Universidad de las Ciencias Informáticas

Facultad 1

Título: Sistema para la verificación de personas en línea a través de huellas dactilares.

Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas.

Autor(es): Eliandis Matos Moreira

Yaricel Guerra Velázquez.

Tutor(es): MsC. Adrian Alberto Machado Cento.

Ing. Ramón Santana Fernández.

La Habana, 18 de junio de 2013

La ciencia es universal y tenemos que aprender del mundo, de la misma manera que debemos estar siempre dispuestos a mostrar el aporte que nosotros podemos obtener.

Fidel Castro Ruz

Declaración de autoría

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.
Para que así conste firmamos la presente a los ____ días del mes de _____ del año 2013.

_____ _____

Autor: Yaricel Guerra Velázquez. **Autor:** Eliandis Matos Moreira.

_____ _____

Tutor: Ing. Ramón Santana Fernández. **Tutor:** MsC. Adrian A. Machado Cento

Datos de contacto

Autor: Eliandis Matos Moreira.
Correo electrónico: ematos@estudiantes.uci.cu

Autor: Yaricel Guerra Velázquez.
Correo electrónico: ygvelazquez@estudiantes.uci.cu

Tutor: Ing. Ramón Santana Fernández.
Ingeniero en Ciencias Informáticas. Líder del proyecto Dactilab del departamento de Biometría del Centro de Identificación y Seguridad Digital (CISED), graduado en el curso 2010-2011. Tiene 2 años de experiencia en el tema. Presentó un trabajo en FESI 2011 y en Uciencia 2012 en el cual obtuvo resultados satisfactorios. Participó en el fórum de ciencia y técnica obteniendo la categoría de destacado. Todos sus trabajos han sido relacionados con los temas de reconstrucción, generación y extracción de características en huellas dactilares.
Correo electrónico: rsfernandez@uci.cu

Tutor: MsC. Adrian Alberto Machado Cento.
Graduado de Ingeniería en Ciencias Informáticas en la UCI en el año 2006. Máster en informática aplicada en la UCI en el año 2008. Profesor del departamento de Biometría del Centro de Identificación y Seguridad Digital.
Correo electrónico: amachado@uci.cu

Resumen.

Los sistemas de verificación biométrica basados en huellas dactilares constituyen herramientas cada vez más confiables para verificar la identidad de las personas. La verificación de identidad de una persona es un proceso de gran importancia para evitar fraudes de identidad, acceso a servicios, obtención de documentos o acceso a entidades por personas no autorizadas. En la actualidad la mayoría de las instituciones no cuentan con mecanismos para verificar la identidad de las personas o los mecanismos que utilizan son altamente vulnerables e inseguros y generalmente los sistemas que ofrecen este servicio son privativos y muy costosos, por lo que son pocas las entidades que pueden hacer uso de ellos, además, en ocasiones son desarrollados para un ambiente específico.

Con la utilización del “Sistema para la verificación de personas en línea a través de huellas dactilares” se logrará brindar servicios de verificación de identidad en línea, que puede ser a través de un identificador o de la huella dactilar, incrementando la seguridad y confiabilidad en las instituciones que lo implementen.

El documento recoge los resultados de la investigación realizada, describiéndose las principales características de los sistemas analizados, así como la arquitectura y el diseño del sistema propuesto. Se describen las herramientas, tecnologías utilizadas y los artefactos generados durante el proceso de desarrollo.

Palabras clave:
Huella dactilar, identidad, servicio en línea, seguridad, verificación.

Índice de contenidos.

Índice de figuras.

Índice de tablas.

Introducción.

Los avances alcanzados por las tecnologías de la información y las comunicaciones han permitido automatizar actividades que tradicionalmente eran realizadas por seres humanos. Dentro de esta amplia gama de actividades que pueden automatizarse, han cobrado importancia las relacionadas con la capacidad para establecer o verificar la identidad de los individuos y como consecuencia directa la biometría se ha transformado en un área fundamental.

El empleo de las técnicas biométricas en sistemas informáticos permite identificar y verificar de forma segura a las personas, a partir de características físicas o conductuales. Entre estas características se encuentran el rostro, el iris, la geometría de la mano, la huella dactilar, la voz, la firma manuscrita y la manera de caminar. Entre todas las técnicas de identificación biométrica existentes en la actualidad, la más ampliamente utilizada es la basada en la huella dactilar (1). Esta se basa en el análisis de la impresión que produce la parte rugosa del dedo de una persona sobre una superficie lisa. Dichas rugosidades se deben a que el dedo presenta una serie de crestas papilares que conforman un dibujo muy característico. Este dibujo se define durante el desarrollo fetal de la persona, no varía en el tiempo excepto por raspaduras, cortadas o quemaduras en la yema del dedo y es único para cada individuo. (2)

Las primeras aplicaciones de los sistemas biométricos basados en huellas dactilares tuvieron lugar dentro del ámbito legal, particularmente en el campo forense. Sin embargo, con el transcurso de los años han sido utilizados en múltiples lugares para garantizar la seguridad, por ejemplo, para el control de acceso a cuentas bancarias, control de fronteras, control de acceso a una institución, documentos electrónicos protegidos o aplicaciones de servicio social.

La suplantación de identidad es una práctica cada vez más frecuente mediante la cual delincuentes, valiéndose de datos personales de un tercero, realizan fraudes tales como la adquisición de productos, servicios o el acceso a una institución. Los casos de suplantación pueden tener implicaciones negativas para las víctimas que van desde los retiros de dinero no solicitados en las cuentas bancarias, la inclusión de reportes negativos en la historia crediticia hasta el robo de información o tecnología valiosa de una institución.

En la actualidad la verificación de identidad de una persona es un proceso de gran importancia, sin embargo la mayoría de las instituciones no cuentan con un mecanismo para verificar la identidad de una persona o los mecanismos que implementan son altamente vulnerables, produciendo como consecuencia fraudes de identidad, acceso a servicios y documentos por personas no autorizadas. El uso de métodos tradicionales como algunos documentos de identificación es altamente susceptible a fallos y la

implantación de soluciones AFIS (*Automated Fingerprint Identification System* o Sistema de Identificación Automática de Huellas Dactilares) puede ser compleja y costosa debido a la infraestructura que requieren.

Por tales razones se plantea el siguiente **problema de investigación:** Los mecanismos de seguridad en las instituciones no garantizan confiablidad en la verificación de identidad de las personas.

Para dar solución al problema propuesto se define como **objeto de estudio** el proceso de verificación de identidad a través de huellas dactilares.

Se define como **objetivo general** desarrollar un sistema que brinde servicios de verificación de personas en línea a partir de imágenes de huellas dactilares para incrementar la seguridad en las instituciones.

Objetivos específicos:

- Realizar un análisis de los sistemas existentes que presten servicios de verificación dactilar en línea.
- Definir las tecnologías, herramientas y la metodología a utilizar para el desarrollo del sistema.
- Realizar el análisis y diseño del sistema.
- Realizar la implementación de los diferentes módulos que forman parte del sistema.
- Realizar pruebas de aceptación al sistema.

Para dar cumplimiento a los objetivos se proponen las siguientes **tareas de investigación**:

- Análisis de los conceptos fundamentales sobre el proceso de verificación dactilar.
- Entrevista a los tutores para obtener información sobre el proceso de verificación dactilar.
- Análisis valorativo de las soluciones existentes que brinden servicios de verificación dactilar en línea.
- Definición de las tecnologías, herramientas y la metodología a utilizar para el desarrollo del sistema.
- Definición de las funcionalidades que debe realizar el sistema.
- Definición de los requisitos no funcionales del sistema.
- Definición de la arquitectura para guiar el desarrollo del sistema.
- Definición de los patrones para el diseño.
- Diseño de la interfaz de usuario del sistema.
- Diseño de la base de datos del sistema.
- Implementación del módulo de administración del sistema.
- Implementación del módulo de configuración del sistema.
- Implementación del módulo de verificación del sistema.

- Implementación del módulo de auditoría y control del sistema.
- Implementación del módulo de reportes del sistema.
- Realización de pruebas de aceptación al sistema.

Métodos teóricos.

- **Analítico – Sintético:** Se utiliza con el objetivo de analizar los diferentes conceptos asociados al dominio del problema. También para extraer los elementos más importantes relacionados con el tema de verificación de identidad a través de las huellas dactilares.
- **Análisis Histórico – Lógico:** Se utiliza para valorar las tendencias actuales de los sistemas que realizan el proceso de verificación de identidad a través de huellas dactilares.

Métodos empíricos.

- **Entrevista:** Se utiliza para realizar entrevistas al tutor y especialistas en huellas dactilares, con el objetivo de obtener información relacionada con el tema de verificación a través de huellas dactilares, precisar el problema a resolver y definir los procesos que se llevan a cabo.

Justificación de la investigación.

Con el desarrollo de esta investigación se proporcionará una herramienta que permitirá brindar servicios de verificación de identidad en línea mediante huellas dactilares para incrementar la seguridad en las instituciones que lo implementen. Como solución alternativa se pretende crear además, toda la infraestructura de software necesaria para gestionar otros servicios de verificación de identidad que posteriormente se desee agregar al sistema. Además se realizará un aporte tecnológico al departamento de Biometría del Centro de Identificación y Seguridad Digital.

El presente trabajo de diploma consta de tres capítulos estructurados de la siguiente manera:

Capítulo 1: **Fundamentación teórica**. En este capítulo se describen los principales conceptos para entender el problema planteado, se realiza un estudio del estado del arte del tema tratado a nivel nacional e internacional y se analizan las principales tecnologías, metodologías y herramientas para darle solución al problema planteado.

Capítulo 2: **Características del sistema**. En este capítulo se definen los requisitos funcionales y no funcionales del sistema, así como la arquitectura y los patrones utilizados para la construcción del mismo. Se modelan los diagramas y otros elementos que requiere la metodología a utilizar.

Capítulo 3: **Implementación y prueba**. En este capítulo se tratan aspectos relacionados con la implementación de la solución propuesta, se modela el diagrama de componentes y se realizan las pruebas, donde se valida el funcionamiento del sistema.

Capítulo 1: Fundamentación teórica.

Introducción.

En este capítulo se describen los principales conceptos asociados al dominio del problema. Se realiza un estudio del estado del arte sobre los sistemas de verificación biométrica basados en huellas dactilares, a nivel nacional e internacional. Se describen las principales tecnologías, metodología y herramientas utilizadas en la actualidad, analizando sus características para seleccionar la más indicada en el desarrollo del sistema.

1.1 Huella dactilar.

Las huellas dactilares son patrones constituidos por las crestas papilares de los dedos de las manos que aparecen visibles en la epidermis, generando configuraciones diversas. Las discontinuidades locales en el patrón del flujo de las crestas son conocidas como minucias. Las huellas dactilares se forman en el período fetal, a partir del sexto mes, manteniéndose invariables a través de la vida del individuo, a menos que sufran alteraciones debido a accidentes tales como cortes o quemaduras.(3) Por las características que presenta se considera como una de las técnicas biométricas más confiables para verificar la identidad de las personas, siendo la principal razón por la cual fue seleccionada para realizar el proceso de verificación biométrica en el sistema que se pretende desarrollar.

Figura 1. Huella dactilar.

1.2 Características de las huellas dactilares.

1.2.1 Valles y Crestas.

Las huellas dactilares están constituidas por rugosidades que forman salientes y depresiones. Las salientes se denominan crestas papilares y las depresiones surcos inter-papilares (o valles). Una cresta está definida como un segmento de curva y un valle es la región entre dos crestas adyacentes. (4)

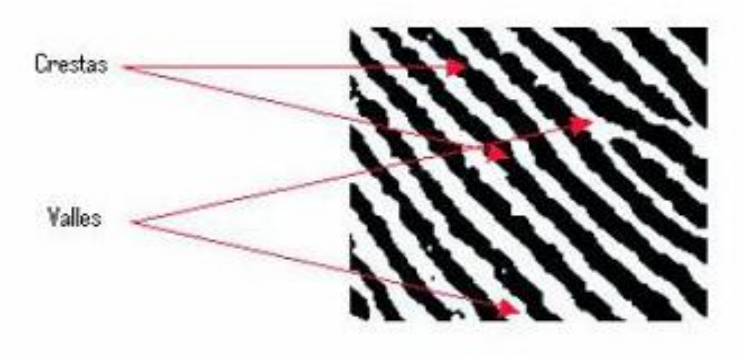

Figura 2. Crestas y Valles.

1.2.2 Minucias.

Las discontinuidades locales son conocidas como minucias. Son los puntos singulares encontrados en el trazo de las crestas. En el proceso de identificación, las que tienen mayor valor significativo son las terminaciones y bifurcaciones. En una huella dactilar existen aproximadamente de 50 a 150 minucias. (5)

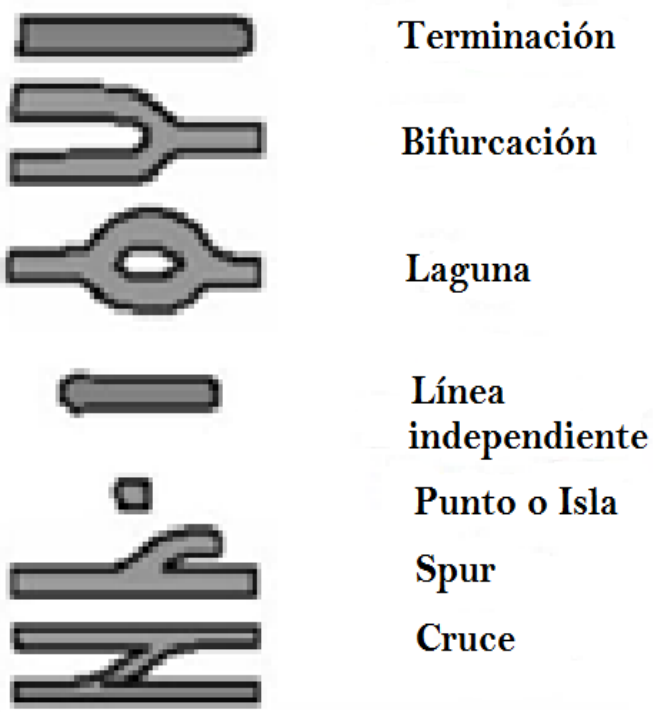

Figura 3 Tipos de minucias.

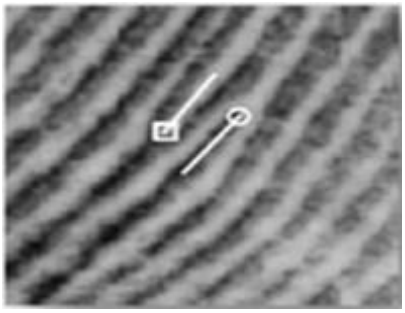

Figura 4. Bifurcación y Terminación.

