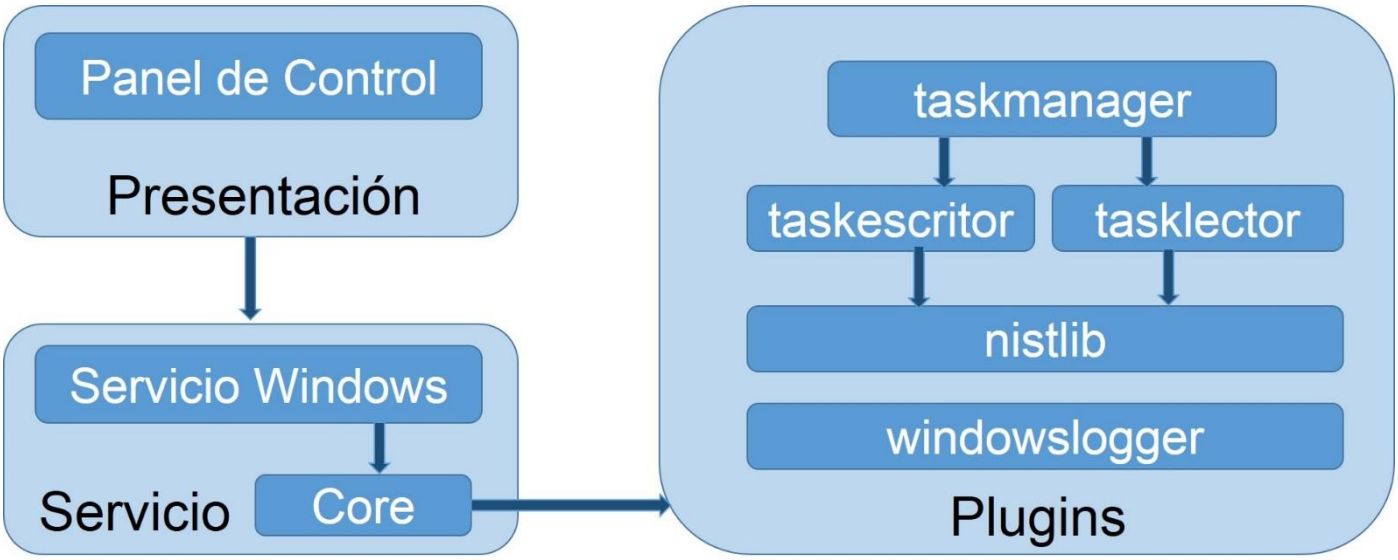


Figura 6: Propuesta de arquitectura del componente

2.6.2. Patrones de diseño

Un patrón de diseño es una solución probada que se puede aplicar con éxito a un determinado tipo de problemas que aparece repetidamente en el desarrollo de sistemas de *software* (37). Se encaminan hacia una línea de esqueleto básico que cada desarrollador luego adapta a sus necesidades y a las peculiares características de su aplicación sus características fundamentales son:

- Son soluciones concretas. Proponen soluciones a problemas concretos, no son teorías genéricas.
- Son soluciones técnicas. Indican resoluciones técnicas basadas en Programación Orientada a Objetos (POO). En ocasiones tienen más utilidad con algunos lenguajes de programación y en otras son aplicables a cualquier lenguaje.
- Se utilizan en situaciones frecuentes. Esto se debe a que se basan en la experiencia acumulada al resolver problemas reiterativos.

- **Patrones GRASP**

Los patrones GRASP (Patrones Generales de *Software* para Asignar Responsabilidades) describen los principios fundamentales de la asignación de responsabilidades a objetos (38). Son una serie de buenas prácticas recomendables en el diseño de *software*.

Seguidamente se muestran los patrones GRASP utilizados durante el diseño.

Patrón Bajo acoplamiento

El bajo acoplamiento estimula tener las clases lo menos ligadas entre sí, de tal forma que, en caso de producirse una modificación en alguna de ellas, tenga la mínima repercusión posible en el resto de las clases, potenciando la reutilización y disminuyendo la dependencia entre las clases (39). En el servicio de comunicación del sistema multibiométrico se emplea este patrón tratando de minimizar, dentro de lo posible, las relaciones entre las clases, sin que esto afecte el funcionamiento correcto de la aplicación.

Patrón Alta cohesión

En este patrón la información que almacena una clase debe ser coherente y debe estar, en la medida de lo posible, relacionada con la clase. Básicamente asigna una responsabilidad de modo que la cohesión siga siendo alta. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme (39). En el desarrollo de la investigación se utiliza este patrón en la clase ***ScheduleTaskLector.cs*** (Figura 7) la cual realiza las tareas asignadas de acuerdo al objetivo con que fue diseñada, logrando que la misma contenga solo la información necesaria para la realización de estas; garantizando de esta manera la sencillez de uso y la reutilización de código.

Figura 7: Patrón Alta cohesión

Patrón Controlador

El patrón Controlador asigna la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase que represente el sistema global. Ofrece una guía para tomar decisiones apropiadas que generalmente se aceptan (39). Sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es la que recibe los datos del usuario y la que los envía a las distintas clases según el método llamado. En el servicio de comunicación para el sistema multibiométrico se emplea este patrón en las relaciones de la clase ***ScheduleTaskManager.cs*** con las clases ***ScheduleTaskLector.cs*** y

ScheduleTaskEscritor.cs dado que la primera cumple la función de controlador de las acciones de estas últimas.

- **Patrones GOF**

Según Erich Gamma, Richard Helm, Ralph Jonson y John Vlissides en el libro "*Design Pattern Elements of Object-Oriented Reusable Software"* (40) los patrones GOF (pandilla de los cuatro, siglas en inglés) son una forma indispensable de enfrentarse a la programación para buscar solución a un problema determinado. Se encuentran organizados en tres categorías:

- Patrones creacionales
- Patrones estructurales
- Patrones de comportamiento

A continuación se describe el patrón GOF utilizado.

Patrón creacional Singleton

En el servicio de comunicación para el sistema multibiométrico se emplea dicho patrón en la clase que precisamente lleva por nombre ***Singleton.cs*** (Figura 8) la cual permite crear una instancia única de un objeto dado su tipo, en el Anexo 3 (Figura 14), se puede ver ejemplificado el uso de este patrón en la solución.

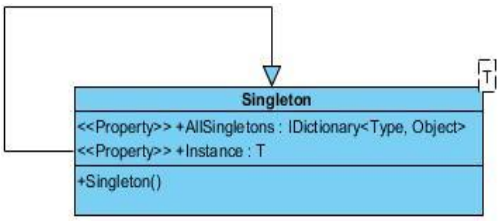


Figura 8: Patrón *Singleton*

- **Otros patrones de diseño utilizados**

Patrón Inversión de Control

El patrón Inversión de Control es un método de programación en el que el flujo de ejecución de un programa se invierte respecto a los métodos de programación tradicionales. Se especifican respuestas deseadas a sucesos o solicitudes de datos concretas, dejando que algún tipo de entidad o arquitectura externa lleve a cabo las acciones de control que se requieran en el orden necesario y para el conjunto de sucesos que tengan que ocurrir. (41)

El control de la aplicación se invierte cuando es el núcleo el que ejecuta las funcionalidades implementadas en los *plugins* en lugar de estos últimos ejecutar las implementadas en el núcleo. En el servicio de

comunicación para el sistema multibiométrico este patrón se emplea mediante el uso del contenedor Autofac el cual implementa este patrón usando la inyección de dependencias.

