Universidad de las Ciencias Informáticas
Facultad 1

SISTEMA INFORMÁTICO PARA LA EVALUACIÓN DE ATRIBUTOS DE CALIDAD DE LOS COMPONENTES BIOMÉTRICOS EN EL CISED

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

Autor: Grettel Susel Incencio Piñeiro

Tutores: Ing. Dolennis Concepción Hidalgo
Ing. Noichel Juan Hernández

La Habana, Cuba.
Junio de 2012

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo “Sistema Informático para la evaluación de atributos de calidad de los componentes biométricos en el CISED”, y autorizo a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de Junio de 2012.

_____

Grettel Susel Incencio Piñeiro

Firma del Autor

_____ _____

Ing. Dolennis Concepción Hidalgo

Firma del Tutor

Ing. Noichel Juan Hernández

Firma del Tutor

DATOS DE CONTACTO

- ✓ **Ing. Dolennis Concepción Hidalgo:** Ingeniera en Ciencias Informáticas, Profesora Instructora. Graduada en la Universidad de las Ciencias Informáticas (UCI) en julio del 2008, donde continúa ejerciendo. Ha estado vinculada al trabajo productivo e investigativo durante más de 4 años en temas relacionados con la Calidad del Software. Actualmente se desempeña como Asesora de Calidad en el Centro de Identificación y Seguridad Digital.

  Correo electrónico: dconcepcion@uci.cu

- ✓ **Ing. Noichel Juan Hernández:** Ingeniero en Ciencias Informáticas, Profesor Instructor. Graduado en la Universidad de las Ciencias Informáticas (UCI) en julio del 2008, donde continúa ejerciendo. Ha estado vinculado al trabajo productivo e investigativo durante más de 4 años en temas relacionados con la Calidad del Software. Actualmente se desempeña al frente del grupo de calidad del Centro de Identificación y Seguridad Digital.

  Correo electrónico: njuan@uci.cu

AGRADECIMIENTOS

A mi mamá por todos estos años de esfuerzo, sacrificio y dedicación para que yo haya podido tener y ser todo lo que hasta hoy he logrado. A mis hermanos Tonito, Giselle y Ediesky, por demostrarme que a pesar de todo, siempre están ahí para mí, los quiero mucho. A mis tías que siempre me han demostrado mucho cariño. A mis abuelos, por ayudarme y apoyarme tanto, por ser parte importante en mi vida y sentirse siempre orgullosos de mí. A mi familia por todo su apoyo y cariño.

A Noichel, por ayudarme siempre, por su comprensión, paciencia y cariño.

A mis amigos, por brindarme su apoyo en todo momento y que han batallado junto a mí siempre.

A todos los profesores que he tenido a lo largo de mi carrera, por prepararme profesionalmente.

A mis tutores por haberme trasmitido sus conocimientos, por guiarme y apoyarme para que este trabajo fuera posible.

A todos Gracias.

DEDICATORIA

Dedico este Trabajo de Diploma a la persona que siempre ha estado conmigo sin fallo alguno en los buenos y malos momentos, que me llevó dentro, que me ha apoyado siempre, la que me ha dado fuerzas para salir adelante, a esa mujer que quiero infinitamente: mi Madre.

RESUMEN

Actualmente en Cuba las empresas dedicadas a la producción de software tienen entre sus principales objetivos desarrollar productos y servicios informáticos de alta calidad. Para alcanzar estos resultados y lograr un lugar y un reconocimiento en el mercado, es necesaria la implantación de Modelos o Estándares de Calidad que garanticen la comprobación objetiva de la evaluación de la calidad de los productos de software, desarrollados en cada una de las entidades. Por lo tanto, el presente trabajo de diploma tiene como objetivo proponer un Modelo de Calidad que permita evaluar los atributos de calidad presentes en los componentes biométricos desarrollados por el Departamento de Biometría del Centro de Identificación y Seguridad Digital (CISED). Para lograr lo antes expuesto, se realizó un estudio de los modelos de calidad estandarizados internacionalmente, en particular la NC - ISO/IEC 9126, a partir del cual se conforma la propuesta de solución.

Se propone además el desarrollo de un sistema informático que permite la evaluación de atributos de calidad de los componentes biométricos del Centro de Identificación y Seguridad Digital. El desarrollo de la aplicación web estuvo guiado por las especificaciones que propone la metodología de desarrollo XP, obteniendo los artefactos de las diferentes fases de trabajo, como la exploración, planificación, diseño, implementación y pruebas. Se emplearon herramientas libres y de código abierto, cuya selección fue el resultado de un estudio comparativo entre las tendencias y tecnologías actuales, y se especifican de forma detallada las funcionalidades que debe brindar el sistema a los usuarios del mismo.

Palabras claves: calidad, componentes biométricos, evaluación, Modelo de Calidad

INDICE DE CONTENIDOS

INTRODUCCIÓN

Con el avance de las tecnologías y el uso de las Tecnologías de la Información y la Comunicación (TIC), se ha experimentado un crecimiento vertiginoso en la industria del software a nivel mundial. Es por esto que muchas de las empresas productoras de software entran en competencia en el mercado de productos. El gobierno revolucionario cubano, pese al bloqueo que se le ha impuesto, ha sabido insertarse en la industria del software, logrando avances notables en este sector.

Cuba ha estado inmersa en un nuevo proceso de transformaciones educacionales y sociales como parte del programa de la Batalla de Ideas, a partir del cual se realizaron y se realizan nuevos programas destinados a elevar el nivel cultural de la población y su calidad de vida. Como resultado de este proceso, en el 2002 el Comandante en Jefe Fidel Castro indicó la idea de convertir el territorio que ocupaba la base militar de radio escucha Lourdes, controlada hasta ese momento por los rusos, en la Universidad de Ciencias Informáticas (UCI).

La UCI es una universidad productiva, cuya misión es producir software y servicios informáticos a partir de la integración de los procesos de formación, investigación, producción y extensión universitaria como modelo de formación. La idea es convertir la informática en una de las ramas más productivas y aportadoras de recursos para la nación. Es básicamente, el empleo a fondo de la inteligencia y del capital humano que tenemos, y principalmente del que podemos crear casi como espina dorsal de la economía (UCI, 2010).

Uno de los productos desarrollados por la UCI son los componentes biométricos del Centro de Identificación y Seguridad Digital (CISED), los cuales son desarrollados por estudiantes y profesores de la Facultad 1 especializados en el tema. Para desarrollar dichos componentes, un factor imprescindible a tenerse en cuenta es la calidad; el producto debe ser lo suficientemente bueno y cumplir con los requerimientos de calidad más exigentes, para poder competir con empresas productoras de software reconocidas internacionalmente.

Situación problémica

Todo producto de software desarrollado debe ser probado, evaluado por sus desarrolladores y un equipo dedicado a garantizar su calidad dentro de la institución que lo desarrolla antes de entregarse al cliente, pues no probar el producto indica que el software no cuente con la calidad requerida, y por lo tanto no

cumpliría con las expectativas del cliente. Una de las vías para garantizar la calidad de un producto de software, es la evaluación del mismo a través de la medición de sus atributos de calidad (internos o externos), la cual es uno de los procesos del ciclo de vida de desarrollo del software.

Actualmente en el Centro de Identificación y Seguridad Digital se utilizan componentes biométricos comprados a empresas extranjeras, por esta razón el Departamento de Biometría del Centro se ha trazado la tarea de desarrollar sus propios componentes biométricos. Además como el proceso del aseguramiento de la calidad se realiza completamente manual no existe una estrategia que permita conocer el estado en que se encuentra la calidad de los componentes de software biométricos, y en este sentido existe la posibilidad de que se introduzcan errores en la realización de futuras evaluaciones a dichos componentes desarrollados en el CISED.

Por otra parte, el tratamiento de los temas de calidad en componentes biométricos, es una cuestión relativamente nueva en el Centro, razón por la cual no se cuenta con una experiencia previa en el tema. Igualmente el personal disponible no cuenta con la capacitación necesaria para garantizar la calidad en los productos de esta línea productiva.

En este caso se han encontrado algunas dificultades, pues en ocasiones los atributos del software especificados por un usuario final durante la fase de análisis de los requisitos, no satisfacen los requisitos del usuario cuando el producto se está utilizando, debido a cambios en los requisitos del usuario y a la dificultad de especificar necesidades implícitas.

Debido a esto surge la necesidad de evaluar los atributos de calidad del software en el Departamento de Biometría, desde el principio de su desarrollo y durante las diferentes fases del ciclo de vida del software, ya que esto constituye un elemento esencial para el buen funcionamiento del software, y es clave a la hora de detectar errores o fallas. Así se evitarían los gastos en corregir los errores, y por consiguiente, se tendría un producto listo en el tiempo previsto y por supuesto, con el presupuesto estimado.

Por todo lo descrito anteriormente, se define el siguiente **problema científico**: ¿Cómo evaluar los atributos de calidad presentes en los componentes biométricos en el Centro de Identificación y Seguridad Digital (CISED)?

El **objeto de estudio** de la problemática planteada, lo constituyen los atributos de calidad en proyectos de desarrollo de software; y se define como **campo de acción** la evaluación de atributos de calidad de los componentes biométricos en el Centro de Identificación y Seguridad Digital.

El **objetivo general** de la presente investigación es definir un Modelo de Calidad que permita evaluar los atributos de calidad de los componentes biométricos desarrollados por el Departamento de Biometría del Centro de Identificación y Seguridad Digital (CISED). Para darle cumplimiento a dicho objetivo, se trazaron los siguientes **objetivos específicos**:

1. Describir el estado del arte de la utilización de Modelos de Calidad para la realización de las evaluaciones de componentes biométricos.
2. Definir los atributos y sub-características de calidad asociadas a los componentes biométricos del Centro.
3. Implementar un sistema que permita automatizar el proceso de evaluación de los atributos de calidad en los componentes biométricos.

Para cumplir con los objetivos, se trazaron las siguientes **tareas**:

1. Estudio de las normas asociadas a la evaluación de calidad basadas en atributos de calidad para componentes biométricos.
2. Estudio de las características, funcionamiento y aplicaciones de las técnicas de identificación y autenticación biométrica.
3. Análisis de los principales estándares de calidad asociados a tecnologías biométricas.
4. Identificación y análisis de métodos o procedimientos para evaluar atributos de calidad en los productos de software.
5. Descripción de las tecnologías, herramientas, metodología y lenguajes de programación a utilizar.
6. Implementación de un sistema automatizado para la realización de las evaluaciones.

Para el desarrollo de la investigación, se parte de la siguiente **idea a defender**: La evaluación de los atributos de calidad de los componentes biométricos del CISED, mejorará la calidad de los mismos, logrando que estos se desempeñen con la eficiencia requerida para productos de este tipo.

Como **métodos de la investigación científica,** se emplearon métodos teóricos y métodos empíricos, como se detalla a continuación:

1. **Métodos teóricos:**

- ✓ **Analítico - Sintético:** *"facilita el entendimiento del fenómeno en el que se trabaja, es más útil la división de este en diferentes fases, y de esta forma descubrir sus características generales, lo que ayuda a seguir una correcta investigación"* (Hernández Meléndrez, 2006). Durante todo el proceso investigativo se realizará un estudio a profundidad de toda la bibliografía relacionada con el tema y a partir del análisis realizado se seleccionará una síntesis de lo estudiado.

- ✓ **Histórico - Lógico:** *"permite estudiar de forma analítica la trayectoria histórica real de los fenómenos, su evolución y desarrollo"* (Hernández Meléndrez, 2006). Este método permitirá conocer sobre el problema a resolver desde sus principales etapas en su trayectoria, la historia de su desarrollo y garantizará el estudio de la estructura del objeto de la investigación junto a la concepción histórica del mismo. Posibilitará saber cuál es la verdadera situación del tema en la actualidad.

- ✓ **Modelación:** ***"****el modelo científico es un instrumento de la investigación de carácter material o teórico, creado por los científicos para reproducir el fenómeno que se está estudiando"* (Hernández Meléndrez, 2006). Se utilizará este método ya que uno de los objetivos propuestos fue el de realizar el diseño de una aplicación web, por lo que a través del modelado se podrá entender la lógica del proceso a automatizar.

2. **Métodos empíricos:**

- ✓ **Observación:** este método permitirá conocer la realidad mediante percepción del objeto de estudio, es decir, cuando se comienza la investigación se basa en la observación para saber cómo son los problemas existentes.

- ✓ **Entrevista:** "*es una conversación planificada entre el investigador y el entrevistado para obtener información*" (Hernández Meléndrez, 2006). Se realizarán a especialistas en desarrollo del software de biometría, ya que ellos están en contacto directo con el software, y además constituirá un medio para el conocimiento cualitativo de los fenómenos.

Resultados esperados:

1. Definición de un Modelo de Calidad para la evaluación de componentes biométricos, basados en sus atributos de calidad.

2. Sistema informático que permita automatizar el procedimiento de evaluación de atributos de calidad en componentes biométricos en el CISED.

La presente investigación está estructurada por introducción, tres (3) capítulos y conclusiones generales:

Capítulo 1: **Fundamentación Teórica**, en este capítulo se establecerá una base teórica para entender el problema planteado. Se describirán los conceptos fundamentales para el dominio del problema, así como las tendencias, técnicas, tecnologías, metodologías y software a utilizar. Se expondrá el estado del arte del tema a nivel nacional e internacional.

Capítulo 2: **Propuesta de solución, exploración y planificación**, en este capítulo se definirá el Modelo de Calidad a utilizar para la evaluación de los componentes biométricos, donde se definirán para el mismo los atributos de calidad, las sub-características, las métricas y lista de chequeo que permitan evaluarlos. Se presentarán además los requerimientos de la propuesta de solución mediante las historias de usuario, y una planificación del tiempo que conllevará la implementación de estas.

Capítulo 3: **Diseño, implementación y prueba del sistema**, en este capítulo se mostrará el diseño del sistema y se describirán los artefactos relacionados con la implementación. Se definirán las tarjetas CRC (Contenido, Responsabilidad, Colaboración) como herramienta de reflexión en el diseño de software orientado a objetos y se detallarán las iteraciones para el desarrollo del sistema, exponiéndose las tareas generadas en cada una de ellas por cada historia de usuario.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1. INTRODUCCIÓN

En el presente capítulo se ofrecerá un enfoque general sobre los aspectos más significativos relacionados con las principales temáticas abordadas acerca del estado del arte. Se proporcionará una panorámica general acerca de los diferentes componentes biométricos existentes, así como los estándares de calidad asociados a los mismos. Se describirán además, los atributos de calidad que pueden ser evaluados en los productos de software, así como los distintos modelos de calidad que se pueden emplear para ello. Finalmente se especificarán los lenguajes de programación, tecnologías, herramientas y metodologías a utilizar en la implementación del sistema propuesto.

1.2. BIOMETRÍA Y COMPONENTES BIOMÉTRICOS

La Biometría es el estudio de métodos automáticos para el reconocimiento único de humanos, basado en uno o más rasgos conductuales o físicos intrínsecos. El término se deriva de los vocablos griegos *bios* (vida) y *metron* (medida).

Es una tecnología de seguridad basada en el reconocimiento de una característica de seguridad, y en el reconocimiento de una característica física e intransferible de las personas. Los sistemas biométricos incluyen un dispositivo de captación y un software biométrico que interpreta la muestra física y la transforma en una secuencia numérica (Homini S.A, 2010).

En este sentido, es necesario definir además qué es un componente de software. Un componente de software es una unidad de composición de aplicaciones software que posee un conjunto de requisitos, y que ha de poder ser desarrollado, adquirido, incorporado al sistema y compuesto con otros componentes, de forma independiente en tiempo y espacio.

Dentro de las principales ventajas del desarrollo de software basado en componentes, se destaca la capacidad de reutilización de los mismos, pues de esta manera los componentes se diseñan y desarrollan con el objetivo de poder ser reutilizados en otras aplicaciones, permitiendo reducir el tiempo de desarrollo y mejorando la fiabilidad del producto final, al utilizar componentes ya probados previamente (Fernández Bertoa, et al., 2010).

Este principio es igualmente aplicable a los componentes biométricos, los cuales se encuentran divididos en dos grupos principalmente, de acuerdo a:

a) **Las características conductuales:** en este grupo se encuentran esencialmente la firma manuscrita y el reconocimiento por voz.

b) **Las características fisiológicas:** en esta clasificación se agrupan la huella dactilar, iris y retina, geometría de la mano y reconocimiento facial.

A continuación se realiza una pequeña descripción de los distintos sistemas de autenticación biométrica.

Firma manuscrita

Suele utilizarse no sólo como señal de autentificación o consentimiento en un escrito, sino también como método que establece la capacidad de una persona en la realización de una transacción (por ejemplo, el pago con tarjeta de crédito). Además la identificación del autor del documento no necesita su presencia física y una vez firmado, no puede negar que prestó su consentimiento al contenido del documento (Ferrer Ballester, 2010).

Reconocimiento por voz

Es una modalidad biométrica que utiliza la voz de un individuo con fines de reconocimiento. Difiere de la tecnología del *speech recognition* (reconocimiento de discurso), que reconoce las palabras a medida que van siendo articuladas. El proceso de reconocimiento de voz depende de las características de la estructura física del tracto vocal del individuo, así como también de sus características de comportamiento (Biometrics.gov, 2010).

Huella dactilar

Una huella dactilar usualmente aparece como una serie de líneas oscuras, que representan los relieves (la porción saliente de las crestas de fricción); mientras los valles entre estas crestas aparecen como espacios en blanco y están en bajo relieve (la porción subyacente de las crestas de fricción). La identificación por huella dactilar está basada principalmente en las minucias, o la ubicación y dirección de los finales y bifurcaciones (separaciones) de las crestas a lo largo su trayectoria (Biometrics.gov, 2010).

Reconocimiento de iris

Es el proceso de reconocer a una persona analizando el patrón al azar del iris. El método automatizado de reconocimiento de iris es relativamente joven, existiendo en patente solamente desde 1994. El iris es un músculo dentro del ojo que regula el tamaño de la pupila, controlando la cantidad de luz que entra en el ojo. Es la porción coloreada del ojo, basando su color en la cantidad del pigmento *Melatonina* dentro del músculo.

Aunque la coloración y la estructura del iris están genéticamente ligadas, los detalles de los patrones no lo están. El iris se desarrolla durante el crecimiento prenatal, con un estricto proceso de formación y plegado de la membrana de tejido fino. Antes del nacimiento, ocurre la degeneración, dando por resultado la abertura de la pupila y los patrones únicos y al azar del iris. Aunque genéticamente idénticos, el iris de un individuo es único y estructuralmente distinto, lo que le permite que sea utilizado para propósitos de reconocimiento (Biometrics.gov, 2010).

Reconocimiento palmar

La identificación palmar, al igual que la identificación dactilar, está basada en la suma de la información presentada en una impresión de surcos de fricción. Esta información combina el flujo de surcos, las características de los surcos y la estructura de los surcos de la porción de la epidermis expuesta. Dado que las huellas palmares y dactilares son únicas y permanentes, han sido utilizadas por más de un siglo como una forma confiable de identificación (Biometrics.gov, 2010).

Un sistema de reconocimiento palmar está diseñado para interpretar el flujo de los surcos en general, asignar una clasificación y luego extraer las minucias (un subconjunto del total de la información disponible), la suficiente como para buscar y compararen una gran base de datos de impresiones palmares.

Reconocimiento facial

Los humanos a menudo utilizan los rostros para reconocer individuos, y los avances en las capacidades de computación en las últimas décadas, ahora permiten reconocimientos similares de forma automática. Los algoritmos de reconocimiento facial anteriores usaban modelos geométricos simples, pero el proceso de reconocimiento actualmente ha madurado en una ciencia de sofisticadas representaciones matemáticas y procesos de coincidencia. Importantes avances e iniciativas en los pasados diez (10) a quince (15) años, han propulsado a la tecnología de reconocimiento facial al centro de la atención (Biometrics.gov, 2010).

1.3. CALIDAD DEL SOFTWARE

La calidad de un producto de software es el conjunto de propiedades o características mediante las cuales se evalúa y describe su calidad. Una característica se puede dividir en múltiples niveles de sub-características (Fernández Bertoa, et al., 2010).

El interés por la calidad del software crece de forma continua, a medida que los clientes se vuelven más selectivos y comienzan a rechazar los productos poco fiables o que realmente no dan respuesta a sus necesidades. Si se desea mejorar en cuanto a la calidad del software, entonces es preciso definir la calidad y sobre todo medirla. Todavía el mayor problema en la Gestión de la Calidad es que el mismo término calidad es ambiguo, tanto que comúnmente es mal entendido.

A la hora de definir la calidad del software se pueden adoptar diferentes conceptos. Primeramente es importante diferenciar entre la calidad del producto software y la calidad del proceso de desarrollo de éste (calidad de diseño y fabricación). No obstante, las metas que se establezcan para la calidad del producto van a determinar los objetivos a establecer para la calidad del proceso de desarrollo, ya que la calidad del primero va a depender, entre otros aspectos, del segundo. Sin un buen proceso de desarrollo es casi imposible obtener un buen producto. Es fundamental resaltar que la calidad de un producto software debe ser considerada en todos sus estados de evolución (especificaciones, diseño, implementación y otros). No basta con verificar la calidad del producto una vez finalizado, cuando los problemas de mala calidad ya no tienen solución o su reparación es muy costosa.

Los desarrolladores de software se encuentran con diferentes problemas a la hora de hablar de la calidad de un producto software, dentro de estos se encuentran: (1) la definición misma de la calidad del software (Kan, 2010), ya que se hace difícil encontrar un conjunto de propiedades que den una indicación de su calidad y para dar solución a este problema surgieron los modelos de calidad; (2) la comprobación de la calidad del software, pues muchas veces no se sabe medir el grado de calidad de un producto software y para ello juega un papel fundamental el control de calidad; (3) la mejora de la calidad del software, en este sentido se hace embarazoso saber utilizar la información disponible acerca de la calidad del producto software para mejorar su calidad a lo largo del ciclo de vida, en este eje aparecen dos conceptos importantes, ellos son la gestión de calidad y el aseguramiento de la calidad (de Antonio, 2010). A continuación se exponen diferentes definiciones de Calidad del Software:

- ✓ Calidad del software se define como la concordancia con los requisitos funcionales y de rendimiento explícitamente establecidos, con los estándares de desarrollo explícitamente documentados, y con las características implícitas que se espera de todo software desarrollado profesionalmente (Pressman, 2002).
- ✓ Calidad del software es el grado con el que un sistema, componente o proceso cumple los requisitos especificados y las necesidades o expectativas del cliente o usuario.
- ✓ Por su parte, la IEEE[1] expresa que Calidad es el grado con el cual el cliente o usuario percibe que el software satisface sus expectativas.

Una conclusión evidente a la que se puede llegar a partir de estas definiciones es que la calidad es algo relativo. Siempre va a depender de los requisitos o necesidades que se desee satisfacer. Por eso, la evaluación de la calidad de un producto siempre va a implicar una comparación entre unos requisitos preestablecidos y el producto realmente desarrollado.

Teniendo esto en cuenta, se puede notar que en un producto software se van a tener diferentes visiones de la calidad: necesaria o requerida (la que quiere el cliente); programada o especificada (la que se ha especificado explícitamente y se intenta conseguir) por último la realizada (la que se ha conseguido).

A la intersección entre la calidad requerida y la calidad realizada se le llama calidad percibida, y es la única que el cliente valora. Toda aquella calidad que se realiza pero no se necesita es un gasto inútil de tiempo y dinero. En esto concuerda *Pressman* al afirmar que la calidad es importante, pero si el usuario no queda satisfecho, ninguna otra cosa realmente importa; y en este sentido es de vital importancia lograr una entrega dentro de presupuesto y del tiempo establecido para alcanzar un producto de software íntegro y de alta calidad.

En los temas de calidad, no basta simplemente con las definiciones antes mencionadas, las cuales pueden resultar un tanto generales e imprecisas a la hora de construir un software de alta calidad, por lo que se hace necesario introducir el tema de los Modelos de Calidad, los cuales ayudan a definir la calidad del software de una forma más precisa y provechosa.

[1]Instituto de Ingenieros Eléctricos y Electrónicos: una asociación técnico-profesional mundial dedicada a la estandarización, entre otras cosas.

1.4. MODELOS DE CALIDAD

Diferentes literaturas coinciden en que un Modelo de Calidad es el conjunto de características y las relaciones entre las mismas, que proveen la base para especificar requisitos de calidad y evaluar la calidad. Implantar modelos de calidad tiene como objetivo principal que las instituciones desarrollen sistemáticamente productos, bienes y servicios de mejor calidad, y cumplan con las necesidades y deseos de los clientes.

Una característica de calidad de un producto software es un conjunto de propiedades mediante las cuales se evalúa y describe su calidad. Una característica se puede refinar en múltiples niveles de sub-características. Un atributo es una propiedad de calidad a la que puede asignársele una métrica, donde una métrica es un procedimiento que examina un componente y produce un dato simple, un símbolo o un número.

La calidad del producto de software se debe evaluar usando un Modelo de Calidad definido. El Modelo de Calidad se usará al fijar los objetivos de calidad para los productos de software y productos de software intermedios. La especificación y la evaluación de la calidad de un producto de software se pueden conseguir definiendo características de calidad apropiadas, tomando en cuenta el objetivo de uso del producto de software en cuestión. Además, el Modelo de Calidad a utilizar depende de los componentes de software a evaluar (Fernández Bertoa, et al., 2010).

Uno de los modelos de calidad más antiguos y extendidos es el de McCall, a partir del cual se han derivado otros modelos, como el de Boehm, FURPS y otros, hasta llegar al propuesto por el estándar ISO/IEC 9126. En general, no existe un consenso a la hora de definir y clasificar las características de calidad que debe presentar un producto de software, por lo cual es necesario estudiar diferentes modelos de calidad, sobre los que se fundamentará la propuesta de solución.

Modelo de Calidad de McCall

McCall y Cavano definieron un juego de once factores de calidad como los primeros pasos hacia el desarrollo de métricas de la calidad del software. El modelo de McCall organiza los factores en tres ejes o puntos de vista, desde los cuales el usuario puede contemplar la calidad de un producto, y cada uno de estos factores se descompone a su vez en criterios. Dichos factores son (Fillottrani, 2007):

- ✓ Operación del producto.
- ✓ Revisión del producto.
- ✓ Transición del producto.