1.3 Procesos de verificación e identificación biométrica.

La verificación es un proceso de combinación de uno a uno, donde se verifica si el individuo es quien dice ser, por lo que se requiere conocer la presunta identidad del mismo. El individuo proporciona su huella dactilar junto con su información de identidad, por ejemplo, su número de identificación. El sistema de comprobación de huella dactilar recupera la plantilla de la huella según el número de identificación y la compara con la huella dactilar del sujeto, adquirida en tiempo real. Si ambas coinciden (muestra previamente registrada y actual), la persona es verificada positivamente. (2) Este proceso fue seleccionado para dar solución al problema planteado debido a que solo se requiere verificar la identidad de las personas que consumen servicios de una determinada institución o acceden a ella.

La identificación es un proceso de combinación de uno a muchos, donde se pretende conocer la identidad del individuo. El mismo proporciona su huella dactilar, la cual es comparada contra una base de datos de individuos conocidos. Sin el conocimiento previo de la identidad de la persona, el sistema de identificación de huella dactilar intenta comparar su huella con aquellas que están en la base de datos. Si encuentra una huella similar, el sujeto es identificado con éxito. (2)

1.4 Servicio.

Un servicio es el conjunto de acciones o actividades generadas con el objetivo de satisfacer un deseo o necesidad de un cliente o usuario. Los servicios de verificación de identidad constituyen un mecanismo importante para garantizar la seguridad. A partir de esta investigación se pretende brindar servicios de verificación de identidad que pueden ser a través de la huella dactilar o de un identificador, permitiendo comprobar que el individuo verificado es quien dice ser.

1.5 Aplicaciones web.

Las aplicaciones web son aquellas herramientas que los usuarios pueden utilizar accediendo a un servidor web a través de Internet o de una intranet haciendo uso de un navegador. En otras palabras, son aplicaciones que se codifican en un lenguaje soportado por los navegadores web.

Los sitios pueden clasificarse en:

Estático: Enfocados a mostrar una información permanente, con ausencia de funcionalidades y cambios en el contenido. Se desarrollan principalmente mediante el lenguaje HTML o XHTML. El usuario no tiene ninguna posibilidad de seleccionar, ordenar o modificar los contenidos o el diseño de la página a su gusto. El proceso de actualización es lento, tedioso y esencialmente manual. No se pueden utilizar funcionalidades tales como bases de datos o foros. (6) La principal ventaja de este tipo de sitios es lo económico que puede resultar debido a que no requiere de muchos recursos.

Dinámico: Requieren mayor complejidad en su programación y en la utilización de bases de datos. El visitante puede alterar el diseño, contenidos o presentación de la página a su gusto. Se desarrollan utilizando diversos lenguajes y técnicas de programación. El proceso de actualización es sumamente sencillo, sin necesidad de entrar en el servidor. Se mantienen en constante actualización y modificación de manera automática. (6) Esta clasificación de sitio web es la más adecuada para el sistema que se pretende desarrollar debido a que permitiría a las diferentes instituciones que contraten el servicio de verificación a través de huellas dactilares, la modificación y actualización automática de los datos de los individuos que acceden a ella.

1.6 Navegador web.

Un navegador es un programa que permite visualizar la información que contiene una página web, interpreta el código de la página y lo presenta, permitiendo al usuario interactuar con su contenido y navegar hacia otros lugares de la red mediante enlaces o hipervínculos. Los documentos pueden estar ubicados en la computadora donde está el usuario, en cualquier otro dispositivo conectado a la misma o en internet.

Una de las limitaciones que poseen los navegadores es que no cuentan con la capacidad de acceder a los recursos locales de la estación de trabajo, incluyendo además a todos los dispositivos de hardware conectados a la misma.

1.7 Estado del arte.

1.7.1 Sistemas que permiten la comunicación con dispositivos de hardware.

Actualmente en el mundo existen sistemas que permiten la comunicación con dispositivos de hardware; tal es el caso del DGM (*Device Grid Manager)* y el XLS2 (*LiveScan Active X*).

Device Grid Manager (DGM).

El DGM es un sistema realizado por el Centro de Identificación y Seguridad Digital de la UCI en colaboración con el Ministerio del Interior de la República de Cuba (MININT). El sistema es usado por el Sistema Único de Identificación Nacional (SUIN) para la comunicación con los dispositivos de captación biométrica. Fue concebido para interactuar con estos dispositivos, tanto desde aplicaciones de escritorio como desde aplicaciones web y posee un *framework* para integrar el acceso a los dispositivos desde la web extensible y modificable. Es compatible con múltiples navegadores. Está compuesto fundamentalmente por dos aplicaciones, una para la interacción con los dispositivos, la cual se encuentra instalada localmente en las estaciones de trabajo y otra aplicación instalada en un servidor que permite llevar un control centralizado de los dispositivos. (7) Entre las características más relevantes del DGM se encuentran las siguientes:

- Permite una integración de forma fácil y segura a cualquier aplicación o módulo, de forma tal que pueda ser usado en los distintos procesos de captación definidos en un sistema.
- Permite la incorporación de nuevos dispositivos en la medida que vayan surgiendo las necesidades para nuevos módulos con solo añadir el controlador del mismo al componente.
- Es totalmente configurable por lo cual se pueden definir los dispositivos habilitados sin necesidad de recompilar la aplicación.
- Contiene complementos que gestionan la autorización para el acceso a los dispositivos.

XLS2.

XLS2 es un componente utilizado en las soluciones exportables del Centro de Identificación y Seguridad Digital (CISED), que automatiza el proceso de captura de imágenes faciales, huellas y firmas. Es compatible solo con el navegador Internet Explorer. Lo provee la compañía francesa SAGEM, actualmente llamada Morpho, y depende de una licencia y una llave de hardware que lo hacen dependiente a un solo modelo de dispositivo. Cuesta alrededor de 100.00 USD por estación. Se utiliza como un componente aislado de la arquitectura de dispositivos del CISED, lo cual representa otra limitante. (8) La dependencia

de esta herramienta con el navegador Internet Explorer y con un solo modelo de dispositivo son características que obstaculizan la facilidad del sistema propuesto para adaptarse a diversos entornos.

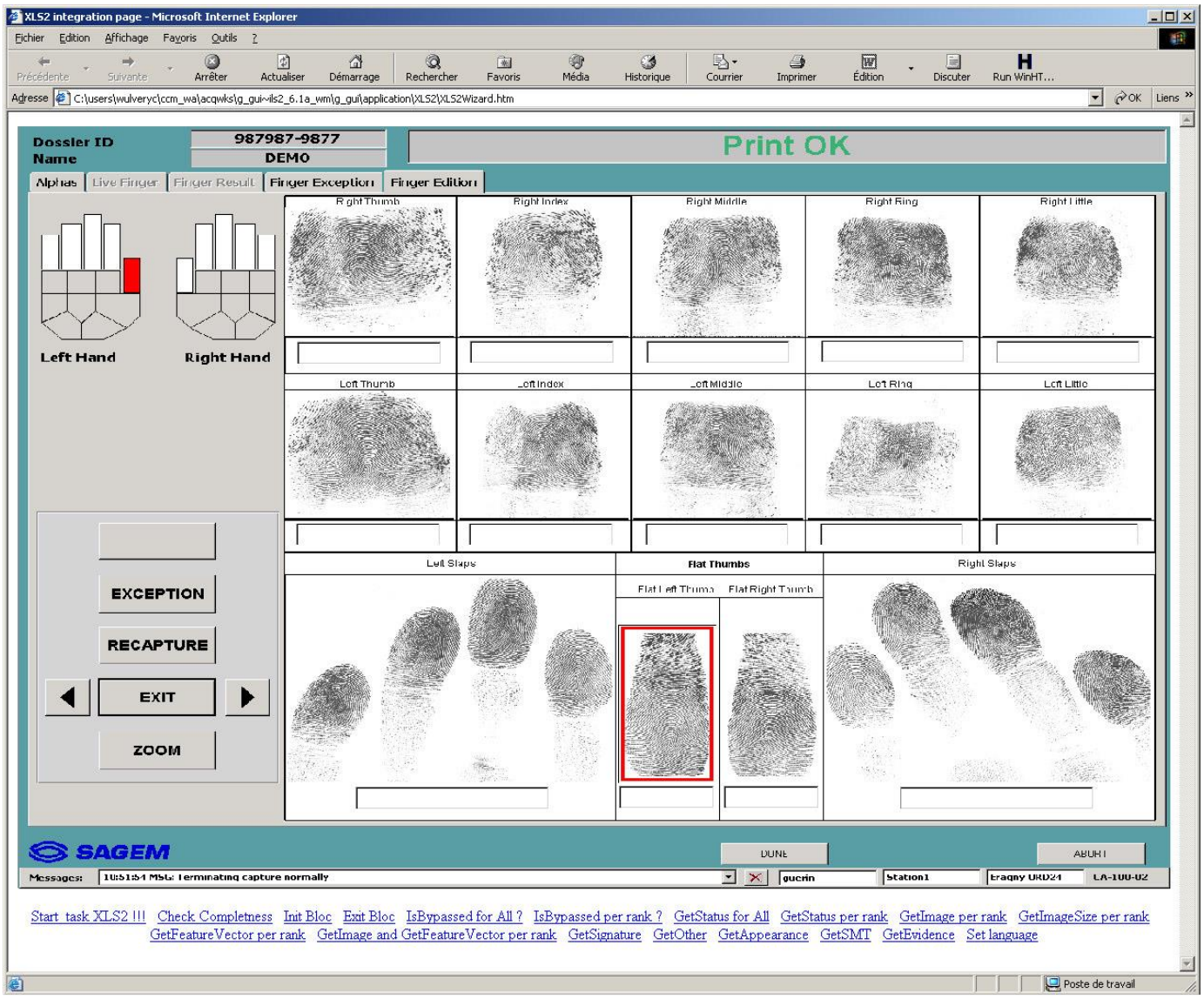

Figura 5. XLS2 en Internet Explorer.

Como parte del proceso que se desarrolla en el “Sistema para la verificación de personas en línea a través de huellas dactilares”, es necesaria la captura de datos a través de dispositivos externos. Luego de analizar estas herramientas a partir de las características y funcionalidades que brindan, haciéndolas coincidir con los requerimientos para los que serán destinados, se decide hacer uso del DGM por ser la herramienta que más se acerca a la solución del problema planteado a pesar de no estar basado en tecnologías libres.

1.7.2 Sistemas que permiten realizar la verificación de identidad a través de huellas dactilares.

Nivel internacional.

Actualmente en el mundo se han desarrollado varios sistemas que permiten verificar la identidad de los individuos a través de huellas dactilares, tal es el caso del sistema TOC y el BioPortal Server.

El **sistema TOC** es un producto desarrollado por la empresa chilena TOC, que permite verificar la identidad de los individuos a través de las huellas dactilares. Se utiliza principalmente en notarías y cooperativas de crédito de forma independiente o sea, no tiene la capacidad de brindar el servicio en línea. Este sistema usa el código de barra bidimensional que posee la cédula de identidad de los chilenos, el cual contiene entre otros datos, las minucias de la huella. Su tecnología opera comparando la huella dactilar de la persona, capturada en vivo por el lector de huellas, con las minucias contenidas en el código de barra bidimensional de la cédula de identidad, esto hace que no requiera base de datos, por lo que es bastante rápido, pero solo se limita a verificar la identidad de las personas. Su licencia es privativa, se debe pagar 0.95 dólares por punto de verificación al mes. (9) Por las características que presenta solo es usable en ambientes donde se cuente con un documento de identificación con las mismas particularidades que se describen.

Figura 6. Representación del funcionamiento de TOC.

BioPortal Server es un sistema privativo desarrollado por la empresa chilena Biometrika, que permite incluir servicios de verificación de identidad para diversos canales tales como aplicaciones Cliente/Servidor o Web (sea una Intranet corporativa, una Extranet o directamente Internet). Permite agregar nuevas tecnologías de verificación, no solamente biométricas. Tiene la capacidad de brindar servicios a múltiples empresas, manteniendo en una misma base de datos información biométrica de individuos pertenecientes a diferentes instituciones, esta característica es fundamental para brindar servicios en línea desde la misma plataforma. Además, soporta el manejo de varios dispositivos como *Digital Person*, *SecuGen*, entre otros y permite generar *tokens* biométricos con claves y certificados digitales, de forma tal que provean alternativas de verificación en la plataforma. Funciona con los más diversos gestores de bases de datos del mercado como SQL Server 2000+, Oracle 8i+, MySql y SyBase.

Por tales características, su licencia está valorada en un alto costo, por lo que no son muchas las instituciones que pueden hacer uso de esta plataforma; el precio inicial equivale a 25 000 dólares. (10) La funcionalidad de permitir agregar nuevos servicios de verificación es muy interesante para enriquecer la solución que se pretende desarrollar debido a que permitiría verificar la identidad de las personas basándose en otras características biométricas como iris, retina, rostro entre otras y de esta manera obtener una plataforma más completa.

Figura 7. Pantalla de verificación del BioPortal Server v4.1.

Nivel nacional.

Cuba cuenta con algunas entidades que intensifican la investigación y desarrollo de tecnologías biométricas, donde han obtenido significativos resultados, como la empresa Datys y el Centro de Aplicaciones de Tecnología Avanzada (CENATAV).

Biomesys AFIS es un sistema automatizado de identificación dactiloscópica desarrollado por la empresa cubana Datys. Este producto está diseñado para reducir los tiempos y aumentar la efectividad en la identificación y autenticación de personas, a partir de las impresiones dactilares, detectando los intentos de suplantación de identidad. Además, permite integrar datos biográficos de las personas registradas en la base de datos y viabiliza la entrega de documentos de identidad, con un grado de confianza acorde con la relevancia de la información que se procesa y según estándares internacionales. Este sistema puede ser utilizado por instituciones públicas o de gobierno encargadas de la identificación ciudadana, control de fronteras, inmigración y extranjería, (policía, bancos, órganos de seguridad nacional). Está soportado sobre el sistema operativo Linux y concebido con un elevado nivel de tolerancia a fallas, tanto por hardware como por software. También brinda herramientas de visualización y monitoreo. (11)

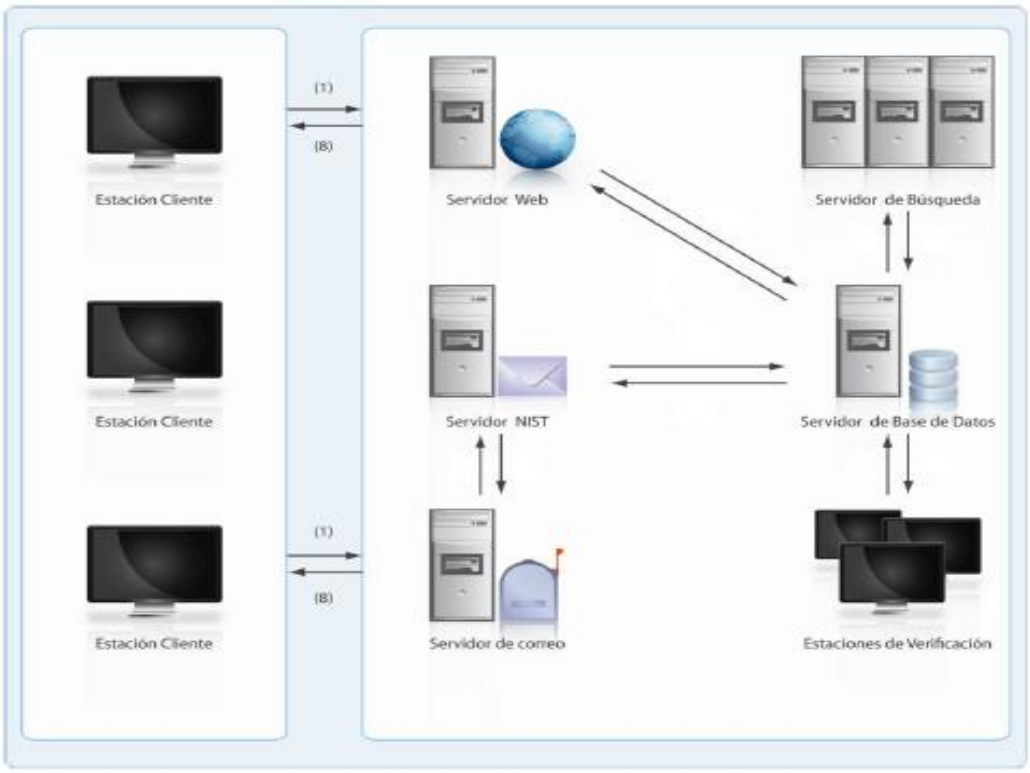

Figura 8. Flujo de solicitudes en Biomesys AFIS.