Patrón Inyección de dependencias

La inyección de dependencias o DI por sus siglas en inglés, es comúnmente utilizada en varios patrones de diseño orientado a objetos, consiste en inyectar comportamiento a componentes. Esto no es más que extraer responsabilidades a un componente para delegarlas en otro, estableciendo un mecanismo a través del cual el nuevo componente pueda ser cambiado en tiempo de ejecución (42). Este concepto básicamente hace que una clase A inyecte objetos en una clase B en lugar de dejar que sea la propia clase B la que se encargue de crear el objeto. La inyección ocurre cuando se desea resolver una instancia de una clase o servicio registrado en el contenedor de Autofac, que depende a la vez de otros servicios registrados en este contenedor de dependencias. En el servicio de comunicación para el sistema multibiométrico este patrón se evidencia en la clase ***WindowLogger.cs*** cuyo comportamiento es inyectado en las clases ***ScheduleTaskLector.cs*** y ***ScheduleTaskEscritor.cs*** quitando la responsabilidad de crear la instancia del objeto a estas últimas, lo que se puede ver reflejado en el Anexo 3, (Figura 15).

2.6.3. Diagrama de clases del diseño

Los diagramas de clases describen la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos. Son los diagramas más comunes en el modelado de sistemas orientados a objetos.

A continuación se presenta el diagrama de clases de la presente investigación (Figura 9).

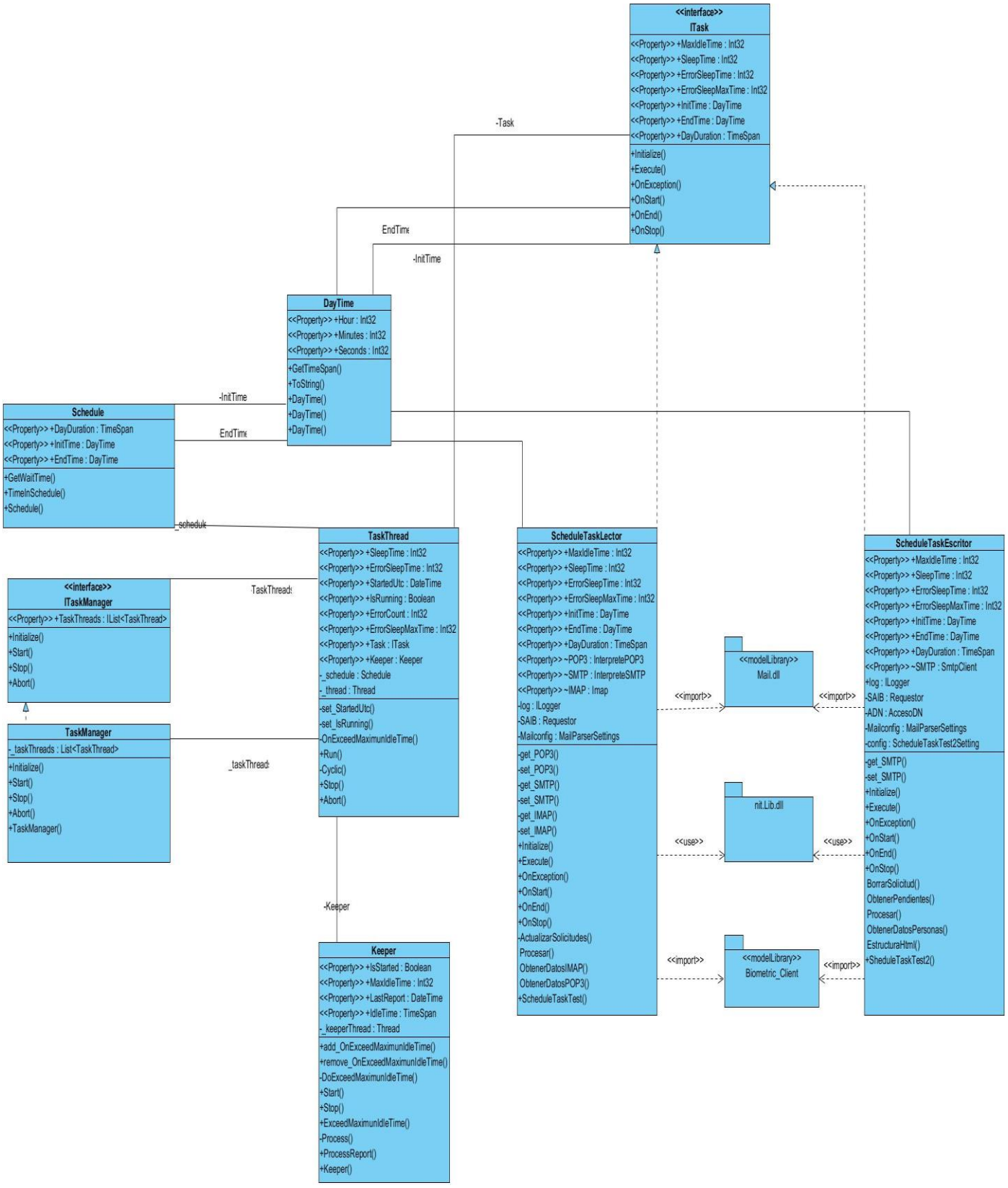


Figura 9: Diagrama de Clases

2.6.4. Tarjetas CRC

La utilización de tarjetas CRC (Clase-Responsabilidad-Colaboración) es una técnica de diseño orientado a objetos propuesta por Kent Beck y Ward Cunningham y constituye uno de los artefactos que genera la metodología XP. El objetivo de la misma es hacer, mediante tarjetas, un inventario de las clases que se necesitan para implementar el sistema y la forma en que interactúan, facilitando el análisis y discusión de las mismas.

A continuación se muestran las tarjetas CRC para las clases *TaskThread* (Tabla 6) y *TaskManager* (Tabla 7), las demás se encuentran en el Anexo 4.

Clase *TaskThread*	
Responsabilidades	**Colaboradores**
Iniciar una nueva tarea.	----------
Correr el programa mientras los parámetros de tiempo configurados lo permitan. (ciclo)	*Schedule, Keeper*
Detener una tarea si existe algún error.	----------
Abortar la tarea.	----------

Tabla 6: Tarjeta CRC *TaskThread*

Clase TaskManager	
Responsabilidades	**Colaboradores**
Crear lista de instancias de la interfaz ITask.	*EngineContext, ITypeFinder, Schedule, Keeper, TaskThread.*
Añadir un hilo de ejecución.	*EngineContext, ITypeFinder, Schedule, Keeper, TaskThread.*
Iniciar un hilo de ejecución.	*EngineContext, ITypeFinder, Schedule, Keeper, TaskThread.*
Detener un hilo de ejecución.	*EngineContext, ITypeFinder, Schedule, Keeper, TaskThread.*
Abortar un hilo de ejecución.	*EngineContext, ITypeFinder, Schedule, Keeper, TaskThread.*
Obtener una lista de hilos de ejecución	*EngineContext, ITypeFinder, Schedule, Keeper, TaskThread.*

Tabla 7: Tarjeta CRC *TaskManager*

Conclusiones parciales

El desarrollo del modelo de dominio permitió visualizar los conceptos más importantes asociados a la investigación, así como las relaciones entre ellos. Se realizó una propuesta de solución que posibilitó capturar los requisitos funcionales y no funcionales del sistema. Las historias de usuario generadas permitieron, entre otros elementos, realizar una planificación real de las iteraciones a llevar a cabo.