Figura 1: Modelo de Calidad de McCall (Fuente: Fillottrani, 2007)

Modelo de Calidad de Boehm

El segundo Modelo de Calidad más conocido es el presentado por Barry Boehm en 1978. Este modelo introduce características de alto nivel, características de nivel intermedio y características primitivas, cada una de las cuales contribuye al nivel general de calidad (Fredy Pérez, et al., 2010). Las características de alto nivel representan requerimientos generales de uso, las características de nivel intermedio representan los factores de calidad de Boehm y el nivel más bajo corresponde a características directamente asociadas a las métricas de calidad.

Este modelo, al igual que el propuesto por McCall, es un modelo fijo sin posibilidad de ser modificado o adaptado por el técnico o el usuario de la aplicación. Los criterios y factores son determinados y fijos de forma que la medida de la calidad debe ajustarse a estas definiciones y a las relaciones entre criterios y factores de calidad que el modelo propone.

El modelo de Boehm junto con el de McCall, representaron los primeros intentos por cuantificar la calidad del software como producto a través de la descomposición jerárquica en árbol (Fillottrani, 2007).

Modelo de Calidad FURPS

Los factores de calidad descritos por McCall y sus colegas representan solo una de las muchas listas de comprobación sugeridas para la calidad del software. Hewlett – Packard ha desarrollado un conjunto de factores de calidad del software al que se le ha dado el acrónimo de FURPS: funcionalidad, facilidad de uso, fiabilidad, rendimiento y capacidad de soporte. Los factores de calidad FURPS provienen de trabajos anteriores y definen atributos para cada uno de los cinco factores principales. Estos factores y sus atributos pueden usarse para establecer métricas de la calidad para todas las actividades del proceso de software.

El modelo FURPS incluye, además de los factores de calidad y los atributos, restricciones de diseño y requerimientos de implementación físicos y de interfaz. Una limitación de este Modelo de Calidad es que no tiene en cuenta la portabilidad de los productos de software que se estén considerando, factor digno de consideración en función de las exigencias actuales que recaen sobre el proceso de desarrollo del software (Wetpaint Team, 2010).

Modelo de Calidad ISO/IEC 9126

La NC-ISO/IEC 9126 divide el Modelo de Calidad para los productos de software en dos partes: a) calidad interna y externa y b) calidad durante el uso. En la primera parte se definen seis (6) características para la calidad interna y externa, las cuales a su vez son divididas en sub-características que se manifiestan de manera externa cuando se usa el software como parte del sistema informatizado, y que además son el resultado de los atributos internos del software. La segunda parte del Modelo de Calidad define cuatro características para la calidad durante el uso, esta parte es la combinación para el usuario de las seis características de calidad del producto de software. Para ver la descripción detallada de cada una de estas características y sub-características remitirse al Anexo 1: Características y sub-características de calidad norma ISO/IEC 9126-1.

Modelo para la calidad interna y externa

Este modelo se ha desarrollado en un intento de identificar los atributos más importantes para la calidad interna y externa en un producto software. El modelo identifica seis características claves de calidad donde cada una de ellas se descompone en un conjunto de sub-características (NC- ISO/IEC 9126-1: 2005).

Los factores ISO/IEC 9126 no necesariamente se utilizan para medidas directas. En cualquier caso, facilitan una valiosa base para medidas indirectas y una excelente lista para determinar la calidad de un sistema (Fernández Bertoa, et al., 2010).

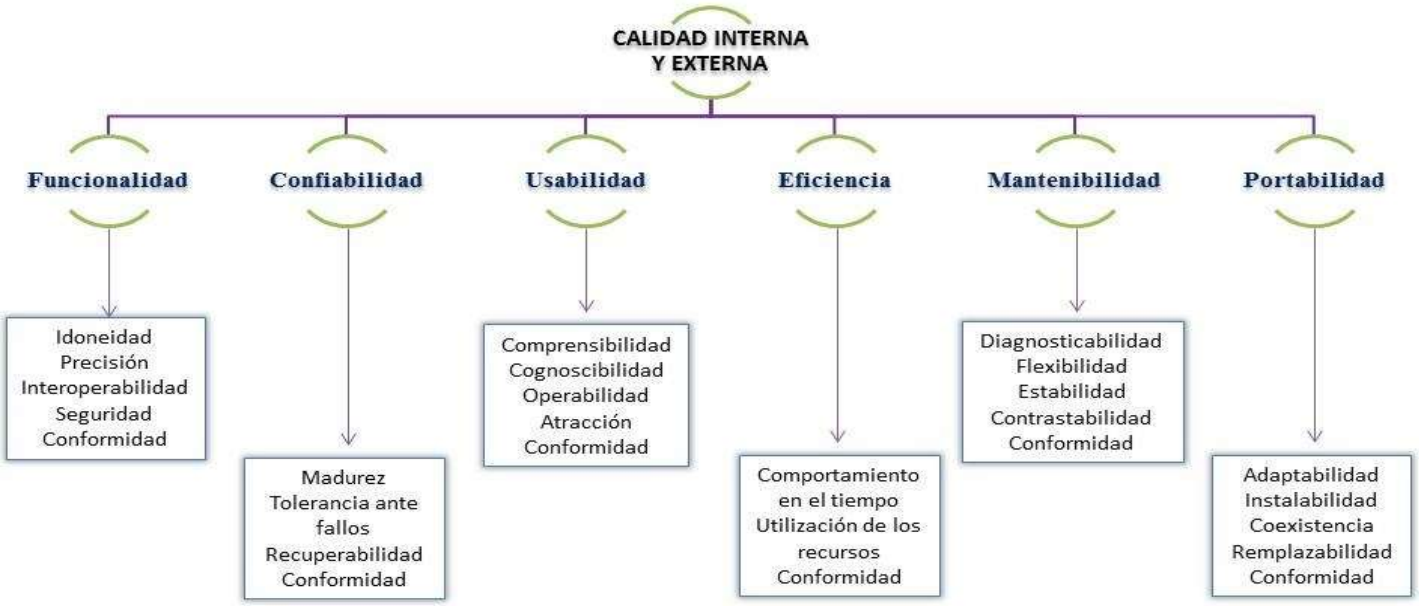

Figura 2: Modelo de Calidad interna y externa (Fuente: (NC- ISO/IEC 9126-1: 2005))

Modelo para la calidad durante el uso

La calidad durante el uso, es la capacidad de un producto de software de permitir que los usuarios especificados, alcancen los objetivos especificados con **efectividad**, **productividad**, **seguridad** y **satisfacción**, en contextos de uso especificados (NC- ISO/IEC 9126-1: 2005).

Figura 3: Modelo de Calidad durante el uso (Fuente: NC- ISO/IEC 9126-1: 2005)

Métricas de software

Todas las organizaciones de software exitosas implementan mediciones como parte de sus actividades cotidianas, pues estas brindan la información objetiva necesaria para la toma de decisiones que tendrá un impacto efectivo en el negocio y desempeño en la ingeniería. Para poder asegurar que un proceso o sus productos resultantes son de calidad o poder compararlos, es necesario asignar valores, descriptores, indicadores o algún otro mecanismo mediante el cual se pueda llevar a cabo dicha comparación. Para entender mejor el concepto de métrica, es necesario aclarar que los términos medida, medición y métrica, no tienen el mismo significado, como se puede observar a continuación:

- ✓ **Medida**: proporciona una indicación cuantitativa de la extensión, cantidad, dimensiones, capacidad o tamaño de algunos atributos de un proceso o producto.
- ✓ **Medición**: la medición es el acto de determinar una medida.
- ✓ **Métrica**: es una medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo dado (Pressman, 2002).

Las métricas de software se definen como la aplicación continua de mediciones basadas en técnicas para el proceso de desarrollo del software y sus productos, para suministrar información relevante a tiempo, así el administrador junto con el empleo de estas técnicas mejorará el proceso y sus productos (González Doria, 2001). Por otra parte, las métricas de calidad son todas las métricas de software que definen de una u otra forma la calidad del software, tales como: exactitud, estructuración o modularidad, pruebas, mantenimiento, reusabilidad, entre otras. Estas son los puntos críticos en el diseño, codificación, pruebas y mantenimiento.

Métricas internas y externas

Las métricas internas pueden ser aplicadas a un producto de software no ejecutable durante el diseño y la programación. Estas miden los atributos internos o indican los atributos externos a través del análisis de las propiedades estáticas del producto de software intermedio o final. Las métricas externas permiten evaluar la calidad del producto de software durante el ensayo o la operación.

Métricas de la calidad en el uso

Las métricas de la calidad en el uso miden hasta qué punto un producto satisface las necesidades de usuarios específicos para lograr las metas especificadas con la eficacia, productividad, seguridad y

satisfacción en un contexto determinado de uso. La evaluación de la calidad en el uso valida la calidad del producto de software en situaciones específicos de las tareas del usuario (NC- ISO/IEC 9126-1: 2005).

1.5. Estándares asociados a tecnologías biométricas

Durante los últimos años las organizaciones encargadas de crear estándares relativos al uso de técnicas biométricas en el ambiente informático, han venido preocupándose debido al gran interés de las industrias a nivel mundial por esta esfera tecnológica, y a los diversos beneficios que su uso aporta. Sin embargo, la estandarización sigue siendo deficiente, lo que trae como resultado que los proveedores de soluciones biométricas continúen suministrado interfaces de software propietarios para sus productos, por lo que se dificulta para las empresas cambiar de productos o el proveedor de los mismos.

En el mundo existen varios organismos coordinadores de las actividades de estandarización biométrica. Para ver los estándares relativos al uso de técnicas biométricas en el ambiente informático Ver Anexo 2: Estándares asociados a tecnologías biométricas.

1.6. Evaluación de software

La evaluación de software es el proceso que tiene como finalidad determinar el grado de eficacia y eficiencia con la que han sido empleados los recursos destinados a alcanzar los objetivos previstos, posibilitando la adopción de medidas correctivas que garanticen el cumplimiento adecuado de las metas presupuestadas (Definición.org, 2010).

¿Por qué evaluar un componente biométrico?

La importancia de realizar evaluaciones a los componentes biométricos radica en el propósito de determinar en qué medida se cumplen los atributos de calidad. Esto significa que se verifique que: el componente realice el trabajo deseado, la habilidad del componente para mantenerse operativo, que satisfaga las necesidades del usuario, que el componente pueda operar en diferentes entornos informáticos y la capacidad del componente para responder a una petición del usuario con la velocidad apropiada. Esto se logra a través de índices medibles que representan las bases para la calidad, el control y el perfeccionamiento del componente.

¿Quiénes participan en una evaluación?

Por lo general las evaluaciones son realizadas por integrantes del equipo de desarrollo contando con la presencia de especialistas de calidad de software que son quienes guiarán el proceso.

Tipos de evaluación de software

En términos generales, se pueden distinguir dos tipos de evaluaciones durante el proceso de desarrollo: verificaciones y validaciones. Según el IEEE *Standards* 729-1983 éstas se definen como:

- ✓ **Verificación**: proceso de determinar si los productos de una cierta fase del desarrollo de software cumplen o no los requisitos establecidos durante la fase anterior.

- ✓ **Validación**: proceso de evaluación del software al final del proceso de desarrollo para asegurar el cumplimiento de las necesidades del cliente.

Técnicas de dirección de calidad de software

Tanto para la realización de verificaciones como de validaciones se pueden utilizar distintos tipos de técnicas. En general, estas técnicas se agrupan en dos categorías:

- ✓ **Técnicas de evaluación estáticas:** buscan faltas sobre el sistema en reposo. Estudian los distintos modelos que componen el sistema de software buscando posibles faltas en los mismos.

Estas técnicas se pueden aplicar tanto a requisitos como a modelos de análisis, diseño y código, generalmente en las etapas tempranas del proceso. Posibilitan la medición de atributos de calidad interna a través de listas de chequeo, cuestionarios, métricas entre otros.

- ✓ **Técnicas de evaluación dinámicas:** generan entradas al sistema con el objetivo de detectar fallos, cuando el sistema ejecuta dichas entradas. Los fallos se observan cuando se detectan incongruencias entre la salida esperada y la salida real.

Estas técnicas se pueden aplicar generalmente sobre código dado que es el único producto ejecutable del desarrollo. Posibilitan la medición de atributos de calidad externa a través de evaluaciones basadas en prototipos, métricas, y escenarios (casos de prueba).

1.7. SOLUCIONES VINCULADAS AL CAMPO DE ACCIÓN

En la actualidad, a pesar de la gran documentación existente sobre el tema de la validación de calidad en los productos de software, existen pocas industrias de software que empleen procesos de evaluación de atributos de calidad para sus productos, pues la evaluación de la calidad de software ha evolucionado hacia modelos basados en métricas para el aseguramiento, control y evaluación de la calidad de un proceso de software.

Es por eso que en el Departamento de Computación CINVESTAV[2] del Instituto Politécnico situado en México, desarrollaron una metodología para la evaluación de la calidad de software en Sistemas de Información en Internet, con el objetivo de que dicha metodología sirva de modelo para cualquier empresa que quiera realizar la validación de los atributos de calidad en sus productos de software. También en el Departamento de Lenguajes y Ciencias de la Computación situado en la Universidad de Málaga, España, se propone un Modelo de Calidad para evaluar componentes de software referenciando sus atributos de calidad. El mismo define las características y atributos de calidad más relevantes para este tipo de producto de software, detallando para cada atributo las métricas que lo pueden cuantificar y que permiten describir y medir no sólo su funcionalidad, sino también sus aspectos extra-funcionales.

En la Universidad de las Ciencias Informáticas (UCI), como institución bandera de todo este movimiento en el país, se realizó en la Facultad 6 un procedimiento para definir los requisitos de calidad en el proceso de desarrollo de software en la Universidad, el cual propone una vía para identificar y especificar los requisitos de calidad luego de haberse obtenido los posibles riesgos técnicos que afectarían al producto. También en la antigua Facultad 8 se realizó una propuesta para evaluar la calidad de los productos de software a través de métricas. La misma propone un conjunto de métricas con el objetivo de que se apliquen durante el proceso de evaluación de la calidad de los productos de software.

1.8. APLICACIONES WEB

Una aplicación web es un sistema que permite al usuario acceder a múltiples informaciones contenidas en la red. Las aplicaciones web se basan en la arquitectura cliente/servidor y ofrecen grandes ventajas que

[2]Centro de Investigación y Estudios Avanzados: realiza actividades de investigación y desarrollo tecnológico en ingeniería eléctrica, electrónica y computación.

pueden ser aprovechadas por muchas organizaciones e instituciones. Las principales ventajas se mencionan a continuación:

- ✓ **Compatibilidad multiplataforma**: varias tecnologías incluyendo Java, Flash, ASP y Ajax permiten un desarrollo efectivo de programas soportando todos los sistemas operativos principales.
- ✓ **Inmediatez de acceso**: las aplicaciones basadas en la web no necesitan ser descargadas, instaladas y configuradas.
- ✓ **Menos requerimientos de memoria**: las aplicaciones basadas en la web tienen muchas más demandas de memoria RAM de parte del usuario final que de los programas instalados localmente.
- ✓ **Múltiples usuarios concurrentes**: las aplicaciones basadas en web realmente pueden ser utilizadas por múltiples usuarios al mismo tiempo.
- ✓ **Alta disponibilidad**: los usuarios de una aplicación web pueden realizar consultas en cualquier parte del mundo donde haya acceso a Internet y en cualquier momento.

Como resultado de los estudios e investigaciones realizadas, se propone la implementación de una aplicación web para la realización del sistema informático que se pretende, ya que son tecnologías actualmente muy prometedoras, seguras y eficaces. Además se propone la arquitectura cliente/servidor ya que está estrechamente relacionada con las aplicaciones web y cuenta con una serie de ventajas que le atribuyen gran importancia. Entre sus principales ventajas podemos citar la posibilidad de existir varios usuarios nutriéndose de información simultáneamente, así como mayor rapidez para el mantenimiento y el desarrollo de aplicaciones; además de la facilidad de integración entre diferentes sistemas, el intercambio de información, entre otras.

1.8.1. LENGUAJES DE PROGRAMACIÓN

Un lenguaje de programación es un idioma artificial diseñado para expresar operaciones que pueden ser llevadas a cabo por máquinas como las computadoras. Pueden usarse para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión o como modo de comunicación humana.

En este caso, el lenguaje de programación a utilizar es un lenguaje de programación web, el cual permite dar dinamismo a las páginas web. Estos se dividen en dos grupos: los que están del lado del cliente y del lado del servidor. Del lado del cliente se encuentran principalmente HTML, Java script, entre otros y del lado

del servidor están PHP, ASP, y otros. El sistema empleará como lenguaje de programación del lado del servidor PHP en su versión 5.3.8.

PHP

PHP (acrónimo de *PHP Hypertext Pre-processor*), es un lenguaje interpretado de alto nivel que se ejecuta del lado del servidor. Este lenguaje permite a los desarrolladores la rápida creación dinámica de páginas web. Con PHP se puede hacer casi cualquier cosa, como el procesamiento de información en formularios, foros de discusión, manipulación de cookies y páginas web dinámicas. También ofrece la integración con varias bibliotecas externas, incorporando una gran cantidad de funciones para realizar útiles tareas relacionadas con la web, dígase generar documentos en PDF hasta analizar código XML, entre otras.

PHP ofrece además una solución simple para las paginaciones dinámicas de la web de fácil programación. Su diseño lo hace más fácil de mantener y ponerse al día en comparación con el código de otros lenguajes. Debido a su amplia distribución y que es un producto de código abierto, PHP está soportado por una gran comunidad de desarrolladores, gozando de la ayuda de un gran grupo de programadores, permitiendo que los fallos de funcionamiento se encuentren y se reparen rápidamente. El código se pone al día continuamente con mejoras y extensiones de lenguaje para ampliar sus capacidades (Welling, et al., 2011).

La seguridad y configurabilidad son aspectos muy importantes, por lo que es recomendable instalarlo sobre servidores Unix o Linux que son conocidos como los más veloces y seguros. PHP permite configurar el servidor de modo que se permita o se rechacen diferentes usos, lo que puede hacer al lenguaje más o menos seguro dependiendo de las necesidades de cada cual. Para la implementación de esta aplicación se ha seleccionado PHP porque:

- ✓ Es un lenguaje multiplataforma, por lo que puede ser utilizado en cualquier sistema operativo.
- ✓ Su velocidad de ejecución es alta, en caso de que sea instalado en un servidor Linux o Unix su velocidad es mayor debido a que se ejecuta en un único espacio de memoria.
- ✓ Es uno de los lenguajes de programación con más seguidores en el mundo, por tal motivo cuenta con una gran comunidad de desarrolladores, lo que posibilita la existencia de una gran cantidad documentación disponible para los desarrolladores que se interesen por el lenguaje.
- ✓ Software de código libre, que puede ser descargado gratuitamente desde internet.

- ✓ Es de fácil uso y posee una curva de aprendizaje relativamente baja y rápida, por lo que no tardará nada en utilizar PHP de manera productiva.

1.8.2. SISTEMA GESTOR DE BASE DE DATOS

Los gestores de base de datos son sistemas formados por un conjunto de datos y un paquete de software para la gestión del mismo, de modo que se controla el almacenamiento de datos redundantes; estos resultan independientes de los programas que los usan y se puede acceder de diversas formas (Matamoros, 2011).

Un sistema gestor de base de datos nos ayuda a realizar las siguientes acciones: (Pérez Valdés, 2011)

- ✓ Definición de los datos.
- ✓ Mantenimiento de la integridad de los datos dentro de la base de datos.
- ✓ Control de la seguridad y privacidad de los datos.
- ✓ Manipulación de los datos.

El sistema empleará como sistema gestor de base de datos MySQL en su versión 5.5.16.

MySQL

MySQL es un sistema para la administración de bases de datos relacional rápido y sólido. El servidor de MySQL controla el acceso a los datos para garantizar el uso simultáneo de varios usuarios, para proporcionar acceso a dichos datos y para asegurarse de que solo obtienen acceso a ellos los usuarios con autorización.

Por lo tanto, MySQL es un servidor multiusuario y de subprocesamiento múltiple. Utiliza SQL (del inglés *Structured Query Language*, Lenguaje de consulta estructurado), como lenguaje estándar para las consultas de bases de datos utilizado en todo el mundo. MySQL se distribuye bajo una licencia de código abierto en la actualidad, pero también existen licencias comerciales (Welling, et al., 2011). MySQL cuenta con muchas otras ventajas, entre las que se encuentran las siguientes:

- ✓ Alto rendimiento y bajo costo.
- ✓ Facilidad de configuración y aprendizaje.
- ✓ Portabilidad.
- ✓ Accesibilidad a código fuente.

1.8.3. METODOLOGÍA DE DESARROLLO DE SOFTWARE

La metodología en el proceso de desarrollo de un software es el plano de apoyo, y es la guía que conducirá a realizar un software de calidad. Además sirve para que el desarrollo y confección del software, no parezca riguroso, complicado, ni difícil de controlar.

Metodologías ágiles

Ante las dificultades para utilizar metodologías tradicionales con restricciones de tiempo y flexibilidad surgen las metodologías ágiles, que al estar especialmente orientadas a proyectos pequeños, constituyen una solución a medida para ese entorno, aportando una elevada simplificación que a pesar de ello no renuncia a las prácticas esenciales para asegurar la calidad del producto (Gadgets, 2010).

Como resultado de esta nueva teoría se crea un manifiesto ágil, cuyas principales características son:

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

Entre las metodologías ágiles se encuentran:

- ✓ eXtreme Programming (XP).
- ✓ Cristal Methods.
- ✓ SCRUM.

Metodología ágil XP

XP 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 retroalimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes y simplicidad en las soluciones implementadas. El ciclo de vida de XP es iterativo y define cuatro fases fundamentales: exploración, planificación, implementación y pruebas. Lo esencial en este proceso de desarrollo es lograr la

comunicación entre desarrolladores y usuarios, la retroalimentación entre ellos y con los usuarios finales y la simplicidad en el código.

¿Por qué se seleccionó la metodología XP?

Después de haberse hecho un estudio y análisis de estas metodologías de desarrollo, se ha llegado a la conclusión de utilizar XP, ya que se adapta al software a desarrollar, así como a las condiciones de trabajo de forma general. La misma consiste en una programación rápida y extrema, cuya particularidad es tener como parte del equipo al usuario final, pues es uno de los requisitos para llegar al éxito del proyecto. Propone que la comunicación y satisfacción del cliente es lo principal. No se hace mucho énfasis en la documentación, sólo es más importante definir los requerimientos y las pruebas de calidad.

Es importante destacar de esta metodología, que dada la poca experiencia que tiene el cliente en temas de aplicaciones web, los requerimientos actuales pueden estar ajustados a futuros cambios, este es un punto donde la metodología es flexible, ya que permite administrar estos cambios de forma óptima. Además, uno de los objetivos de importancia que aplica esta metodología, es potenciar al máximo el trabajo en grupo, donde los jefes de proyecto, los clientes y desarrolladores son parte del equipo y están involucrados en el desarrollo del software.

Otra de las razones fundamentales para escoger esta metodología es que el proyecto es pequeño, y XP está concebida para ser utilizada dentro de proyectos pequeños y de desarrollo rápido; y por otra parte la existencia de pocos roles, ya que esta metodología está dirigida a grupos de desarrollo pequeños y con pocos roles. Para ver más información sobre el análisis realizado para esta selección ver Anexo 3: Comparación de la metodología utilizada con metodologías pesadas y con otras similares.

1.8.4. LENGUAJE DE MODELADO

Un lenguaje para el modelado de objetos es un conjunto estandarizado de símbolos y de modos de disponerlos para modelar un diseño de software orientado a objetos. El uso de un lenguaje de modelado es más sencillo que la programación, pues existen menos medios para verificar efectivamente el funcionamiento adecuado del modelo.

Esto puede suponer también que las interacciones entre partes del programa, de lugar a sorpresas cuando el modelo ha sido convertido en un software que funciona. Toda metodología de desarrollo de

software utiliza un lenguaje para el modelado de objetos, para la representación de sus diagramas y artefactos. Para el modelado del sistema en cuestión se utilizará UML.

UML (*Unified Modeling Language*)

UML es un lenguaje estándar para el modelado de software utilizado para visualizar, especificar y documentar los artefactos del sistema. Permite a los desarrolladores visualizar el producto de su trabajo (artefacto) en esquemas o diagramas estandarizados.

UML es la interrelación Elementos – Relaciones – Diagramas, es un modelo gráfico que incluye aspectos conceptuales (negocio y sistema) y aspectos concretos (expresiones del lenguaje de programación, esquema de base de datos y componentes de software reutilizables). Se emplea para definir un sistema, sus artefactos, documentos y construir el software.

El UML estándar está compuesto por tres partes: bloques de construcción (tales como clases, objetos, mensajes), relaciones entre bloques (asociación, generación) y diagramas (por ejemplo el diagrama de actividades). Los perfiles de UML son las extensiones a las notaciones estándares del UML utilizando los mecanismos de extensión del UML: los estereotipos, los valores etiquetados y restricciones (Pressman, 2002). Se escoge UML porque:

- ✓ Permite modelar sistemas utilizando técnicas orientadas a objetos.
- ✓ Permite especificar todas las decisiones de análisis y diseño, construyéndose así modelos precisos, no ambiguos y completos.
- ✓ Puede conectarse con lenguajes de programación (Ingeniería directa e inversa).
- ✓ Permite documentar todos los artefactos de un proceso de desarrollo (requisitos, arquitectura, pruebas, versiones, etc.).
- ✓ Es un lenguaje muy expresivo que cubre todas las vistas necesarias para desarrollar y luego desplegar los sistemas.
- ✓ Existe un equilibrio entre expresividad y simplicidad, pues no es difícil de aprender ni de utilizar.

1.8.5. HERRAMIENTA CASE

Las herramientas de modelado de objetos, son fundamentales para el análisis del sistema. Existen varias herramientas creadas para el desarrollo de la Ingeniería de Software. Estas existen con el fin de desarrollar

programas, utilizando técnicas de diseño y metodologías bien definidas, soportadas por herramientas automáticas. Estas presentan varios beneficios como los citados a continuación (López Pecho, et al., 2008):

- ✓ Facilidad para la revisión de aplicaciones.
- ✓ Soporte para el desarrollo de prototipos de sistemas.
- ✓ Generación de código.
- ✓ Mejora en la habilidad para satisfacer los requerimientos del usuario.
- ✓ Soporte interactivo para el proceso de desarrollo.