La Universidad de las Ciencias Informáticas (UCI), cuenta con un departamento de Biometría perteneciente al centro CISED, el cual se dedica a la investigación y desarrollo de soluciones biométricas, pero aunque ha obtenido algunos resultados no cuenta con un sistema capaz de brindar servicio de verificación de identidad en línea a través de huellas dactilares.

En la investigación realizada no se encuentra ningún sistema que constituya una solución total al problema y se observa que la mayoría son propietarios, hay que pagar para poder utilizarlos, lo cual va en contra del paradigma de independencia tecnológica que persigue el país; sin embargo, el estudio de estas aplicaciones permitió obtener una visión más amplia de la solución y proporcionó funcionalidades que pueden ser de interés para enriquecer la solución, como permitir agregar nuevos servicios de verificación de identidad, mostrar datos personales de los individuos verificados y guardar información biométrica de individuos pertenecientes a diferentes instituciones. Se puede señalar que el sistema **TOC,** se centra solo en verificar la identidad de las personas, no permite consultar otros datos que pueden ser de interés y como su proceso de verificación depende de la información que posee el código de barra bidimensional de la cédula de identidad, no es capaz de brindar el servicio en ambientes donde no exista este documento, por lo que limita su utilidad. El **BioPortal Server** a pesar de ser una plataforma con grandes ventajas y

muy semejante al producto que se pretende desarrollar en esta investigación, requiere de un alto costo para pagar su licencia, por lo que no todas las instituciones pueden hacer uso de tal servicio. El **Biomesys AFIS** posee grandes funcionalidades pero requiere una compleja infraestructura por lo que puede resultar muy costoso. Es por ello que se hace necesario desarrollar un sistema propio del centro CISED que responda a las necesidades planteadas.

1.8 Metodología, tecnologías y herramientas.

1.8.1 Metodología de desarrollo de software.

Las metodologías de desarrollo describen los pasos a seguir para desarrollar un producto de software, definen qué es lo que se debe hacer, cómo y quién debe hacerlo. Precisan el orden de las tareas, los artefactos que van a ser generados, quiénes son los responsables y cómo deben hacerse cada una de estas tareas durante el ciclo de desarrollo del proyecto.

La selección de la metodología a utilizar se hace en base a las características del equipo y las necesidades específicas de la situación. Para elegir una metodología de desarrollo de software se debe tener en cuenta dos factores fundamentales: el tipo de proyecto que se desea desarrollar y el tiempo que se dispone para desarrollar el mismo.

Actualmente existen dos grandes grupos de metodologías de desarrollo, las metodologías tradicionales y las ágiles; las primeras se centran en el uso exhaustivo de documentación durante todo el ciclo de vida del proyecto, mientras que las segundas dan mayor importancia a la capacidad de respuesta a los cambios. A continuación se presenta una breve comparación entre ellas.

Las metodologías ágiles dan mayor valor a la colaboración con el cliente y al desarrollo incremental del software con iteraciones muy cortas. Se basa en la filosofía de que es más importante desarrollar software que funcione y responder ante un cambio, que conseguir una buena documentación y estrictamente un plan. (12) El empleo de esta metodología para guiar el desarrollo del sistema propuesto permitiría obtener un producto aceptable en menor tiempo.

Las metodologías tradicionales llevan un control estricto del proceso, estableciendo rigurosamente las actividades involucradas, los artefactos que se deben producir y las notaciones que se usarán. Centran su atención en llevar una documentación exhaustiva de todo el proyecto y en cumplir con un plan de proyecto. (12)

Después de haber realizado un análisis sobre las principales características de las metodologías ágiles y las tradicionales se observa que las primeras se adecuan más para guiar el desarrollo del sistema

propuesto, principalmente porque el proyecto no es altamente complejo, se cuenta con pocos roles y no se dispone de mucho tiempo. Entre las metodologías ágiles más conocidas se encuentran: *eXtreme Programming* (XP), *Scrum y Feature Driven Development* (FDD).

Extreme Programming (XP) (13).

XP es una metodología de desarrollo de software diseñada para entregar el software que el cliente solicita, en el momento que lo necesita, además promueve el uso de prácticas para aumentar la productividad del equipo de desarrollo y mejorar la adaptabilidad a los frecuentes cambios dentro del ciclo de vida del proyecto.

Algunas de las prácticas más usadas son:

- Entregas pequeñas y frecuentes.
- Cliente in-situ.
- Diseños simples.
- Integración continua.
- Programación en parejas.
- Desarrollo guiado por pruebas.
- Estándares y refactorización de código.

Características de la metodología XP (14):

- Empieza en pequeño y añade funcionalidad con retroalimentación continua.
- El cliente o el usuario se convierten en miembro del equipo.
- El proyecto es pequeño.
- Propiedad colectiva del código.
- Comunicación de los programadores a través del código.

Ventajas:

- Apropiado para entornos volátiles, equipos de desarrollo pequeños y proyectos de alto riesgo.
- Permite una mejor adaptabilidad a los cambios, que se traduce en una reducción de costos.
- Planificación a corto plazo y más transparente para los clientes, ya que conocen las fechas de entrega de funcionalidades vitales para su negocio.
- Permite tener retroalimentación continua de los usuarios a través de las entregas frecuentes.
- La presión está a lo largo de todo el proyecto y no en una entrega final.

Desventajas:

- A veces cuesta delimitar el alcance del proyecto con el cliente.

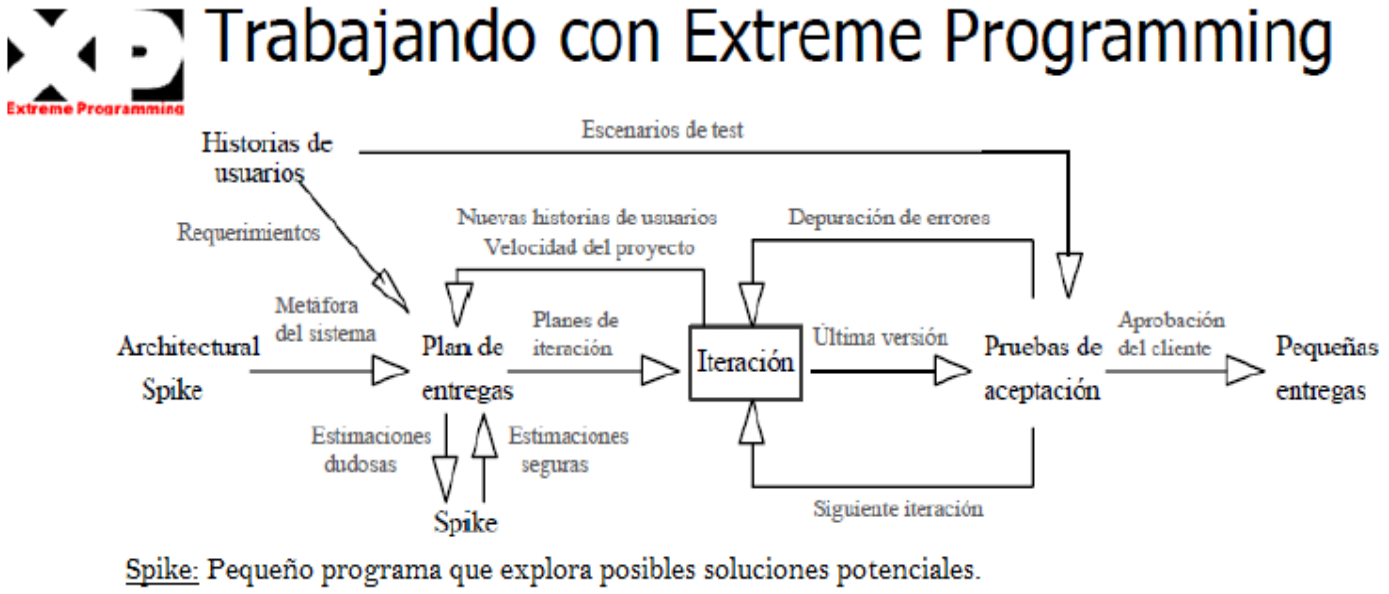

Figura 9. Eventos principales de un proyecto con XP.

Scrum.

Scrum es un marco de trabajo iterativo e incremental para el desarrollo de proyectos, productos y aplicaciones. Estructura el desarrollo en ciclos de trabajo llamados Sprint, estas son iteraciones de 1 a 4 semanas, y se van ejecutando una detrás de otra con duración fija, terminan en una fecha específica aunque no se haya finalizado el trabajo y nunca se alargan, por lo que se limitan en tiempo. Al comienzo de cada Sprint, un equipo multifuncional selecciona los elementos de una lista priorizada y al final, el equipo lo revisa con los interesados en el proyecto y les enseña lo que han construido, obteniendo comentarios y observaciones que se puede incorporar al siguiente.

Scrum hace énfasis en que los productos funcionen al final de cada iteración; en el caso del software significa que el código esté integrado, completamente probado y potencialmente para entregar. El desarrollo inevitablemente implica aprender, innovación y sorpresas, por eso esta metodología hace hincapié en dar un pequeño paso de desarrollo e inspeccionar el producto resultante y la eficacia de las prácticas actuales para adaptar el objetivo del producto y las prácticas del proceso, y volver a repetir. (15) Como se ha descrito anteriormente las iteraciones en esta metodología son demasiado inflexibles lo que puede traer como resultados proyectos incompletos. Implicaría contar con un equipo de desarrollo

preparado y con experiencia para cumplir con las fechas definidas para cada ciclo de trabajo a fin de obtener un producto completamente probado en el tiempo requerido.

Figura 10. Eventos principales de un proyecto con Scrum.

Feature Driven Development (FDD).

La metodología FDD se utiliza en proyectos pequeños de corta duración. Su funcionamiento está basado en iteraciones cortas, produciendo un software que permite monitorizar los procesos en cuanto al avance obtenido. Las iteraciones se deciden a través de las funcionalidades que son pequeñas partes del software. (16)

La metodología FDD se divide en 5 fases:

- Desarrollo de un modelo general.
- Construcción de una lista de funcionalidades.
- Plan de entregas en base a las funcionalidades a implementar.
- Diseñar en base a las funcionalidades.
- Implementar en base a las funcionalidades.

FDD define métricas para controlar cómo va el desarrollo del proyecto y que se pueda estimar mejor el proyecto en un futuro. El principal problema de este proceso está representado por la necesidad de contar con miembros experimentados en el equipo, miembros que puedan definir los roles y funciones de cada uno y que marquen el camino a seguir desde el principio con la elaboración del modelo global. Al contrario

de XP, el código fuente tiene propietario. Los equipos varían en cuanto a la funcionalidad a implementar. El conocimiento de la aplicación se reparte a través del trabajo en equipo y revisiones ya que dicha metodología presenta una jerarquía dentro del equipo de desarrollo.

Fundamentación de la metodología a utilizar.

Se identificaron ciertos factores que permitieron la selección de XP como metodología a utilizar, los elementos determinantes fueron:

- El desarrollo en este caso es realizado por pocas personas, XP está dirigida a grupos de desarrollo pequeños y con pocos roles.
- El desarrollo del sistema comienza a partir de los requerimientos básicos y a partir de ahí se van añadiendo funcionalidades que tanto el desarrollador como el cliente entiendan necesarias. XP añade funcionalidad con retroalimentación continua.
- Se requiere de un desarrollo realizando pequeñas y continuas mejoras. XP plantea que a medida que el proyecto avanza pueden surgir nuevas expectativas o ideas que pueden ser incorporadas fácilmente permitiéndole mayor adaptabilidad al producto.
- Propiedad colectiva del código. XP plantea que todos los programadores pueden realizar cambios en cualquier parte del código en cualquier momento.

1.8.2 Framework de desarrollo web.

Un *framework*, constituye una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Los objetivos principales que posee es acelerar el proceso de desarrollo de un sistema, reutilizar código ya existente y promover buenas prácticas como el uso de patrones de diseños (17). Existen una variada cantidad de *frameworks* que son destinados solamente para la creación de aplicaciones web. Dentro de los más usados y conocidos se encuentran ASP.NET, Symfony, Grails entre muchos otros.

ASP.NET

ASP (*Active Server Pages o* Páginas de Servidor Activas) es una tecnología de Microsoft del tipo lado del servidor para páginas web generadas dinámicamente. Uno de los objetivos principales de ASP.NET es aprovechar toda la capacidad del CLR (*Common Languaje Runtime* o Entorno en Tiempo de Ejecución de Lenguaje Común), el cual compila en algún punto todos los códigos de aplicaciones en códigos naturales de máquina. En este *framework* el código se separa de la lógica de la aplicación, brinda la posibilidad de escoger el lenguaje que desee el programador, por defecto trae integrado C# y VB.NET. Otra

característica importante es que tiene integrado librerías de clases que son comunes en toda la plataforma .NET, esto permite crear aplicaciones con un considerable ahorro de líneas de código, por sus múltiples funciones es más lento que la mayoría de los *frameworks* de desarrollo web, pero sin lugar a dudas el mayor inconveniente que presenta, es su condición de propietario (18).

Symfony

Symfony es un *framework* diseñado para optimizar el desarrollo de aplicaciones web. Se encuentra desarrollado completamente con PHP 5. Separa la lógica del negocio, la lógica del servidor y la presentación de la aplicación. Reduce considerablemente el tiempo de desarrollo de una aplicación web compleja. Automatiza las tareas más comunes y reutiliza el código cada vez que se crea una nueva aplicación web. Tiene compatibilidad con la mayoría de los gestores de base de datos. Es multiplataforma, sin embargo, su mapeo de objeto relacional para conectarse a las bases de datos es propenso a lentitud, pues conlleva a gastos innecesarios en la generación y configuración del código. Al estar desarrollado con PHP 5 se debe tener un avanzado conocimiento de este lenguaje. (19)

Grails

Grails es un *framework* de desarrollo para aplicaciones web, dentro de la plataforma Java, que utiliza el lenguaje de programación Groovy, se basa en el patrón Modelo Vista Controlador (MVC) y soporta Ajax. Permite que se le agreguen extensiones desarrolladas por terceros y generar el controlador y las vistas necesarias para realizar operaciones CRUD (acrónimo para las operaciones básicas de Creación, Lectura, Actualización y Borrado). Abarca las tres capas del desarrollo, acceso a la base de datos, capa de negocio y presentación.