La definición de la arquitectura a utilizar, el empleo de distintos patrones GRASP y GOF, así como la creación del diagrama de clases, permitieron tener la base necesaria para comenzar la implementación del sistema. La creación de las tarjetas CRC indicó las responsabilidades y colaboradores que poseen cada clase.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBAS

Introducción

En el siguiente capítulo se muestran las particularidades necesarias para describir el proceso de la implementación. Para ello se definen los estándares de codificación a tener en cuenta por el equipo de desarrollo. Se diseñan diferentes diagramas que ayudan a explicar el funcionamiento de la aplicación desarrollada, además de presentarse las interfaces de usuario creadas para la administración del servicio de comunicación. Para comprobar el cumplimiento de los requisitos funcionales se realizan pruebas unitarias, de aceptación y pruebas de carga.

3.1. Codificación

Después de concluir la planificación y el diseño del sistema se pasa a la fase de codificación con el objetivo de tener una entrega real del producto, e ir avanzando de forma iterativa e incremental hacia la solución definitiva.

Un concepto clave que propone la metodología XP en esta fase es la programación en pareja. Recomienda que dos personas trabajen juntas en una estación de trabajo para crear el código de una historia de usuario. Esto genera un mecanismo para enfrentar un problema determinado y asegura mejor calidad en el producto final.

3.1.1. Estándares de codificación

XP enfatiza en la comunicación de los programadores a través del código, con lo cual es indispensable que se sigan ciertos estándares de programación para mantener el código legible entre los miembros del equipo de desarrollo facilitando de esta manera los cambios que se puedan presentar.

En la propuesta de solución se tendrá en cuenta los siguientes estándares de codificación:

- El nombre que se le otorgue a una variable, método o clase debe expresar el propósito con que fue creado dicho elemento.
- Se utilizará para los nombres de los métodos y las clases el estilo Pascal de mayúsculas y minúsculas donde la primera letra en el identificador y la primera letra de cada subsiguiente palabra concatenada se capitalizan, ejemplo la clase ***ScheduleTaskManager.cs***.
- Los nombres de variables lógicas deben contener *Is*, lo que implicaría valores del tipo *Yes/No True/False*, como por ejemplo *_isStarted*.
- Se escribe únicamente una declaración por línea.

- Se escribe únicamente una instrucción por línea.
- A los nombres de las funciones se le añadirá una descripción del valor que vaya a ser devuelto, como por ejemplo el método *MaxIdleTime* que retorna el tiempo máximo de inactividad de un hilo de ejecución.

3.1.2. Diagrama de componentes

Un diagrama de componentes representa cómo un sistema de software es dividido en componentes y muestra las dependencias entre ellos. Está compuesto por simples archivos, paquetes, bibliotecas cargadas dinámicamente, bases de datos, etc. Al igual que ocurre con las clases, los componentes pueden ofrecer una interfaz, para que otros componentes puedan realizar las operaciones que esta ofrece. (43)

El diseño de la arquitectura en tres capas, la generación del diagrama de clases y la codificación de la solución propuesta, han facilitado identificar distintos elementos que funcionan de manera independiente y que se relacionan entre sí. A continuación (Figura 10) se muestra el diagrama de componentes definido en la investigación.

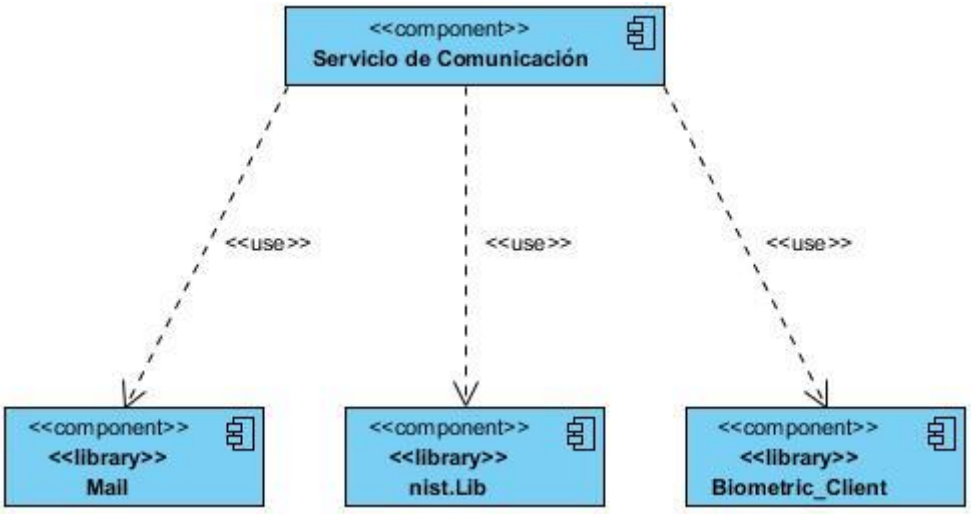


Figura 10: Diagrama de componentes

***Mail*:** Biblioteca encargada de establecer la comunicación con el servidor de correo electrónico.

nist.Lib: Biblioteca encargada de deserializar y serializar los archivos ANSI/NIST

***Biometric_Client*:** Biblioteca encargada de la comunicación entre el servicio de comunicación y el SAIBio.

3.1.3. Diagrama de despliegue

Los diagramas de despliegues muestran las relaciones físicas entre los componentes físicos y lógicos del sistema final. Presentan como características fundamentales:

- Los elementos usados por este tipo de diagrama son nodos (representados como un prisma), componentes (representados como una caja rectangular con dos protuberancias del lado izquierdo) y asociaciones.
- Representa los artefactos del sistema como nodos, los cuales son conectados a través de caminos de comunicación para crear redes de sistemas de complejidad arbitraria.
- Describe la arquitectura en tiempo de ejecución de procesadores, dispositivos y los componentes de *software* que ejecutan esta arquitectura.
- Describe una topología del sistema, estructura del *hardware* y el *software* que se ejecuta en cada unidad. (43)

En el servicio de comunicación para el sistema multibiométrico el diagrama de despliegue queda como se muestra en la (Figura11).