Dentro de las herramientas CASE tenemos: Microsoft Project, Rational Rose, JDeveloper, Magic Draw, Visual Paradigm, Microsoft Visio, Enterprise Architect, Poseidon y Erwin. Como herramienta CASE para el modelado se utilizará Visual Paradigm.

Visual Paradigm

Visual Paradigm es una herramienta CASE (Ingeniería de Software Asistida por Ordenador, del inglés *Computer Aided Software Engineering*) que utiliza UML como lenguaje de modelado. Soporta el ciclo de vida completo de desarrollo de un software, desde la fase de análisis hasta el despliegue del mismo. Permite realizar ingeniería directa o inversa sobre el software y es capaz de, a partir de un modelo relacional en diferentes sistemas gestores de base de datos, desplegar todas las clases asociadas a las tablas.

Visual Paradigm for UML está diseñada para usuarios interesados en sistemas de software de gran escala con el uso del acercamiento orientado a objeto, además apoya los estándares más recientes de las notaciones de Java y de UML. Incorpora el soporte para trabajo en equipo, que permite que varios desarrolladores trabajen a la vez en el mismo diagrama y vean en tiempo real los cambios hechos por sus compañeros (Team Project) (Visual Paradigm, 2011). Presenta una serie de ventajas las cuales se relacionan a continuación:

- ✓ Presenta licencia gratuita y comercial.
- ✓ Buena integración con IDE’s de desarrollo.
- ✓ Soporta muchos más lenguajes de programación.
- ✓ Soporte ORM: generación de objetos Java desde la base de datos.

- ✓ Generación de bases de datos: transformación de diagramas de entidad-relación en tablas de base de datos.
- ✓ Ingeniería inversa de bases de datos: desde sistemas gestores de bases de datos existentes a diagramas de entidad-relación.
- ✓ Generador de informes para generación de documentación.
- ✓ Distribución automática de diagramas: reorganización de las figuras y conectores de los diagramas UML.
- ✓ Importación y exportación de ficheros XML.
- ✓ Ayuda a construir aplicaciones de calidad más rápido, mejor y a más bajo costo.

Desventajas

- ✓ No permite su uso en proyectos comerciales e incluye marca de agua recordando este hecho.
- ✓ Muestra muchas otras funcionalidades no disponibles como gancho para las versiones de pago.
- ✓ Las imágenes y reportes generados, no son de muy buena calidad.

¿Por qué se seleccionó para el modelado la herramienta CASE Visual Paradigm?

Luego de un detallado estudio se decidió escoger para el modelado del sistema la herramienta CASE Visual Paradigm, ya que esta brinda una respuesta rápida y bajos requisitos de memoria del motor de persistencia, sólo requiere de una configuración de escritorio, soporta la ingeniería inversa de múltiples formas y permite modelar todos los diagramas de clases. Para ver más información sobre las herramientas remitirse al Anexo 4: Herramientas CASE.

1.8.6. Herramienta IDE

Un entorno de desarrollo integrado IDE (del inglés *Integrated Development Environment*), está compuesto por un conjunto de herramientas para agilizar el proceso de desarrollo del programador. Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica GUI. Estos pueden ser aplicaciones por si solas o pueden ser parte de aplicaciones existentes. Además, proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación. En algunos lenguajes puede funcionar como un sistema en tiempo de ejecución, en donde se permite utilizar el lenguaje de programación en forma interactiva. Para la implementación del sistema se seleccionó NetBeans IDE 7.1.2.

NetBeans

El IDE NetBeans es un reconocido entorno de desarrollo integrado disponible para Windows, Mac, Linux y Solaris. El proyecto NetBeans está formado por un IDE de código abierto y una plataforma de aplicación que permite a los desarrolladores crear con rapidez aplicaciones web, empresariales, de escritorio y para móviles utilizando la plataforma Java, así como JavaFX, PHP, JavaScript y Ajax, C/C++, y otras (NetBeans.org, 2011).

El proyecto de NetBeans está apoyado por una comunidad de desarrolladores dinámica y ofrece documentación y recursos de formación exhaustivos, así como una amplia selección de complementos de terceros. Dentro de las funciones más importantes que se pueden encontrar en el NetBeans se pueden citar:

Lenguajes Web: HTML, CSS, JavaScript

- ✓ Reestructuración y búsqueda de usos para CSS y lenguajes parecidos a HTML.
- ✓ Autocompletado y enlace para atributos id y class.
- ✓ Reestructuración de estilos en línea de CSS.

PHP

- ✓ Compatibilidad con PHP Zend Framework.
- ✓ Anotaciones *Overrides/Implements* y *Is Overridden/Implemented*.
- ✓ Nuevo formateador con más reglas de formateo.

Java

- ✓ Agrupación de puntos de interrupción en el depurador de Java, histórico de parámetros adjuntados al depurador.
- ✓ Compatibilidad para procesadores de anotaciones para el editor, configurable en las propiedades del proyecto.
- ✓ Compatibilidad para nuevos applets y Web Starts.
- ✓ Navegación mejorada en el analizador de trazas de pila y URL's.

1.8.7. FRAMEWORK DE DESARROLLO

Un *framework* es una estructura de soporte definida, en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, un *framework* puede incluir soporte de programas, librerías y un

lenguaje de scripting, además de otros programas para ayudar a desarrollar y unir los diferentes componentes de un proyecto. Este representa una arquitectura de software que modela las relaciones generales de las entidades del dominio. Provee una estructura y una metodología de trabajo la cual extiende o utiliza las aplicaciones del dominio. Además proporciona estructura al código fuente, forzando al programador a crear código más legible y más fácil de mantener. Para el desarrollo del sistema se seleccionó el *framework* Symfony en su versión 1.4.18.

Symfony

Symfony es un *framework* PHP que facilita el desarrollo de las aplicaciones web. Se encarga de todos los aspectos comunes de las aplicaciones web, dejando que el programador se dedique a aportar valor agregado, desarrollando las características únicas de cada proyecto.

También aumenta la productividad y ayuda a mejorar la calidad de las aplicaciones web aplicando todas las buenas prácticas y patrones de diseño que se han definido para la web. Es además el *framework* más documentado del mundo, ya que cuenta con miles de páginas de documentación distribuidas en varios libros gratuitos y decenas de tutoriales. Las principales características que posee este *framework* son las siguientes (Eguiluz, 2011):

- ✓ Fácil de instalar y configurar en sistemas Windows, Mac y Linux.
- ✓ Funciona con todas las bases de datos comunes (MySQL, PostgreSQL, SQLite, Oracle, MS SQL Server).
- ✓ Compatible solamente con PHP 5, para asegurar el mayor rendimiento y acceso a las características más avanzadas de PHP.
- ✓ Basado en la premisa de convenir en vez de configurar, en la que el desarrollador solo debe configurar aquello que no es convencional.
- ✓ Preparado para aplicaciones empresariales, ya que se puede adaptar con facilidad a las políticas y arquitecturas propias de cada empresa u organización.
- ✓ Flexible hasta cualquier límite y extensible mediante un completo mecanismo de *plugins*.

- ✓ Publicado bajo licencia MIT[3] de software libre y apoyado por una empresa comprometida con su desarrollo.
- ✓ Traducido a más de 40 idiomas y fácilmente traducible a cualquier otro idioma.

1.8.8. SERVIDOR WEB

Un servidor web o servidor HTTP es un programa que procesa cualquier aplicación del lado del servidor, realizando conexiones bidireccionales y/o unidireccionales y síncronas o asíncronas con el cliente, generando o cediendo una respuesta en cualquier lenguaje o aplicación del lado del cliente. El código recibido por el cliente suele ser compilado y ejecutado por un navegador web. Para la transmisión de todos estos datos suele utilizarse algún protocolo; generalmente se utiliza el protocolo HTTP para estas comunicaciones, perteneciente a la capa de aplicación del modelo OSI. Para hospedar la aplicación se utilizará el servidor web Apache 2.2.21.

Servidor web Apache

En la actualidad el servidor web Apache es el más utilizado del mundo, encontrándose muy por encima de sus competidores, tanto gratuitos como comerciales. Es un software de código abierto que funciona sobre cualquier plataforma.

El servidor Apache es un software que está estructurado en módulos, es decir, está dividido en muchas porciones de código que hacen referencia a diferentes aspectos o funcionalidades del servidor web. Esta modularidad es intencionada ya que la configuración de cada módulo se hace mediante la configuración de las directivas que están contenidas dentro del módulo. Los módulos del Apache se pueden clasificar en tres categorías:

- ✓ **Módulos base**: módulos con las funciones básicas del Apache.
- ✓ **Módulos multiproceso**: son los responsables de la unión con los puertos de la máquina, aceptando las peticiones y enviando a los hijos a atender a las peticiones.
- ✓ **Módulos adicionales**: cualquier otro módulo que le añada una funcionalidad al servidor.

[3] MIT: El **Instituto Tecnológico de Massachusetts** (**MIT**, del inglés *Massachusetts Institute of Technology*) es una institución de educación superior privada situada en Cambridge, Massachusetts (Estados Unidos).

El resto de funcionalidades del servidor se consigue por medio de módulos adicionales que se pueden cargar. Para añadir un conjunto de utilidades al servidor, simplemente hay que añadirle un módulo, de forma que no es necesario volver a instalar el software. Entre las principales ventajas del servidor web Apache se pueden citar (Ciberaula, 2010):

- ✓ Multiplataforma, flexible, rápido y eficiente.
- ✓ Se desarrolla de forma abierta.
- ✓ Modular, ya que puede ser adaptado a diferentes entornos y necesidades, cuenta con diferentes módulos de apoyo y con la API de programación de módulos para el desarrollo de módulos específicos.
- ✓ Trabaja con gran cantidad de lenguajes de script como son PHP, Perl, y otros.
- ✓ Incentiva la retroalimentación de los usuarios, obteniendo nuevas ideas, informes de fallos y parches para la solución de los mismos.
- ✓ Tiene una alta configurabilidad en la creación y gestión de *logs*.
- ✓ Continuamente actualizado y evolucionando a gran velocidad.

XAMPP

Es un servidor independiente de plataforma, de software libre, que consiste principalmente en la base de datos MySQL, el servidor Web Apache y los intérpretes para lenguajes de script PHP y Perl. El nombre proviene del acrónimo de X (para cualquiera de los diferentes sistemas operativos), Apache, MySQL, PHP, Perl. El programa está liberado bajo la licencia GNU y actúa como un servidor web libre, fácil de usar y capaz de interpretar páginas dinámicas. Actualmente XAMPP está disponible para Microsoft Windows, GNU/Linux, Solaris, y MacOS (Oswald Seidler, 2011).

1.9. CONCLUSIONES PARCIALES

Una vez analizados los elementos vinculados a la problemática, se puede concluir que el aseguramiento de la calidad de procesos o productos, como el caso en cuestión, constituye una arista esencial en el proceso de desarrollo de software. Los componentes biométricos no escapan a esta tendencia, razón por la cual se hace necesario poder evaluar los mismos, una vez que estos sean desarrollados por el Departamento de Biometría del Centro, para garantizar así la máxima excelencia posible en su funcionamiento.

En este sentido, el estudio realizado permitirá en un primer momento, definir un Modelo de Calidad para evaluar componentes biométricos, tomando como base los atributos de calidad propios de este tipo de software; y finalmente, la implementación de un sistema informático para automatizar dicho proceso, garantizando una completa fiabilidad en la realización del mismo.

CAPÍTULO 2: PROPUESTA DE SOLUCIÓN, EXPLORACIÓN Y PLANIFICACIÓN

2.1. INTRODUCCIÓN

En este capítulo se realizará la descripción de la propuesta de solución que tiene como objetivo el presente trabajo. Para ello se definirán los atributos de calidad necesarios para la realización de la evaluación, así como las respectivas sub-características medibles en los componentes biométricos. También se definirán las métricas asociadas a cada sub-característica seleccionada, así como una lista de chequeo.

En cuanto al sistema informático, se identificará el negocio en general y se definirán las historias de usuario como elemento fundamental para conocer los requisitos. Se describirán además, los requisitos no funcionales del sistema a implementar y se define el plan de entregas para la implementación de las mismas, así como el plan de iteraciones con su correspondiente descripción.

2.2. ¿CUÁNDO REALIZAR LA EVALUACIÓN?

El proceso de evaluación se basa fundamentalmente en la evaluación del producto no del proceso de desarrollo del mismo, además de que el interés radica principalmente en la evaluación del mismo antes de su puesta en funcionamiento real con el cliente al cual va dirigido.

Para RUP

Actualmente el Departamento de Biometría del Centro tiene definida como metodología de desarrollo base, el Proceso Unificado de Rational (RUP). En este caso la evaluación de los atributos de calidad de los componentes biométricos se realizará en la etapa de transición del producto, que es donde se libera el producto y se entrega al usuario para su uso real.

Para SXP

En caso de usar la metodología SXP la evaluación se realizará en la fase de entrega, en la cual se realiza la puesta en marcha del producto.

Para XP

En caso de usar la metodología XP la evaluación se realizará al finalizar la fase de iteraciones, dicha evaluación se realizará al final de la última iteración, en la cual estará listo para entregar la primera versión del producto al cliente.

2.3. ROLES Y RESPONSABILIDADES EN LA EVALUACIÓN

El proceso de evaluación de los componentes biométricos va a estar guiado por los roles definidos a continuación:

- ✓ **Líder del equipo de evaluación de la calidad del software:** responsable de todo el proceso de la evaluación, debe estar capacitado y autorizado por el asesor de calidad del Centro, y debe asegurar que el proceso de evaluación sea realizado conforme al Modelo de Calidad propuesto, y que los resultados de éste sean representativos. Garantizará además, que los datos de la evaluación sean registrados en el formato adecuado.

- ✓ **Equipo evaluador:** personas con experiencias previas en el tema, el número de integrantes dependerá de la complejidad del proyecto o la urgencia con que se realice la evaluación. Estos realizan las siguientes actividades: realización de la evaluación e información del estado actual del proyecto.

Sin embargo puede haber también situaciones en las que intervengan personas especialistas en el tema. Otro involucrado que también se interesa por los resultados de la evaluación es el cliente, ya que en dependencia de los resultados, puede tomar decisiones de continuar o no con el desarrollo del componente.

2.4. ESTABLECIMIENTO DEL MODELO DE CALIDAD

El Modelo de Calidad propuesto está basado principalmente en la NC/ISO 9126 - 1 ya que esta propone un Modelo de Calidad bastante completo. Una dificultad que este modelo presenta es que no todas las características y sub-características son aplicables al desarrollo de componentes de software, por lo que es necesario en este caso adaptarlo para componentes biométricos. Igualmente se toman en consideración elementos establecidos por los distintos estándares biométricos, pues esto es un requerimiento indispensable para el Modelo de Calidad que se desea obtener.

En este sentido fueron seleccionadas las características y sub-características más significativas de acuerdo al análisis realizado, teniendo en cuenta los principales estándares establecidos en cuanto a tecnologías biométricas en el mundo, así como la opinión de los especialistas sobre el tema en el Centro.

Selección de los atributos de calidad

De los atributos de calidad propuestos por la norma NC/ISO 9126 – 1 para la calidad interna y externa, el atributo usabilidad no se tiene en cuenta en el Modelo de Calidad propuesto. Este atributo es más adecuado para evaluar portales y aplicaciones web, ya que estos están diseñados para ser utilizados por los usuarios, y existe una interacción entre los usuarios y el sistema web. Esto permitirá agilizar el proceso de evaluación y lo hará además menos engorroso.

Debido a que en este caso se va a estar evaluando componentes biométricos, los cuales fundamentan sus decisiones de reconocimiento mediante una característica personal que puede ser reconocida o verificada de manera automatizada, el resto de los atributos de calidad si pueden ser evaluados en los mismos. Se adiciona además, la característica precisión del algoritmo, la cual representa un punto clave en la efectividad del mismo durante su utilización en ambientes reales de explotación.

El Modelo de Calidad queda compuesto entonces, por las siguientes características de calidad: Funcionalidad, Confiabilidad, Eficiencia, Mantenibilidad, Portabilidad y Precisión del algoritmo, como se puede apreciar en la figura 4.

Selección de las sub-características

Como no todas las sub-características son aplicables a los componentes de software, en la figura 4 se muestra la selección realizada, a través de la propuesta del Modelo de Calidad para componentes biométricos, en el cual no se tienen en cuenta algunas sub-características tales como analizabilidad, ensayabilidad y coexistencia, entre otras. La sub-característica Conformidad perteneciente a los atributos Funcionalidad, Confiabilidad, Eficiencia, Portabilidad y Mantenibilidad, no se incluye por no estar bien establecida y no conocerse las reglas que se deben cumplir.

En el caso de la característica Precisión del algoritmo, se establecen como sub-características la Tasa de falsos aceptados (FAR) y la Tasa de falsos rechazados (FRR). Se escogen estas sub-características por ser los indicadores más significativos en lo que a precisión de algoritmos se refiere.

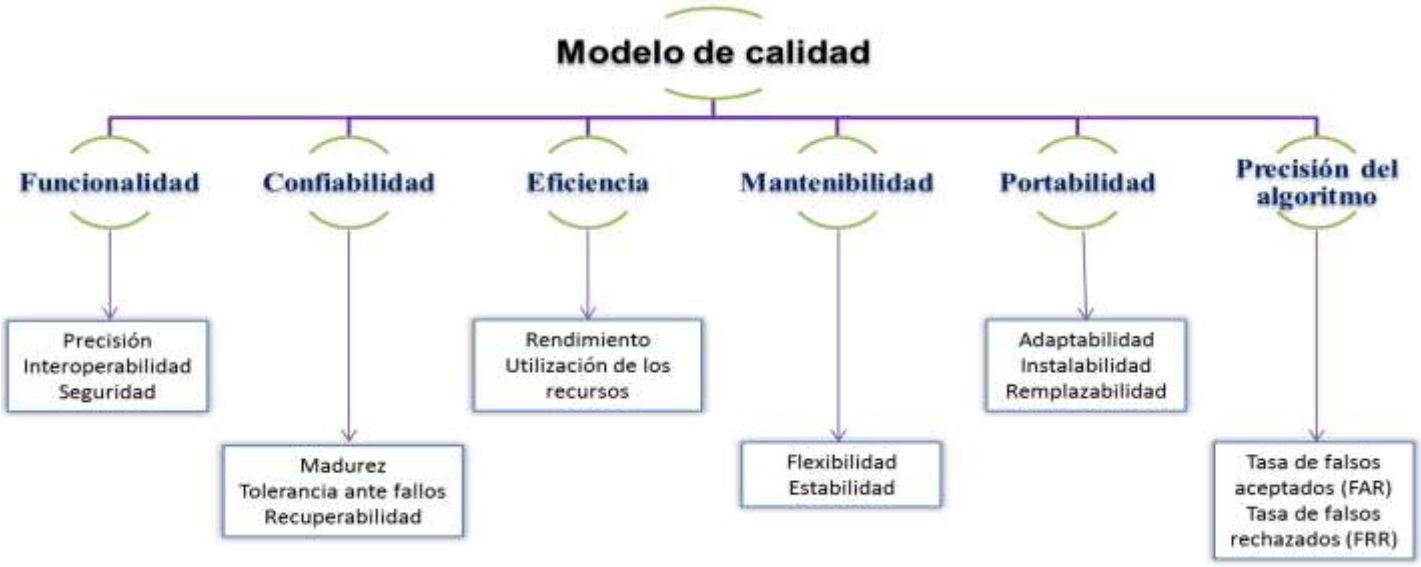

Figura 4: Modelo de Calidad para componentes biométricos (Fuente: Elaboración propia)

2.4.1. SELECCIÓN DE MÉTRICAS

Para la selección de las métricas, fue analizada de forma exhaustiva la norma ISO\IEC 9126 – 3, la cual propone las métricas para el Modelo de Calidad interna y externa establecido en la parte uno (1) de la propia norma. Es necesario aclarar que para cada sub-característica, en algunos casos existen varias métricas que pueden ser utilizadas; igualmente en ocasiones las métricas existentes no son aplicables o significativas a las características de los componentes de software, por lo que selección depende del atributo de calidad a evaluar así como de las condiciones de desarrollo y operación del componente biométrico.

Estas métricas fueron las menos complejas, donde el significado de las variables comprendidas en las fórmulas de medición fueran las más claras para el evaluador, para así agilizar y hacer menos engorroso el proceso de evaluación de los componentes biométricos. Para las sub-características de remplazabilidad y adaptabilidad, no se propone una métrica específica, ya que estas deben ser chequeadas directamente en el ambiente de explotación del componente en cuestión.

- ✓ **Métricas de funcionalidad**

Las métricas de funcionalidad deben ser capaces de medir el comportamiento funcional del sistema en el cual el componente está presente.

Funcionalidad				
Nombre de la métrica	**Propósito de la métrica**	**Método de aplicación**	**Fórmula**	**Interpretación del valor obtenido**
Sub-característica: Precisión				
Exactitud esperada	¿Existen diferencias entre los resultados actuales y los razonablemente esperados?	Ejecutar los casos de pruebas de entrada versus salida y comparar los resultados actuales y los razonablemente esperados. Contar el número de casos encontrados con diferencias inaceptables en relación con los resultados razonablemente esperados.	X = A/T A - Número de casos encontrados con diferencias entre los resultados razonablemente esperados y aquellos resultantes más allá de lo permisible. T - Tiempo de operación.	0 <= X A mayor cercanía al 0 resultará mejor.
Sub-característica: Interoperabilidad				
Intercambiabilidad de datos, en base a su formato	¿Cuán correctamente ha sido implementado el intercambio de funciones de interfaces para una transferencia de datos específica?	Ejecutar las pruebas a cada registro de salida de las funciones de interfaces de acuerdo con la especificación de los campos de datos. Contar el número de formatos de datos que deben ser	X = A / B A - Número de formatos de datos intercambiados exitosamente con otro software o sistemas durante las pruebas del intercambio de datos. B - Número total de	0 <= X <= 1 A mayor cercanía al 1 resultará mayor intercambiabilidad.

		intercambiados con otro software o sistemas durante las pruebas en comparación con el número total.	formatos de datos a intercambiar.	
Sub-característica: Seguridad				
Controlabilidad de acceso	¿Cuán controlable es el acceso al sistema?	Contar la cantidad de requerimientos de control de acceso que fueron implementados correctamente y compararlo con la cantidad de requerimientos de control de acceso descritos en la especificación.	X = A / B A- Número de historias de usuario que presentan problemas con el control de acceso. B - Número total de historias de usuario probadas.	0 <= X <= 1 A mayor cercanía al 1 resultará más seguro.

Tabla 1: Métricas de Funcionalidad (Fuente: Calisoft, 2009)

- ✓ **Métricas de confiabilidad**

Las métricas de confiabilidad deben ser capaces de medir el comportamiento del sistema del cual el componente forma parte durante la ejecución de las pruebas, para indicar la magnitud de la confiabilidad, o sea, seguridad de funcionamiento del software durante la operación del sistema, con las que en la mayor parte de los casos no se distingue entre el software y el sistema.

Confiabilidad				
Nombre de la métrica	**Propósito de la métrica**	**Método de aplicación**	**Fórmula**	**Interpretación del valor obtenido**
Sub-característica: Madurez				

Latencia estimada de la intensidad de fallos	¿Cuántos problemas aún existen que pueden emerger como futuros fallos?	Contar el número de fallos detectados durante el período definido de pruebas y pronosticar el número potencial de futuros fallos utilizando un modelo de estimación de la fiabilidad.	X = (ABS (A1- A2)) / B ABS () - Valor absoluto. A1 - Número total de fallos latentes predecibles en el producto de software. A2 -Número total de fallos detectados realmente. B - Tamaño del producto.	0 <= X A mayor cercanía al 0 resultará mejor.
Sub-característica: Tolerancia ante fallos				
Evitación de desastre	¿Cuán frecuentemente el componente de software causa un desastre en el ambiente de la producción total?	Contar el número de desastres detectados con respecto al número de fallos totales.	X = 1- A /B A - Número de desastres. B - Número de fallos totales. **Nota:** Se entiende por desastre cuando la ejecución de cualquier tarea del usuario es suspendida hasta que el sistema sea reiniciado o se ha perdido el control, lo que obliga a apagar el sistema.	0 <= X <= 1 A mayor cercanía al 1 resultará mejor.
Sub-característica: Recuperabilidad				

Recargabilidad	¿Cuán frecuentemente el sistema logra recargar luego de una ejecución perdida proveyendo nuevamente los servicios a los usuarios en el plazo de tiempo previsto?	Contar el número de veces que el sistema se reinicia o se recarga y ofrece exitosamente sus servicios a los usuarios en el plazo de tiempo previsto, y compárelo con el número de veces que el sistema se reinició como resultado de una caída en inactividad durante un período de prueba especificado.	X = A / B A - Número de veces que se provocó el reinicio o recarga en el plazo de tiempo previsto en la prueba especificada o en la implantación. B - Número total de veces que se provocó el reinicio o recarga durante la prueba especificada o la implantación.	0 <= X <= 1 A mayor cercanía al 1 mejor, el usuario puede reiniciar o recargar más fácilmente.

Tabla 2: Métricas de Confiabilidad (Fuente: Calisoft, 2009)

✓ **Métricas de eficiencia**

Las métricas de eficiencia deben ser capaces de medir el rendimiento o consumo de tiempo y el comportamiento en la utilización de los recursos del sistema, incluyendo al software, durante las pruebas y la implantación.

Eficiencia				
Nombre de la métrica	**Propósito de la métrica**	**Método de aplicación**	**Fórmula**	**Interpretación del valor obtenido**
Sub-característica: Rendimiento				
Tiempo de espera	¿Qué proporción del tiempo los usuarios dedican a esperar por una respuesta del	Ejecutar los casos de pruebas en situaciones características y tareas concurrentes. Medir el tiempo que toma	X = Ta/Tb Ta - Tiempo total dedicado a esperar. Tb - Tiempo consumido	0 <= X Cuanto menor resultará mejor

	sistema?	completar las operaciones seleccionadas. Guardar un registro de cada intento y calcular el tiempo medio para cada caso o situación.	por la tarea.	
Sub-característica: Utilización de los recursos				
Relación entre los errores de memoria y el tiempo	¿Cuántos errores de memoria fueron identificados durante el período de tiempo especificado, y con una utilización de recursos especificada?	Calibrar las condiciones de prueba. Emular una condición en que el sistema haya alcanzado un máximo de carga. Ejecutar la aplicación y registre el número de errores debido a fallos de memoria.	X = A/T A - número de mensajes de alerta o fallos del sistema. T - tiempo de operación del usuario.	0 < X Más pequeña, mejor.