Grails consigue un mayor rendimiento basándose en otros *frameworks* de desarrollo de código abierto, principalmente Spring e Hibernate que son potentes en su campo y está basado en los paradigmas “convención sobre configuración” y “DRY” (*don’t repite yourself*) o no te repitas, que permite al programador evitar la duplicidad de código y el mayor número de cambios en distintas partes del código para conseguir una mejor evolución y mantenimiento de la aplicación. Su principal característica es que está diseñado para que se programe en Groovy, el cual es un lenguaje dinámico basado en Java, pero que incorpora muchas funcionalidades nuevas permitiendo programar con menos código. Grails es una plataforma completa, puesto que incluye también un contenedor web, bases de datos, sistemas de empaquetado de la aplicación y un completo sistema para la realización de pruebas sobre la aplicación. (20)

Fundamentación del framework seleccionado.

El *framework* seleccionado fue Grails principalmente con el objetivo de facilitar el proceso de integración de los *applet* de java que realizan los procesos de extracción y comparación de minucias. Posee una corta gestión de configuración para el comienzo de proyectos, lo que permite crear aplicaciones web en menor tiempo. Además utiliza el lenguaje de programación Groovy, que permite la creación de propiedades y métodos dinámicos en los objetos de la aplicación, por lo que se necesita menos código para obtener el mismo resultado que si se hiciera en java tradicional. Permite generar el controlador y las vistas necesarias para realizar operaciones CRUD y abarca las tres capas del desarrollo, acceso a la base de datos, capa de negocio y presentación.

1.8.3 Lenguajes de programación.

Groovy

Groovy es un lenguaje dinámico y flexible basado en java, con él se puede ahorrar casi un 90% de líneas de código respecto al lenguaje Java. Incorpora además poderosas características de lenguajes como Phyton, Ruby y Smalltalk. Permite la integración con todos los objetos y librerías existentes en Java. Entre sus principales características se encuentran (21):

- Closures: bloque de código anónimo definido entre llaves, el cual toma argumentos y retorna valores.
- Sintaxis nativas para *List y Maps.*
- Soporte nativo para expresiones regulares.
- Soporte para tipificación dinámica y estática.
- Soporta código embebido dentro de cadenas.

JavaScript

JavaScript es un lenguaje de *script* basado en objetos, utilizado para acceder a objetos en aplicaciones. Principalmente se utiliza integrado en un navegador web permitiendo el desarrollo de interfaces de usuario mejoradas y páginas web dinámicas. Aunque comparte muchas de las características y de las estructuras del lenguaje Java, fue desarrollado independientemente. El lenguaje JavaScript puede interactuar con el código HTML, permitiendo a los desarrolladores web utilizar contenido dinámico. No requiere de compilación ya que el lenguaje funciona del lado del cliente, los navegadores son los encargados de interpretar estos códigos. Es utilizado para crear pequeños programas que luego son insertados en una página web y en programas más grandes, orientados a objetos mucho más complejos.

1.8.4 Entorno de Desarrollo Integrado (IDE).

Un entorno de desarrollo integrado o *Integrated Development Environment* (IDE), es un programa compuesto por un conjunto de herramientas para un programador. Puede dedicarse en exclusiva a un solo lenguaje de programación o utilizarse para varios. Los IDEs proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación tales como C++, Java, C#, Delphi, Visual Basic, entre otros. (22) Entre los IDEs que existen para el trabajo con el lenguaje de programación se encuentran Eclipse y NetBeans.

Eclipse.

Eclipse es un entorno de desarrollo integrado de código abierto. Permite que nuevos componentes puedan utilizar distintos tipos de contenido, para realizar determinadas tareas con contenidos existentes. La Plataforma Eclipse permite descubrir e invocar funcionalidad implementada en componentes llamados *plugins*. Soporta las herramientas proporcionadas por diferentes fabricantes de aplicaciones informáticas independientes. Contiene funciones que permiten manipular diferentes contenidos como son HTML, Java, C, JSP, EJB, XML, y GIF. Facilita una integración transparente entre todas las herramientas y tipos de contenidos sin tener en cuenta al proveedor. (23)

Netbeans.

El **IDE Netbeans** es una herramienta libre y gratuita sin restricciones de uso, de código abierto, pensada para escribir, compilar, depurar y ejecutar programas. Está escrito en Java, pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extender el IDE Netbeans. El código fuente está disponible para su reutilización de acuerdo con la Licencia de Desarrollo y Distribución Común v1.0 (CDDL por sus siglas en inglés) y la GPL (*General Public License* o Licencia Pública General) v2. Este IDE soporta el desarrollo de todos los tipos de aplicaciones Java (J2SE, web, EJB y aplicaciones móviles). NetBeans IDE proporciona a los programadores un paquete completo de aplicaciones que necesitan para crear aplicaciones de escritorio profesionales, multiplataforma, de empresa, web y móviles. Viene integrado con servidores de aplicaciones GlassFish v3, Apache Tomcat y maneja Bases de Datos MySQL, PostgreSQL y cualquiera que se conecte con JDBC como Oracle, SQL Server entre otros. (24) Permitiría desarrollar el sistema propuesto a partir de un conjunto de componentes llamados módulos, facilitando que pueda ser extendido por otros desarrolladores de software.

NetBeans IDE 7.2 ofrece un rendimiento significativamente mejorado y la experiencia de codificación, con las nuevas capacidades de análisis de código estático en el Editor de Java. Escanea de forma inteligente el proyecto para corregir cualquier tipo de fallo. Incluye características notables, como la integración con el Generador de Escena para los gráficos JavaFX, apoyo a marcos de PHP múltiples; mejora del rendimiento de sistemas de archivos remotos y muchas otras en Java EE, Maven, C / C + + y la plataforma NetBeans. (25)

Fundamentación del IDE seleccionado.

El entorno de desarrollo integrado seleccionado es NetBeans en su versión 7.2, por ser un producto libre y gratuito sin restricciones de uso y por todas las funcionalidades y la potencia que este posee a la hora de trabajar con el lenguaje de programación Groovy y el *framework* Grails. Además, la plataforma Netbeans ofrece a los desarrolladores numerosas ventajas en la creación de nuevas aplicaciones multiplataforma. Consta con gran éxito, una gran base de usuarios, una comunidad en constante crecimiento y el equipo de desarrollo se encuentra más familiarizado con este IDE.

1.8.5 Sistema gestor de bases de datos.

Los sistemas de gestión de bases de datos o SGBD son un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan. Su función general es manejar de manera clara, sencilla y ordenada un conjunto de datos que posteriormente se convertirán en información relevante para una organización.

PostgreSQL.

PostgreSQL es un gestor de bases de datos relacional, libre y gratuito. Está liberado bajo la licencia BSD (*Berkeley Software Distribution* o Distribución de Software Berkeley), lo que significa que se puede disponer de su código fuente, modificarlo a voluntad y redistribuirlo libremente.

Características generales. (26)

- **Objeto-Relacional:** PostgreSQL aproxima los datos a un modelo objeto-relacional, y es capaz de manejar complejas rutinas y reglas. Ejemplos de su avanzada funcionalidad son consultas SQL declarativas, control de concurrencia multi-versión, soporte multi-usuario, transacciones, optimización de consultas, herencia, y arreglos.
- **Altamente extensible:** Soporta operadores, funciones, métodos de acceso y brinda la posibilidad a los usuarios de crear sus propios tipos de datos.

- **Integridad referencial:** Soporta integridad referencial, la cual es utilizada para garantizar la validez de los datos de la base de datos.
- **API flexible:** La flexibilidad del API de PostgreSQL ha permitido proporcionar fácilmente el soporte al desarrollo para el Sistema de Gestión de Bases de Datos Objeto-Relacional (ORDBMS) PostgreSQL. Estas interfaces incluyen Object Pascal, Python, Perl, PHP, Java/JDBC, C/C++, y Pike.
- **Lenguajes procedurales:** PostgreSQL tiene soporte para lenguajes procedurales internos, incluyendo un lenguaje nativo denominado PL/pgSQL. Este lenguaje es comparable al lenguaje procedural de Oracle, PL/SQL.
- ***Multi-Version Concurrency Control* MVCC (Control de Concurrencia Multi-Versión):** Tecnología que PostgreSQL utiliza para evitar bloqueos innecesarios. MVCC posibilita que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos, ya que mantiene una ruta para todas las transacciones realizadas por los usuarios de la base de datos. Esta estrategia es superior al uso de bloqueos por tabla o por filas comunes en otras bases de datos, eliminando la necesidad del uso de bloqueos explícitos.
- ***Write Ahead Logging* WAL (Registro Anticipado de Escritura):** Esta característica incrementa la dependencia de la base de datos al registro de cambios antes de que estos sean escritos en la base de datos. Esto garantiza que en caso de que la base de datos tenga fallos, existirá un registro de las transacciones a partir del cual se puede restaurar la base de datos. Una vez que el sistema ha quedado restaurado, un usuario puede continuar trabajando desde el punto en que lo dejó cuando falló la base de datos.

PostgreSQL 9.0 incluye más de una docena de otros cambios importantes que mejorarán muchos aspectos de diseño y rendimiento de aplicaciones de base de datos, entre los que se cuentan: soporte Windows 64-bit, *triggers* condicionales y por columna, actualización in-situ desde las versiones 8.3 y 8.4, restricciones de unicidad postergables y paso de mensajes de alto rendimiento. (27)

MySQL.

MySQL es un gestor de base de datos sencillo de usar e increíblemente rápido. Es gratis para aplicaciones no comerciales.

Principales características de MySQL: (28)

- Aprovecha la potencia de sistemas multiprocesadores, gracias a su implementación multi-hilo.
- Soporta gran cantidad de tipos de datos para las columnas.
- Dispone de API's en gran cantidad de lenguajes (C, C++, Java, PHP).
- Gran portabilidad entre sistemas.
- Soporta hasta 32 índices por tabla.
- Gestión de usuarios y contraseñas, manteniendo un muy buen nivel de seguridad en los datos.

Fundamentación del sistema gestor de base de datos seleccionado.

Se seleccionó PostgreSQL en su versión 9.0 debido a que es un Sistema de Gestión de Bases de Datos Objeto Relacional, capaz de manejar complejas rutinas y reglas. Ejemplos de su avanzada funcionalidad son consultas SQL declarativas, control de concurrencia multi-versión, soporte multi-usuario, transacciones, optimización de consultas, herencia, y arreglos. Posee una gran escalabilidad, haciéndolo idóneo para ser usado en aplicaciones que realicen varias peticiones al día. Permite la gestión de usuarios, como también los permisos asignados a cada uno de ellos. Posee alta concurrencia permitiendo que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos.

1.8.6 Herramientas de modelado.

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.

UML (*Unified Modeling Language o Lenguaje de Modelado Unificado*): Se define como un lenguaje que permite especificar, visualizar y construir los artefactos de los sistemas de software. Es importante resaltar que UML es un "lenguaje" para especificar y no para describir métodos o procesos. UML está consolidado como el lenguaje estándar en el análisis y diseño de sistemas de cómputo. Permite modelar, construir y documentar los elementos que forman un sistema de software orientado a objetos. (29)

Herramienta CASE.

Las herramientas CASE (Computer Aided Software Engineering o Ingeniería de Software Asistida por Ordenador) representan una forma que permite modelar los procesos de negocios de las empresas y desarrollar los sistemas de información gerenciales.

Visual Paradigm es un potente conjunto de herramientas CASE que utiliza UML como lenguaje de modelado, soporta el ciclo de vida completo del desarrollo del software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Su diseño está enfocado al negocio que genera un software de mayor calidad, permite realizar ingeniería directa e inversa sobre el software, también dibujar todos los tipos de diagramas de clases, además de generar documentación. Proporciona abundantes tutoriales, proyectos y demostraciones interactivas de UML. Presenta licencia gratuita y comercial, además que es fácil de instalar, actualizar y compatible entre ediciones. (30) Su utilización permitirá realizar todos los diagramas que requiera la metodología seleccionada haciendo uso del lenguaje de modelado UML.

1.8.7 Servidor web.

Un servidor web es un programa que se ejecuta continuamente en un computador, diseñado para transferir hipertextos, páginas web o páginas HTML, manteniéndose a la espera de peticiones de ejecución realizadas por un cliente. El servidor web se encarga de contestar a estas peticiones de forma adecuada a través del protocolo HTTP, entregando como resultado una página web o información de todo tipo de acuerdo a los comandos solicitados. Apache es el servidor web hecho por excelencia, su robustez, seguridad y estabilidad hacen que cada vez millones de servidores usen este programa.

Apache.

El servidor web Apache fue desarrollado por el *Apache Server Project* (Proyecto Servidor Apache). Su función principal es analizar todos los archivos solicitados por un navegador y mostrar los resultados correctos de acuerdo con el código dentro de ese archivo. Es un servidor web muy potente y flexible, y puede realizar prácticamente cualquier tarea que el usuario requiera. Cuenta con una arquitectura modular que le permite personalizarse mejor para las necesidades de cada sitio web y facilita a los usuarios adicionar cómodamente funcionalidad a entornos específicos. Es multiplataforma, pues trabaja sobre todas las versiones recientes de UNIX y Linux, Windows y BeOS. Facilita la visualización de código HTML. Una de las grandes ventajas que posee es su capacidad de autenticación, lo que proporciona el control de acceso de los usuarios y estaciones de trabajo a varios sitios web. (31) Esta característica permitirá una regulación en la Internet en cuanto a los usuarios que están capacitados o no de conocer la información perteneciente a una institución.

Conclusiones del capítulo.

En el desarrollo de este capítulo se obtuvo un mejor dimensionamiento del problema a partir del análisis de los principales conceptos asociados a la solución. El estudio de varios sistemas existentes que realizan la verificación de personas a través de huellas dactilares, permitió obtener una mejor visión de la solución y aportó nuevas funcionalidades que podrían ser de interés. Para la implementación del software fueron seleccionadas un conjunto de herramientas y tecnologías mayormente basadas en licencias de software libre, encaminado a obtener un producto de alta independencia tecnológica y utilizable en diferentes plataformas.

Capítulo 2: Características del sistema.

Introducción.

En el presente capítulo se describen las principales características del sistema propuesto, donde se ejecutan las fases de Exploración y Planificación definidas en la metodología de desarrollo de software *Extreme Programming* (XP), así como los diferentes artefactos generados en cada una de ellas. Se presenta la propuesta de solución del sistema, se definen los requisitos funcionales y no funcionales y las historias de usuario expresando las necesidades de sistema y finalmente se realiza una estimación de tiempo y se define la arquitectura del sistema.