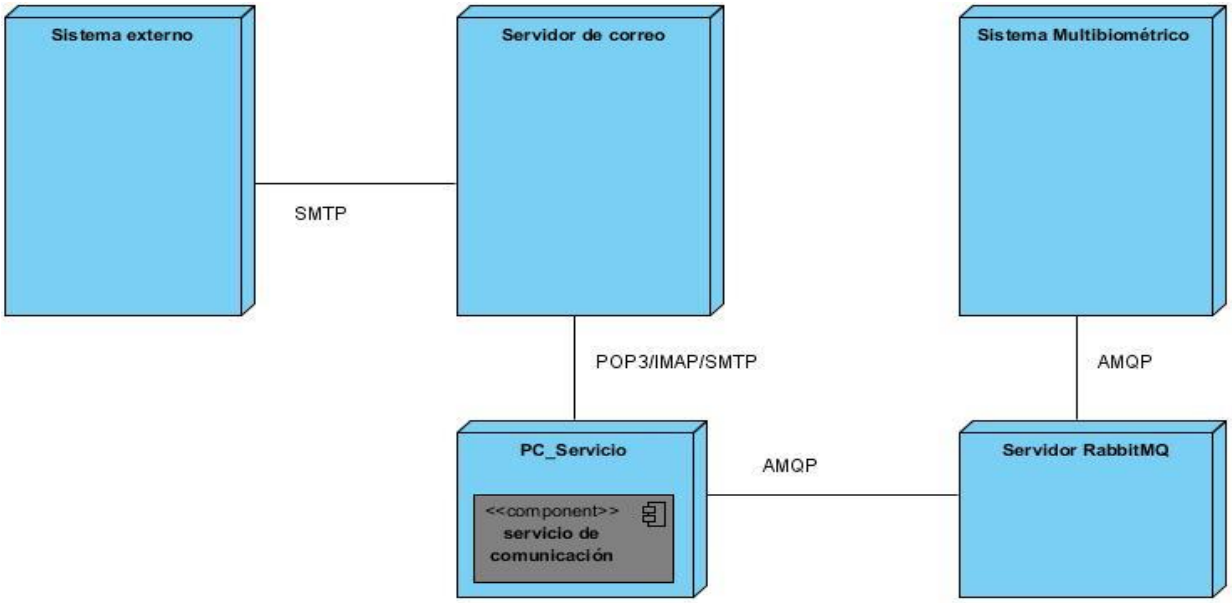


Figura 11: Diagrama de despliegue

El sistema externo enviará una solicitud a un buzón de correo electrónico. El servicio de comunicación monitorizará la llegada de nuevas solicitudes, las cuales se colocarán en la cola del servidor RabbitMQ. Luego estas solicitudes son procesadas por el servidor multibiométrico, que envía la respuesta al servidor RabbitMQ y de este gestor de colas, el servicio de comunicación estará leyendo, para crear el mensaje y enviarlo a la dirección de correo desde donde fue recibida la solicitud.

3.2. Tratamiento de fallos

Todos los sistemas informáticos pueden fallar y son los desarrolladores del sistema los que tienen la responsabilidad de planificar las consecuencias de posibles fallos. En el caso específico del servicio de comunicación con el Sistema Multibiométrico se debe tener en cuenta que lo que se transporta son datos de identificación por lo que elaborar una correcta estrategia de tratamientos de fallos resulta de especial interés.

Detección y recuperación de fallos

Para corregir un fallo, primeramente es necesario conocer que está ocurriendo un evento opuesto al normal funcionamiento del sistema. De esta forma el servicio de comunicación cuenta con una clase llamada *Keeper*, que gestiona los tiempos de cada tarea que se está ejecutando en ese momento. Uno de sus atributos es *MaxIdleTime*, una propiedad editable que tiene la responsabilidad de otorgar un máximo de tiempo a cada tarea en ejecución.

De forma general la tarea estará tratando de ejecutarse mientras el tiempo transcurrido desde el inicio de la misma no supere el valor de la propiedad *MaxIdleTime*, una vez superado este, se detiene la tarea por un lapso determinado, definido por otro atributo llamado *SleepTime* que representa un tiempo de espera para la tarea en cuestión. Una vez concluido ese tiempo de inactividad se vuelve a cargar la tarea de forma automática y se inicia el mismo proceso. De esta manera el sistema obtiene una estrategia en caso de que exista un fallo en la red o que el servidor RabbitMQ o el de correo electrónico no respondan.

3.3. Pruebas

El instrumento más adecuado para determinar el estado de la calidad de un producto de *software* es el proceso de pruebas. En este proceso se ejecutan pruebas dirigidas a componentes del *software* o al sistema de *software* en su totalidad, con el objetivo de medir el grado en que el software cumple con los requerimientos. (44)

3.3.1. Pruebas unitarias

Las pruebas unitarias son una forma de probar el correcto funcionamiento de un módulo de código, asegurando que cada parte independiente de un sistema funcione de manera esperada. Se realizan de manera sistemática durante la implementación de todo el proyecto, una vez que se termine un determinado módulo y hasta que su efectividad no sea la máxima, el código no podrá integrarse al sistema. Las pruebas unitarias brindan una inmediata retroalimentación en la realización del trabajo, permitiendo al programador saber si una determinada funcionalidad se puede agregar al sistema existente sin alterar el funcionamiento

actual del mismo. Se sugiere que el uso de las pruebas unitarias sea de forma automática de modo que se puedan ejecutar en reiteradas ocasiones y de una manera fácil. (45)

Para realizar las pruebas al servicio de comunicación se utilizó el *ReScharper* (R#), un *Add-in* de Visual Studio que permite detectar errores y propone posibles soluciones a los mismos.

Los resultados de estas pruebas pueden consultarse en el Anexo 5.

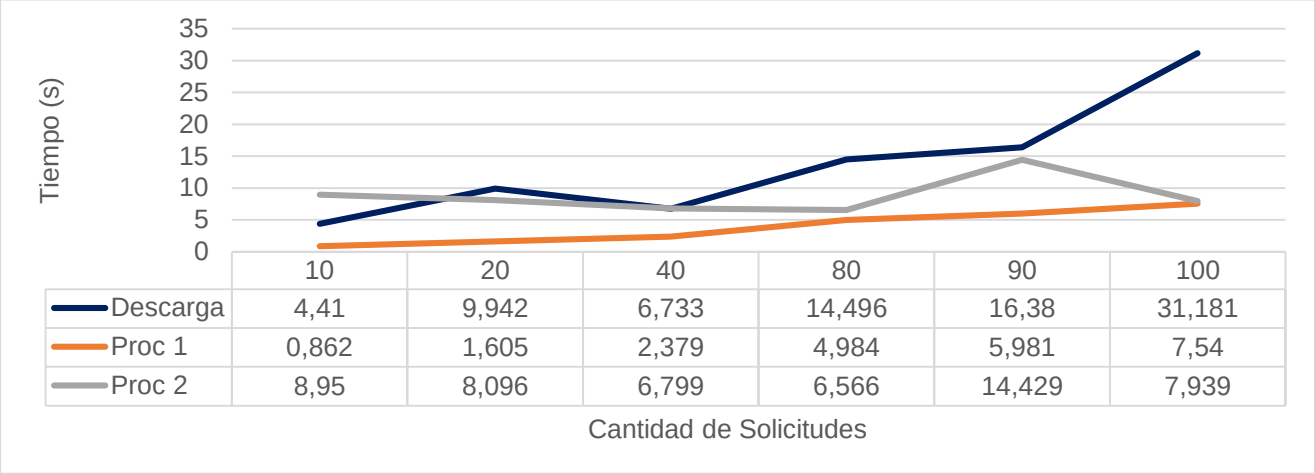
3.3.2. Pruebas de carga

El rendimiento es una de las características más importantes de los sistemas informáticos modernos. Las pruebas de carga deben realizarse siempre antes de que el sistema entre en explotación. Esta prueba reduce la probabilidad de que un sistema funcione de manera incorrecta cuando empiece a utilizarse. Sin este tipo de pruebas es imposible saber qué ocurre cuando la aplicación tiene que funcionar bajo cargas elevadas. (46)

Específicamente el componente desarrollado puede estar sometido a varias peticiones de solicitud biométrica que un sistema externo en un momento determinado haya realizado. El servicio de comunicación, en el menor tiempo posible, debe tener la capacidad de atender todas las solicitudes recibidas y enviar las respuestas generadas por el sistema multibiométrico nuevamente al sistema externo. El gráfico que se muestra a continuación (Figura 12) expone el tiempo empleado en cada una de las etapas del procesamiento de una solicitud.