Tabla 3: Métricas de Eficiencia (Fuente: Calisoft, 2009)

✓ **Métricas de mantenibilidad**

Las métricas de mantenibilidad deben ser capaces de medir el comportamiento del serviciador, el usuario o el propio sistema, incluyendo el software, cuando el software es objeto de mantenimiento o modificación, durante las pruebas y el mantenimiento.

Mantenibilidad				
Nombre de la métrica	**Propósito de la métrica**	**Método de aplicación**	**Fórmula**	**Interpretación del valor obtenido**
Sub-característica: Flexibilidad				
Tiempo de espera del usuario ante la	¿Cuál es el impacto de la utilización de	Ejecutar una amplio número de tareas	T = Tiempo empleado en esperar por la culminación	0 < T

utilización de los equipos de E/S	los equipos de E/S en el tiempo de espera de los usuarios?	concurrentes y medir el tiempo de espera del usuario ante la utilización de los equipos de E/S.	de una operación de un equipo de E/S.	Más corto, resultará mejor.
Sub-característica: Estabilidad				
Impacto del cambio	¿Cuál es la frecuencia de efectos adversos después de la modificación?	Contar el número de efectos adversos detectados después de la modificación y compararlo con el número de modificaciones realizadas.	X = 1 - A/B A - número de efectos adversos detectados en las modificaciones. B - número de modificaciones realizadas.	0 <= X <= 1 A mayor cercanía al 1 mejor.

Tabla 4: Métricas de Mantenibilidad (Fuente: Calisoft, 2009)

✓ **Métricas de portabilidad**

Las métricas de portabilidad deben ser capaces de medir el comportamiento del operador y el sistema durante las actividades de implementación y distribución del producto de software.

Portabilidad				
Nombre de la métrica	**Propósito de la métrica**	**Método de aplicación**	**Fórmula**	**Interpretación del valor obtenido**
Sub-característica: Instalabilidad				
Facilidad de instalación	¿Puede el usuario o serviciador instalar fácilmente el producto de software en su	Observar el comportamiento del usuario o serviciador cuando tratan de instalar el producto de	X = A/B A - número de casos en los cuales el usuario tiene éxito en las operaciones	0 <= X <= 1 A mayor cercanía al 1, resultará mejor.

	ambiente de operación?	software en su ambiente de operación.	para una instalación adecuada a su conveniencia. B - número total de casos en los cuales el usuario intenta adecuar la instalación a su conveniencia.	

Tabla 5: Métricas de Portabilidad (Fuente: Calisoft, 2009)

✓ **Métricas de Precisión del algoritmo**

Las métricas de precisión del algoritmo deben ser capaces de medir el comportamiento del algoritmo biométrico en un ambiente real de verificación. Para esto se utilizará el software "Plataforma de pruebas para algoritmos biométricos", implementado por estudiantes del Centro de Identificación y Seguridad Digital (CISED) en el 2011.

Precisión del algoritmo				
Nombre de la métrica	**Propósito de la métrica**	**Método de aplicación**	**Fórmula**	**Interpretación del valor obtenido**
***Sub-característica:* Tasa de falsos aceptados (FAR)**				
Tasa de falsos aceptados (FAR)	¿Cuál es la probabilidad de que un individuo no autorizado sea autenticado?	Entre más bajo sea el valor de la tasa de falsa aceptación, más alta es la precisión del sistema biométrico.	FAR = NFA/Nii NFA – número de falsos aceptados. Nii – número de intentos del impostor.	0 <= FAR <= 0.1 A mayor cercanía al 0, resultará mejor.
***Sub-característica:* Tasa de falsos rechazados (FRR)**				
Tasa de falsos rechazados (FRR)	¿Cuál es la probabilidad de que	Entre más bajo sea el valor de la tasa de falso	FRR = NFR/Nii NFR – número de falsos	0 <= FRR <= 1 A mayor cercanía al

	un individuo autorizado sea inapropiadamente rechazado?	rechazo, más alta es la precisión del sistema biométrico.	rechazados. Nii – número de intentos del impostor.	0, resultará mejor.

Tabla 6: Métricas de precisión (Fuente: Younis, 2007)

2.4.2. LISTA DE CHEQUEO

A continuación se muestra una lista de chequeo que consta de preguntas correspondientes a cada característica y sub-característica de calidad definidas anteriormente, las cuales están escritas de forma clara y de fácil entendimiento. Esta lista de chequeo permitirá definir a través de una gráfica, el nivel de calidad del componente biométrico.

1	**Funcionalidad**
1.1	***Interoperabilidad***
1.1.1	¿El componente es capaz de leer datos provenientes de otros sistemas?
1.1.2	¿El componente es capaz de producir datos para otro sistema?
1.2	***Precisión***
1.2.1	¿Los resultados ofrecidos por el componente son exactos?
1.3	***Seguridad***
1.3.1	¿Es importante una seguridad extrema en su sistema?
1.3.2	¿El sistema asegura que el componente no pierda datos ante un ataque interno o externo?
2	**Confiabilidad**
2.1	***Madurez***
2.1.1	¿El componente permite entradas de datos incorrectas?

2.2	***Tolerancia ante fallos***
2.2.1	¿Es importante que se estructure el sistema de forma tal que se minimicen los daños si falla una parte del mismo?
2.2.2	¿El sistema debe seguir funcionando si se produce algún fallo?
2.3	***Recuperabilidad***
2.3.1	¿El componente puede continuar prestando sus servicios si se produce algún fallo?
3	**Eficiencia**
3.1	***Rendimiento***
3.1.1	¿El componente tiene potencial para identificar a individuos con un grado de certeza alto?
3.1.2	¿Se realizan transacciones de datos en pequeños instantes de tiempo?
3.1.3	¿El componente da respuestas rápidas?
3.2	***Utilización de recursos***
3.2.1	¿El componente puede compartir recursos adecuadamente?
3.2.2	¿El componente necesitará recursos extras para su funcionamiento?
3.2.3	¿Los recursos con los que debe trabajar se consideran de vital importancia para el buen funcionamiento del componente?
4	**Mantenibilidad**
4.1	***Flexibilidad***
4.1.1	¿El componente brinda facilidad para modificarlo?
4.2	***Estabilidad***

4.2.1	¿El componente se adapta con facilidad ante una modificación realizada?
4.2.2	¿El componente facilita la sustitución/adaptación del mismo?
5	**Portabilidad**
5.1	***Adaptabilidad***
5.1.1	¿El componente continúa funcionando correctamente aun cuando los servicios de los provistos por el ambiente varíen?
5.2	***Instalabilidad***
5.2.1	¿El componente puede instalarse fácilmente en todos los ambientes donde debe funcionar?
5.2.2	¿Es responsabilidad del usuario final instalar el sistema?
5.2.3	¿Es de interés del usuario que la aplicación sea portable?
5.3	***Remplazabilidad***
5.3.1	¿El componente se adapta fácilmente en otros ambientes donde no debe funcionar?
6	**Precisión del algoritmo**
6.1	**Tasa de falsos aceptados (FAR)**
6.1.1	La tasa de falsos aceptados se encuentra dentro de los límites establecidos comercialmente.
6.2	**Tasa de falsos rechazados (FRR)**
6.2.1	La tasa de falsos rechazados se encuentra dentro de los límites establecidos comercialmente.

Tabla 7: Lista de chequeo para componentes biométricos (Fuente: Elaboración propia)

2.5. DESCRIPCIÓN DEL PROCESO DE EVALUACIÓN DE LAS MÉTRICAS DE CALIDAD

Primeramente se escoge el componente biométrico a evaluar. Luego se realiza el cálculo de las métricas de calidad propuestas, donde se le da valor a cada una de las variables en dependencia de lo que se está evaluando. Este proceso se realiza para cada una de las características de calidad definidas.

Valoración de los resultados de las métricas

Después de realizar el cálculo de las métricas a cada una de las sub-características, se realiza una valoración de dichos resultados para darle un valor y evaluación a las características de calidad. Esto significa que se obtiene un valor de cada sub-característica, luego se hace un resumen del resultado de cada característica; esto sería hallar el promedio entre los valores de las sub-característica correspondientes a cada característica. Una vez realizado este procedimiento se llega a una conclusión sobre el grado de conformidad que tiene cada característica sobre el componente biométrico que se está evaluando, este grado de conformidad se determina por el resultado del promedio y finalmente, según el grado de conformidad, se emite un criterio de evaluación para dicha característica.

Resultado de las características de calidad

El resultado de un atributo de calidad se obtiene a través de la suma de los resultados de las métricas de cada sub-característica, entre la cantidad de sub-características correspondientes a cada característica de calidad. Este cálculo se realiza para cada una de las características de calidad definidas. A continuación se describe detalladamente el proceso:

- ✓ **Funcionalidad** = (resultado de la métrica de idoneidad + resultado de la métrica de precisión + resultado de la métrica de interoperabilidad) / 3.
- ✓ **Confiabilidad** = (resultado de la métrica de madurez + resultado de la métrica de tolerancia ante fallos + resultado de la métrica de recuperabilidad) /3.
- ✓ **Usabilidad** = (resultado de la métrica de operabilidad + resultado de la métrica de atracción) / 2.
- ✓ **Mantenibilidad** = (resultado de la métrica de flexibilidad + resultado de la métrica de estabilidad) / 2.
- ✓ **Portabilidad** = (resultado de la métrica de instalabilidad + resultado de la métrica de coexistencia + resultado de la métrica de remplazabilidad) / 3.

- ✓ **Precisión del algoritmo** = (resultado de la métrica de FAR + resultado de la métrica de FRR) / 2.

Los atributos de calidad se evalúan de la siguiente manera: N (*Not*), P (*Partially*) y F (*Fully*), siendo:

- ✓ **N:** No alcanzado: poca o ninguna evidencia de la consecución del atributo.
- ✓ **P:** Parcialmente alcanzado: evidencia de un enfoque sistemático y de la consecución del atributo aunque algunos aspectos de la consecución pueden ser impredecibles.
- ✓ **F:** Totalmente alcanzado: evidencia de un enfoque completo y sistemático y de la consecución plena del atributo.

Rango del resultado	Criterio de evaluación
0 - 0,3	Totalmente alcanzado (F)
0,4 - 0,7	Parcialmente alcanzado (P)
0,8 - 1	No alcanzado (N)

Tabla 8: Criterios de evaluación (Fuente: Elaboración propia)

Evaluación final

Finalmente se realiza la conclusión final del proceso de evaluación, el cual consiste en dar a conocer el grado de aceptación del componente biométrico que se está evaluando. Es decir, después de haberse hecho un resumen individual del criterio de evaluación de cada característica evaluada, se llega a un veredicto conclusivo donde se obtiene como resultado si el componente biométrico evaluado tiene un nivel de calidad adecuado. Esta evaluación final está dada por la función promedio del valor de cada una de las características de calidad.

- ✓ Promedio = (Funcionalidad + Confiabilidad + Eficiencia + Mantenibilidad + Portabilidad + Precisión del algoritmo) / 6.

Para obtener un nivel de calidad del componente biométrico se toma el resultado obtenido de la función promedio y se lleva a la Escala del nivel de calidad que se muestra a continuación:

Rango	Nivel de Calidad
0 - 0,3	Adecuado
0,4 - 0,7	Medianamente adecuado
0,8 - 1	No adecuado

Tabla 9: Escala del nivel de calidad (Fuente: Elaboración propia)

Cuando se tenga definido el grado de conformidad y se hayan detectado las deficiencias en el componente se dará un criterio de evaluación para el componente, según lo muestra la siguiente tabla.

Criterio de Evaluación
( ) Sin modificaciones
( ) Pequeñas modificaciones
( ) Grandes modificaciones

Tabla 10: Criterio de evaluación (Fuente: Elaboración propia)

2.6. DESCRIPCIÓN DEL SISTEMA PROPUESTO

Como propuesta de solución al problema científico se propone el desarrollo de una aplicación web que garantice el proceso de evaluación de los atributos de calidad a los componentes biométricos del Centro de Identificación y Seguridad Digital (CISED).

Una vez autenticado el usuario, la aplicación le proporcionará de acuerdo a su rol una serie de funcionalidades. Dicha aplicación estará conformada por dos (2) módulos principales los cuales son: administración del sistema y evaluación de componentes.

Además la aplicación estará conformada por una serie de secciones con el objetivo de brindar a todos los usuarios la siguiente información: datos referentes al Centro de Identificación y Seguridad Digital tales como: Misión, Visión, Estructura, Líneas de desarrollo, Contactos. La información que mostrará la aplicación será

accesible para todos los navegantes y permitirá consultar una serie de sitios relacionados con el Centro que sean considerados de interés.

La sección autenticación de usuario consistirá en validar el usuario y la contraseña introducida por el usuario con las almacenadas en la base de datos y una vez autenticado, podrá acceder a las secciones que tenga acceso de acuerdo con el rol que le corresponda al usuario autenticado, y poseerá ciertos privilegios y permisos que le permitirán tener acceso a las diferentes funcionalidades de la aplicación.

La aplicación contará con dos (2) tipos de usuarios principalmente. El administrador del sistema será el encargado de formalizar toda la información referente al sistema y posibilitará la actualización de todas las informaciones contenidas en el mismo. El evaluador será el encargado de seleccionar el componente biométrico que desea evaluar, una vez seleccionado el mismo podrá realizar la evaluación del componente guiándose por una serie de parámetros que le permitirán realizar una evaluación satisfactoria. Además este podrá ver evaluaciones realizadas a este u otros componentes en otras ocasiones, pero no podrá modificar los datos de las mismas.

La base de datos registrará de forma unificada todas las evaluaciones realizadas, manteniendo así la base de datos de la aplicación actualizada con la información necesaria, garantizando la estabilidad y fácil mantenimiento del sistema.

Personas relacionadas con el sistema

Un factor importante a tener en cuenta cuando se comienza el desarrollo de un sistema informático es determinar a quién va dirigido el mismo. Teniendo en cuenta la anterior descripción, el sistema propuesto dispondrá de los siguientes roles de usuario:

Personas relacionadas con el sistema	Descripción
Administrador del sistema	Es la persona facultada para la gestión del sistema. Es el encargado de administrar todas las cuentas de los usuarios, además de gestionar todo tipo de información en el sistema.

Evaluador de componentes biométricos	Tiene acceso a la gestión de ciertas funcionalidades de la aplicación excepto las administrativas. Es la persona que podrá evaluar los componentes biométricos y podrá visualizar datos de evaluaciones realizadas anteriormente.

Tabla 11: Personas relacionadas con el sistema (Fuente: Elaboración propia)

2.7. REQUERIMIENTOS NO FUNCIONALES DEL SISTEMA

Los requerimientos no funcionales son propiedades o cualidades que debe tener el producto, las cuales representan las características que hacen al producto atractivo, rápido y confiable. Seguidamente se especifican los requerimientos no funcionales del sistema propuesto:

Apariencia o interfaz externa:

- ✓ Optimizado para una resolución de 1024x768.
- ✓ Interfaz amigable y sencilla, con colores suaves a la vista y sin cúmulo de objetos que le sean incómodos al usuario a la hora de interactuar con el sistema.
- ✓ Un menú principal que le brinde al usuario rapidez y facilidad para obtener la información que desea.

Software:

Para el cliente:

- ✓ Sistema operativo multiplataforma con interfaz gráfica.
- ✓ Navegador web (Internet Explorer 6.0 o superior, Mozilla Firefox 3.0 o superior, Opera, Safari).

Para el Servidor:

- ✓ Framework Symfony 1.4.10 o superior.
- ✓ Servidor web Apache 2.2.12 o superior.
- ✓ Versión de PHP 5.3.0 o superior.
- ✓ Gestor de Base de Datos MySQL 5.1.37 o superior.
- ✓ Software controlador de versiones: CollabNetSubversion-client-1.6.15 o superior.

Hardware:

Para el desarrollo:

- ✓ PC con las siguientes características: Intel Pentium 4 o superior, CPU 3GHz o superior, 1GB RAM o superior, 80 GB HDD o superior.

Para explotación del cliente:

- ✓ PC con las siguientes características: Intel Pentium 4 o superior, CPU 3GHz o superior, 256 MB RAM mínimo, 512 RAM recomendada o superior.

Para explotación del servidor:

- ✓ PC con las siguientes características: Intel Pentium 4 o superior, RAM: 1 GB mínimo (Recomendado 2 GB), 80 GB de disco duro.

Usabilidad:

- ✓ Para hacer uso del sistema es necesario poseer conocimientos elementales de computación y sobre el ambiente web en sentido general.
- ✓ El diseño debe tener buena visibilidad en los principales navegadores.
- ✓ El sistema poseerá estructura y diseño homogéneos en todas sus pantallas, que facilite la navegación tales como menús laterales y/o desplegables que permitan el acceso rápido a la información.

Soporte:

- ✓ Sistema multiplataforma.
- ✓ Los sistemas deberán estar abiertos a las modificaciones que se requieran, ya sea ante la detección de un fallo o por nuevos requisitos. Después de su puesta en explotación se le dará mantenimiento una vez al año.

Seguridad:

- ✓ Chequear que el usuario esté autenticado antes de que pueda realizar alguna acción sobre el sistema.
- ✓ La información manejada por el sistema estará protegida de acceso no autorizado y divulgación.
- ✓ Garantizar que las funcionalidades del sistema se muestren de acuerdo al tipo de usuario que esté activo.

- ✓ Realizar salvas periódicas de la información y base de datos en otros dispositivos, como solución ante la ocurrencia de problemas.
- ✓ Llevar un registro de sucesos donde se archiven los eventos del sistema incluyendo los eventos de error, inicio de sesión, cierre de sesión y modificación de la información.

Disponibilidad:

- ✓ El sistema deberá tener un 100% de disponibilidad por lo que podrá ser usado las 24 horas del día por todos sus clientes.

2.8. SEGURIDAD DEL SISTEMA

La seguridad es uno de los aspectos más importantes a tener en cuenta a la hora de proteger la información. Actualmente Symfony propone numerosas estrategias y utilidades para hacer frente a los ataques XSS[4] y CSRF[5] con el objetivo de mantener la integridad, confidencialidad y disponibilidad de la información.

Algunas de estas estrategias son: las opciones allow_extra_fields y filter_extra_fields las cuales permiten controlar si todos los campos del formulario deben tener asignado un validador y si se deben filtrar todos los campos adicionales, por lo que no tienen efecto los ataques que intentan inyectar campos que no pertenecen al formulario original. Por otra parte se ha creado un *plugin* llamado pdCSRFPlugin para solucionar los problemas contra ataques de tipo CSRF.

2.9. FASE DE EXPLORACIÓN

En esta primera fase de la metodología XP es donde el cliente define sus necesidades a través de las historias de usuario (HU), las cuales son de interés para la primera entrega del producto. Al mismo tiempo, el equipo de desarrollo se familiariza con las tecnologías, herramientas y prácticas que se utilizarán en el proyecto. Se prueba la tecnología y se exploran las posibilidades de la arquitectura del sistema construyendo

[4] XSS: del inglés ***Cross-site scripting*** es un tipo de inseguridad informática o agujero de seguridad basado en la explotación de vulnerabilidades del sistema de validación de HTML incrustado.

[5]CSRF del inglés ***Cross-site request forgery*** o falsificación de petición en sitios cruzados: es un tipo de *exploit* (una secuencia de comandos que se aprovecha de un error, fallo o vulnerabilidad, a fin de causar un comportamiento no deseado o imprevisto en los programas informáticos) malicioso de un sitio web en el que comandos no autorizados son transmitidos por un usuario en el cual el sitio web confía.

un prototipo. Esta fase tarda poco de acuerdo al tamaño y familiaridad que tengan los programadores con la tecnología.

2.9.1. HISTORIAS DE USUARIO (HU)

Como factor importante en el desarrollo de la aplicación se encuentra la administración de requisitos, a la cual XP da cobertura mediante la confección de las historias de usuario, para luego ser desarrolladas en un proceso iterativo e incremental. Estas expresan las necesidades del sistema y tienen el mismo propósito que los casos de uso, aunque cuentan con algunas diferencias tales como: constan de 3 ó 4 líneas escritas por el cliente en un lenguaje no técnico sin hacer mucho hincapié en los detalles; no se debe hablar ni de posibles algoritmos para su implementación ni de diseños de base de datos adecuados etc.

Las historias de usuario son utilizadas para estimar tiempos de desarrollo de la parte de la aplicación que describen. También se utilizan en la fase de pruebas, para verificar si el programa cumple con lo que especifica la historia de usuario. Cuando llega la hora de implementar una historia de usuario, el cliente y los desarrolladores se reúnen para concretar y detallar lo que tiene que hacer dicha historia. El tiempo de desarrollo ideal para una historia de usuario es entre 1 y 3 semanas (Menéndez, et al., 2011).

Durante el proceso de análisis en la fase de exploración se identificaron 7 HU, cada una de estas, respondiendo a las diferentes funcionalidades solicitadas por el cliente y dando una idea al resto del equipo de desarrollo de cómo debe ser su implementación posteriormente. A continuación se detalla la HU Gestionar componentes biométricos. Para ver el resto de las historias de usuario remitirse al Anexo 5: Historias de Usuario.

Historia de Usuario	
Número: 2	**Nombre:** Gestionar componentes biométricos
Usuario: Administrador del sistema	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 1	**Iteración asignada:** 1

Descripción: El usuario podrá adicionar, eliminar y mostrar un determinado componente así como la modificación de sus datos: ✓ Nombre del componente biométrico. ✓ Breve descripción del componente. ✓ Desarrollador.
Observaciones:

Tabla 12: HU Gestionar componentes biométricos (Fuente: Elaboración propia)

2.10. PLANIFICACIÓN DE LA ENTREGA

En esta fase el cliente establece la prioridad de cada historia de usuario, y correspondientemente los programadores realizan una estimación del esfuerzo necesario para cada una de ellas. También se toman acuerdos sobre el contenido de cada entrega. A partir de la prioridad de las historias de usuario, se decide cuáles de ellas se implementarán en las primeras iteraciones.

Estimación de esfuerzo por historia de usuario

Las historias de usuarios deben ser programadas en un tiempo entre una (1) y tres (3) semanas. Si la estimación es superior a tres (3) semanas, debe ser dividida en dos (2) o más historias. Para ello se decide realizar la estimación de esfuerzo que arroja cada historia de usuario con el objetivo de obtener un mejor desarrollo del sistema. Se realizó una estimación de las historias de usuario identificadas, donde se obtuvieron los siguientes resultados:

Historias de usuario	Puntos de estimación (semanas)
Autenticar usuario	0.2
Gestionar componentes biométricos	1
Gestionar atributos de calidad	1
Gestionar sub-características de calidad	1

Gestionar lista de chequeo	1
Evaluar componente biométrico	2
Ver evaluaciones realizadas	0.3
Total	6.5

Tabla 13: Estimación de esfuerzo por Historia de Usuario (Fuente: Elaboración propia)

Los puntos de estimación de esfuerzo por historias de usuarios es el tiempo de implementación para cada una de estas, cada punto representa una semana de trabajo, por lo que si se le da un punto a una HU determinada, esto significa que solo se necesita una semana de trabajo para esta HU, y así sucesivamente. (Una semana es 8 horas al día, 5 días a la semana).

2.11. PLAN DE ITERACIONES

Las HU seleccionadas para cada entrega son desarrolladas y probadas en un ciclo de iteración, de acuerdo al orden preestablecido. Al comienzo de cada ciclo, se realiza una reunión de planificación de la iteración. Cada HU se traduce en tareas específicas de programación. Así mismo, para cada HU se establecen las pruebas de aceptación. Estas pruebas se realizan al final del ciclo en el que se desarrollan, pero también al final de cada uno de los ciclos siguientes. Por todo lo antes mencionado se decide realizar la aplicación en dos (2) iteraciones, las cuales se describen a continuación:

Iteración 1

En esta primera iteración se le dará cumplimiento a las historias de usuarios 1, 2, 3, 4 y 5 las cuales serán de vital importancia para la aplicación, pues ellas conformarán la base de la estructura del negocio. Estas recogen las principales funcionalidades del sistema. Una vez que se terminen de implementar todas las funcionalidades de esta iteración, se procederá a liberar un ejecutable, el cual será mostrado al cliente con el objetivo de obtener una retroalimentación para el grupo de trabajo.

Iteración 2

En esta iteración se la implementarán de las historias de usuarios restantes, las cuales son: 6 y 7. Una vez que se termine de implementar estas funcionalidades ya se liberará la versión 1.0 de la aplicación.

2.12. PLAN DE DURACIÓN DE LAS ITERACIONES

Para una mayor organización del trabajo y como parte del ciclo de vida de un proyecto que utiliza la metodología XP se crea el plan de duración de cada una de las iteraciones, en este caso se hace para el único equipo de desarrollo con el cual se cuenta. Este plan tiene como objetivo mostrar la duración de cada iteración, así como el orden en que serán implementadas las HU en cada una de las mismas, lo que ayuda a obtener una idea aproximada del tiempo que durará la confección del sistema en su totalidad.

Iteraciones	Orden de las HU a implementar	Duración total de las iteraciones
1	1. Autenticar usuario 2. Gestionar componentes biométricos 3. Gestionar atributos de calidad 4. Gestionar sub-características de calidad 5. Gestionar lista de chequeo	4.2 semanas
2	6. Evaluar componente biométrico 7. Ver evaluaciones realizadas	2.3 semanas

Tabla 14: Plan de iteraciones (Fuente: Elaboración propia)

2.13. PLAN DE ENTREGAS

Como propuesta del plan de entregas se harán versiones al sistema en las fechas aproximadas que se indican a continuación:

Producto	**Final 1ra Iteración** 4ta semana de Abril	**Final 2da Iteración** 1ra semana de Mayo
Sistema informático para la evaluación de atributos de calidad de componentes biométricos	V beta	V 1.0

Tabla 15: Plan de entregas (Fuente: Elaboración propia)

2.14. CONCLUSIONES PARCIALES

La creación de un Modelo de Calidad que permita evaluar atributos de calidad en componentes biométricos, representa un paso de avance en el camino para la obtención de la excelencia en el desarrollo de software en el CISED. En este sentido la investigación realizada y la solución propuesta permitirán avanzar en la construcción del sistema informático que se pretende.

Para ello se generaron los artefactos que propone la metodología de desarrollo XP, historias de usuario, plan de entregas y plan de iteraciones. Se definieron además los requerimientos no funcionales y finalmente se realizaron las estimaciones de esfuerzo y el tiempo que tomará la implementación de la aplicación.