2.1 Propuesta de solución.

En la presente investigación se propone el desarrollo de un sistema de gestión de servicios de verificación que permita verificar la identidad de las personas en línea a través de huellas dactilares, a fin de obtener un mecanismo confiable que incremente la seguridad en las instituciones. El mismo permitirá agregar nuevos servicios de verificación (puede ser a través de rostro, iris, retina, firma u otras características biométricas) en función de las necesidades que puedan surgir a partir de una estructura basada en *plugins* y contará con una sencilla configuración que le permitirá adaptarse fácilmente a diversos entornos.

Entre las diferentes operaciones que el sistema permitirá realizar se encuentran las siguientes:

1. Verificación de identidad.
2. Búsqueda de personas y usuarios.
3. Gestión de usuarios, instituciones y licencias.
4. Autenticación de usuarios.
5. Registro y visualización de datos de interés de las personas (foto, identificación, nombres, apellidos, edad y sexo).
6. Visualización de los reportes.
7. Configuración de servicios.

El usuario podrá verificar la identidad de las personas dependiendo de la cantidad de verificaciones que le hayan sido asignadas en la licencia contratada por la institución, al agotarse la cantidad de verificaciones asignadas no podrá realizar más verificaciones y se le bloqueará el servicio. Para verificar la identidad de una persona, primeramente el usuario que realiza esta operación debe autenticarse en el sistema y se debe contar con una base de datos con la información biométrica de las personas que se deseen verificar. Para dar inicio al proceso, la persona muestra el documento de identificación que anteriormente fue

definido por la institución, esta información se toma como entrada en el sistema para buscar su huella dactilar en la base de datos según el número de identificación. Luego se procede a tomar la huella dactilar del mismo haciendo uso de un lector de huellas, la cual es comparada con la huella encontrada anteriormente. Si ambas coinciden es decir, la muestra previamente registrada y la actual, el individuo es verificado con éxito y se muestran sus datos personales, en otro caso el sistema muestra un aviso alertando que ocurrió una verificación no exitosa.

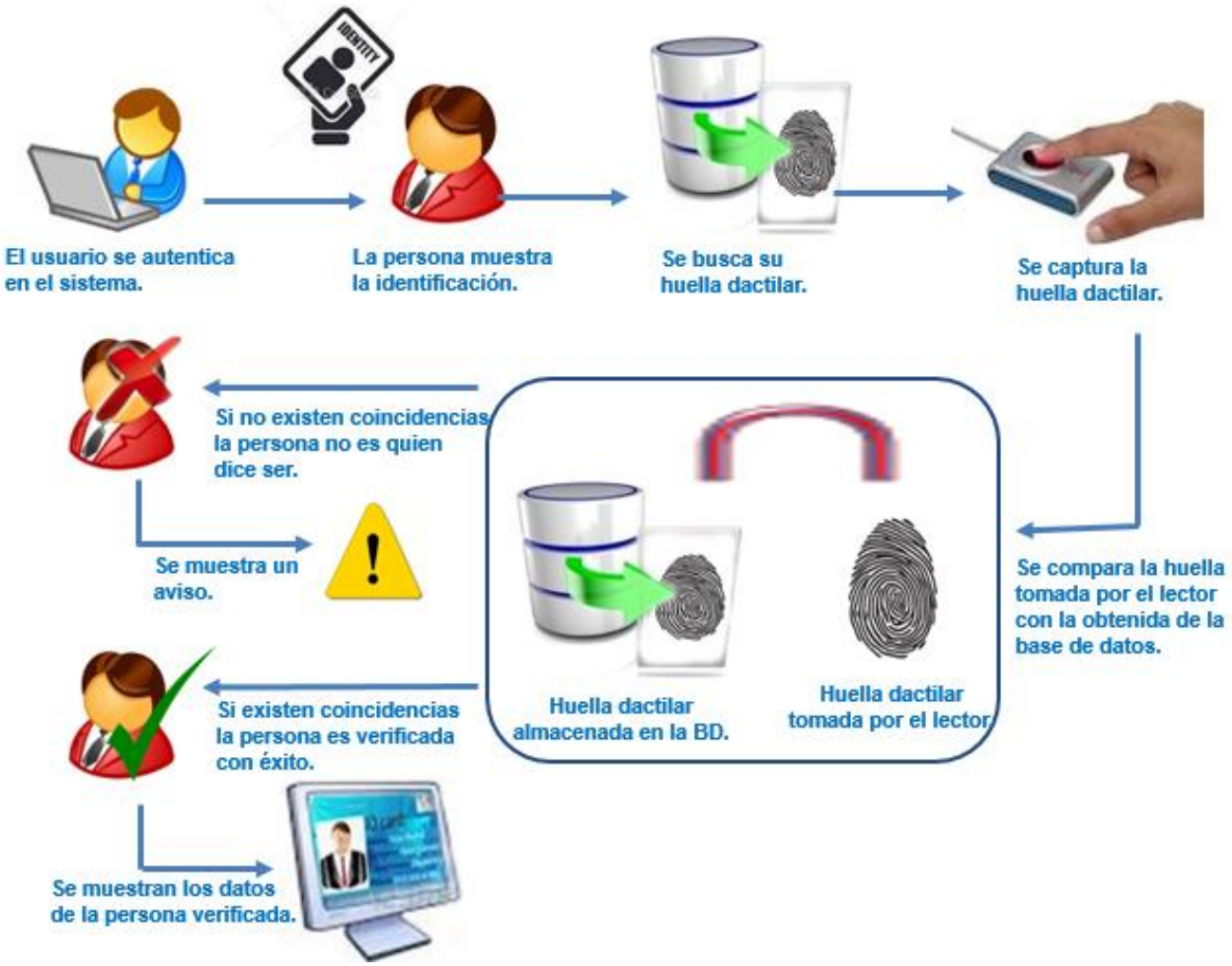

Figura 11. Flujo del proceso de verificación de personas a través de huellas dactilares en el sistema.

2.2 Modelo de dominio.

Teniendo en cuenta que el problema que pretende resolver el sistema no está determinado a partir de procesos bien definidos que permitan modelar el funcionamiento del mismo, se ha procedido a crear un modelo de dominio. Este artefacto se crea para lograr una mayor comprensión del dominio del problema, el mismo no es más que una representación visual de los conceptos u objetos del mundo real que son

significativos para el problema o el área que se analiza; representando las clases conceptuales, no los componentes de software. Puede verse como un modelo que comunica a los interesados, cuáles son los términos importantes y cómo se relacionan entre sí.

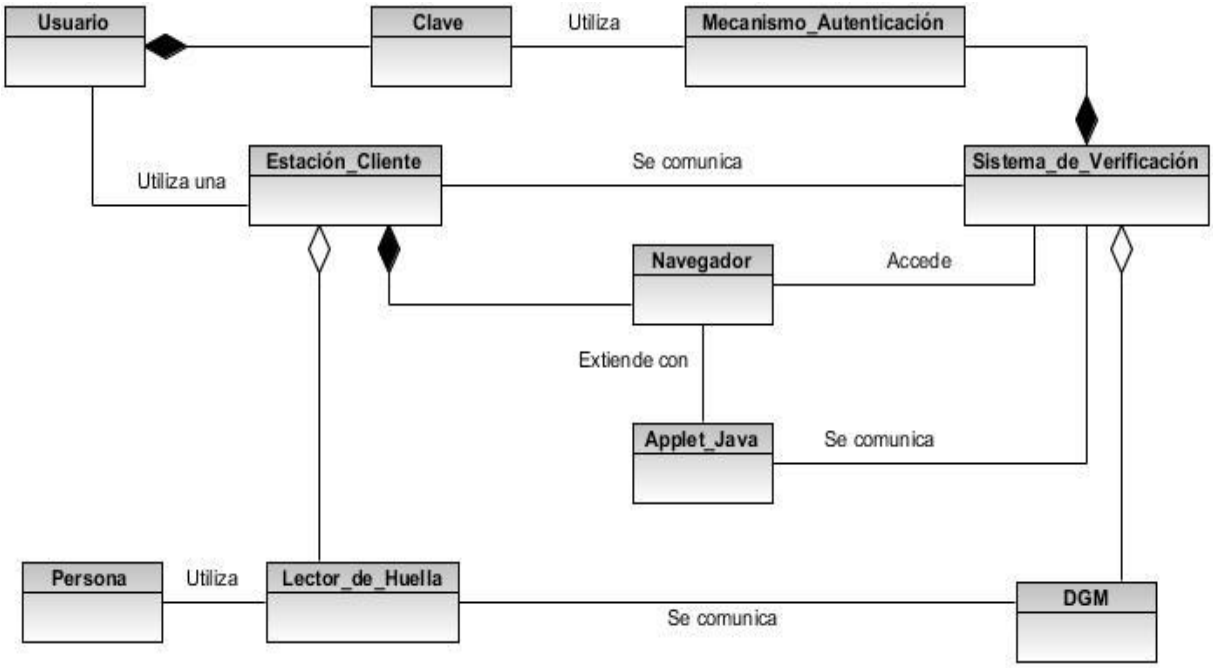

Figura 12. Modelo de dominio.

Glosario de términos del modelo de dominio.

Usuario: Persona que interactúa con el sistema.

Persona: Persona que va a ser verificada en el sistema.

Clave: Cadena de caracteres usada para identificarse y poder acceder al sistema.

Mecanismo_Autenticación: Mecanismo de autenticación que sirve al sistema para comprobar si la persona que intenta acceder al mismo está autorizada para ello.

Sistema_de_Verificación: Encargado de controlar y gestionar todas las acciones en el sistema.

Estación_Cliente: Computadora que utiliza el usuario.

Navegador: Permite visualizar la información que contiene la página web y a través de él se accederá al servicio de verificación, será la interfaz gráfica para la comunicación con el usuario.

Applet_Java: Hace referencias al *applets* de java ejecutado en el navegador, encargado de realizar la extracción y comparación de características de las huellas dactilares de la persona a verificar.

DGM: Controlador de dispositivos, encargado de efectuar la comunicación con el lector de huellas.

Lector_de_Huella: Dispositivo encargado de leer la huella dactilar de la persona a verificar.

2.3 Requisitos funcionales.

Módulo de administración.

RF 1: Buscar usuario.

RF 2: Adicionar usuario.

RF 3: Modificar usuario.

RF 4: Eliminar usuario.

RF 5: Autenticar usuario.

RF 6: Verificar permisos para el uso del sistema.

Módulo de configuración.

RF 7: Adicionar institución.

RF 8: Modificar institución.

RF 9: Eliminar institución.

RF 10: Adicionar licencia.

RF 11: Modificar licencia.

RF 12: Eliminar licencia.

RF 13: Configurar servicio.

Módulo de verificación.

RF 14: Realizar conexión con el controlador de dispositivos.

RF 15: Capturar la imagen de la huella dactilar.

RF 16: Verificar identidad del individuo por la huella dactilar.

RF 17: Verificar identidad del individuo por identificador.

RF 18: Mostrar datos del individuo.

Módulo de auditoría y control.

RF 19: Habilitar acceso a servicios.

RF 20: Deshabilitar acceso a servicios.

Módulo de reportes.

RF 21: Mostrar historial de verificaciones por usuario.

RF 22: Mostrar historial de verificaciones por persona.

RF 23: Reportar vencimiento de licencia.

RF 24: Reportar verificación no exitosa.

2.4 Requisitos no funcionales del sistema.

RNF 1: Usabilidad

El sistema debe ser manipulado por personas con conocimientos básicos en el manejo de la computadora y de un ambiente web.

RNF 2: Rendimiento

El sistema deberá ser capaz de atender solicitudes de verificación y enviar respuestas a diferentes clientes al mismo tiempo.

RNF 3: Confiabilidad y seguridad

- El sistema se mantendrá disponible 24 horas durante los 7 días de la semana.
- No se realizarán mantenimientos preventivos en horario laboral, deberán ejecutarse los fines de semana o días feriados.
- Se garantizará la consistencia de los datos, se realizarán comprobaciones y validaciones automáticas en todos los casos posibles.
- Mantener seguridad y control a nivel de usuarios, garantizando el acceso de cada uno de ellos solo a los niveles establecidos para el rol que ejecute. Las contraseñas de los usuarios solo podrán cambiarse con el administrador del sistema.

RNF 4: Software.

- Servidor.
  - Servidor de base de datos: PostgreSQL 9.0 o superior.
  - Servidor web: Apache 2.2 o superior.
  - Máquina virtual de java.
- Cliente.
  - Máquina virtual de java.
  - Navegador web.
  - DGM

RNF 5: Hardware.

- Servidor de base de datos (basado en los requerimientos mínimos de PostgreSQL).

- Periféricos: Mouse y Teclado.
- Procesador Intel Pentium 4 o superior.
- 256 MB de RAM o superior.
- 36 MB de espacio libre en HDD (30 MB para el código fuente, 5 MB de espacio para la instalación de los ejecutables y 1 MB para las bases de datos básicas).
- CPU 2.4 GHZ o superior.

- Servidor web (basado en los requerimientos mínimos de Apache).

  - Periféricos: Mouse y Teclado.
  - 2 GB de espacio libre en HDD o superior.
  - 1 GB Memoria RAM o superior.
  - CPU 1.8 GHZ o superior.

- Cliente.

  - Periféricos: Mouse y Teclado.
  - HDD 40 GB o superior.
  - 1 GB Memoria RAM o superior.
  - CPU 1.6 GHZ o superior.
  - Dispositivo de captura de huellas dactilares.

2.5 Fase de exploración.

La metodología de desarrollo XP comienza con la fase de Exploración, donde se plantean las historias de usuario (HU) a través de un proceso de identificación, que es muy importante para la primera entrega del producto. En esta fase se exploran las posibilidades de la arquitectura del sistema construyendo un prototipo. En esta metodología, una iteración de desarrollo es un período de tiempo en el que se realiza un conjunto de funcionalidades determinadas, que corresponden a un conjunto de historias de usuario.

2.5.1 Historias de usuario.

Dentro de los artefactos más importantes que genera la metodología XP se encuentran las Historias de Usuario (HU). Estas presentan el mismo propósito que los casos de uso, que son los encargados de representar los requisitos funcionales del sistema, es decir, son como un fragmento de funcionalidad del sistema que proporciona al usuario un valor añadido. Las mismas expresan su punto de vista en cuanto a las necesidades del sistema. Son descripciones cortas y escritas en el lenguaje del usuario sin terminología técnica. Otra de sus características es que solamente proporcionan los detalles sobre la

estimación del riesgo y cuánto tiempo conllevará su implementación. Su nivel de detalle debe ser el mínimo posible, de manera que permita hacerse una ligera idea de cuánto costará implementar el sistema. A continuación se describen algunas HU, el resto se pueden encontrar en el anexo 1.

Historia de usuario	
Número: HU_1	**Usuario:** Administrador.
Nombre de historia: Buscar usuario.	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Medio
Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable: Yaricel Guerra Velázquez.	
Descripción: El administrador del sistema tendrá la posibilidad de buscar usuarios en el sistema.	
Observaciones: No tiene observaciones.	