En el gráfico se encuentran representados los tres procesos fundamentales llevados a cabo por el servicio: la descarga, donde se obtienen los adjuntos contenidos en los mensajes de correo electrónico, el segundo (Proc 1) representa el tiempo de escritura de las nuevas solicitudes en la cola del RabbitMQ y por último (Proc 2) la recepción de la respuesta enviada por el SAIBio, conformación del mensaje con la respuesta en formato ANSI/NIST y envío de este al servidor del cual se recibió la solicitud.

Estos tiempos están condicionados por la infraestructura de la red y la capacidad de los servidores y estaciones de trabajo donde se despliegue la aplicación. Para la realización de las pruebas se utilizó como servidor de correo el de la UCI y como estación de trabajo en la cual se ejecutó el servicio, una computadora con las siguientes características: 2GB RAM DDR-2 y Microprocesador Intel *Core Quad* a 2.10 GHz.



	10	20	40	80	90	100
Descarga	4,41	9,942	6,733	14,496	16,38	31,181
Proc 1	0,862	1,605	2,379	4,984	5,981	7,54
Proc 2	8,95	8,096	6,799	6,566	14,429	7,939

Figura 12: Pruebas de carga

3.3.3. Pruebas de aceptación

Las pruebas de aceptación de la programación extrema, las especifica el cliente y se enfocan en las características generales y la funcionalidad del sistema, elementos visibles y revisables por el cliente. Se derivan de las historias de usuario que se han implementado como parte del lanzamiento del software (23). Las pruebas de aceptación es el último tipo de pruebas que debe superar una aplicación dentro de un plan. Una vez que ya se ha probado que cada módulo funciona bien por separado, que la aplicación puede utilizarse bajo condiciones de operación extremas, que todos los módulos se integran correctamente y que el software ofrezca las funciones esperadas, llega el momento de que el cliente logre probar su producto final.

Específamente al servicio de comunicación desarrollado se le aplicaron dos tipos de pruebas de aceptación: las pruebas alfas y las pruebas betas. Las pruebas alfas se le realizaron al componente en la primera y segunda iteración, fase en la que el sistema no estaba bien conformado y trabajaba de manera inestable. Las pruebas betas se le realizaron en la tercera iteración, cuando se obtuvo la primera versión del producto completamente funcional y se requería de la integración con el sistema multibiométrico.

A continuación se muestra el Caso de Prueba de Aceptación correspondiente a la historia de usuario Recibir el mensaje del servidor de correos electrónicos (Tabla 8). Los demás casos de prueba se encuentran en el Anexo 6 del documento.

Caso de Prueba de Aceptación	
Código de caso de prueba: HU1_CP1	**Nombre de la historia de usuario:** Recibir el mensaje del servidor de correos electrónicos.

Responsable de la prueba: Carlos Iván García Delgado.
Descripción de la prueba: Prueba de funcionalidad para verificar que se ha recibido correctamente el correo electrónico.
Condiciones de ejecución: El sistema externo debe enviar una petición de solicitud biométrica al sistema multibiométrico
Entrada/Pasos de ejecución: • Establecer sesión POP3/IMAP con el servidor de correos electrónicos. • Enviar comandos del protocolo POP3/IMAP al servidor de correos electrónicos.
Resultado esperado: Se recibe el correo electrónico
Evaluación de la prueba: Prueba satisfactoria

Tabla 8: Caso de prueba de aceptación HU1_CP1

En las pruebas de aceptación después de definir con el cliente los 10 casos de prueba que tendría la aplicación en un total de 3 iteraciones (Figura 13) se detectaron 10, 6 y 2 no conformidades respectivamente para un total de 18 no conformidades que en su mayoría correspondían a errores de baja responsabilidad en el código del sistema. Todas estas dificultades fueron corregidas en un tiempo máximo de 8 días. Posteriormente se realizó una nueva iteración con el objetivo de detectar alguna no conformidad, esta iteración arrojó como resultado cero no conformidades por lo que evidenció que el servicio de comunicación cumple con las funcionalidades definidas.

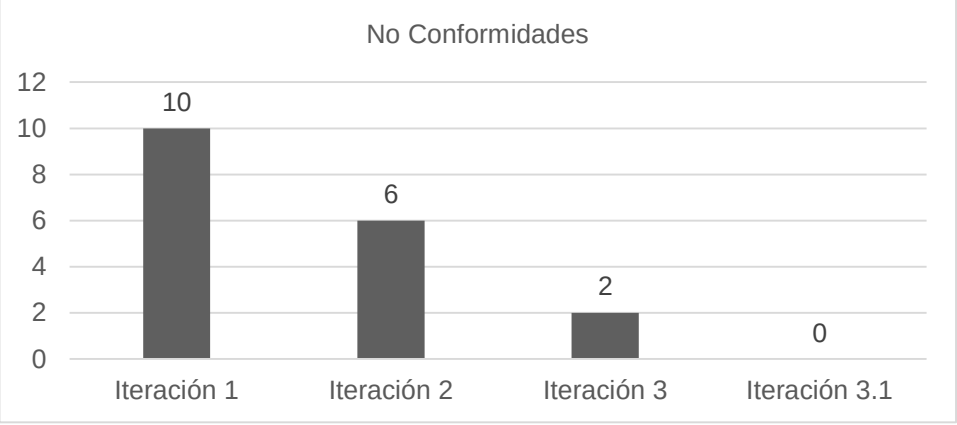


Figura 13: Resultado de las pruebas

Conclusiones parciales

La utilización de los diferentes estándares de codificación facilita el uso del componente desarrollado en otras soluciones. Con la realización del diagrama de componentes y el diagrama de despliegue se obtuvo

una representación de la estructura del sistema y el entorno donde estará trabajando. La realización de pruebas al sistema validó y verificó las funcionalidades desarrolladas.

CONCLUSIONES

- El uso del estándar ANSI/NIST ITL 1-2011 en la solución desarrollada garantiza el envío de varios patrones biométricos en un mismo formato.
- La definición de una arquitectura en capas basada en *plugins* permitirá que el componente desarrollado pueda ser utilizado para posteriores soluciones de comunicación entre servidores biométricos y multibiométricos que estén ubicados en distintos ambientes de red.
- Se desarrolló el servicio de comunicación y se integró al sistema multibiométrico.
- Se validó la aplicación mediante la realización de pruebas unitarias al código utilizado y pruebas de aceptación que lograron satisfacer las expectativas del cliente.
- La realización de pruebas de carga logró validar el rendimiento de la aplicación evidenciando su capacidad de atender grandes volúmenes de solicitudes biométricas en un tiempo relativamente corto.

RECOMENDACIONES

- Establecer nuevos mecanismos de seguridad y recuperación ante errores que mejoren la confidencialidad e integridad de los datos que se transmiten.
- Continuar incrementando la cantidad de registros que el servicio de comunicación puede procesar.