CAPÍTULO 3: DISEÑO, IMPLEMENTACIÓNY PRUEBA

3.1. INTRODUCCIÓN

En este capítulo se abordarán los aspectos fundamentales acerca de la construcción de la aplicación web. Igualmente se hará una representación a través del diagrama de despliegue de cómo será distribuida la aplicación, llevándose a cabo la fase de implementación a través de las tareas de ingeniería. En esta última etapa se definirán y realizarán las pruebas necesarias al software, para que tenga la calidad requerida y cumpla así con las expectativas del cliente.

3.2. DISEÑO DEL SISTEMA

Para el diseño de la aplicación web se utilizarán las tarjetas CRC (Contenido, Responsabilidad y Colaboración). Estas tarjetas permiten al programador centrarse en el desarrollo orientado a objetos. Cada tarjeta identifica una clase, sus propiedades y relaciones con otra tarjeta. Se elaboró una tarjeta representativa para cada clase. A continuación se detalla la tarjeta CRC Evaluación, generada en esta fase en su correspondiente iteración. Para ver el resto de las tarjetas CRC remitirse al Anexo 6: Tarjetas CRC.

Evaluación	
Funcionalidades	**Colaboraciones**
Realizar evaluación Mostrar datos evaluación	Usuario Componente Atributo Sub-característica Lista de chequeo Permiso Grupo

Tabla 16: Tarjeta CRC Evaluación (Fuente: Elaboración propia)

3.2.1. DISEÑO DE LA BASE DE DATOS

Para cualquier aplicación donde se gestione información, la base de datos desempeña un papel importante, sobre todo en el caso de las aplicaciones web, por lo que se hace necesario que los datos se almacenen de forma coherente y organizada, para evitar que dicha información se pierda. Es por esto que se realiza el diseño de la base de datos en cuestión, para lograr la persistencia de los datos de los usuarios, así como la información de los recursos que se gestionan por los mismos.

La persistencia es la capacidad de un objeto de mantener su valor en el tiempo y el espacio. No todas las clases identificadas durante el análisis son persistentes, pues existen clases temporales que son manejadas y almacenadas por el sistema en tiempo de ejecución, dejando de existir cuando termina el programa.

La propuesta de base de datos que se expone a continuación, satisface las necesidades de persistencia de los datos que el sistema requiere, en cumplimiento de sus requerimientos funcionales. Para diseñar la base de datos del sistema, se utilizan los modelos lógico y físico de datos. A continuación se presenta el modelo físico de datos.

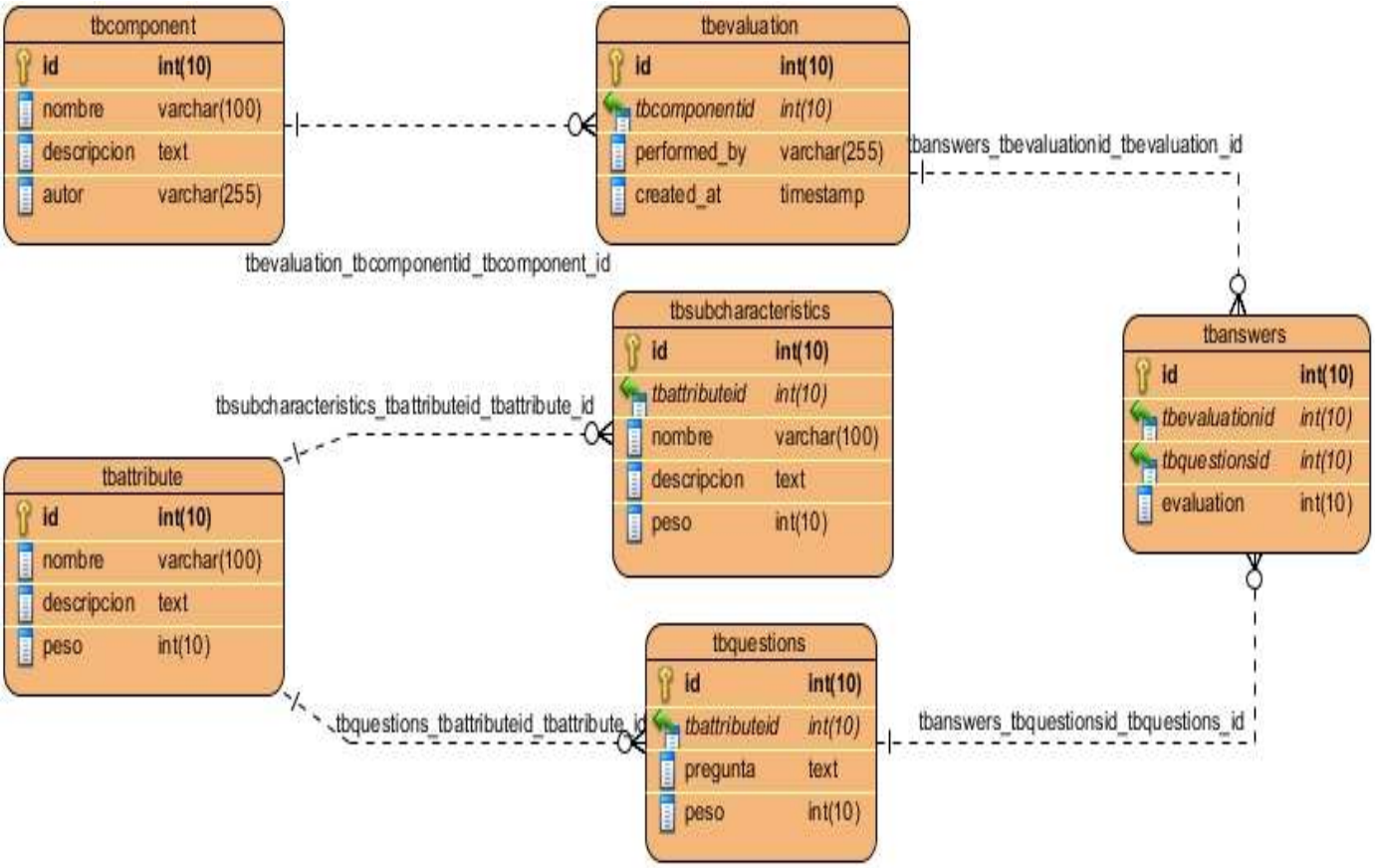

Figura 5: Diagrama de Entidad – Relación (1) (Fuente: Elaboración propia)

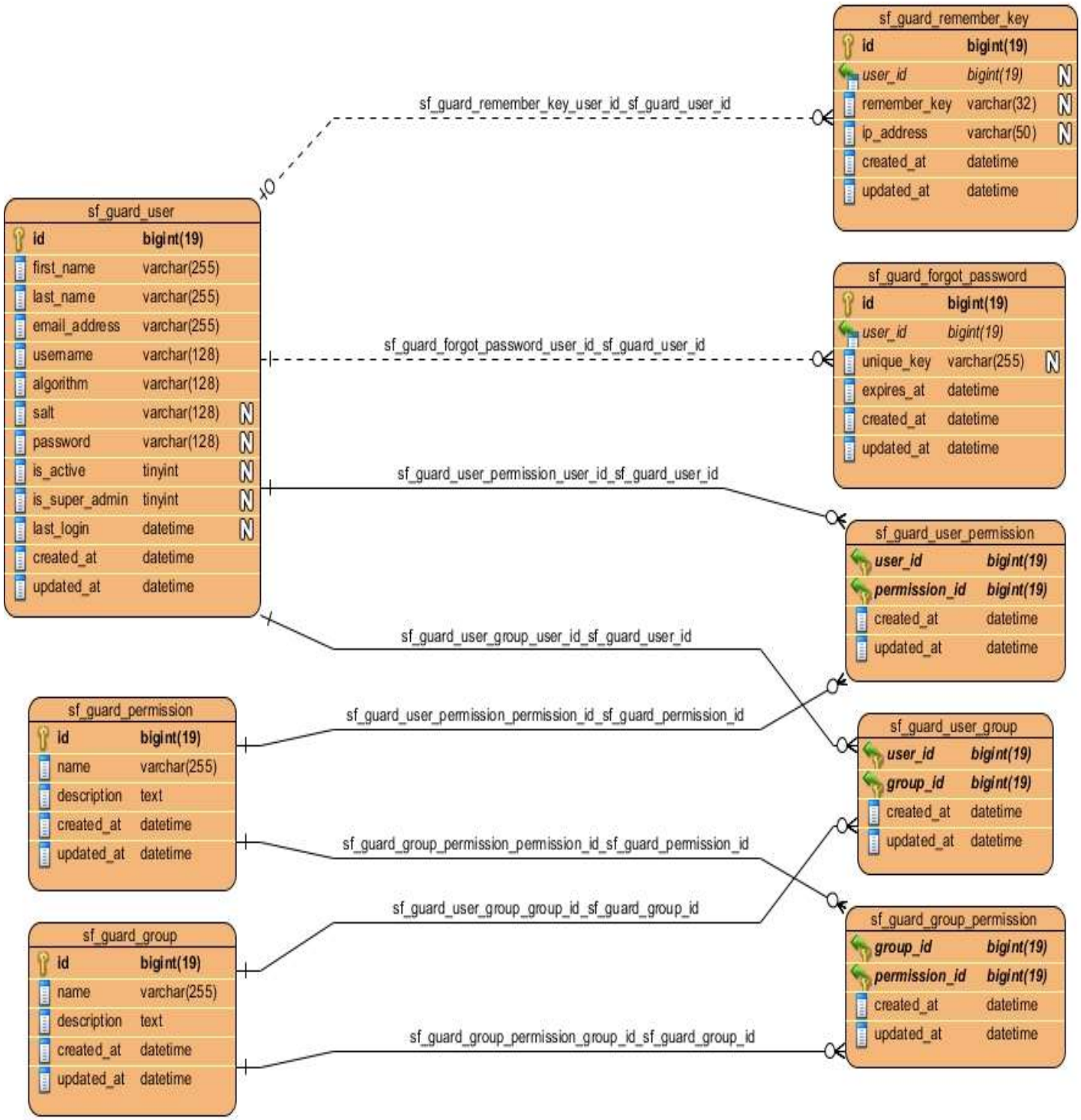

Figura 6: Diagrama de Entidad – Relación (2) (Fuente: Elaboración propia)

3.2.2. ESTRUCTURA DEL MARCO DE TRABAJO SYMFONY

Symfony está basado en el patrón arquitectónico Modelo - Vista - Controlador (MVC), implementando todas las ventajas de dicho patrón, ya que separa la vista (interfaz) y el modelo (base de datos) mediante el controlador que es el encargado de procesar las interacciones del usuario y realizar los cambios apropiados en el modelo o en la vista, pero además hace otra división en el modelo.

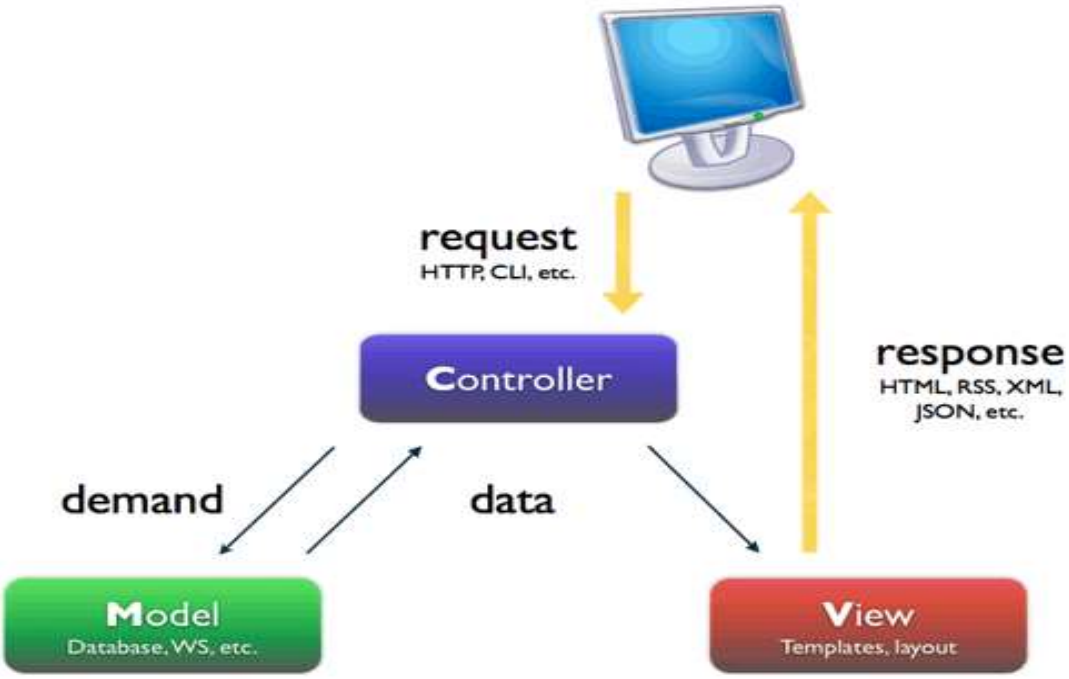

Figura 7: Funcionamiento básico del patrón MVC de Symfony (Fuente: Potencier, et al., 2010)

Modelo

Symfony divide el modelo en una capa de acceso a datos y otra de abstracción de datos. La abstracción indica que se quiere de la base de datos, y la capa de acceso hace las consultas necesarias para obtener esa información, de esta forma, si se cambia de base de datos, solamente se cambiaría la capa de acceso, y la capa de abstracción podría seguir haciendo las mismas operaciones.

Vista

En la presentación de la mayoría de las páginas existen varios elementos comunes como son: la cabecera, la navegación, el pie de página y la plantilla global conocido como *layout*, cambiando tan solo el interior o contenido de la página. Así estos tres (3) elementos quedan separados.

Controlador

El trabajo del controlador se repite para muchas acciones. Symfony crea un controlador frontal, único en la aplicación, que está encargado de realizar labores comunes como son: el manejo de las peticiones del usuario, el manejo de la seguridad, cargar la configuración de la aplicación y otras tareas similares.

Además como tantos otros marcos de trabajos, utiliza un Mapeador Objeto Relacional (ORM) para gestionar el acceso a la base de datos. Esto significa que se puede guardar, obtener, modificar y borrar información en una base de datos sin crear sentencias SQL a mano y sin tener que descender hasta los detalles más técnicos de cada base de datos. Además, la otra gran ventaja de los ORM es que se puede cambiar de una base de datos a otra simplemente cambiando una opción en un archivo de configuración (Potencier, et al., 2010).

3.2.3. PATRONES UTILIZADOS EN EL DISEÑO

Un patrón es una solución a un problema en un contexto, codifica conocimiento específico acumulado por la experiencia en un dominio. Symfony en su implementación usa una serie de patrones para su funcionamiento los cuales se mencionan a continuación:

Patrones GRASP usados:

- ✓ **Creador:** en la clase *Actions* de cada uno de los módulos se encuentran definidas las acciones del sistema y se ejecutan cada una de ellas. En las acciones se crean los objetos de las clases que representan las entidades, evidenciando de este modo que la clase *Actions* es creador de dichas entidades.

- ✓ **Experto:** este es uno de los más utilizados, puesto que Doctrine es la librería externa que utiliza Symfony para realizar su capa de abstracción en el modelo. Symfony divide el modelo en una capa de abstracción de datos y otra de acceso a datos. En estas clases se encapsula toda la lógica de los datos y son generadas las clases con todas las funcionalidades comunes de las entidades.

- ✓ **Alta cohesión:** Symfony permite asignar responsabilidades con una alta cohesión, por ejemplo la clase *Actions* tiene la responsabilidad de definir las acciones para las plantillas y colabora con otras para realizar diferentes operaciones como instanciar objetos y acceder a las propiedades, es decir,

está formada por diferentes funcionalidades que se encuentran estrechamente relacionadas proporcionando que el software sea flexible frente a grandes cambios.

- ✓ **Controlador:** todas las peticiones son manejadas por un solo controlador frontal (*sfActions*), que es el punto de entrada único de toda la aplicación en un entorno determinado. Cuando el controlador frontal recibe una petición, utiliza el sistema de enrutamiento para asociar el nombre de una acción y el nombre de un módulo con la URL entrada por el usuario.

- ✓ **Bajo Acoplamiento:** este patrón se manifiesta en cada uno de los módulos del sistema, la clase *Actions* hereda solamente de *sfActions* para lograr un bajo acoplamiento de clases.

Patrones GOF empleados:

Creacionales:

- ✓ **Singleton (Instancia única):** garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia. En el controlador frontal hay una llamada a sfContext: getInstance (). En una acción, el método getContext (), un objeto muy útil que guarda una referencia a todos los objetos del núcleo de Symfony.

- ✓ **Abstract Factory (Fábrica abstracta):** permite trabajar con objetos de distintas familias de manera que las familias no se mezclen entre sí y haciendo transparente el tipo de familia concreta que se esté usando. Cuando el framework necesita por ejemplo crear un nuevo objeto para una petición, busca en la definición de la factoría el nombre de la clase que se debe utilizar para esta tarea.

Estructurales:

- ✓ **Decorator (Envoltorio):** añade funcionalidad a una clase dinámicamente. El archivo *layout.php*, que también se denomina plantilla global, almacena el código HTML que es común a todas las páginas de la aplicación, para no tener que repetirlo en cada página. El contenido de la plantilla se integra en el layout, o si se mira desde el otro punto de vista, el layout decora la plantilla. La siguiente imagen ilustra lo anteriormente descrito.

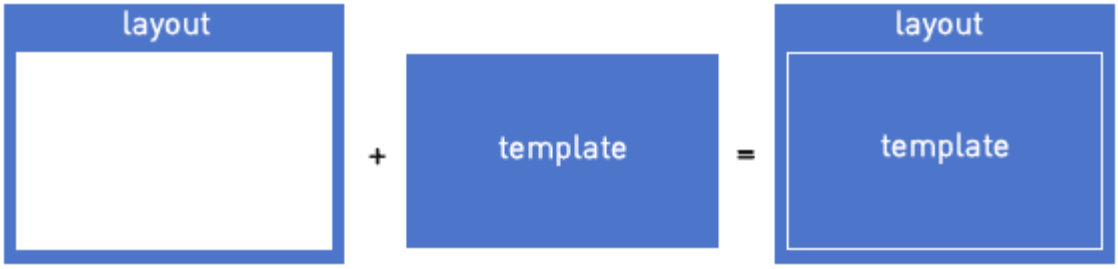

Figura 8: Funcionamiento del patrón decorador en Symfony (Fuente: García, 2005)

- ✓ **Composite (Objeto compuesto):** permite tratar objetos compuestos como si de un objeto simple se tratase. Sirve para construir objetos complejos a partir de otros más simples y similares entre sí, gracias a la composición recursiva y a una estructura en forma de árbol. Esto simplifica el tratamiento de los objetos creados, ya que al poseer todos ellos una interfaz común, se tratan todos de la misma manera.

Comportamiento:

- ✓ **Command (Acción):** permite que diferentes objetos puedan ejecutar la misma acción sin necesidad de repetir su declaración e implementación (García, 2005).

3.3. FASE DE IMPLEMENTACIÓN DEL SISTEMA

En esta fase se plantea la implementación de las historias de usuario en su correspondiente iteración, obteniéndose en cada una de ellas una versión funcional del producto. Desde el inicio se realizan revisiones del plan de iteraciones, el cual se modifica y es mejorado en caso de ser necesario, esto se hace de manera continua a lo largo del proyecto.

3.3.1. TAREAS DE LA PROGRAMACIÓN

Cada una de las historias de usuario se transformará en tareas a desarrollar, las cuales serán creadas para ayudar a organizar la implementación exitosa de las mismas. Estas tareas serán para el uso estricto de los programadores. Estas pueden ser escritas en lenguaje técnico y no necesariamente entendible por el cliente. Teniendo en cuenta la planificación realizada anteriormente, se llevaron a cabo dos iteraciones sobre el sistema, obteniéndose como finalidad un producto funcional con todas las restricciones y características deseadas.

Iteración 1

En esta iteración se desarrollan las HU de mayor prioridad en el sistema, con el objetivo de obtener una primera versión del producto con las principales características y funcionalidades para ser mostrada al cliente.

Historia de Usuario		Tarea
1	Autenticar usuario	Creación de la tabla sfGuardUser en la BD. Implementar la interfaz de autentificación.
2	Gestionar componente	Creación de la tabla tbcomponent en la BD. Implementar el adicionar componente. Implementar el modificar componente. Implementar el eliminar componente. Implementar el mostrar componente.
3	Gestionar atributo	Creación de la tabla tbatribute en la BD. Implementar el adicionar atributo de calidad. Implementar el modificar atributo de calidad. Implementar el eliminar atributo de calidad. Implementar el mostrar atributo de calidad.
4	Gestionar sub-característica	Creación de la tabla tbsubcharacteristics en la BD. Implementar el adicionar sub-característica de calidad. Implementar el modificar sub-característica de calidad.

		Implementar el eliminar sub-característica de calidad. Implementar el mostrar sub-característica de calidad.
5	Gestionar lista de chequeo	Creación de la tabla tbchecklist en la BD. Implementar el adicionar lista de chequeo. Implementar el modificar lista de chequeo. Implementar el eliminar lista de chequeo. Implementar el mostrar lista de chequeo.

Tabla 17: HU abordadas en la primera iteración (Fuente: Elaboración propia)

Iteración 2

En esta iteración se desarrollan las HU restantes, con el objetivo de obtener una primera versión del producto, con sus respectivas funcionalidades para ser mostrada al cliente.

Historia de Usuario		Tarea
1	Evaluar componente	Creación de la tabla tbevaluation en la BD. Implementar el evaluar componente.
2	Ver evaluación realizada	Implementar el mostrar evaluación realizada.

Tabla 18: HU abordadas en la segunda iteración (Fuente: Elaboración propia)

Para ver la descripción de cada tarea de ingeniería remitirse al Anexo 7: Tareas de la ingeniería.

3.4. DIAGRAMA DE DESPLIEGUE

Para modelar el hardware utilizado en las implementaciones del sistema y las relaciones entre sus componentes se utiliza el Diagrama de Despliegue, ya que este permite apreciar de forma visual cómo se encuentran relacionados físicamente los componentes de la aplicación. En este caso la aplicación se

encuentra hospedada en un servidor Web, el cual se comunica con un sistema de gestión de base de datos como se aprecia en la siguiente figura.

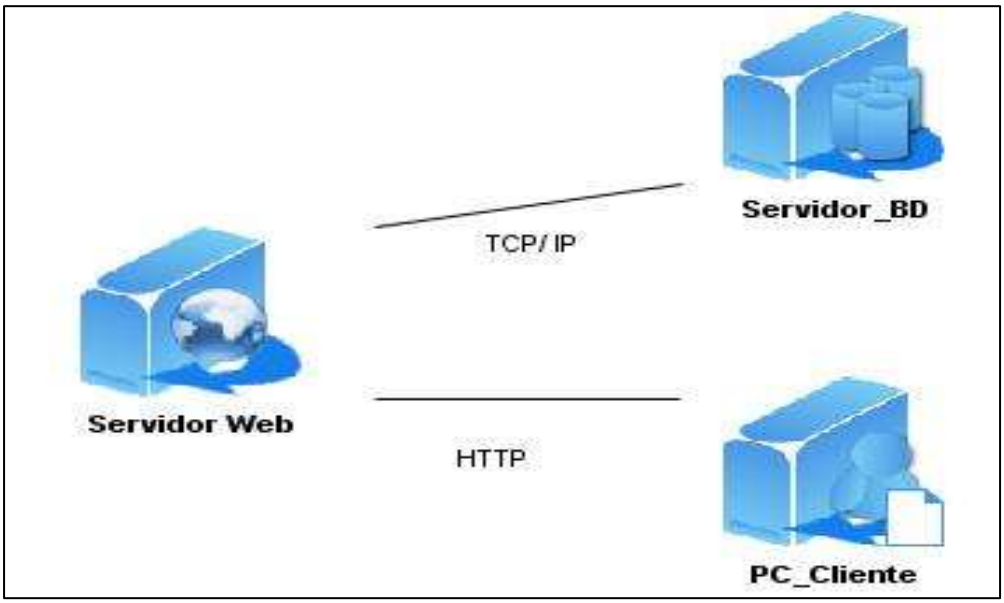

Figura 9: Diagrama de despliegue del sistema (Fuente: Elaboración propia)

Descripción de los nodos

- ✓ **Cliente:** representa una computadora desde la cual el usuario podrá acceder a la aplicación.
- ✓ **Servidor Web:** representa una estación donde estará montado el servidor Apache sobre el cual se estará ejecutando la aplicación.
- ✓ **Servidor de BD:** representa el servidor donde estará el sistema gestor de base de datos MySQL que dará respuesta a las peticiones hechas por la aplicación.

Protocolos

- ✓ **TCP/IP:** se utiliza en la comunicación entre el servidor y la base de datos para realizar operaciones sobre la información de las tablas.
- ✓ **HTTP (Protocolo de transferencia de hipertexto):** establece un esquema de comunicación cliente/servidor. El cliente es el navegador web que realiza las peticiones a las que el servidor se encarga de dar respuesta.

3.5. INTERFACES DE LA APLICACIÓN

A continuación se muestra la interfaz del módulo de administración del sistema. El resto de las interfaces de la aplicación se pueden consultar en el Anexo 8: Interfaces de la aplicación.