Tabla 1. HU_1 Buscar usuario.

Historia de usuario	
Número: HU_2	**Usuario:** Administrador.
Nombre de historia: Gestionar usuario.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable: Eliandis Matos Moreira.	
Descripción: El administrador del sistema en la estación cliente tendrá la posibilidad de adicionar, modificar y eliminar usuarios en el sistema.	
Observaciones: No tiene observaciones.	

Tabla 2. HU_2 Gestionar usuario.

Historia de usuario	
Número: HU_3	**Usuario:** Sistema.
Nombre de historia: Autenticar usuario.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable: Yaricel Guerra Velázquez.	
Descripción: Permite autenticar a los usuarios a través de nombre de usuario y contraseña para acceder al sistema.	
Observaciones: En caso de que el usuario no se encuentre registrado en la base de datos será notificado por un aviso del sistema.	

Tabla 3. HU_3 Autenticar usuario.

Historia de usuario	
Número: HU_4	**Usuario:** Sistema.
Nombre de historia: Verificar permisos.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable: Eliandis Matos Moreira.	
Descripción: Permite verificar los permisos del usuario según su rol para el uso del sistema.	
Observaciones: No tiene observaciones.	

Tabla 4. HU_4 Verificar permisos.

Historia de usuario

Número: HU_5	**Usuario:** Administrador.
Nombre de historia: Gestionar institución.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable: Eliandis Matos Moreira.	
Descripción: El administrador del sistema en el servidor tendrá la posibilidad de adicionar, modificar y desactivar las instituciones clientes.	
Observaciones: No tiene observaciones.	

Tabla 5. HU_5 Gestionar institución.

2.5.2 Metáfora.

Después de haberse definido las funcionalidades que el sistema debe cumplir y los requisitos no funcionales, se da lugar a la creación de la metáfora, esta es una breve descripción de cómo debe funcionar el sistema en su totalidad y es encargada de guiar todo el desarrollo como una gran historia de usuario, ayudando a entender los elementos básicos del sistema y sus relaciones.

El sistema para la verificación de personas en línea a través de huellas dactilares, permitirá brindar servicios de verificación de identidad en toda aquella institución donde se desee hacer uso del mismo y se cuente con la infraestructura adecuada. Para hacer uso de tales servicios se debe acceder a una dirección URL a través del navegador instalado en la estación cliente, donde debe estar conectado el lector de huellas. El sistema proporciona un componente (DGM) que deberá ser instalado en la estación cliente, el mismo será el encargado de controlar la conexión con el lector de huellas y de proveer la imagen de la huella dactilar capturada por este. El cliente o entidad debe solicitar una licencia con una cantidad limitada de verificaciones, cuando esta cantidad se agote o sea, ocurra el vencimiento de la licencia no puede realizar más verificaciones porque el servicio es bloqueado. En cada solicitud de verificación que se realice, el sistema siempre enviará una notificación que puede ser exitosa o no.

2.5.3 Arquitectura del sistema.

Para guiar el desarrollo del Sistema para la verificación de personas en línea a través de huellas dactilares, se definió la arquitectura cliente/servidor.

Teniendo en cuenta las características de la aplicación, se identifica una capa cliente que contiene un componente (DGM) que controla la conexión con el lector de huellas y provee la imagen de la huella dactilar capturada, los *applets* de java, encargados de realizar la extracción y comparación de características y un archivo JavaScript para la comunicación con los *applets*. Además se tiene la capa servidor, donde se encuentran el módulo de control de acceso, módulo de configuración y la base de datos entre otros componentes del sistema, los cuales se relacionan entre sí para resolver las solicitudes que vienen desde el cliente.

Figura 13. Arquitectura del sistema.

2.5.4 Patrón arquitectónico.

Se utilizó el patrón arquitectónico Modelo Vista Controlador (MVC). El estilo de llamada y retorno MVC, se ve frecuentemente en aplicaciones web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página. El modelo es la lógica de negocio y el controlador es el responsable de recibir los eventos de entrada desde la vista.

El flujo que sigue MVC aplicado al sistema es el siguiente:

1. El usuario interactúa con la interfaz de usuario solicitando alguna operación.
2. El controlador del cliente (componente que gestiona la interfaz) recibe la notificación de la operación solicitada por el usuario y lo envía al servidor.
3. El controlador del servidor (componente que gestiona las solicitudes del cliente) realiza la operación y se comunica con el controlador en el cliente, enviando el resultado de la misma.

4. La aplicación web muestra los resultados al usuario a través de la página web y espera nuevas interacciones del usuario, comenzando el ciclo nuevamente.

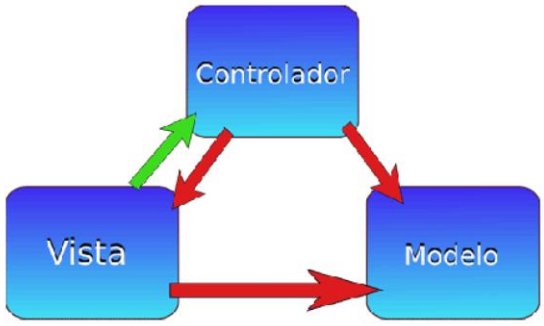

Figura 14 Estructura del patrón Modelo-Vista-Controlador (MVC).

2.5.5 Patrones de diseño.

Un patrón es una solución a un problema en un contexto, codifica conocimiento específico acumulado por la experiencia en un dominio. Los desarrolladores lo usan como una forma de reutilizar la experiencia, clasificando las soluciones con términos de común denominación y van formando un amplio repertorio de principios generales y de expresiones que los guían al crear un software. En el diseño de la aplicación se usarán los patrones GRASP (*General Responsibility Asignment Software Patterns*) que significa patrones generales de software para asignar responsabilidades y los GoF (*Gang Of Four*).

A continuación se describen los patrones GRASP que se usarán en el desarrollo del sistema. (32)

Experto: Permite dar solución al problema, el cuál sería el principio fundamental para asignar responsabilidades en el diseño orientado a objetos. La solución factible es asignar la responsabilidad a la clase contenedora de la información necesaria para cumplir la responsabilidad. El cumplimiento de una responsabilidad requiere a menudo información distribuida en varias clases de objetos. El uso de este patrón permitirá a los objetos valerse de su propia información para hacer lo que se les pide, favorece la existencia de mínimas relaciones entre las clases, lo que permite contar con un sistema sólido y fácil de mantener. Este patrón se pone de manifiesto en todas las clases.

Creador: Se considera para la asignación de responsabilidades a las clases relacionadas con la creación de objetos, de forma tal que una instancia de un objeto solo pueda ser creada por la clase que contiene la información necesaria para ello. El uso de este patrón admite crear las dependencias mínimas necesarias entre las clases, beneficiando el mantenimiento del sistema y brindando mejores oportunidades de

reutilización. Este patrón se pone de manifiesto en los controladores ya que son los encargados de crear cada una de las clases del dominio con los que se relacionan.

Controlador: Es un objeto de interfaz no destinada al usuario que se encarga de efectuar las asignaciones en cuanto al manejo de los eventos del sistema y definir sus operaciones. Crea una nueva instancia por la clase que tiene la información necesaria para realizar la creación del objeto, usa directamente las instancias creadas del objeto, almacena o maneja varias instancias de la clase y contiene o agrega la clase. El sistema cuenta con clases controladoras capaces de gestionar todo el flujo de datos de la aplicación, así como manejar varias instancias de objetos.

Alta cohesión: La cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme. Una baja cohesión hace muchas cosas no afines o realiza trabajo excesivo. En resumen, este patrón se observa cuando una clase tiene la responsabilidad de realizar una labor dentro del sistema, no desempeñada por el resto de los componentes del diseño. Este patrón se evidencia en conjunto con el patrón bajo acoplamiento, de forma tal que cada clase realiza sus acciones y se evita que otra clase realice acciones correspondientes a la clase con la que está relacionada.

Bajo acoplamiento: El acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases. Una clase con bajo (o débil) acoplamiento no depende de muchas otras. En el sistema se hace uso de este patrón cuando las clases entidades manejan todas las operaciones referentes a ellas por sí sola y las controladoras de cada entidad solo hacen uso de estas operaciones, para que no implique cambios en la controladora en caso de haber un cambio en el diseño de una entidad, y de implicar cambios estos sean mínimos.

También se utiliza para el diseño de la aplicación los patrones **GoF** (*Gang Of Four*). Este se clasifica en tres grandes categorías (33):

Creacionales: Los patrones creacionales tratan con las formas de crear instancias de objetos. Su objetivo es abstraer el proceso de instanciación y ocultar los detalles de cómo los objetos son creados o inicializados. (Fábrica abstracta, Constructor, Método de fabricación, Prototipo, Instancia única).

Estructurales: Los patrones estructurales describen como las clases y objetos pueden ser combinados para formar grandes estructuras y proporcionar nuevas funcionalidades. Estos objetos adicionados

pueden ser incluso objetos simples u objetos compuestos. (Adaptador, Puente, Compuesto, Decorador, Fachada, Peso ligero).

Comportamiento: Los patrones de comportamiento ayudan a definir la comunicación e iteración entre los objetos de un sistema. El propósito de este patrón es reducir el acoplamiento entre los objetos. (Cadena de responsabilidad, Orden, Intérprete, Iterador, Mediador, Observador, Estado, Estrategia, Método plantilla, Visitante).

Para el diseño del sistema de verificación de personas en línea a través de huellas dactilares se utilizaron los siguientes patrones GoF:

Decorador: Añade responsabilidades adicionales a un objeto dinámicamente, proporcionando una alternativa flexible a la especialización mediante herencia, cuando se trata de añadir funcionalidades. El uso de este patrón se centra en la utilización de *SiteMesh* integrado en la arquitectura del *framework* Grails. El mismo permite a partir de una plantilla inicial de la página incorporar modificaciones a cada vista en particular.

Instancia única: Garantiza que una clase sólo tenga una instancia, y proporciona un punto de acceso global a ella. Se utiliza en la instanciación de los servicios en los controladores. Por defecto en Grails todos los servicios son de instancia única.

2.6 Fase de planificación.

En la metodología XP, la planificación se plantea como un diálogo continuo entre las partes involucradas en el proyecto. Es una fase corta, donde se acuerda el orden en que deberán implementarse las historias de usuario, así como su entrega. Esta planificación debe de ser flexible para que pueda adaptarse a los posibles cambios que puedan surgir.

2.6.1 Estimación de tiempo.

Se realiza una estimación del tiempo que se necesita para desarrollar cada historia de usuario, este valor se expresa en semanas. Con el transcurso de las iteraciones, se irá acercando a la realidad. La tabla se muestra en el anexo 2.

2.6.2 Plan de iteraciones.

Como parte del ciclo de vida de un proyecto usando la metodología XP, se crea el plan de duración de cada una de las iteraciones que se han definido, que tiene como objetivo mostrar la duración y el orden en

que serán implementadas las historias de usuario dentro de cada iteración. La tabla se muestra a continuación.

Iteración	No. HU	Historia de Usuario	Duración estimada
Iteración 1	HU_1	Buscar usuario.	7 semanas
	HU_2	Gestionar usuario.	
	HU_3	Autenticar usuario.	
	HU_4	Verificar permisos.	
	HU_5	Gestionar institución.	
	HU_6	Gestionar licencia.	
	HU_7	Configurar servicio.	
Iteración 2	HU_8	Realizar conexión con el DGM.	5 semanas
	HU_9	Capturar imagen de huella dactilar.	
	HU_10	Verificar identidad del individuo.	
	HU_11	Mostrar datos del individuo.	
Iteración 3	HU_12	Cambiar estado de los servicios.	3 semanas
	HU_13	Mostrar historial de verificaciones.	
	HU_14	Mostrar reportes.	

Tabla 6. Plan de iteraciones.

2.6.3 Plan de entrega.

El plan de entrega se elabora con la intención de fijar qué período de tiempo puede tardar la implementación de cada una de las historias de usuario, definiéndose las fechas en que serán liberadas las versiones funcionales del producto.

Entregable	Final Iteración 1	Final Iteración 2	Final Iteración 3
Verifier	Marzo 2013	Abril 2013	Mayo 2013

Tabla 7. Plan de entregas.

Conclusiones del capítulo.

En este capítulo se logró una visión clara y objetiva de los requerimientos del cliente, generando los artefactos que propone la metodología de desarrollo XP como las historias de usuarios, el plan de entrega y el plan de iteraciones. El análisis de los principales conceptos y sus relaciones reflejado en el modelo de dominio, se obtuvo una visión más clara de la solución, llegándose a la conclusión de que la misma abarca el desarrollo de dos componentes fundamentales: uno para el cliente y otro para el servidor. Se definieron los requisitos funcionales y los no funcionales. La propuesta de solución que se generó cumple con las funcionalidades acorde a las necesidades del cliente.

Capítulo 3: Implementación y prueba.

Introducción

En este capítulo se abordan los aspectos fundamentales acerca del desarrollo del sistema y la aprobación del mismo. Se diseña el diagrama de componentes, donde se representan los principales componentes del sistema y las relaciones entre ellos, además, se representa su distribución física a través del diagrama de despliegue. Se visualizan las interfaces de usuario y se proporciona una explicación de cada una de ellas para un mayor entendimiento del funcionamiento del sistema. Se realizan las pruebas necesarias al software para que tenga la calidad requerida.

3.1 Diagrama de despliegue.

El diagrama de despliegue se utiliza para modelar el hardware utilizado en las implementaciones de sistemas y las relaciones entre sus componentes. Los diagramas de despliegue muestran las relaciones físicas de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos. La vista de despliegue representa la disposición de las instancias de componentes de ejecución en instancias de nodos conectados por enlaces de comunicación. La siguiente figura muestra cómo será desplegado el software.

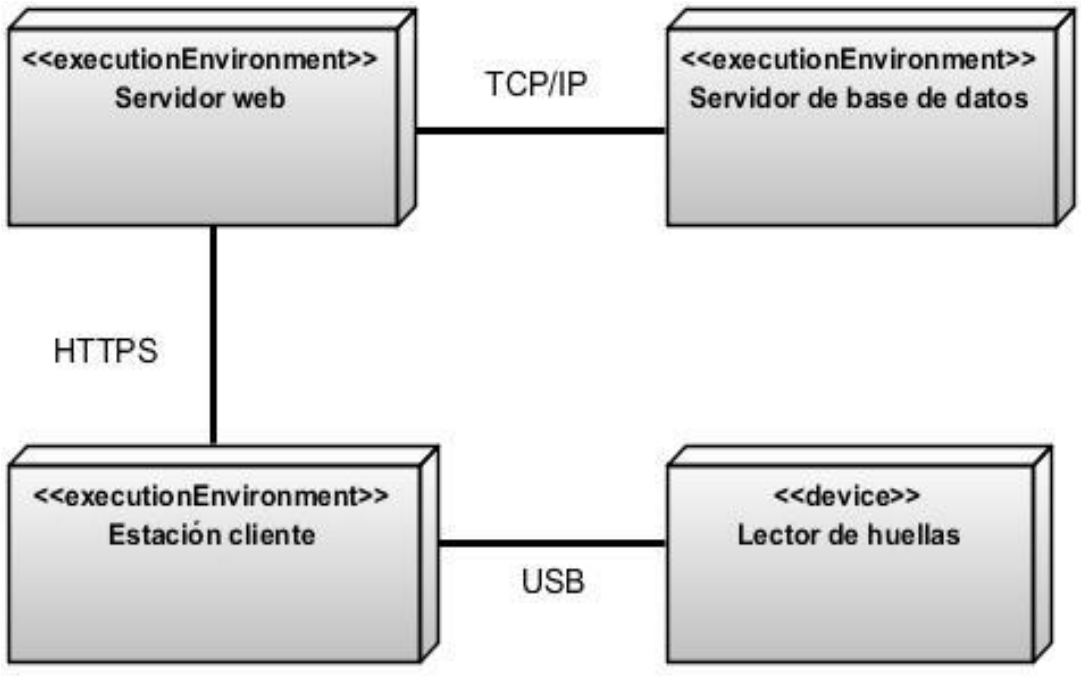

Figura 15. Diagrama de despliegue.