GLOSARIO DE TÉRMINOS

- **AFIS:** Sistema Automatizado de Identificación por Huellas Dactilares.
- **ANSI:** Instituto Nacional Americano de Estándares.
- **CISED:** Centro de Identificación y Seguridad Digital.
- **Consorcio:** Asociación económica en la que una serie de empresas buscan desarrollar una actividad conjunta.
- **Cuantización:** Proceso para construir una teoría cuántica que explica la comprensión clásica de un fenómeno físico.
- **IMAP:** Protocolo de Acceso a Mensajes de Internet.
- **ITL:** Laboratorio de Tecnologías de la Información.
- **NIST:** Instituto Nacional de Normas y Tecnología.
- **Ondeletas:** Tipo especial de transformada de Fourier que representa una señal en términos de versiones trasladadas y dilatadas de una onda finita.
- ***Plugin*:** Complemento que puede anexarse a un software para aumentar sus funcionalidades (generalmente sin afectar otras funciones ni afectar la aplicación principal).
- **POP3:** Protocolo de Oficina de Correos.
- **ppmm:** Píxeles por milímetro.
- **SMTP:** Protocolo Simple de Transmisión de Correo.
- **TIC:** Tecnologías de la Información y las Comunicaciones.

BIBLIOGRAFÍA REFERENCIADA

1. **César Tolosa Borja, Álvaro Giz Bueno.** Universidad de Castilla-La Mancha. [En línea] [Citado el: 30 de noviembre de 2013.] http://www.dsi.uclm.es/personal/MiguelFGraciani/mikicurri/Docencia/Bioinformatica/web_BIO/Documentacion/Trabajos/Biometria/Trabajo%20Biometria.pdf.

2. **HOMINI.** Plataforma Biométrica Homini. [En línea] Homini, 2004. [Citado el: 23 de octubre de 2013.] http://www.homini.com/new_page_5.htm.

3. **ICR SL.** IdosE Ingeniería, Informática y Electrónica. [En línea] 2009. [Citado el: 12 de noviembre de 2013.] http://www.idose.es/biometria.

4. **Universidad Nacional de Entre Ríos.** Buenas tareas. [En línea] agosto de 2011. [Citado el: 11 de diciembre de 2013.] http://www.buenastareas.com/ensayos/Desarrollo-De-Un-Sistema-Multi-Biom%C3%A9trico-Mediante/2594627.html.

5. **Landi, Juan Carlos.** Universidad Politécnica Salesiana. [En línea] 2004. [Citado el: 30 de octubre de 2013.] http://dspace.ups.edu.ec/bitstream/.

6. **UNAM.** Facultad de Ingeniería Biométrica Informática. [En línea] [Citado el: 13 de noviembre de 2013.] http://redyseguridad.fi-p.unam.mx/proyectos/biometria/estandares/estandar.html.

7. **ScienceDaily.** ScienceDaily. [En línea] 12 de diciembre de 2011. [Citado el: 15 de diciembre de 2013.] http://www.sciencedaily.com/releases/2011/12/111207105433.htm.

8. **National Institute of Standards and Technology.** *ANSI/NIST-ITL 1-2011 Data Format for the Interchange of Fingerprint, Facial & Other Biometric Information.* 2011.

9. **Arburola Valverde, Dr. Allan.** MailxMail.com. [En línea] 11 de 08 de 2008. [Citado el: 04 de febrero de 2014.] http://www.mailxmail.com/curso-dactiloscopia-disciplina-criminalistica/tipos-huellas-dactilares.

10. **Definición abc.** Definición abc. [En línea] 2007. [Citado el: 10 de diciembre de 2013.] http://www.definicionabc.com/tecnologia/correo-electronico.php.

11. **Díaz de Cossío, Roger y Cerón Roa, Armando.** Curso Básico del uso de la computadora e Internet. [En línea] julio de 2004. [Citado el: 19 de enero de 2014.]

http://inepja.inea.gob.mx/cursos/computacion/cursocomputo/cursocomputomac/correo/Correo_1.htm.

12. **Universitat Jaume I de Castelló(UJI).** Tema 4 Correo Electrónico. [En línea] [Citado el: 25 de noviembre de 2013.] www3.uji.es/~huerta/j11/tema4Email.pdf.

13. **León Carri, Carolina.** www-2.dc.uba.ar. [En línea] junio de 2006. [Citado el: 11 de enero de 2014.] www-2.dc.uba.ar/materias/tc/downloads/apuntes/smtp_pop_imap.pdf.

14. **MX Vanger.** Avangermx. [En línea] 12 de julio de 2011. [Citado el: 2 de diciembre de 2013.] http://www.vangermx.com/2011/07/diferencias-ventajas-y-desventajas-entre-imap-y-pop/.

15. **Alonso Gómez, Jairo.** INBIOSYS Biometria. [En línea] INBIOSYS, 16 de septiembre de 2009. [Citado el: 2 de diciembre de 2013.] http://inbiosys.wordpress.com/2009/09/16/historia-de-la-biometria/.

16. **National Institute of Standards and Tecnology.** National Institute of Standards and Tecnology. [En línea] 23 de noviembre de 2011. [Citado el: 14 de enero de 2014.] http://www.nist.gov/itl/iad/ig/ansi_standard-history.cfm.

17. **AWARE.** Aware. [En línea] 2008. [Citado el: 4 de noviembre de 2013.] http://www.aware.com/es/biometrics/nistpack.html.

18. **Rodríguez Grille, Yilian y Verdecia Viltres, Yunierki .** *Servicios de comunicación de sistemas clientes con el AFIS Civil de SAIME, en software libre.* La Habana : s.n., 2008.

19. **Safran Morpho.** Iafis Home. [En línea] [Citado el: 23 de enero de 2014.] http://www.iafisgroup.com/.

20. **Morpho.** MORPHOBIS. [En línea] [Citado el: 19 de enero de 2014.] http://www.iafisgroup.com/files/productos/1320802263_Morphobis.pdf.

21. **DATYS.** *ESPECIFICACIONES TÉCNICAS CARGA MASIVA.* 2010.

22. **Departamento de Sistemas Informáticos y Computación.** *Universidad Politécnica de Valencia.* Valencia : s.n., 2011.

23. **Pressman, Roger S.** *Ingeniería del software (sexta edición).* s.l. : McGraw-Hill, 2005.

24. **Palacio, Juan.** *El modelo Scrum.* 2006.

25. **Letelier, Patricio y Penadés, Carmen.** Ingeniería del software y sistemas de información. [En línea] [Citado el: 20 de enero de 2014.] http://issi.dsic.upv.es/archives/f-1069167248521/actas.pdf.

26. **Cornejo González, José Enrique.** DocIRS. [En línea] enero de 2008. [Citado el: 25 de enero de 2014.] http://www.docirs.cl/uml.htm.

27. **Palomino, Esau Chavez.** Scribd. [En línea] 2014. [Citado el: 18 de abril de 2014.] http://es.scribd.com/doc/92545312/Historia-de-La-Herramienta-Case-Rational-Rose.

28. **Universidad Carlos III de Madrid.** Knowledge Reuse Group. [En línea] 2013. [Citado el: 12 de febrero de 2014.] http://www.ie.inf.uc3m.es/grupo/docencia/reglada/Is1y2/PracticaVP.pdf.

29. **Scribd.** Scribd. [En línea] 2014. [Citado el: 23 de enero de 2014.] http://es.scribd.com/doc/7411856/Caracteristicas-de-C.

30. **Babylon.** Babylon. [En línea] 1997-2014. [Citado el: 15 de enero de 2014.] http://diccionario.babylon.com/java?&tl=/.

31. **Definición.** Definición. [En línea] 2008-2014. [Citado el: 25 de febrero de 2014.] http://definicion.de/.

32. **NeatBeans.** NeatBeans. [En línea] 2013. [Citado el: 15 de abril de 2014.] https://netbeans.org/index_es.html.