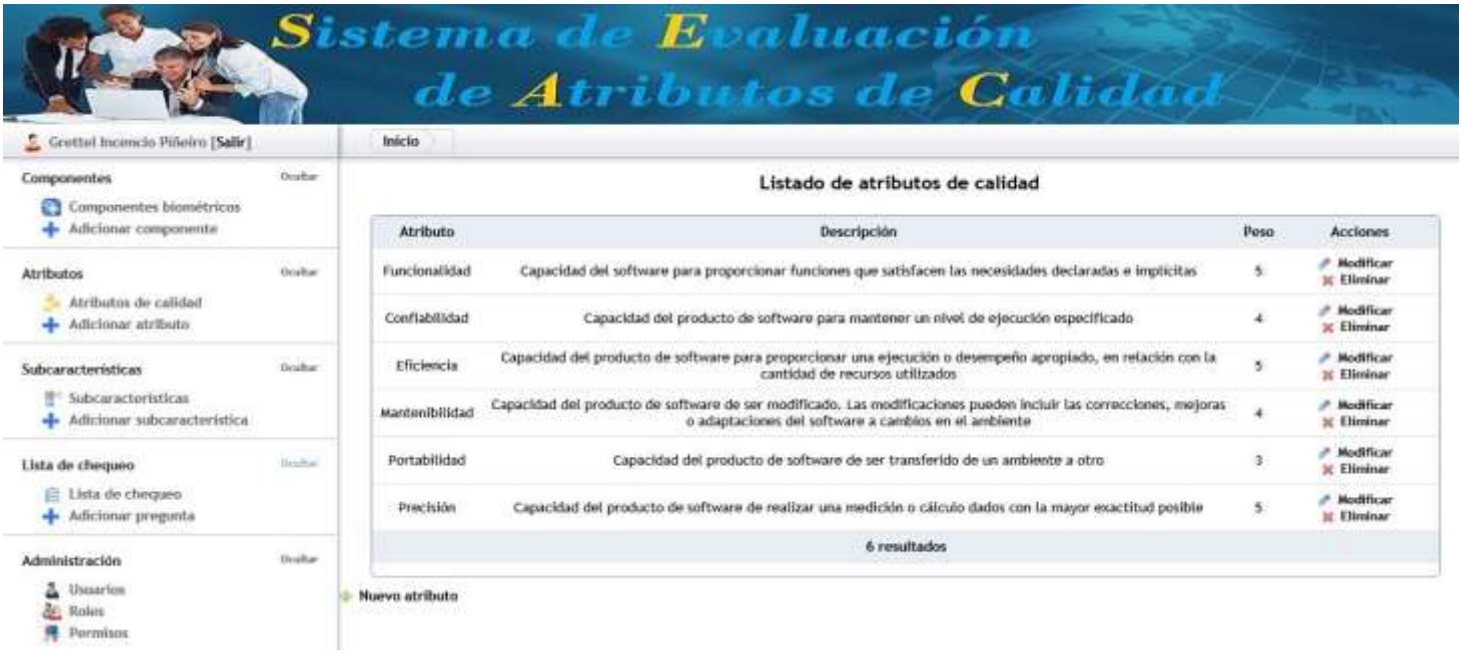

Figura 10: Interfaz del módulo de Administración del sistema (Fuente: Elaboración propia).

3.6. PRUEBAS

Una de las partes fundamentales de la metodología XP lo constituye el proceso de pruebas, el cual anima a los desarrolladores a probar constantemente. Mediante esta filosofía se minimiza la cantidad de errores no detectados y también el tiempo transcurrido entre la introducción de este en la aplicación y su detección. Todo esto contribuye a elevar la calidad del producto y a la seguridad de los desarrolladores a la hora de hacer cambios o modificaciones. La metodología XP divide las pruebas en dos grupos:

- ✓ **Pruebas unitarias:** desarrolladas por los programadores y encargadas de verificar el código de forma automática.

- ✓ **Pruebas de aceptación:** destinadas a evaluar si al final de una iteración se obtuvo la funcionalidad requerida, además de comprobar que dicha funcionalidad sea la esperada por el cliente.

Pruebas unitarias

Las pruebas unitarias se establecen antes de escribir el código y son ejecutadas constantemente ante cada modificación del sistema, esto se basa en hacer pruebas a pequeños fragmentos de código encargados de una tarea específica. El objetivo de realizar este tipo de prueba, es demostrar realizándole inspección al código de que no contiene errores, y en caso de contener proceder a su pronta eliminación, asegurando que cada módulo implementado funcione adecuadamente.

Planificación: Se planificaron dos iteraciones de pruebas por cada iteración de desarrollo, quedando planificada la realización de dos pruebas por cada funcionalidad, aplicando dos herramientas para pruebas individualmente.

En el desarrollo de las pruebas se utilizó la herramienta *Selenium IDE*, la cual es un complemento para el navegador Mozilla Firefox que permite realizar pruebas funcionales a aplicaciones web. La otra herramienta utilizada fue *SQL Inject Me*, para realizar inyecciones SQL a los formularios.

Resultados de las pruebas Unitarias

Con *Selenium IDE* se fueron grabando acciones sobre los formularios tales como: seleccionar una opción, introducir un texto, hacer clic en un botón aceptar, entre otras. Luego se repitió la prueba varias veces después de realizar varias modificaciones a las mismas funcionalidades probadas.

Con la herramienta *SQL Inject Me* se efectuó un ataque masivo a cada uno de los formularios de la aplicación, donde se detectaron dos (2) fallas en consultas que fueron corregidas utilizando de manera correcta las funciones del marco de trabajo para obtener elementos de la base de datos.

Una cuantificación de las no conformidades detectadas en cada iteración de pruebas se puede ver en el Anexo 9: Resultados de las pruebas unitarias.

Pruebas de aceptación

Las pruebas de aceptación son pruebas que se crean a partir de las historias de usuario. En ellas se especifican, desde la perspectiva del cliente, los escenarios para probar que una HU ha sido implementada correctamente. Una HU puede tener todas las pruebas de aceptación que necesite para asegurar su correcto funcionamiento. El objetivo final de estas pruebas es garantizar que los requerimientos hayan sido

cumplidos y que la aplicación es realmente lo que el cliente quería. Una HU no se considera terminada hasta que no ha pasado sus pruebas de aceptación (Pruebas de Software, 2005).

Las pruebas de aceptación se elaboran a lo largo de la iteración, en paralelo con el desarrollo del sistema, y adaptándose a los cambios que el sistema sufra. Son más importantes que las pruebas unitarias dado que significan la satisfacción del cliente con el producto desarrollado al final de una iteración. Por esta razón se decidió utilizar este tipo de prueba para garantizar el buen funcionamiento de la aplicación y así la satisfacción del cliente con el desarrollo del mismo. Para consultar los casos de pruebas realizados a la aplicación ver el Anexo 10: Casos de prueba de aceptación.

Resultados de las pruebas de Aceptación

Después de haberse realizado los casos de pruebas se confeccionó una tabla donde se recogieron las no conformidades detectadas por cada historia de usuario en cada iteración realizada al sistema. Entre los errores más relevantes se encontraron: campos no validados, mensajes de error no apropiados de acuerdo a la funcionalidad probada entre otros. Todas las dificultades fueron corregidas, ejecutándose pruebas de regresión lográndose de esta manera implementar todas las HU favorablemente, obteniéndose una aplicación que cumple con las expectativas del cliente.

Casos de prueba de aceptación	Ip1	Ip2
Autenticar usuario	3	1
Adicionar componente	2	1
Modificar componente	1	0
Eliminar componente	0	0
Adicionar atributo	3	1
Modificar atributo	1	1
Eliminar atributo	0	0

Adicionar sub-característica	1	0
Modificar sub-característica	0	0
Eliminar sub-característica	0	0
Adicionar lista de chequeo	2	1
Modificar lista de chequeo	0	0
Eliminar lista de chequeo	1	0
Evaluar componente	2	1
Mostrar evaluación	3	1

Tabla 19: No conformidades detectadas por cada caso de prueba de aceptación (Fuente: Elaboración propia)

CONCLUSIONES PARCIALES

Una vez concluidas las fases de exploración, planificación, implementación y pruebas se puede arribar a la conclusión de que el desglose de las historias de usuario en tareas de la ingeniería permitió facilitar la implementación de las funcionalidades. El desarrollo guiado por pruebas aseguró el cumplimiento de los objetivos trazados en las historias de usuario. La utilización del marco de trabajo Symfony facilitó la correcta implementación del patrón arquitectónico Modelo – Vista – Controlador, asegurando la integridad de los datos contra posibles ataques a la aplicación. El sistema, a partir de los resultados obtenidos durante la fase de prueba, se encuentra listo para desplegarse.

CONCLUSIONES GENERALES

La realización del presente trabajo ha posibilitado cumplir con los objetivos y tareas propuestas, por lo que se plantean las siguientes conclusiones:

- ✓ La investigación realizada a lo largo del proceso, permitió conocer las particularidades de los modelos de calidad más reconocidos en el área de investigación, así como las principales características del tipo de software hacia el cual va dirigida la investigación.

- ✓ Se estableció un Modelo de Calidad para evaluar atributos de calidad en componentes biométricos, lo cual permitirá garantizar la calidad de estos al ser desarrollados en el Centro. Igualmente fue descrito el proceso de evaluación especificándose los roles involucrados entre otras cuestiones.

- ✓ Se dotó al Centro de Identificación y Seguridad Digital de la Universidad de las Ciencias Informáticas de una aplicación web como herramienta complementaria, que permite la evaluación de los atributos de calidad de los componentes biométricos a través de la lista de chequeo establecida en el Modelo de Calidad establecido.

- ✓ La realización de pruebas al sistema permitió verificar la calidad del mismo, garantizando que la solución desarrollada proporcione a los usuarios una información correcta y de utilidad para el proceso de evaluación de componentes biométricos en el CISED.

RECOMENDACIONES

Las recomendaciones de la investigación están dirigidas a sugerir acciones para complementar el producto obtenido. Los objetivos de este trabajo han sido logrados de manera satisfactoria, teniendo en cuenta que se cumplieron todos los requerimientos planteados. Como resultado del proceso de investigación y construcción de la aplicación han surgido ideas que serían recomendables tener en cuenta para una futura versión de la misma, para ello se recomienda:

- ✓ Trabajar en mejoras para el diseño de la aplicación, teniendo en cuenta la opinión de los usuarios.
- ✓ Continuar el desarrollo de este sistema, adicionándole nuevas funcionalidades e integrándole servicios que sean necesarios para el Centro.
- ✓ Tener en cuenta para futuras versiones de la aplicación la posibilidad de agregarle la funcionalidad de evaluar componentes biométricos a través de las métricas establecidas en el Modelo de Calidad.
- ✓ Migrar la aplicación hacia la versión 2.0.x del marco de trabajo Symfony, recientemente salido al mercado.

REFERENCIAS BIBLIOGRÁFICAS

1. **Fillottrani, Pablo R. 2007.** Departamento de Ciencias e Ingenieria de la Programación. Modelos de calidad de software. [En línea] 2007. [Citado el: 12 de Febrero de 2011.] Disponible en: http://www.cs.uns.edu.ar/~prf/teaching/SQ07/clase6.pdf.
2. **Menéndez, Rafael y Asensio, Barzanallana. 2011.** Ingeniería del software. [En línea] 2011. [Citado el: 11 de Marzo de 2011.] Disponible en: http://www.um.es/docencia/barzana/IAGP/Enlaces/CASE_principales.html.
3. **Araujo, Yuriana , y otros. 2010.** República Bolivariana de Venezuela. [En línea] Mayo de 2010. [Citado el: 28 de Noviembre de 2010.] Disponible en: http://www.scribd.com/doc/31440864/Metodologia-RUP.
4. **Biometrics.gov. 2010.** Biometrics.gov. [En línea] 2010. [Citado el: 10 de Diciembre de 2010.] Disponible en: http://www.biometrics.gov.
5. **Calisoft. 2009.** *Métricas para Calidad de Software.* [Documento pdf] 2009.
6. **Chova, Gabriel . 2010.** Javascripts.Comunidad Astalaweb. [En línea] 2010. [Citado el: 20 de Enero de 2010.] Disponible en: http://javascripts.astalaweb.com/_inicio/Qu%C3%A9esjavascript.asp.
7. **Ciberaula. 2010.** Ciberaula. *Una introducción a Apache.* [En línea] 2010. [Citado el: 18 de Enero de 2011.] Disponible en: http://linux.ciberaula.com/articulo/linux_apache_intro/.
8. **de Antonio, Angélica. 2010.** G_Calidad.Gestión, Control y Garantía de la Calidad del Software. [En línea] 2010. [Citado el: 12 de Diciembre de 2010.] Disponible en: http://www.inf.uach.cl/rvega/asignaturas/info265/G_Calidad.pdf.
9. **Definición.org. 2010.** Definiciones en tu web. [En línea] 2010. [Citado el: 10 de Diciembre de 2010.] Disponible en: http://www.definicion.org/evaluacion.
10. **Eguiluz, Javier. 2011.** Symfony.es. [En línea] 2011. [Citado el: 18 de Enero de 2011.] Disponible en: http://www.symfony.es/que-es-symfony/.
11. **eumed.net. 2010.** Biblioteca Virtual de Derecho, Economía y Ciencias Sociales. Metodologías tradicionales y metodologías ágiles. [En línea] 2010. [Citado el: 24 de Noviembre de 2010.] Disponible en: http://www.eumed.net/libros/2009c/584/Metodologias%20tradicionales%20y%20metodologias%20agiles.htm

12. **Fernández Bertoa, Manuel, María Troya, José y Vallecillo, Antonio. 2010.** Universidad de los Andes Web del Profesor. [En línea] 2010. [Citado el: 22 de Noviembre de 2010.] Disponible en: webdelprofesor.ula.ve/.../CAC03%20Desarrollo%20de%20componentes.pdf.
13. **Ferrer Ballester, Miguel Ángel. 2010.** Verificación automática de firmas manuscritas. [En línea] 2010. [Citado el: 12 de Diciembre de 2010.] Disponible en: http://www.iec.csic.es/criptonomicon/articulos/expertos70.html.
14. **Fredy Pérez, Jhon y Martínez, Wilson. 2010.** Modelos de calidad del software. [En línea] 2010. [Citado el: 25 de Febrero de 2010.] Disponible en:http://www.slideshare.net/rolmary/modelo-de-calidaddelsoftware1.
15. **Gadgets. 2010.** mis-gadgets-scrumwave. Introducción para la gestión de proyectos con Scrum a través de google wave. [En línea] 2010. [Citado el: 28 de Octubre de 2010.] Disponible en: code.google.com/p/mis-gadgets-scrumwave/wiki/Introduccion.
16. **García, Joaquin. 2005.** IngenieroSoftware. *Análisis y Diseño. Patrones de diseño.* [En línea] 27 de Mayo de 2005. [Citado el: 24 de Abril de 2012.] Disponible en: http://www.ingenierosoftware.com/analisisydiseno/patrones-diseno.php.
17. **González Doria, Heidi. 2001.** Universidad de las Américas Puebla. Las Métricas de Software y su Uso en la Región. Tesis Licenciatura. [En línea] 2001. [Citado el: 15 de Enero de 2010.] Disponible en: http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/gonzalez_d_h/indice.html.
18. **Grupo Soluciones Innova S.A. 2007.** Grupo Soluciones Innova(GSI). Rational Rose Enterprise. [En línea] 2007. [Citado el: 8 de Marzo de 2011.] Disponible en: http://www.rational.com.ar/herramientas/roseenterprise.html.
19. **Hernández Meléndrez, Edelsys. 2006.** Repositorio de ficheros. INFOMED. Metodología de la Investigación. [En línea] 2006. [Citado el: 3 de Noviembre de 2010.] Disponible en: http://files.sld.cu/rehabilitacion/files/2010/09/como_escribir_tesis-06.pdf.
20. **Homini S.A. 2010.** Plataforma Biométrica Homini. [En línea] 2010. [Citado el: 22 de Noviembre de 2010.] Disponible en: http://www.homini.com/new_page_5.htm.
21. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James. 2000.** *El Proceso Unificado de Desarrollo de Software.* 2000. Disponible en: http://biblioteca.uci.cu/sbd/biuci/index.html.
22. **Kan, Stephen H. 2010.** Metrics and Models in Software Quality Engineering. Segunda Edición. [En línea] 2010. [Citado el: 29 de Noviembre de 2010.] Disponible en: http://www.flazx.com/ebook5838.php.

23. **López Pecho, Ramiro y Ballesteros, Julio César. 2008.** Herramientas CASE. Herramientas CASE trabajo para ingenieria de software. [En línea] 2008. [Citado el: 22 de Enero de 2011.] Disponible en: http://tpsis324.blogspot.com/2008/09/4-ventajas-y-desventajas.html.
24. **Matamoros, Soldiamar. 2011.** *Bases de Datos. Introducción a la Informática.* 2011. Disponible en: http://www.dspace.espol.edu.ec/bitstream/123456789/1416/1/2751.pdf.
25. **NC- ISO/IEC 9126-1: 2005.** Software engineering—Product quality—Part 1: Quality Model.
26. **NetBeans.org. 2011.** Sitio oficial NetBeans. [En línea] 2011. [Citado el: 18 de Enero de 2011.] Disponible en: http://netbeans.org/index_es.html.
27. **Oswald Seidler, Kai. 2011.** Apache Friends. Xampp. [En línea] 2011. [Citado el: 18 de Enero de 2011.] Disponible en: http://www.apachefriends.org/es/xampp.html.
28. **Peñalver Romero, Gladys Marsi, García De La Puente, Sergio Jesús y Meneses Abad, Abel. 2011.** Red Iberoamericana de Ingeniería de Proyectos.SXP, metodología ágil para el desarrollo de software. [En línea] 2011. [Citado el: 14 de Enero de 2011.] Disponible en: http://usbvirtual.usbcali.edu.co/ijpm/index.php?option=com_content&view=article&id=31:sxp-metodologia-agil-para-el-desarrollo-de-software&catid=2:volumen2&Itemid=2.
29. **Pérez Valdés, Damián. 2011.** Maestros del Web.¿Qué son las bases de datos? [En línea] 2011. [Citado el: 19 de Enero de 2011.] Disponible en: http://www.maestrosdelweb.com/principiantes/%C2%BFque-son-las-bases-de-datos.
30. **Potencier, Fabien y Zaninotto, François . 2010.** *Symfony 1.0, la guía definitiva.* s.l. : Apress (ISBN-13: 978-1590597866), 2010. Disponible en: http://www.librosweb.es/symfony_1_0/capitulo2/el_patron_mvc.html.
31. **Pressman, Roger S. 2002.** *Ingeniería de Software, un enfoque práctico.Quinta edición.* 2002. Disponible en: http://biblioteca.uci.cu/sbd/biuci/index.html.
32. **Pruebas de Software. 2005.** Pruebas de Software. *Gestión de Calidad y pruebas de software.* [En línea] 2005. [Citado el: 24 de Abril de 2012.] Disponible en: http://pruebasdesoftware.com/pruebadeaceptacion.htm.
33. **UCI. 2010.** Portal UCI en Internet. [En línea] 2010. [Citado el: 18 de Octubre de 2010.] Disponible en: http://www.uci.cu/.
34. **Visual Paradigm. 2011.** Visual Paradigm for UML. [En línea] 2011. [Citado el: 17 de Enero de 2011.] Disponible en: http://www.visual-paradigm.com/product/vpuml/.

35. **—. 2011.** Visual Paradigm for UML. [En línea] 2011. [Citado el: 17 de Enero de 2011.] Disponible en: http://www.visual-paradigm.com/product/vpuml/.
36. **Welling, Luke y Thomson, Laura. 2011.** *Desarrollo web con PHP y MySQL.* 2011. Disponible en: http://bibliodoc.uci.cu/pdf/reg02819.pdf..
37. **Wetpaint Team. 2010.** Ingeniería de Software. FURPS. [En línea] 2010. [Citado el: 25 de Febrero de 2010.] Disponible en: http://clases3gingsof.wetpaint.com/page/FURPS.
38. **Younis, Hamed Elmadany. 2007.** *Performance Evaluation of Biometric System.* 2007.

BIBLIOGRAFÍA CONSULTADA

1. **Ariza Rojas, M., & Molina García, J. C. (2004).** Introducción y principios básicos del desarrollo de software basado en componentes.
2. **Cabrera Montoya , Lizzet y Acosta Hinojo, Virgen Yuliet . 2008.** *Métricas estandarizadas internacionalmente. Propuestas para evaluar la calidad de los productos de software.* La Habana : s.n., 2008.
3. **Dávila Nicanor, Leticia y Mejía Alvarez, Pedro.** *Evaluacion de la Calidad de Software en Sistemas de Informacion en Internet.* Zacatenco. México : s.n.
4. **Fernández Bertoa, Manuel, María Troya, José y Vallecillo, Antonio. 2010.** Universidad de los Andes Web del Profesor. [En línea] 2010. [Citado el: 22 de Noviembre de 2010.] Disponible en: webdelprofesor.ula.ve/.../CAC03%20Desarrollo%20de%20componentes.pdf.
5. **ISO/IEC TR 9126-2. (2003).** Software engineering-Product quality-Part 2: External metrics.
6. **ISO/IEC TR 9126-3. (2003).** Software engineering-Product quality-Part 3: Internal metrics.
7. **ISO/IEC TR 9126-4. (2004).** Software engineering-Product quality-Part 4: Quality in use metrics.
8. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James. 2000.** El Proceso Unificado de Desarrollo de Software. 2000. Disponible en: http://biblioteca.uci.cu/sbd/biuci/index.html.
9. **Juez Aldana , Liliana y Izquierdo Reinoso, Alianny . 2010.** *Procedimiento para definir los requisitos de calidad en el proceso de desarrollo de software en la Universidad de las Ciencias Informáticas.* La Habana, Cuba : s.n., 2010.
10. **Potencier, Fabien y Zaninotto, François . 2010.** Symfony 1.0, la guía definitiva. s.l. : Apress (ISBN-13: 978-1590597866), 2010. Disponible en: http://www.librosweb.es/symfony_1_0/capitulo2/el_patron_mvc.html.
11. **SensioLabs. (2009).** Doctrine ORM for PHP. Guide to Doctrine for PHP.
12. **Uffo Lima, G. D., & Achang Rivas, C. (Junio de 2011).** Plataforma de pruebas para algoritmos biométricos. Biblioteca de la Universidad de las Ciencias Informáticas: Disponible en: http://repositorio_institucional.uci.cu/jspui/bitstream/ident/TD_04926_11/1/TD_04926_11.pdf

GLOSARIO DE TÉRMINOS

Ajax: acrónimo inglés de *Asynchronous JavaScript And XML (*JavaScript y XML asíncronos*).* Es una técnica de desarrollo web para crear aplicaciones interactivas.

API: acrónimo inglés de *Application Programming Interface* (Interfaz de Programación de Aplicaciones). Es el conjunto de funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción. Son usadas generalmente en las bibliotecas (también denominadas vulgarmente "librerías").

ASP: acrónimo inglés de *Active Server Pages*, es una tecnología de Microsoft del tipo "lado del servidor" para páginas web generadas dinámicamente.

Código Abierto: es una tendencia internacional del desarrollo de software que basada en la distribución del código junto a las aplicaciones.

CSS: acrónimo inglés de *Cascading Style Sheets* (Hojas de Estilo en Cascada). Son un lenguaje formal usado para definir la presentación de un documento estructurado escrito en HTML o XML.

GNU: Licencia Pública General de GNU, está orientada principalmente a proteger la libre distribución, modificación y uso de software. Su propósito es declarar que el software cubierto por esta licencia es software libre y protegerlo de intentos de apropiación que restrinjan esas libertades a los usuarios.

GUI: acrónimo inglés de G*raphical User Interface* (Interfaz Gráfica de Usuario). Es un programa informático que actúa de interfaz de usuario, utilizando un conjunto de imágenes y objetos gráficos para representar la información y acciones disponibles en la interfaz.

HTML: lenguaje estático para el desarrollo de sitios web (acrónimo en inglés de *HyperText Markup Language*, en español Lenguaje de Marcas Hipertextuales).

HTTP: Protocolo de Transmisión Hipertexto. Protocolo de comunicaciones utilizado por los programas clientes y servidores de WWW para intercambiar archivos (texto, gráfica, imágenes, sonido, video y otros archivos multimedia).

IEC: acrónimo inglés de *International Electrotechnical Commission* (Comisión Electrotécnica Internacional). Es una

organización de normalización en los campos eléctrico, electrónico y tecnologías relacionadas.

ISO: acrónimo inglés de *International Organization for Standardization* (Organización Internacional para la Estandarización).Organismo encargado de promover el desarrollo de normas internacionales.

Modelo OSI: el modelo de interconexión de sistemas abiertos (OSI) en inglés *Open System Interconnection* es un marco de referencia para la definición de arquitecturas de interconexión de sistemas de comunicaciones.

Módulo: pieza o conjunto unitario de piezas que se repiten en una construcción de cualquier tipo, para hacerla más fácil, regular y económica.

Navegador web: es un programa que permite ver la información que contiene una página web. Es un tipo de software con una interfaz gráfica que incluye botones de navegación, una barra de direcciones y una barra de estado.

ORM: acrónimo inglés de *Object-Relational Mapping* (Mapeo Objeto-Relacional). Es una interfaz que traduce la lógica de objetos a la lógica relacional debido al uso de un lenguaje orientado a objetos que se relaciona con una base de datos relacional.

Patrones GRASP: son patrones generales de software para asignación de responsabilidades. Son una serie de "buenas prácticas" de aplicación recomendable en el diseño de software.

Patrones GOF: Gang of four, grupo compuesto por Erich Gamma, Richard Helm, Ralph Johnson y John Vlisides, publicaron el libro Design Patterns en el que se recogían 23 patrones diseño comunes.

RAM: acrónimo inglés de *Random-Access Memory* (Memoria de Acceso Aleatorio). Es la memoria desde donde el procesador recibe las instrucciones y guarda los resultados.

Software Libre: es el software que, una vez obtenido, puede ser usado, copiado, estudiado, modificado y redistribuido libremente.

TCP/IP: son las siglas de Protocolo de Control de Transmisión/Protocolo de Internet (en inglés *Transmission Control Protocol/Internet Protocol*), un sistema de protocolos que hacen posibles servicios Telnet, FTP, E-mail, y otros entre ordenadores que no pertenecen a la misma red.

URL: acrónimo inglés de *Uniform Resource Locator* (Localizador Uniforme de Recurso). Es una secuencia de caracteres, de acuerdo a un formato estándar, que se usa para nombrar

recursos, como documentos e imágenes en Internet, por su localización.

WWW: red de documentos HTML intercomunicados y distribuidos entre servidores del mundo entero.

XUL: acrónimo inglés de XML-*Based User-interface Language*, (lenguaje basado en XML para la interfaz de usuario). Es la aplicación de XML a la descripción de la interfaz de usuario en el navegador Mozilla.

XULRunner: es un entorno de tiempo de ejecución elaborado por la Fundación Mozilla para ofrecer un back-end común para aplicaciones basadas en XUL.