Descripción de los nodos y protocolos de comunicación.

Estación cliente: Representa la computadora desde la cual el usuario podrá acceder a la aplicación.
Servidor de aplicaciones: Representa una estación donde estará montado el servidor Apache sobre el cual se estará ejecutando la aplicación.
Servidor de base de datos: Representa el servidor donde estará el SGBD PostgreSQL que dará respuesta a las peticiones realizadas por el sistema.
TCP/IP: Protocolo que se utiliza en la comunicación entre el servidor web y la base de datos para realizar operaciones sobre la información de las tablas.
HTTPS: (*Hypertext Transfer Protocol Secure* o Protocolo Seguro de Transferencia de Hipertexto) es un protocolo que establece un esquema seguro de comunicación cliente/servidor.
USB: Protocolo para la conexión entre la estación cliente y el lector de huellas. El USB *(Universal Serial Bus* o Bus Universal en Serie) es un puerto que se utiliza para conectar periféricos a un ordenador.

3.2 Diagrama de componentes.

El diagrama de componentes modela la vista estática de un sistema y describe los elementos físicos, los cuales pueden ser archivos, librerías, módulos, ejecutables, o paquetes y pueden ser usados para modelar y documentar cualquier arquitectura de sistema. Además, en los diagramas de componentes se muestran los elementos de diseño de un sistema y por otra parte permiten visualizar con más facilidad la estructura general del sistema y el comportamiento del servicio que estos componentes proporcionan y utilizan a través de las interfaces.

Descripción de los principales componentes:

AppletExtracciónMinucias: Componente que realiza la extracción de características de las huellas dactilares.
AppletComparaciónMinucias: Componente que realiza la comparación de características de las huellas dactilares.
DeviceGridManager: Componente que controla la conexión con el lector de huellas y provee la imagen de la huella dactilar capturada por el lector.
Captura.gps: Interfaz que interactúa con el componente de extracción y comparación de características y con el *Device Grid Manager*.
Views: Paquete que contiene las vistas o interfaces de la aplicación.
Controllers: Paquete que contiene las clases controladoras del sistema.
Domain classes: Paquete que contiene las clases modelo o sea la lógica del negocio del sistema.

Base de datos: Componente que representa la base de datos del sistema.

Verifier: Paquete que representa al sistema.

Verifier_API.zip: Componente que representa el API del sistema.

Huella_dactilar.zip: Componente que representa el *plugin* de verificación a través de huellas dactilares.

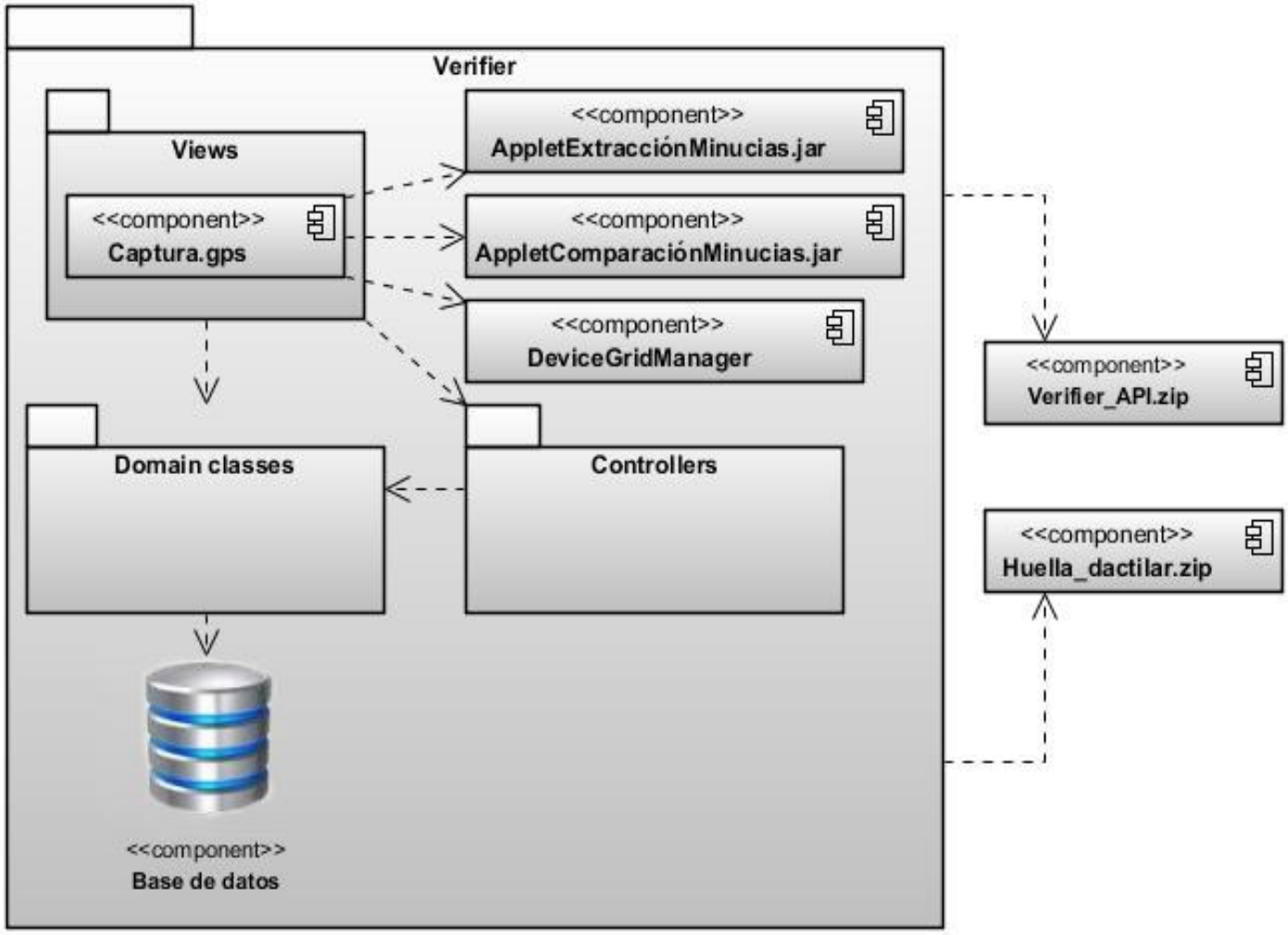

Figura 16. Diagrama de componentes.

3.3 Diseño de la base de datos.

La base de datos desempeña una función importante en el desarrollo de las aplicaciones web, ya que permite que los datos se almacenen de forma coherente y organizada, y no exista pérdida de los mismos. La base de datos que se muestra a continuación, satisface las necesidades de persistencia de los datos que el sistema requiere, en cumplimiento de sus requisitos funcionales.

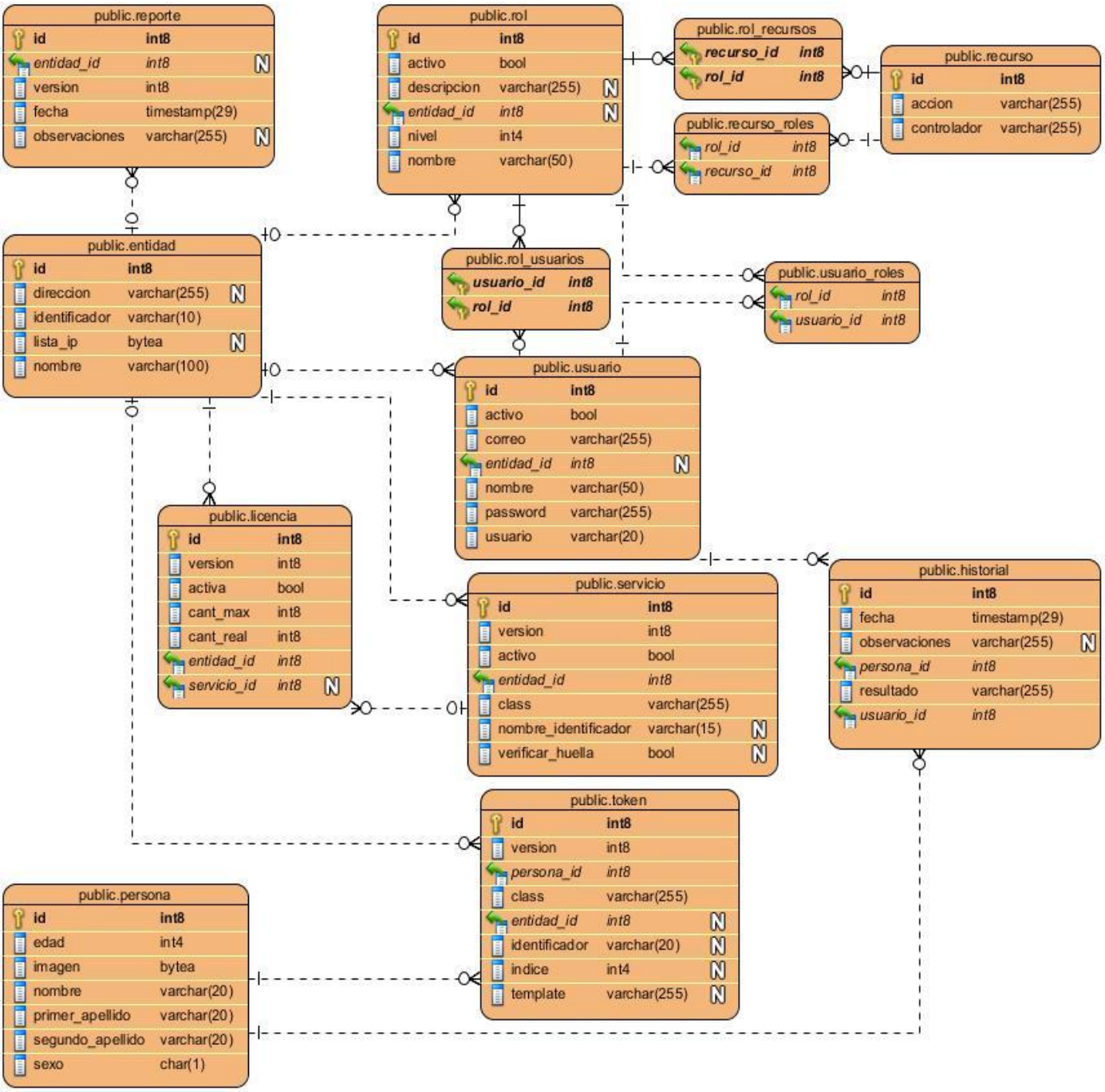

Figura 17. Diagrama Entidad-Relación.

3.4 Fase de implementación.

Según la metodología XP la implementación o desarrollo del sistema es la parte más importante, porque permite obtener un resultado al culminar cada iteración y de esta manera una versión del producto funcional, el cual debe ser mostrado y probado al cliente, sirviendo de retroalimentación para el equipo de

trabajo. A continuación se exponen detalladamente las tres iteraciones generadas por la planificación descrita anteriormente así como las tareas que se plantearon para la realización de cada una de las historias de usuario.

3.4.1 Tareas de ingeniería.

Las tareas de ingeniería serán de uso únicamente para el programador, ya que estas son creadas para ayudar a organizar la implementación exitosa de las historias de usuario. Las historias de usuario no ofrecen el nivel de detalle requerido para llevar a cabo esta acometida, es por eso que son divididas generalmente en más de una tarea de ingeniería, las cuales pueden ser escritas en lenguaje técnico y no necesariamente entendible por el cliente. Según el plan de iteraciones trazado, las historias de usuario se agruparon en tres ciclos de iteraciones. A continuación se exponen las tareas de ingeniería y sus descripciones se pueden encontrar en el anexo 3.

Iteración 1

En esta iteración se desarrollan las historias de usuario de mayor prioridad para la seguridad en el sistema, con el objetivo de obtener una primera versión del producto para ser mostrada al cliente.

Historias de usuario	Tareas
Buscar usuario.	- Crear la tabla usuario en la BD. - Implementar el buscar usuario.
Gestionar usuario.	- Implementar el adicionar usuario. - Implementar el eliminar usuario. - Implementar el modificar usuario. - Implementar el asignar rol al usuario.
Autenticar usuario.	- Implementar la interfaz de autenticación.
Verificar permisos.	- Implementar el verificar permisos.
Gestionar institución.	- Implementar el adicionar institución. - Implementar el eliminar institución.

	- Implementar el modificar institución.
Gestionar licencia.	- Implementar el adicionar licencia. - Implementar el eliminar licencia. - Implementar el modificar licencia.
Configurar servicios.	-Implementar la configuración de los servicios.

Tabla 8. Tareas de ingeniería de la iteración 1.

Iteración 2

En el transcurso de esta iteración se implementan las historias de usuario que tienen un nivel de prioridad alto para la aplicación. Esto permite agregar nuevas funcionalidades al sistema y obtener una nueva versión, de esta manera se tiene una visión más completa de cómo va a quedar el producto final.

Historias de usuario	Tareas
Realizar conexión con el DGM.	- Integrar el componente DGM. - Realizar conexión con el DGM.
Capturar imagen de huella dactilar.	- Capturar imagen de huella dactilar.
Verificar identidad del individuo.	- Integrar el componente de extracción de características. - Integrar el componente de comparación de características.
Mostrar datos del individuo.	- Mostrar datos del individuo.

Tabla 9. Tareas de ingeniería de la iteración 2.

Iteración 3

En esta iteración se desarrollan las historias de usuario restantes, con el objetivo de obtener la primera versión final del producto, con todas sus funcionalidades para ser mostradas al cliente.

Historias de usuario	Tareas

Cambiar estado de los servicios.	- Habilitar servicios. - Deshabilitar servicios.
Mostrar historial de verificaciones.	- Mostrar historial de verificaciones según usuario. - Mostrar historial de verificaciones según persona.
Mostrar reportes.	- Mostrar reportes de vencimiento de licencia. - Mostrar reportes de verificación no exitosa.

Tabla 10. Tareas de ingeniería de la iteración 3.