33. **Microsoft.** Visual Studio. [En línea] Microsoft. [Citado el: 22 de enero de 2014.] http://www.visualstudio.com/es-es/visual-studio-homepage-vs.aspx.

34. **Quijano, Juan.** Genbeta:dev desarrollo y software. [En línea] Genbeta, 26 de diciembre de 2013. [Citado el: 15 de enero de 2014.] http://www.genbetadev.com/herramientas/visual-studio-2013.

35. **eumednet.** Enciclopedia y Biblioteca Virtual de las Ciencias Sociales, Económicas y Jurídicas. [En línea] Universidad de Málaga. [Citado el: 22 de enero de 2014.] http://www.eumed.net/libros-gratis/2009c/585/Descripcion%20del%20modelo%20del%20dominio.htm.

36. **Bass, Len, Clements, Paul y Kazman, Rick.** *Software Architecture in Practice.* s.l. : Addison-Wesley, 2003. 0321154959.

37. **Lleonart Martín, Eva y García-Menacho Rovira, Asunción .** *PATRONES.* Valencia. España : s.n., 2006.

38. **Grosso, Andrés.** Prácticas de software. [En línea] 2011. [Citado el: 8 de febrero de 2014.] http://www.practicasdesoftware.com.ar/2011/03/patrones-grasp/.

39. **Larman, Craig.** Parte IV Fase del Diseño(1). *UML y Patrones.* Ciudad de México : Prentice Hall, 1999.

40. **Gamma, Erich , y otros.** *Design Patterns: Elements of Reusable Object-Oriented Software.* 2005. ISBN 0-201-63361-2.

41. **Cisneros Gonález, Javier.** archetypeuma. [En línea] marzo de 2011. [Citado el: 22 de marzo de 2014.] http://code.google.com/p/archetypeuma/downloads/detail?name=PFC-ArchetypeUma.doc.

42. **Ruiz Pacheco, Juan Carlos.** Developer Network. [En línea] Microsoft. [Citado el: 11 de marzo de 2014.] http://msdn.microsoft.com/es-es/library/jj635998.aspx.

43. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James.** *El proceso unificado de desarrollo de software.* Estados Unidos : Addison-Wesley, 2000. ISBN 9788478290369.

44. **Salazar Martínez, Ing. Eduardo.** Informática jurídica. [En línea] 2011. [Citado el: 15 de marzo de 2014.] http://www.informatica-juridica.com/trabajos/Propuesta_Procedimiento_para_realizar_pruebas_Caja_Blanca_aplicaciones_desarrollan_lenguaje_Python.asp.

45. **Barrientos, Pablo Andrés.** SEDICI, Repositorio Institucional de la UNLP. [En línea] Universidad Nacional de la PLata, 2014. [Citado el: 17 de abril de 2014.] http://sedici.unlp.edu.ar/handle/10915/34969.

46. **ITI.** Instituto Tecnológico de Informática, Valencia España. [En línea] 30 de diciembre de 2010. [Citado el: 11 de abril de 2014.] http://www.iti.es/servicios/servicio/resource/7240/index.html.

BIBLIOGRAFÍA CONSULTADA

1. **César Tolosa Borja, Álvaro Giz Bueno.** Universidad de Castilla-La Mancha. [En línea] [Citado el: 30 de noviembre de 2013.] http://www.dsi.uclm.es/personal/MiguelFGraciani/mikicurri/Docencia/Bioinformatica/web_BIO/Documentacion/Trabajos/Biometria/Trabajo%20Biometria.pdf.

2. **HOMINI.** Plataforma Biométrica Homini. [En línea] Homini, 2004. [Citado el: 23 de octubre de 2013.] http://www.homini.com/new_page_5.htm.

3. **ICR SL.** IdosE Ingeniería, Informática y Electrónica. [En línea] 2009. [Citado el: 12 de noviembre de 2013.] http://www.idose.es/biometria.

4. **Universidad Nacional de Entre Ríos.** Buenas tareas. [En línea] agosto de 2011. [Citado el: 11 de diciembre de 2013.] http://www.buenastareas.com/ensayos/Desarrollo-De-Un-Sistema-Multi-Biom%C3%A9trico-Mediante/2594627.html.

5. **Landi, Juan Carlos.** Universidad Politécnica Salesiana. [En línea] 2004. [Citado el: 30 de octubre de 2013.] http://dspace.ups.edu.ec/bitstream/.

6. **UNAM.** Facultad de Ingeniería Biométrica Informática. [En línea] [Citado el: 13 de noviembre de 2013.] http://redyseguridad.fi-p.unam.mx/proyectos/biometria/estandares/estandar.html.

7. **ScienceDaily.** ScienceDaily. [En línea] 12 de diciembre de 2011. [Citado el: 15 de diciembre de 2013.] http://www.sciencedaily.com/releases/2011/12/111207105433.htm.

8. **National Institute of Standards and Technology.** *ANSI/NIST-ITL 1-2011 Data Format for the Interchange of Fingerprint, Facial & Other Biometric Information.* 2011.

9. **Arburola Valverde, Dr. Allan.** MailxMail.com. [En línea] 11 de 08 de 2008. [Citado el: 04 de febrero de 2014.] http://www.mailxmail.com/curso-dactiloscopia-disciplina-criminalistica/tipos-huellas-dactilares.

10. **Definición abc.** Definición abc. [En línea] 2007. [Citado el: 10 de diciembre de 2013.] http://www.definicionabc.com/tecnologia/correo-electronico.php.

11. **Díaz de Cossío, Roger y Cerón Roa, Armando.** Curso Básico del uso de la computadora e Internet. [En línea] julio de 2004. [Citado el: 19 de enero de 2014.]

http://inepja.inea.gob.mx/cursos/computacion/cursocomputo/cursocomputomac/correo/Correo_1.htm.

12. **Universitat Jaume I de Castelló(UJI).** Tema 4 Correo Electrónico. [En línea] [Citado el: 25 de noviembre de 2013.] www3.uji.es/~huerta/j11/tema4Email.pdf.

13. **León Carri, Carolina.** www-2.dc.uba.ar. [En línea] junio de 2006. [Citado el: 11 de enero de 2014.] www-2.dc.uba.ar/materias/tc/downloads/apuntes/smtp_pop_imap.pdf.

14. **MX Vanger.** Avangermx. [En línea] 12 de julio de 2011. [Citado el: 2 de diciembre de 2013.] http://www.vangermx.com/2011/07/diferencias-ventajas-y-desventajas-entre-imap-y-pop/.

15. **Alonso Gómez, Jairo.** INBIOSYS Biometria. [En línea] INBIOSYS, 16 de septiembre de 2009. [Citado el: 2 de diciembre de 2013.] http://inbiosys.wordpress.com/2009/09/16/historia-de-la-biometria/.

16. **National Institute of Standards and Tecnology.** National Institute of Standards and Tecnology. [En línea] 23 de noviembre de 2011. [Citado el: 14 de enero de 2014.] http://www.nist.gov/itl/iad/ig/ansi_standard-history.cfm.

17. **AWARE.** Aware. [En línea] 2008. [Citado el: 4 de noviembre de 2013.] http://www.aware.com/es/biometrics/nistpack.html.

18. **Rodríguez Grille, Yilian y Verdecia Viltres, Yunierki .** *Servicios de comunicación de sistemas clientes con el AFIS Civil de SAIME, en software libre.* La Habana : s.n., 2008.