ANEXOS

ANEXO 1: CARACTERÍSTICAS Y SUB-CARACTERÍSTICAS DE CALIDAD NORMA ISO/IEC 9126-1

1. Funcionalidad

Es la capacidad del software para proporcionar funciones que satisfacen las necesidades declaradas e implícitas cuándo el software se usa bajo las condiciones especificadas. Está determinada por la siguientes sub-características:

- ✓ **Idoneidad:** capacidad del software para mantener un conjunto apropiado de funciones para las tareas y los objetivos del usuario especificados.
- ✓ **Precisión:** capacidad del software para proporcionar efectos o resultados correctos o convenidos con el grado de exactitud necesario.
- ✓ **Interoperabilidad:** capacidad del producto de software para interactuar recíprocamente con uno o más sistemas especificados.
- ✓ **Seguridad (informática):** capacidad del producto de software para proteger información y los datos, para que personas o sistemas desautorizados no puedan leer o pueden modificar los mismos, y las personas o sistemas autorizados tenga el acceso a ellos.
- ✓ **Conformidad con la funcionalidad:** capacidad del software para adherirse a las normas que se le apliquen, convenciones, regulaciones, leyes y las prescripciones similares relativos a la funcionalidad.

2. Confiabilidad

La capacidad del producto de software para mantener un nivel de ejecución especificado cuando se usa bajo las condiciones especificadas.

- ✓ **Madurez:** capacidad del producto de software de evitar un fallo total como resultado de haberse producido un fallo del software.
- ✓ **Tolerancia ante fallos:** capacidad del producto de software de mantener un nivel de ejecución o desempeño especificado en caso de fallos del software o de infracción de su interface especificada.

- ✓ **Recuperabilidad:** capacidad del producto de software de restablecer un nivel de ejecución especificado y recuperar los datos directamente afectados en caso de fallo total.

- ✓ **Conformidad con la confiabilidad:** capacidad del producto de software para adherirse a las normas que se le apliquen, convenciones, regulaciones, leyes y las prescripciones similares relativos a la confiabilidad.

3. Usabilidad

Capacidad del producto de software de ser comprendido, aprendido, utilizado y de ser atractivo para el usuario, cuando se utilice bajo las condiciones especificadas.

- ✓ **Comprensibilidad:** capacidad del producto de software para permitirle al usuario entender si el software es idóneo, y cómo puede usarse para las tareas y condiciones de uso particulares.

- ✓ **Cognoscibilidad:** capacidad del producto del software para permitirle al usuario aprender su aplicación.

- ✓ **Operabilidad:** capacidad del producto del software para permitirle al usuario operarlo y controlarlo.

- ✓ **Atracción:** capacidad del producto del software de ser atractivo o amigable para el usuario.

- ✓ **Conformidad con la usabilidad:** capacidad del producto de software para adherirse a las normas, convenciones, guías de estilo o regulaciones relativas a la usabilidad.

4. Eficiencia

Capacidad del producto de software para proporcionar una ejecución o desempeño apropiado, en relación con la cantidad de recursos utilizados usados, bajo condiciones establecidas.

- ✓ **Rendimiento:** capacidad del producto de software para proporcionar apropiados tiempos de respuesta y procesamiento, así como tasas de producción de resultados, al realizar su función bajo condiciones establecidas.

- ✓ **Utilización de recursos:** capacidad del producto de software para utilizar la cantidad y el tipo apropiado de recursos cuando el software realiza su función bajo las condiciones establecidas.

- ✓ **Conformidad de la eficiencia:** capacidad del producto de software de adherirse a las normas o convenciones que se relacionan con la eficiencia.

5. Mantenibilidad

Capacidad del producto de software de ser modificado. Las modificaciones pueden incluir las correcciones, mejoras o adaptaciones del software a cambios en el ambiente, así como en los requisitos y las especificaciones funcionales.

- ✓ **Diagnosticabilidad:** capacidad del producto del software de ser objeto de un diagnóstico para detectar deficiencias o causas de los fallos totales en el software, o para identificar las partes que van a ser modificadas.

- ✓ **Flexibilidad:** capacidad del producto del software para permitir la aplicación de una modificación especificada.

- ✓ **Estabilidad:** capacidad del producto de software para minimizar los efectos inesperados de las modificaciones realizadas al software.

- ✓ **Contrastabilidad:** capacidad del producto del software para permitir la validación de un software modificado.

- ✓ **Conformidad de la mantenibilidad:** capacidad del producto de software para adherirse a las normas o convenciones que se relacionan con la mantenibilidad.

6. Portabilidad

Capacidad de producto de software de ser transferido de un ambiente a otro. (NC- ISO/IEC 9126-1: 2005)

- ✓ **Adaptabilidad:** capacidad del producto de software de ser adaptado a los ambientes especificados sin aplicar acciones o medios de otra manera que aquellos suministrados con el propósito de que el software cumpla sus fines.

- ✓ **Instalabilidad:** capacidad del producto de software de ser instalado en un ambiente especificado.

- ✓ **Coexistencia:** capacidad del producto de software de coexistir con otro software independiente en un ambiente común y compartir los recursos comunes.
- ✓ **Remplazabilidad:** capacidad del producto de software de ser usado en lugar de otro producto de software especificado para los mismos fines y en el mismo ambiente.
- ✓ **Conformidad con la portabilidad:** capacidad del producto de software de adherirse a las normas o convenciones relativas a la portabilidad.

Calidad durante el uso

Donde cada característica se define como sigue:

- ✓ **Eficacia:** capacidad del producto de software de permitir que los usuarios logren objetivos especificados con precisión e integridad en un contexto especificado.
- ✓ **Productividad:** capacidad del producto de software de permitir que los usuarios dediquen una cantidad de recursos apropiada en relación con la eficacia alcanzada en un contexto de uso especificado.
- ✓ **Seguridad:** capacidad del producto de software de alcanzar niveles aceptables de riesgo de daños a las personas, el negocio, el software, la propiedad o el ambiente en un contexto de uso especificado.
- ✓ **Satisfacción:** capacidad del producto de software de satisfacer a los usuarios en un contexto de uso especificado.

ANEXO 2: ESTÁNDARES ASOCIADOS A TECNOLOGÍAS BIOMÉTRICAS

Los estándares de interoperabilidad en biometría, aunque existen, se encuentran en evolución, pudiéndose distinguir entre otros, los relacionados con la interfaz de aplicación, los de formatos de intercambio de datos biométricos y los de perfiles de aplicación. El principal organismo coordinador de las actividades de estandarización biométrica a nivel mundial es el Sub-Comité 17 (SC17) del *Joint Technical Committee on Information Technology* (ISO/IEC JTC1), perteneciente a la *International Organization for Standardization* (ISO), y el *International Electrotechnical Commission* (IEC).

Mientras que en los Estados Unidos realizan un papel similar el Comité Técnico M1 del INCITS (*International Committee for Information Technology Standards*), el *National Institute of Standards and Technology* (NIST) y el *American National Standards Institute* (ANSI).

Por otra parte, existen otros organismos impulsando iniciativas en materias biométricas tales como: *Biometrics Consortium*, *International Biometrics Groups* y BioAPI[6]. Este último se estableció en Estados Unidos en 1998 compuesto por las empresas *Bioscrypt*, *Compaq*, *Iridiam*, *Infineon*, NIST, *Saflink* y *Unisis*. El Consorcio BioAPI desarrolló conjuntamente con otros consorcios y asociaciones, un estándar que promoviera la conexión entre los dispositivos biométricos y los diferentes tipos de programas de aplicación, además de promover el crecimiento de los mercados biométricos. Algunos de los estándares más importantes son (Biometrics.gov, 2010):

- ✓ **Estándar ANSI X.9.84:** creado por la ANSI (*American National Standards Institute*), define las condiciones de los sistemas biométricos para la industria de servicios financieros haciendo referencia a la transmisión y almacenamiento seguro de información biométrica, y a la seguridad del hardware asociado.

- ✓ **Estándar ANSI / INCITS 358:** creado por ANSI y BioAPI *Consortium*, este presenta una interfaz de programación de aplicación que garantiza que los productos y sistemas que cumplen este estándar son interoperables entre sí.

- ✓ **Estándar NISTIR 6529:** conocido también como CBEFF (*Common Biometric Exchange File Format*) es un estándar creado por NIST y *Biometrics Consortium* que propone un formato estandarizado (estructura lógica de archivos de datos) para el intercambio de información biométrica.

- ✓ **Estándar ANSI 378:** creado por la ANSI, establece criterios para representar e intercambiar la información de las huellas dactilares a través del uso de minucias. El propósito de esta norma es que un sistema biométrico dactilar pueda realizar procesos de verificación de identidad e identificación, empleando información biométrica proveniente de otros sistemas.

[6]Biometric Application Programming Interface: es una parte clave de las Normas Internacionales de los sistemas de apoyo que realizan la inscripción biométrica y verificación (o identificación).

- ✓ **Estándar ISO 19794-2:** creado por la ISO/IEC con propósitos similares a la norma ANSI 378, respecto a la que guarda mucha similitud.

- ✓ **Estándar PIV-071006:** creado por el NIST y el FBI en el contexto de la norma FIPS 201 del gobierno de EE.UU, establece los criterios de calidad de imagen que deben cumplir los lectores de huellas dactilares para poder ser usados en procesos de verificación de identidad en agencias federales.

ANEXO 3: COMPARACIÓN DE LA METODOLOGÍA UTILIZADA CON METODOLOGÍAS PESADAS Y CON OTRAS SIMILARES

A continuación el análisis realizado para elegir la metodología de desarrollo de software que será utilizada:

Metodologías Tradicionales o Pesadas

En el desarrollo de las metodologías, aquellas con mayor énfasis en la planificación y control del proyecto, en especificación precisa de requisitos y modelado, reciben el apelativo de Metodologías Tradicionales o Pesadas.

Estas metodologías tradicionales imponen una disciplina de trabajo sobre el proceso de desarrollo del software, con el fin de conseguir un software 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 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. Además, 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 (eumed.net, 2010).

Entre las metodologías tradicionales o pesadas se encuentran:

- ✓ RUP (Rational Unified Procces).
- ✓ MSF (Microsoft Solution Framework).
- ✓ Win-Win Spiral Model.
- ✓ Iconix.

Metodología tradicional RUP

El Proceso Unificado de Rational (RUP) es una propuesta para el desarrollo de software orientado a objetos que utiliza el UML para describir todo el proceso. El ciclo de vida organiza las tareas en fases e iteraciones. RUP divide el proceso en cuatro fases, 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 (Araujo, et al., 2010).

Entre sus principales elementos se encuentran:

- ✓ Trabajadores (quién): Define el papel de un individuo, grupo de individuos, sistema automatizado o máquina, que trabajan en conjunto como un equipo. Ellos realizan las actividades y son propietarios de elementos.
- ✓ Actividades (cómo): Es una tarea que tiene un propósito claro, es realizada por un trabajador y manipula elementos.
- ✓ Artefactos (qué): Productos tangibles del proyecto que son producidos, modificados y usados por las actividades. Pueden ser modelos, elementos dentro del modelo, código fuente y ejecutables.
- ✓ Flujo de actividades (cuándo): Secuencia de actividades realizadas por trabajadores y que produce un resultado de valor observable (Jacobson, et al., 2000).

Sus características principales son:

- ✓ Guiado por casos de uso.
- ✓ Iterativo e Incremental.
- ✓ Centrado en la Arquitectura.

Metodología ágil

Las metodologías ágiles están centradas en otras dimensiones, como por ejemplo, el factor humano o el producto de software, las cuales dan mayor valor al individuo, a la colaboración con el cliente y al desarrollo incremental del software con iteraciones muy cortas.

Metodología ágil Scrum

SCRUM, más que una metodología de desarrollo de software, es una forma de auto-gestión de los equipos de programadores. Un grupo de programadores deciden cómo hacer sus tareas y cuánto van a

tardar en ello. SCRUM ayuda a que trabajen todos juntos, en la misma dirección, con un objetivo claro. Permite además seguir de forma clara el avance de las tareas a realizar, de forma que los jefes pueden ver día a día cómo progresa el trabajo. Es una de las más conocidas metodologías ágiles, y se basa en un enfoque iterativo, donde cada iteración se denomina Sprint. El principio básico es que es muy difícil contar desde el principio con un catálogo completo de funcionalidades, ya que los requisitos van surgiendo conforme al propietario de la aplicación y los usuarios de la misma van haciendo sucesivas aportaciones. Así pues, SCRUM plantea el desarrollo de sucesivas versiones ampliadas, todas ellas plenamente usables y evaluables por el usuario. SCRUM es, además, una metodología especialmente indicada para pequeños equipos de desarrollo y se orienta a una entrega rápida de resultados y una alta flexibilidad.

Metodología ágil SXP

Hoy en día, es necesario realizar productos de software en el menor tiempo posible y elaborar sólo la documentación que sea necesaria. Por lo que para el desarrollo del sistema informático se utiliza una metodología de procedimientos ágiles. SXP es un híbrido cubano de metodologías ágiles, dicha metodología está integrada por Scrum y XP como metodologías bases, las cuales permiten actualizar los procesos de desarrollo de software para lograr un mejor resultado en el proceso de producción.

Esta metodología está dividida en cuatro fases, dichas fases son: Planificación-Definición, en esta fase se realiza el aseguramiento del financiamiento del proyecto, Desarrollo es donde se realiza la implementación del sistema, Entrega y Mantenimiento donde se realiza la puesta en marcha y el soporte. Cada una de estas fases está desplegada en flujos de trabajos de los cuales se realizan numerosas actividades que generan artefactos necesarios para respaldar la documentación de cada uno de los sistemas logrando que no queden sistemas sin ser documentados ni analizados (Ver figura 11). SXP permite fortalecer el trabajo en equipo y además continuar con el avance de las tareas a realizar; a partir de la integración de procedimientos ágiles que posibilitan la actualización de los procesos de software incrementando el interés del equipo de trabajo (Peñalver Romero, et al., 2011).

Figura 11: Fases y flujos de trabajo de SXP (Fuente: Peñalver Romero, y otros, 2011)

SXP está diseñada principalmente para proyectos de pequeños grupos de trabajo, también está orientada a una rápida entrega de resultados y que posean una alta flexibilidad, permitiendo que todos los integrantes del equipo trabajen en conjunto.

Características de SXP

- ✓ La metodología establece el uso de sistemas automatizados para la generación de algunos artefactos. Y además los recomienda de manera explícita.
- ✓ La descripción de una historia de usuario suele ser más sencilla que la de un caso de uso.
- ✓ Excelente para equipos pequeños.

- ✓ Siempre hay más código que documentación. Mientras que con la metodología RUP suele haber mucho papel y poco código (Peñalver Romero, et al., 2011).

La siguiente tabla muestra la comparación entre las metodologías ágiles y tradicionales.

Metodologías ágiles	Metodologías tradicionales
Centran su atención en la producción de código.	Centran su atención en llevar una documentación exhaustiva de todo el proyecto.
Especialmente preparados para cambios durante el proyecto.	Cierta resistencia a los cambios.
Impuestas internamente (por el equipo).	Impuestas externamente.
Proceso menos controlado, con pocos principios.	Proceso mucho más controlado, con numerosas políticas/normas.
No existe contrato tradicional o al menos es bastante flexible.	Existe un contrato prefijado.
El cliente es parte del equipo de desarrollo.	El cliente interactúa con el equipo de desarrollo mediante reuniones.
Grupos pequeños (menor de 10 integrantes) y trabajando en el mismo sitio.	Grupos grandes y posiblemente distribuidos.
Pocos artefactos.	Más artefactos.
Pocos roles.	Más roles.
Menos énfasis en la arquitectura del software.	La arquitectura del software es esencial y se expresa mediante modelos.

Tabla 20: Comparación entre metodologías ágiles y tradicionales (Fuente: eumed.net, 2010)

Comparación entre las metodologías Scrum y XP:

Semejanzas:

- ✓ Ambas son metodologías de desarrollo ágiles, basadas en los valores del manifestó ágil.
- ✓ Ambas utilizan las historias de usuario, las cuales son escritas por el propio cliente. Son una forma rápida de administrar los requerimientos de los usuarios, sin tener que elaborar gran cantidad de documentos formales y sin requerir de mucho tiempo para administrarlos. Además permiten responder rápidamente a los requerimientos cambiantes.
- ✓ Realizan continuamente entregas al cliente en cortos periodo de tiempo.

Scrum	eXtreme Programming
Es un proceso de gestión de proyectos.	Es una práctica de programación.
Cada miembro del Scrum Team trabaja de forma individual.	Se centra más en la propia programación o administración del proyecto (creación del producto).
Es un proceso de gestión de proyectos, XP es una práctica de programación.	Los miembros programan en parejas en un proyecto de XP.

Tabla 21: Diferencias entre metodologías Scrum y XP (Fuente: Elaboración propia)

ANEXO 4: HERRAMIENTAS CASE

Rational Rose

Rational Rose es una herramienta de producción y comercialización. Rose es un instrumento operativo conjunto que utiliza el Lenguaje Unificado (UML) como medio para facilitar la captura de dominio de la semántica, la arquitectura y el diseño. Esta herramienta propone la utilización de cuatro tipos de modelo para realizar un diseño del sistema, utilizando una vista estática y otra dinámica de los modelos del sistema, uno lógico y otro físico. Permite crear y refinar estas vistas creando de esta forma un modelo completo que representa el dominio del problema y el sistema de software (Grupo Soluciones Innova S.A., 2007).

Ventajas

- ✓ Modelado UML para trabajar en diseños de base de datos, con capacidad de representar la integración de los datos y los requerimientos de aplicación a través de diseños lógicos y físicos.

- ✓ Capacidad de crear definiciones de tipo de documento XML (DTD) para el uso en la aplicación.
- ✓ Integración con otras herramientas de desarrollo de Rational.
- ✓ Publicación web y generación de informes para optimizar la comunicación dentro del equipo.
- ✓ Admite la integración con otras herramientas de desarrollo (IDE's).
- ✓ Mantiene la consistencia de los modelos del sistema software.
- ✓ Generación de documentación automáticamente.
- ✓ Generación de código a partir de los modelos.
- ✓ Ingeniería inversa (crear modelo a partir código y viceversa).

Desventajas

Rational Rose presenta una pequeña desventaja y es que necesita de mucha memoria para poder de alguna forma ser manejado de forma rápida y eficiente, ya que la mayoría de los lenguajes orientados a objetos imponen una carga bastante pesada a la computadora.

ANEXO 5: HISTORIAS DE USUARIO

Historia de Usuario	
Número: 1	**Nombre:** Autenticar usuario
Usuario: Administrador del sistema y Evaluador de componentes biométricos	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 0.2	**Iteración asignada:** 1
Descripción: Se brinda la posibilidad de que el usuario que accede al sistema introduzca sus datos (usuario-contraseña) con la finalidad de verificar y otorgarle los permisos según el rol que ocupa dentro de la aplicación.	
Observaciones: El usuario y contraseña serán validados de acuerdo a lo almacenado en la respectiva tabla de la base de datos, y no contra ningún dominio específico.	

Tabla 22: HU Autenticar usuario (Fuente: Elaboración propia)

Historia de Usuario	
Número: 3	**Nombre:** Gestionar atributos de calidad
Usuario: Administrador del sistema	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 1	**Iteración asignada:** 1
Descripción: El usuario podrá adicionar, eliminar y mostrar determinados atributos así de calidad, como la modificación de sus datos: ✓ Nombre del atributo. ✓ Descripción del atributo. ✓ Peso.	
Observaciones:	

Tabla 23: HU Gestionar atributos de calidad (Fuente: Elaboración propia)

Historia de Usuario	
Número: 4	**Nombre:** Gestionar sub-características de calidad
Usuario: Administrador del sistema	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 1	**Iteración asignada:** 1
Descripción: El usuario podrá adicionar, eliminar y mostrar las sub-características de calidad así como la modificación de sus datos: ✓ Nombre de la sub-característica. ✓ Atributo al que corresponde.	

✓ Descripción. ✓ Peso.
Observaciones:

Tabla 24: HU Gestionar lista de chequeo (Fuente: Elaboración propia)

Historia de Usuario	
Número: 5	**Nombre:** Gestionar lista de chequeo
Usuario: Administrador del sistema	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 1	**Iteración asignada:** 1
Descripción: El usuario podrá adicionar, eliminar y mostrar las preguntas de la lista de chequeo así como la modificación de sus datos: ✓ Atributo. ✓ Pregunta. ✓ Peso.	
Observaciones:	

Tabla 25: HU Gestionar lista de chequeo (Fuente: Elaboración propia)

Historia de Usuario	
Número: 6	**Nombre:** Evaluar componente biométrico
Usuario: Evaluador	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta

Puntos estimados: 2	**Iteración asignada:** 2
Descripción: Inicia cuando el usuario selecciona un componente biométrico para ser evaluado. Una vez seleccionado se mostrará la lista de chequeo, la cual muestra los parámetros a evaluar. Los datos de la evaluación realizada serán almacenados por el sistema.	
Observaciones:	

Tabla 26: HU Evaluar componente (Fuente: Elaboración propia)

Historia de Usuario	
Número: 7	**Nombre:** Ver evaluaciones realizadas
Usuario: Evaluador	
Prioridad en negocio: Medio	**Riesgo en desarrollo:** Bajo
Puntos estimados: 0.3	**Iteración asignada:** 2
Descripción: Brinda la posibilidad de listar todas las evaluaciones realizadas hasta la fecha actual y mostrar los datos de la que seleccione el usuario.	
Observaciones:	

Tabla 27: HU Ver evaluaciones realizadas (Fuente: Elaboración propia)

ANEXO 6: TARJETAS CRC

Usuario	
Funcionalidades	**Colaboraciones**
Autenticación Terminar sesión	Componente Atributo Lista de chequeo Evaluación Permiso Grupo Sub-característica

Tabla 28: Tarjeta CRC Usuario (Fuente: Elaboración propia)

Componente	
Funcionalidades	**Colaboraciones**
Adicionar componente Modificar componente Eliminar componente Listar componentes	Usuario Permiso Grupo Evaluación

Tabla 29: Tarjeta CRC Componente (Fuente: Elaboración propia)

Atributo	
Funcionalidades	**Colaboraciones**
Adicionar atributo Modificar atributo Eliminar atributo Listar atributos	Usuario Permiso Grupo Evaluación Lista de chequeo Sub-característica

Tabla 30: Tarjeta CRC Atributo (Fuente: Elaboración propia)

Sub-característica	
Funcionalidades	**Colaboraciones**
Adicionar sub-característica Modificar sub-característica Eliminar sub-característica Listar sub-característica	Usuario Permiso Grupo Evaluación Lista de chequeo Atributo

Tabla 31: Tarjeta CRC Sub-característica (Fuente: Elaboración propia)

Lista de chequeo	
Funcionalidades	**Colaboraciones**
Mostrar preguntas de la lista Adicionar pregunta Modificar pregunta Eliminar pregunta	Atributo Usuario Permiso Grupo Evaluación Sub-característica

Tabla 32: Tarjeta CRC Lista de chequeo (Fuente: Elaboración propia)

Permiso	
Funcionalidades	**Colaboraciones**
Crear permiso Modificar permisos Eliminar una permios	Usuario Grupo

Tabla 33: Tarjeta CRC Permiso (Fuente: Elaboración propia)

Grupo	
Funcionalidades	**Colaboraciones**
Crear grupos	Usuario

Modificar grupos Eliminar grupos	Permiso

Tabla 34: Tarjeta CRC Grupo (Fuente: Elaboración propia)

ANEXO 7: TAREAS DE LA INGENIERÍA

A continuación se exponen cada una de las tareas de desarrollo en la que se descompuso cada historia de usuario en su iteración correspondiente.

Tareas de las historias de usuarios abordadas en la primera Iteración.

Tarea	
Número de tarea: 1	**Número de HU:** 1
Nombre de la tarea: Creación de la tabla sfGuardUser en la BD.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.1
Fecha inicio: 26 de marzo del 2012	**Fecha fin:** 26 de marzo del 2012
Programador responsable: Grettel Susel Incencio Piñeiro.	
Descripción: Se crea en la BD la tabla correspondiente a los usuarios registrados.	

Tabla 35: HU_1 Tarea 1 Creación de la tabla sfGuardUser en la BD (Fuente: Elaboración propia)

Tarea	
Número de tarea: 2	**Número de HU:** 1
Nombre de la tarea: Implementar la interfaz de autenticación.	
Tipo de tarea: Desarrollo	**Puntos estimados:**0.1
Fecha inicio: 27 de marzo del 2012	**Fecha fin:** 27 de marzo del 2012

Programador responsable: Grettel Susel Incencio Piñeiro.
Descripción: Se crea una interfaz para que el usuario inserte sus datos (usuario y contraseña).

Tabla 36: HU_1 Tarea 2 Implementar la interfaz de autenticación (Fuente: Elaboración propia)

Tarea	
Número de tarea: 1	**Número de HU:** 2
Nombre de la tarea: Creación de la tabla tbcomponent en la BD.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.1
Fecha inicio: 28 de marzo del 2012	**Fecha fin:** 28 de marzo del 2012
Programador responsable: Grettel Susel Incencio Piñeiro.	
Descripción: Se crea en la BD la tabla correspondiente a los componentes biométricos.	

Tabla 37: HU_2 Tarea 1 Creación de la tabla tbcomponent en la BD (Fuente: Elaboración propia)

Tarea	
Número de tarea: 2	**Número de HU:** 2
Nombre de la tarea: Implementar el adicionar componente.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.1
Fecha inicio: 29 de marzo del 2012	**Fecha fin:** 29 de marzo del 2012
Programador responsable: Grettel Susel Incencio Piñeiro.	
Descripción: Se implementan las funcionalidades que permitan adicionar un componente. Luego se realiza la validación de los datos, de forma tal que los mismos sean insertados de la forma esperada, sin dejarse algún campo del formulario requerido vacío.	

Tabla 38: HU_2 Tarea 2 Implementar el adicionar componente (Fuente: Elaboración propia)

Tarea	
Número de tarea: 3	**Número de HU:** 2
Nombre de la tarea: Implementar el modificar componente.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.1
Fecha inicio: 30 de marzo del 2012	**Fecha fin:** 30 de marzo del 2012
Programador responsable: Grettel Susel Incencio Piñeiro.	
Descripción: Se implementan las funcionalidades que permitan modificar un componente.	

Tabla 39: HU_2 Tarea 3 Implementar el modificar componente (Fuente: Elaboración propia)

Tarea	
Número de tarea: 4	**Número de HU:** 2
Nombre de la tarea: Implementar el eliminar componente.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.1
Fecha inicio: 2 de abril del 2012	**Fecha fin:** 2 de abril del 2012
Programador responsable: Grettel Susel Incencio Piñeiro.	
Descripción: Se implementan las funcionalidades que permitan eliminar un componente.	