19. **Safran Morpho.** Iafis Home. [En línea] [Citado el: 23 de enero de 2014.] http://www.iafisgroup.com/.

20. **Morpho.** MORPHOBIS. [En línea] [Citado el: 19 de enero de 2014.] http://www.iafisgroup.com/files/productos/1320802263_Morphobis.pdf.

21. **DATYS.** *ESPECIFICACIONES TÉCNICAS CARGA MASIVA.* 2010.

22. **Departamento de Sistemas Informáticos y Computación.** *Universidad Politécnica de Valencia.* Valencia : s.n., 2011.

23. **Pressman, Roger S.** *Ingeniería del software (sexta edición).* s.l. : McGraw-Hill, 2005.

24. **Palacio, Juan.** *El modelo Scrum.* 2006.

25. **Letelier, Patricio y Penadés, Carmen.** Ingeniería del software y sistemas de información. [En línea] [Citado el: 20 de enero de 2014.] http://issi.dsic.upv.es/archives/f-1069167248521/actas.pdf.

26. **Cornejo González, José Enrique.** DocIRS. [En línea] enero de 2008. [Citado el: 25 de enero de 2014.] http://www.docirs.cl/uml.htm.

27. **Palomino, Esau Chavez.** Scribd. [En línea] 2014. [Citado el: 18 de abril de 2014.] http://es.scribd.com/doc/92545312/Historia-de-La-Herramienta-Case-Rational-Rose.

28. **Universidad Carlos III de Madrid.** Knowledge Reuse Group. [En línea] 2013. [Citado el: 12 de febrero de 2014.] http://www.ie.inf.uc3m.es/grupo/docencia/reglada/Is1y2/PracticaVP.pdf.

29. **Scribd.** Scribd. [En línea] 2014. [Citado el: 23 de enero de 2014.] http://es.scribd.com/doc/7411856/Caracteristicas-de-C.

30. **Babylon.** Babylon. [En línea] 1997-2014. [Citado el: 15 de enero de 2014.] http://diccionario.babylon.com/java?&tl=/.

31. **Definición.** Definición. [En línea] 2008-2014. [Citado el: 25 de febrero de 2014.] http://definicion.de/.

32. **NeatBeans.** NeatBeans. [En línea] 2013. [Citado el: 15 de abril de 2014.] https://netbeans.org/index_es.html.

33. **Microsoft.** Visual Studio. [En línea] Microsoft. [Citado el: 22 de enero de 2014.] http://www.visualstudio.com/es-es/visual-studio-homepage-vs.aspx.

34. **Quijano, Juan.** Genbeta:dev desarrollo y software. [En línea] Genbeta, 26 de diciembre de 2013. [Citado el: 15 de enero de 2014.] http://www.genbetadev.com/herramientas/visual-studio-2013.

35. **eumednet.** Enciclopedia y Biblioteca Virtual de las Ciencias Sociales, Económicas y Jurídicas. [En línea] Universidad de Málaga. [Citado el: 22 de enero de 2014.] http://www.eumed.net/libros-gratis/2009c/585/Descripcion%20del%20modelo%20del%20dominio.htm.

36. **Bass, Len, Clements, Paul y Kazman, Rick.** *Software Architecture in Practice.* s.l. : Addison-Wesley, 2003. 0321154959.

37. **Lleonart Martín, Eva y García-Menacho Rovira, Asunción .** *PATRONES.* Valencia. España : s.n., 2006.

38. **Grosso, Andrés.** Prácticas de software. [En línea] 2011. [Citado el: 8 de febrero de 2014.] http://www.practicasdesoftware.com.ar/2011/03/patrones-grasp/.

39. **Larman, Craig.** Parte IV Fase del Diseño(1). *UML y Patrones.* Ciudad de México : Prentice Hall, 1999.

40. **Gamma, Erich , y otros.** *Design Patterns: Elements of Reusable Object-Oriented Software.* 2005. ISBN 0-201-63361-2.

41. **Cisneros Gonález, Javier.** archetypeuma. [En línea] marzo de 2011. [Citado el: 22 de marzo de 2014.] http://code.google.com/p/archetypeuma/downloads/detail?name=PFC-ArchetypeUma.doc.

42. **Ruiz Pacheco, Juan Carlos.** Developer Network. [En línea] Microsoft. [Citado el: 11 de marzo de 2014.] http://msdn.microsoft.com/es-es/library/jj635998.aspx.

43. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James.** *El proceso unificado de desarrollo de software.* Estados Unidos : Addison-Wesley, 2000. ISBN 9788478290369.

44. **Salazar Martínez, Ing. Eduardo.** Informática jurídica. [En línea] 2011. [Citado el: 15 de marzo de 2014.] http://www.informatica-juridica.com/trabajos/Propuesta_Procedimiento_para_realizar_pruebas_Caja_Blanca_aplicaciones_desarrollan_lenguaje_Python.asp.

45. **Barrientos, Pablo Andrés.** SEDICI, Repositorio Institucional de la UNLP. [En línea] Universidad Nacional de la PLata, 2014. [Citado el: 17 de abril de 2014.] http://sedici.unlp.edu.ar/handle/10915/34969.

46. **ITI.** Instituto Tecnológico de Informática, Valencia España. [En línea] 30 de diciembre de 2010. [Citado el: 11 de abril de 2014.] http://www.iti.es/servicios/servicio/resource/7240/index.html.

47. **NIST.** National Institute of Standard and Technology. [En línea] NIST. [Citado el: 2 de diciembre de 2013.] http://www.nist.gov/.

48. **Quintero, Juan Bernardo.** Aprenda en Línea. [En línea] [Citado el: 15 de febrero de 2014.] http://aprendeenlinea.udea.edu.co/.

49. **DATYS.** Datys Tecnología & Sistemas. [En línea] [Citado el: 23 de noviembre de 2013.] http://www.datys.cu/wpinfproducto.aspx?18.

50. **Definición.de.** Definición.de. [En línea] [Citado el: 18 de enero de 2014.] http://definicion.de/java/.

51. **Álvarez, Sara.** desarrolloweb. [En línea] 17 de octubre de 2012. [Citado el: 24 de enero de 2014.] http://www.desarrolloweb.com/articulos/protocolo-smtp-pop.html.

52. **Download 3k.** Download 3k. [En línea] 2011. [Citado el: 11 de noviembre de 2013.] http://www.download3k.es/Foto-y-Grafico/Visualizadores/Download-NIST-ANSI-NIST-ITL-1-2000-viewer.html.

53. **Oracle Corporation.** NetBeans. [En línea] 2013. [Citado el: 1 de abril de 2014.] https://netbeans.org/community/releases/61/index_es.html.

54. **Olivares Rojas, Msc. Juan Carlos.** *Patrones de Diseño.* México : s.n.

55. **Sparx Systems.** Sparx System. [En línea] Sparx Systems Pty Ltd. [Citado el: 16 de abril de 2014.] http://www.sparxsystems.com.au/resources/uml2_tutorial/uml2_componentdiagram.html.

56. **Gil, Manuel Torres.** Universidad de Almería. [En línea] [Citado el: 14 de marzo de 2014.] http://indalog.ual.es/mtorres/LP/Prueba.pdf.

57. **Universidad Union Bolivariana.** Universidad Union Bolivariana. [En línea] [Citado el: 1 de diciembre de 2013.] http://ingenieriadesoftware.mex.tl/52753_XP---Extreme-Programing.html.