Tabla 40: HU_2 Tarea 4 Implementar el eliminar componente (Fuente: Elaboración propia)

Tarea	
Número de tarea: 5	**Número de HU:** 2
Nombre de la tarea: Implementar el mostrar componente.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.1
Fecha inicio: 3 de abril del 2012	**Fecha fin:** 3 de abril del 2012
Programador responsable: Grettel Susel Incencio Piñeiro.	
Descripción: Se implementan las funcionalidades que permitan mostrar un componente.	

Tabla 41: HU_2 Tarea 5 Implementar el mostrar componente (Fuente: Elaboración propia)

Tarea	
Número de tarea: 1	**Número de HU:** 3
Nombre de la tarea: Creación de la tabla tbatribute en la BD.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.1
Fecha inicio: 4 de abril del 2012	**Fecha fin:** 4 de abril del 2012
Programador responsable: Grettel Susel Incencio Piñeiro.	
Descripción: Se crea en la BD la tabla correspondiente a los atributos de calidad con que cuenta la aplicación.	

Tabla 42: HU_3 Tarea 1 Creación de la tabla tbatribute en la BD (Fuente: Elaboración propia)

Tarea	
Número de tarea: 2	**Número de HU:** 3
Nombre de la tarea: Implementar el adicionar atributo de calidad.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.1
Fecha inicio: 5 de abril del 2012	**Fecha fin:** 5 de abril del 2012
Programador responsable: Grettel Susel Incencio Piñeiro.	
Descripción: Se implementa la funcionalidad que permita adicionar un atributo de calidad en la aplicación. Luego se realiza la validación de los datos, de forma tal que los mismos sean insertados de la forma esperada, sin dejarse algún campo del formulario requerido vacío.	

Tabla 43: HU_3 Tarea 2 Implementar el adicionar atributo de calidad (Fuente: Elaboración propia)

Tarea	
Número de tarea: 3	**Número de HU:** 3
Nombre de la tarea: Implementar el modificar atributo de calidad.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.1
Fecha inicio: 6 de abril del 2012	**Fecha fin:** 6 de abril del 2012
Programador responsable: Grettel Susel Incencio Piñeiro.	
Descripción: Se implementa la funcionalidad que permita modificar un atributo de calidad en la aplicación.	

Tabla 44: HU_3 Tarea 3 Implementar el modificar atributo de calidad (Fuente: Elaboración propia)

Tarea	
Número de tarea: 4	**Número de HU:** 3
Nombre de la tarea: Implementar el eliminar atributo de calidad.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.1
Fecha inicio: 9 de abril del 2012	**Fecha fin:** 9 de abril del 2012
Programador responsable: Grettel Susel Incencio Piñeiro.	
Descripción: Se implementa la funcionalidad que permita eliminar un atributo de calidad en la aplicación.	

Tabla 45: HU_3 Tarea 4 Implementar el eliminar atributo de calidad (Fuente: Elaboración propia)

Tarea	
Número de tarea: 5	**Número de HU:** 3
Nombre de la tarea: Implementar el mostrar atributo de calidad.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.1
Fecha inicio: 10 de abril del 2012	**Fecha fin:** 10 de abril del 2012
Programador responsable: Grettel Susel Incencio Piñeiro.	
Descripción: Se implementa la funcionalidad que permita mostrar un atributo de calidad en la aplicación.	

Tabla 46: HU_3 Tarea 5 Implementar el mostrar atributo de calidad (Fuente: Elaboración propia)

Tarea	
Número de tarea: 1	**Número de HU:** 4
Nombre de la tarea: Creación de la tabla tbsubcharacteristic en la BD.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.1
Fecha inicio: 11 de abril del 2012	**Fecha fin:** 11 de abril del 2012
Programador responsable: Grettel Susel Incencio Piñeiro.	
Descripción: Se crea en la BD la tabla correspondiente a las sub-características de calidad con que cuenta la aplicación.	

Tabla 47: HU_4 Tarea 1 Creación de la tabla tbsubcharacteristic en la BD (Fuente: Elaboración propia)

Tarea	
Número de tarea: 2	**Número de HU:** 4
Nombre de la tarea: Implementar el adicionar sub-característica de calidad.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.1
Fecha inicio: 12 de abril del 2012	**Fecha fin:** 12 de abril del 2012
Programador responsable: Grettel Susel Incencio Piñeiro.	
Descripción: Se implementa la funcionalidad que permita adicionar una sub-característica de calidad en la aplicación. Luego se realiza la validación de los datos, de forma tal que los mismos sean insertados de la forma esperada, sin dejarse algún campo del formulario requerido vacío.	

Tabla 48: HU_4 Tarea 2 Implementar el adicionar sub-característica de calidad (Fuente: Elaboración propia)

Tarea	
Número de tarea: 3	**Número de HU:** 4
Nombre de la tarea: Implementar el modificar sub-característica de calidad.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.1
Fecha inicio: 13 de abril del 2012	**Fecha fin:** 13 de abril del 2012
Programador responsable: Grettel Susel Incencio Piñeiro.	
Descripción: Se implementa la funcionalidad que permita modificar una sub-característica de calidad en la aplicación.	

Tabla 49: HU_4 Tarea 3 Implementar el modificar sub-característica de calidad (Fuente: Elaboración propia)

Tarea	
Número de tarea: 4	**Número de HU:** 4
Nombre de la tarea: Implementar el eliminar sub-característica de calidad.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.1
Fecha inicio: 16 de abril del 2012	**Fecha fin:** 16 de abril del 2012
Programador responsable: Grettel Susel Incencio Piñeiro.	
Descripción: Se implementa la funcionalidad que permita eliminar una sub-característica de calidad en la aplicación.	

Tabla 50: HU_4 Tarea 4 Implementar el eliminar sub-característica de calidad (Fuente: Elaboración propia)

Tarea	
Número de tarea: 5	**Número de HU:** 4
Nombre de la tarea: Implementar el mostrar atributo de calidad.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.1
Fecha inicio: 17 de abril del 2012	**Fecha fin:** 17 de abril del 2012
Programador responsable: Grettel Susel Incencio Piñeiro.	
Descripción: Se implementa la funcionalidad que permita mostrar una sub-característica de calidad en la aplicación.	

Tabla 51: HU_4 Tarea 5 Implementar el mostrar sub-característica de calidad (Fuente: Elaboración propia)

Tarea	
Número de tarea: 1	**Número de HU:** 5
Nombre de la tarea: Creación de la tabla tbchecklist en la BD.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.1
Fecha inicio: 18 de abril del 2012	**Fecha fin:** 18 de abril del 2012
Programador responsable: Grettel Susel Incencio Piñeiro.	
Descripción: Se crea en la BD la tabla correspondiente a la lista de chequeo con que cuenta la aplicación.	

Tabla 52: HU_5 Tarea 1 Creación de la tabla tbchecklist en la BD (Fuente: Elaboración propia)

Tarea	
Número de tarea: 2	**Número de HU:** 5
Nombre de la tarea: Implementar el adicionar lista de chequeo.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.1
Fecha inicio: 19 de abril del 2012	**Fecha fin:** 19 de abril del 2012
Programador responsable: Grettel Susel Incencio Piñeiro.	
Descripción: Se implementa la funcionalidad que permita adicionar una lista de chequeo en la aplicación. Luego se realiza la validación de los datos, de forma tal que los mismos sean insertados de la forma esperada, sin dejarse algún campo del formulario requerido vacío.	

Tabla 53: HU_5 Tarea 2 Implementar el adicionar lista de chequeo (Fuente: Elaboración propia)

Tarea	
Número de tarea: 3	**Número de HU:** 5
Nombre de la tarea: Implementar el modificar lista de chequeo.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.1
Fecha inicio: 20 de abril del 2012	**Fecha fin:** 20 de abril del 2012
Programador responsable: Grettel Susel Incencio Piñeiro.	
Descripción: Se implementa la funcionalidad que permita modificar una lista de chequeo en la aplicación.	

Tabla 54: HU_5 Tarea 3 Implementar el modificar lista de chequeo (Fuente: Elaboración propia)

Tarea	
Número de tarea: 4	**Número de HU:** 5
Nombre de la tarea: Implementar el eliminar lista de chequeo.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.1
Fecha inicio: 23 de abril del 2012	**Fecha fin:** 23 de abril del 2012
Programador responsable: Grettel Susel Incencio Piñeiro.	
Descripción: Se implementa la funcionalidad que permita eliminar una lista de chequeo en la aplicación.	

Tabla 55: HU_5 Tarea 4 Implementar el eliminar lista de chequeo (Fuente: Elaboración propia)

Tarea	
Número de tarea: 5	**Número de HU:** 5
Nombre de la tarea: Implementar el mostrar lista de chequeo.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.1
Fecha inicio: 24 de abril del 2012	**Fecha fin:** 24 de abril del 2012
Programador responsable: Grettel Susel Incencio Piñeiro.	
Descripción: Se implementa la funcionalidad que permita mostrar una lista de chequeo en la aplicación.	

Tabla 56: HU_5 Tarea 5 Implementar el mostrar lista de chequeo (Fuente: Elaboración propia)

Tareas de las historias de usuarios abordadas en la segunda Iteración.

Tarea	
Número de tarea: 1	**Número de HU:** 6
Nombre de la tarea: Creación de la tabla tbevaluation en la BD.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha inicio: 25 de abril del 2012	**Fecha fin:** 26 de abril del 2012
Programador responsable: Grettel Susel Incencio Piñeiro.	
Descripción: Se crea en la BD la tabla correspondiente a las evaluaciones.	

Tabla 57: HU_6 Tarea 1 Creación de la tabla tbevaluation en la BD (Fuente: Elaboración propia)

Tarea	
Número de tarea: 2	**Número de HU:** 6
Nombre de la tarea: Implementar el evaluar componente.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1.8
Fecha inicio: 30 de abril del 2012	**Fecha fin:** 1 de mayo del 2012
Programador responsable: Grettel Susel Incencio Piñeiro.	
Descripción: Se implementan las funcionalidades que posibilitan la captura de los datos que corresponden a la evaluación del componente.	

Tabla 58: HU_6 Tarea 2 Implementar el evaluar componente (Fuente: Elaboración propia)

Tarea	
Número de tarea: 1	**Número de HU:** 7
Nombre de la tarea: Implementar el mostrar evaluación realizada.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.3
Fecha inicio: 2 de mayo del 2012	**Fecha fin:** 4 de mayo del 2012
Programador responsable: Grettel Susel Incencio Piñeiro	
Descripción: Se implementa la funcionalidad que permita mostrar evaluaciones realizadas en la aplicación.	

Tabla 59: HU_7 Tarea 1 Implementar el mostrar evaluación realizada (Fuente: Elaboración propia)

ANEXO 8: INTERFACES DE LA APLICACIÓN

Figura 12: Interfaz "Autenticar usuario" (Fuente: Elaboración propia).

Figura 13: Interfaz "Adicionar atributo de calidad" (Fuente: Elaboración propia).

Figura 14: Interfaz "Modificar componente biométrico" (Fuente: Elaboración propia).

Figura 15: Interfaz "Eliminar componente biométrico" (Fuente: Elaboración propia).

Figura 16: Interfaz "Listar evaluaciones" (Fuente: Elaboración propia).

Figura 17: Interfaz "Nueva evaluación" (Fuente: Elaboración propia).

ANEXO 9: RESULTADOS DE LAS PRUEBAS UNITARIAS

En la siguiente tabla se muestran las no conformidades encontradas en cada una de las iteraciones de pruebas (Ipn), siendo n el número de la iteración.

Historias de Usuario			Herramientas para pruebas de software			
			Selenium IDE		SQL Inject Me	
			Ip1	Ip2	Ip1	Ip2
Iteración de desarrollo	1	Autenticar usuario	3	1	1	0
		Gestionar componentes biométricos	2	1	0	0
		Gestionar atributos de calidad	2	0	0	0
		Gestionar sub-características de calidad	2	1	0	0
		Gestionar lista de chequeo	3	1	0	0
	2	Evaluar componente biométrico	4	1	1	0
		Ver evaluaciones realizadas	3	1	0	0

Tabla 60: No conformidades detectadas por cada iteración de pruebas unitarias (Fuente: Elaboración propia)

Como en las primeras iteraciones de desarrollo están definidas las funcionalidades base para el funcionamiento de la aplicación, siendo estas las más sencillas, se obtuvo un menor número de no conformidades. En la última iteración la cantidad de no conformidades incrementó, junto a la complejidad de las funcionalidades implementadas.

ANEXO 10: CASOS DE PRUEBA DE ACEPTACIÓN

Caso de prueba de aceptación " Autenticar usuario"		
Código: HU1_P1		**Historia de Usuario:** Autenticar usuario
Descripción de la funcionalidad: Probar que se pueda autenticar un usuario en el sistema.		
Condiciones de ejecución: Usuario y contraseña		
Flujo central: En la página principal: "Entrar". Ingresar: Usuario y Contraseña. Ejecutar la opción "Entrar".		
Clases válidas	**Clases inválidas**	**Resultado esperado**
Introducción correcta de los datos.		El usuario es autenticado correctamente.
	Introducción incorrecta de los datos.	Se muestra un mensaje indicando que alguno de los datos es incorrecto, permitiendo introducirlo de nuevo.
	Campos vacíos.	Se muestra un mensaje indicando que alguno de los campos está vacío, permitiendo introducirlo de nuevo.

Tabla 61: Caso de prueba de aceptación HU1_P1 "Autenticar usuario" (Fuente: Elaboración propia)

Caso de prueba de aceptación " Adicionar componente"	
Código: HU2_P1	**Historia de Usuario:** Gestionar componentes biométricos
Descripción de la funcionalidad: Probar la funcionalidad adicionar componente.	

Condiciones de ejecución: El usuario autenticado debe ser administrativo.		
Flujo central: Ir al panel “Componentes”, escoger la opción “Adicionar componente”. Introducir los datos “Nombre”, “Descripción”, “Desarrollador”. Ejecutar la acción “Adicionar”.		
Clases válidas	**Clases inválidas**	**Resultado esperado**
Introducción correcta de los datos.		El componente es agregado correctamente.
	Introducción incorrecta de los datos.	Se muestra un mensaje indicando que alguno de los datos es incorrecto, permitiendo introducirlo de nuevo.
	Campos vacíos.	Se muestra un mensaje indicando que alguno de los campos está vacío, permitiendo introducirlo de nuevo.

Tabla 62: Caso de prueba de aceptación HU2_P1 “Adicionar componente” (Fuente: Elaboración propia)

Caso de prueba de aceptación “ Modificar componente”		
Código: HU2_P2		**Historia de Usuario:** Gestionar componentes biométricos
Descripción de la funcionalidad: Probar la funcionalidad modificar componente.		
Condiciones de ejecución: El usuario autenticado debe ser administrativo.		
Flujo central: Ir al panel “Componentes”, de la lista mostrada, seleccionar el componente a modificar y ejecutar la opción “Modificar”.		
Clases válidas	**Clases inválidas**	**Resultado esperado**
Introducción correcta de los datos.		El componente es modificado correctamente.
	Introducción incorrecta	Se muestra un mensaje indicando que alguno de los datos es incorrecto, permitiendo introducirlo de nuevo.

	de los datos.	
	Campos vacíos.	Se muestra un mensaje indicando que alguno de los campos está vacío, permitiendo introducirlo de nuevo.

Tabla 63: Caso de prueba de aceptación HU2_P2 "Modificar componente" (Fuente: Elaboración propia)

Caso de prueba de aceptación " Eliminar componente"		
Código: HU2_P3		**Historia de Usuario:** Gestionar componentes biométricos
Descripción de la funcionalidad: Probar la funcionalidad eliminar componente.		
Condiciones de ejecución: El usuario autenticado debe ser administrativo.		
Flujo central: Ir al panel "Componentes", de la lista mostrada, seleccionar el componente a eliminar y ejecutar la opción "Eliminar".		
Clases válidas	**Clases inválidas**	**Resultado esperado**
Selección del componente a eliminar.		El componente es eliminado correctamente.

Tabla 64: Caso de prueba de aceptación HU2_P3 "Eliminar componente" (Fuente: Elaboración propia)

Caso de prueba de aceptación " Adicionar atributo"	
Código: HU3_P1	**Historia de Usuario:** Gestionar atributos de calidad
Descripción de la funcionalidad: Probar la funcionalidad adicionar atributo de calidad.	
Condiciones de ejecución: El usuario autenticado debe ser administrativo.	
Flujo central: Ir al panel "Atributos", escoger la opción "Adicionar atributo". Introducir los datos "Nombre", "Descripción", "Peso". Ejecutar la acción "Adicionar".	

Clases válidas	Clases inválidas	Resultado esperado
Introducción correcta de los datos.		El atributo es agregado correctamente.
	Introducción incorrecta de los datos.	Se muestra un mensaje indicando que alguno de los datos es incorrecto, permitiendo introducirlo de nuevo.
	Campos vacíos.	Se muestra un mensaje indicando que alguno de los campos está vacío, permitiendo introducirlo de nuevo.

Tabla 65: Caso de prueba de aceptación HU3_P1 "Adicionar atributo" (Fuente: Elaboración propia)

Caso de prueba de aceptación " Modificar atributo"		
Código: HU3_P2		**Historia de Usuario: :** Gestionar atributos de calidad
Descripción de la funcionalidad: Probar la funcionalidad modificar atributo de calidad.		
Condiciones de ejecución: El usuario autenticado debe ser administrativo.		
Flujo central: Ir al panel "Atributos", de la lista mostrada, seleccionar el atributo a modificar y ejecutar la opción "Modificar".		
Clases válidas	**Clases inválidas**	**Resultado esperado**
Introducción correcta de los datos.		El atributo es modificado correctamente.
	Introducción incorrecta de los datos.	Se muestra un mensaje indicando que alguno de los datos es incorrecto, permitiendo introducirlo de nuevo.
	Campos vacíos.	Se muestra un mensaje indicando que alguno de los campos está vacío, permitiendo introducirlo de nuevo.

Tabla 66: Caso de prueba de aceptación HU3_P2 "Modificar atributo" (Fuente: Elaboración propia)

Caso de prueba de aceptación " Eliminar atributo"		
Código: HU3_P3		**Historia de Usuario:** Gestionar atributos de calidad
Descripción de la funcionalidad: Probar la funcionalidad eliminar atributo de calidad.		
Condiciones de ejecución: El usuario autenticado debe ser administrativo.		
Flujo central: Ir al panel "Atributos", de la lista mostrada, seleccionar el atributo a eliminar y ejecutar la opción "Eliminar".		
Clases válidas	**Clases inválidas**	**Resultado esperado**
Selección del atributo a eliminar.		El atributo es eliminado correctamente.

Tabla 67: Caso de prueba de aceptación HU3_P3 "Eliminar atributo" (Fuente: Elaboración propia)

Caso de prueba de aceptación " Adicionar sub-característica"		
Código: HU4_P1		**Historia de Usuario:** Gestionar sub-característica de calidad
Descripción de la funcionalidad: Probar la funcionalidad adicionar sub-característica de calidad.		
Condiciones de ejecución: El usuario autenticado debe ser administrativo.		
Flujo central: Ir al panel "Subcaracterísticas", escoger la opción "Adicionar sub-característica". Introducir los datos "Nombre", "Atributo", "Descripción", "Peso". Ejecutar la acción "Adicionar".		
Clases válidas	**Clases inválidas**	**Resultado esperado**
Introducción correcta de los datos.		La sub-característica es agregada correctamente.
	Introducción incorrecta	Se muestra un mensaje indicando que alguno de los datos es incorrecto, permitiendo introducirlo de

	de los datos.	nuevo.
	Campos vacíos.	Se muestra un mensaje indicando que alguno de los campos está vacío, permitiendo introducirlo de nuevo.

Tabla 68: Caso de prueba de aceptación HU4_P1 "Adicionar sub-característica" (Fuente: Elaboración propia)

Caso de prueba de aceptación " Modificar sub-característica"		
Código: HU4_P2		**Historia de Usuario: :** Gestionar sub-característica de calidad
Descripción de la funcionalidad: Probar la funcionalidad modificar sub-característica de calidad.		
Condiciones de ejecución: El usuario autenticado debe ser administrativo.		
Flujo central: Ir al panel "Subcaracterísticas", de la lista mostrada, seleccionar la sub-característica a modificar y ejecutar la opción "Modificar".		
Clases válidas	**Clases inválidas**	**Resultado esperado**
Introducción correcta de los datos.		La sub-característica es modificada correctamente.
	Introducción incorrecta de los datos.	Se muestra un mensaje indicando que alguno de los datos es incorrecto, permitiendo introducirlo de nuevo.
	Campos vacíos.	Se muestra un mensaje indicando que alguno de los campos está vacío, permitiendo introducirlo de nuevo.

Tabla 69: Caso de prueba de aceptación HU4_P2 "Modificar sub-característica" (Fuente: Elaboración propia)

Caso de prueba de aceptación " Eliminar sub-característica"		
Código: HU4_P3		**Historia de Usuario:** Gestionar sub-característica de calidad
Descripción de la funcionalidad: Probar la funcionalidad eliminar sub-característica de calidad.		
Condiciones de ejecución: El usuario autenticado debe ser administrativo.		
Flujo central: Ir al panel "Subcaracterísticas", de la lista mostrada, seleccionar la sub-característica a eliminar y ejecutar la opción "Eliminar".		
Clases válidas	**Clases inválidas**	**Resultado esperado**
Selección de la sub-característica a eliminar.		La sub-característica es eliminada correctamente.

Tabla 70: Caso de prueba de aceptación HU4_P3 "Eliminar sub-característica" (Fuente: Elaboración propia)

Caso de prueba de aceptación " Adicionar lista de chequeo"		
Código: HU5_P1		**Historia de Usuario:** Gestionar lista de chequeo
Descripción de la funcionalidad: Probar la funcionalidad adicionar lista de chequeo.		
Condiciones de ejecución: El usuario autenticado debe ser administrativo.		
Flujo central: Ir al panel "Lista de chequeo", escoger la opción "Adicionar pregunta". Introducir los datos "Atributo", "Pregunta", "Peso". Ejecutar la acción "Adicionar".		
Clases válidas	**Clases inválidas**	**Resultado esperado**
Introducción correcta de los datos.		La lista de chequeo es agregada correctamente.
	Introducción incorrecta	Se muestra un mensaje indicando que alguno de los datos es incorrecto, permitiendo introducirlo de

	de los datos.	nuevo.
	Campos vacíos.	Se muestra un mensaje indicando que alguno de los campos está vacío, permitiendo introducirlo de nuevo.

Tabla 71: Caso de prueba de aceptación HU5_P1 "Adicionar lista de chequeo" (Fuente: Elaboración propia)

Caso de prueba de aceptación " Modificar lista de chequeo"		
Código: HU5_P2		**Historia de Usuario: :** Gestionar lista de chequeo
Descripción de la funcionalidad: Probar la funcionalidad modificar lista de chequeo.		
Condiciones de ejecución: El usuario autenticado debe ser administrativo.		
Flujo central: Ir al panel "Lista de chequeo", de la lista mostrada, seleccionar la pregunta a modificar y ejecutar la opción "Modificar".		
Clases válidas	**Clases inválidas**	**Resultado esperado**
Introducción correcta de los datos.		La lista de chequeo es modificada correctamente.
	Introducción incorrecta de los datos.	Se muestra un mensaje indicando que alguno de los datos es incorrecto, permitiendo introducirlo de nuevo.
	Campos vacíos.	Se muestra un mensaje indicando que alguno de los campos está vacío, permitiendo introducirlo de nuevo.

Tabla 72: Caso de prueba de aceptación HU5_P2 "Modificar lista de chequeo" (Fuente: Elaboración propia)

Caso de prueba de aceptación " Eliminar lista de chequeo "	
Código: HU5_P3	**Historia de Usuario:** Gestionar lista de chequeo
Descripción de la funcionalidad: Probar la funcionalidad eliminar lista de chequeo.	
Condiciones de ejecución: El usuario autenticado debe ser administrativo.	

Flujo central: Ir al panel "Lista de chequeo", de la lista mostrada, seleccionar la pregunta a eliminar y ejecutar la opción "Eliminar".		
Clases válidas	**Clases inválidas**	**Resultado esperado**
Selección de la lista de chequeo a eliminar.		La lista de chequeo es eliminada correctamente.

Tabla 73: Caso de prueba de aceptación HU5_P3 "Eliminar lista de chequeo" (Fuente: Elaboración propia)

Caso de prueba de aceptación " Evaluar componente"		
Código: HU6_P1		**Historia de Usuario: :** Evaluar componente biométrico
Descripción de la funcionalidad: Probar la funcionalidad evaluar componente.		
Condiciones de ejecución: El usuario autenticado debe tener los permisos pertinentes.		
Flujo central: Ir al panel " Realizar evaluación ", seleccionar el componente a evaluar, introducir los datos "Nombre del evaluador", ejecutar la opción "Enviar datos ",realizar la evaluación y ejecutar la opción "Enviar datos ".		
Clases válidas	**Clases inválidas**	**Resultado esperado**
Realización de la evaluación del componente seleccionado.		El componente es evaluado correctamente.
	Campos vacíos.	Se muestra un mensaje indicando que alguno de los campos está vacío, permitiendo introducirlo de nuevo.

Tabla 74: Caso de prueba de aceptación HU6_P1 "Evaluar componente" (Fuente: Elaboración propia)

Caso de prueba de aceptación " Mostrar evaluación"		
Código: HU7_P1		**Historia de Usuario: :** Ver evaluaciones realizadas
Descripción de la funcionalidad: Probar la funcionalidad mostrar evaluación.		
Condiciones de ejecución: El usuario autenticado debe tener los permisos pertinentes. Se debe de haber realizado al menos una evaluación.		
Flujo central: Ir al panel "Evaluaciones realizadas", de la lista mostrada seleccionar el componente al cual se desea ver la evaluación y ejecutar la opción "Detalles".		
Clases válidas	**Clases inválidas**	**Resultado esperado**
Selección de la evaluación que se desea conocer el resultado.		Se muestran los resultados de la evaluación realizada.
	No se selecciona ninguna evaluación.	Se muestra un mensaje indicando que se debe seleccionar una evaluación.
	Se selecciona más de una evaluación.	Se muestra un mensaje indicando que se debe seleccionar solo una evaluación.

Tabla 75: Caso de prueba de aceptación HU7_P1 "Mostrar evaluación" (Fuente: Elaboración propia)