3.4.2 Implementación del sistema.

El sistema de verificación de identidad a través de huellas dactilares está constituido fundamentalmente por tres componentes: “Sistema de gestión”, “API del sistema” y el *plugin* “Verificación de identidad a través de huellas dactilares”. El mismo se encuentra estructurado a partir de *plugins*, facilitando la integración de nuevos servicios de verificación de identidad.

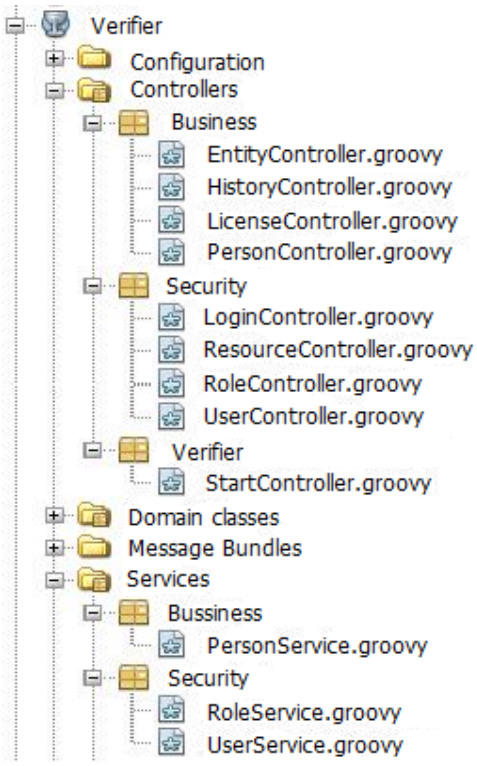

Figura 18. Estructura del sistema.

El sistema de gestión está estructurado como se muestra en la figura 18. Dentro de los controladores se encuentran los paquetes que a continuación se describen.

Business: Contiene las clases *EntityController, HistoryController, LicenseController, PersonController,* encargadas del proceso de gestión de entidades, historiales, licencias y personas respectivamente.

Security: Contiene las clases *LoginController, ResourceController, RoleController, UserController,* encargadas del proceso de gestión de autenticación, recursos del sistema, roles y usuarios respectivamente.

Verifier: Contiene la clase *StartController,* encargada de mostrar la página principal del sistema.

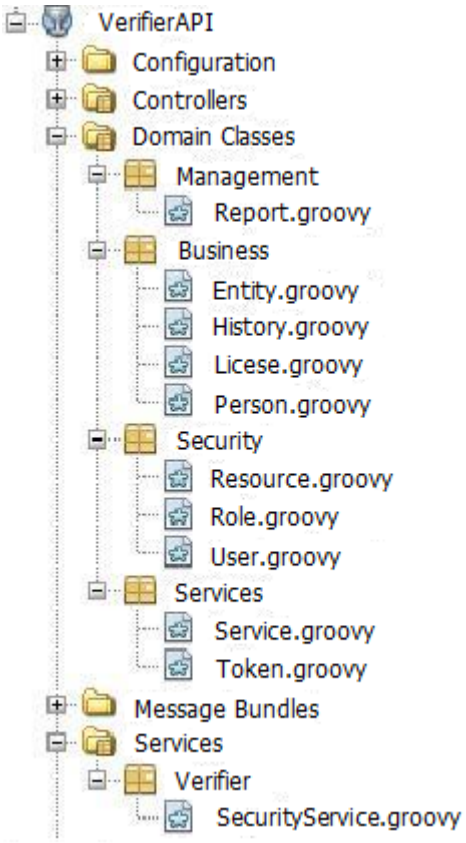

Figura 19. Estructura del API del sistema

El API del sistema es un *plugin* que debe ser instalado en el sistema de gestión y en el *plugin* de verificación a través de huellas dactilares. El mismo se encuentra estructurado como se muestra en la figura 29. Dentro de las clases del dominio se encuentran los siguientes paquetes: Management, Business, Security y Services; en el contenido de estos paquetes se encuentran las clases encargadas de acceso a los datos. La capa *Services* tiene como fin ayudar en la lógica de la aplicación, evitando colocar

demasiado código en los controladores, la misma contiene la clase *SecurityService,* encargada de manejar la seguridad en el sistema.

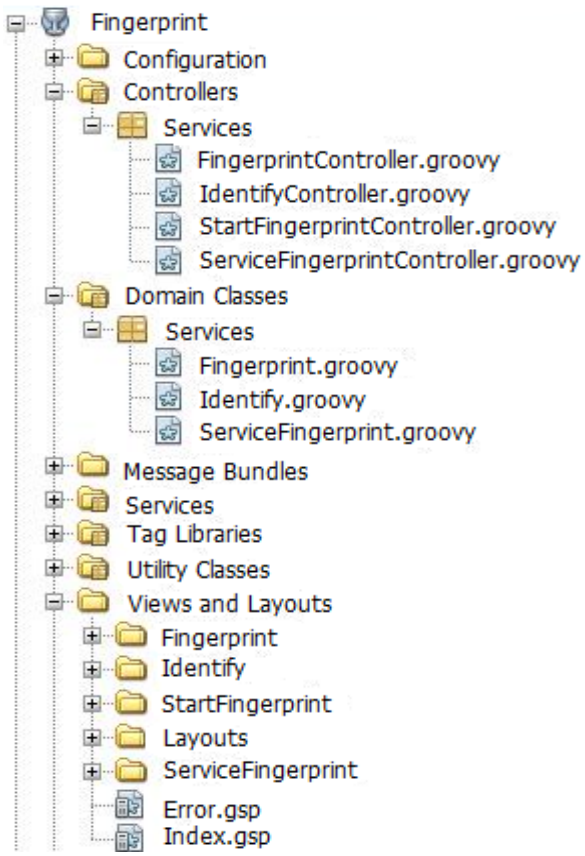

Figura 20. Estructura del plugin de verificación de identidad a través de huellas dactilares.

El *plugin* de verificación de identidad a través de huellas dactilares es un componente que debe tener instalado el API del sistema. El mismo se encuentra estructurado como se muestra en la figura 20. Dentro de los controladores encargados de llevar la lógica del negocio en este módulo se encuentran los siguientes: *FingerprintController, IdentifyController, StartFingerprintController y ServiceFingerprintController*. Dentro del paquete de servicios de las clases del dominio se encuentran las siguientes clases: *ServiceFingerprint*, *Fingerprint e Identify,* donde la primera hereda de la clase *Servicio* y las dos últimas de la clase *Token* que se encuentra en el API del sistema como muestra la figura 21.

```
class HuellaDactilar extends Token{
    int indice
    String template
    static constraints = {
        indice(nullable:true)
        version(nullable:true)
    }
    static mapping = {
        version false
    }
}
```

Figura 21. Herencia de la clase Token.

Agregar un nuevo servicio de verificación de identidad.

Para agregar un nuevo servicio de verificación de identidad (puede ser a través de rostro, iris, retina, firma u otras características biométricas) es necesario crear un *plugin* haciendo uso del *framework* Grails en su versión 2.0.1. Luego se debe instalar el API del sistema. Todas las clases que se necesiten desarrollar deben pertenecer al paquete de servicios. Posteriormente se crea la clase del dominio relacionada con el servicio que se desea agregar, dicha clase debe heredar de la clase *Servicio* que se encuentra en el API del sistema. Si para el desarrollo de este nuevo servicio se necesita un *token* adicional, este debe heredar de la clase *Token* que pertenece al API como se muestra en la figura 21.

3.4.3 Interfaces del sistema.

La interfaz gráfica del usuario es el medio por el cual este interactúa con el sistema, por lo que su diseño debe ser amigable, consistente y que no requiera usuarios con un alto nivel informático. Una aplicación con una interfaz bien diseñada, además de un buen diseño gráfico, debe tener una buena navegabilidad, usabilidad y distribución de los contenidos.

A continuación se muestran dos figuras, una representa la página principal de un usuario administrador, el cual tendrá la posibilidad de gestionar los distintos usuarios que tienen acceso al sistema, así como gestionar las licencias e instituciones que hacen uso del mismo y cambiar el estado de los servicios, la otra figura representa una de las funcionalidades que gestiona este usuario.

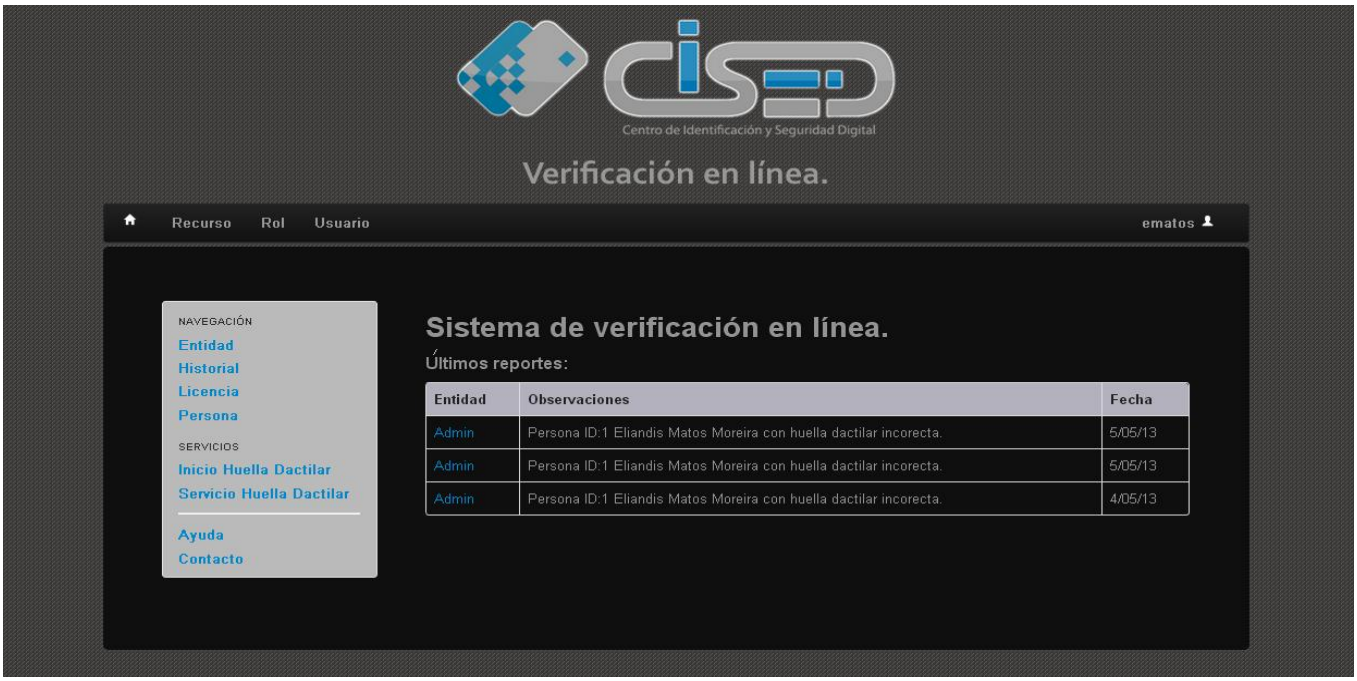

Figura 22. Página principal del administrador.

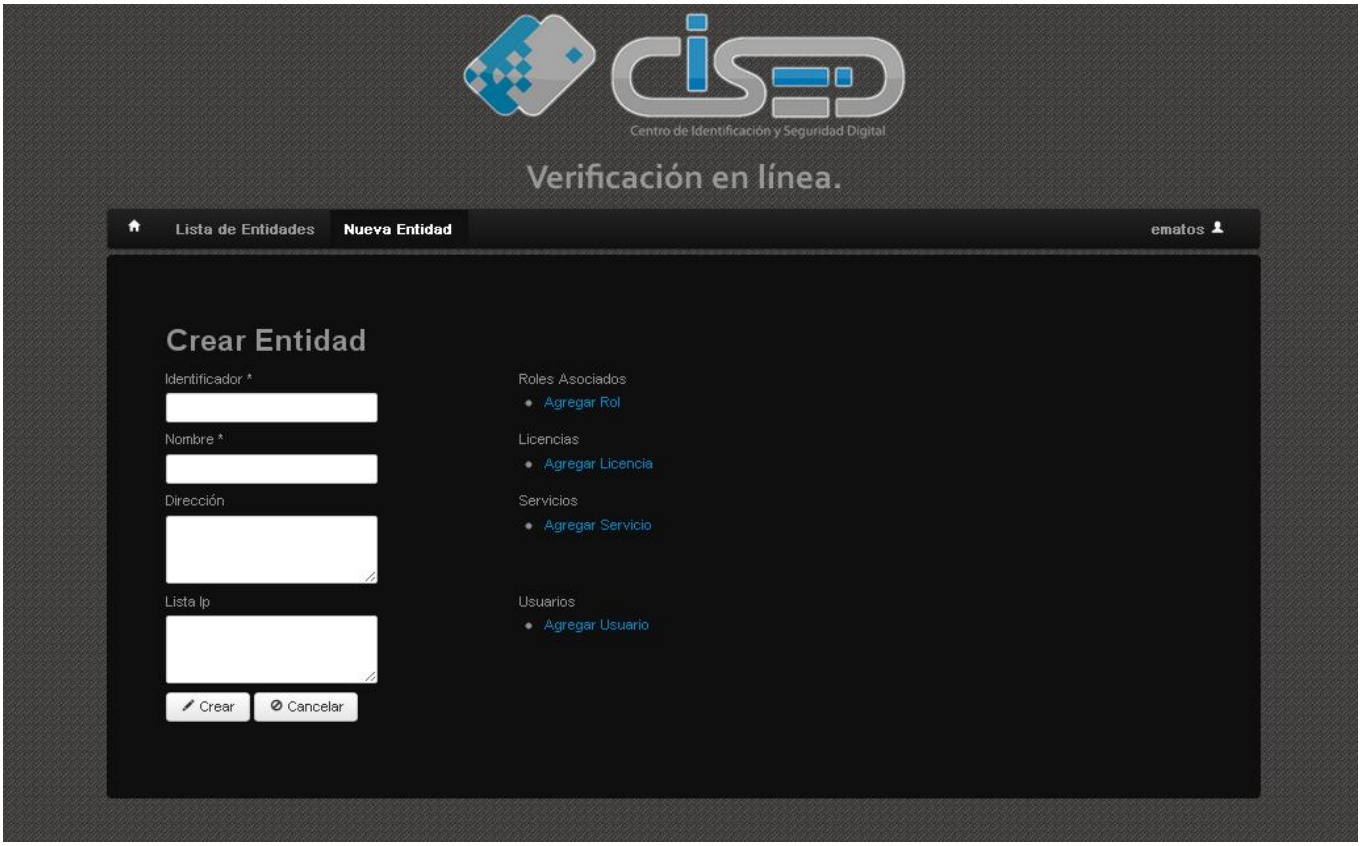

Figura 23. Adicionar institución.

A continuación se muestran 2 figuras, una de la página de verificación de identidad por huellas dactilares y la otra de la página donde se muestra el resultado de una verificación exitosa.

Figura 24. Verificar identidad por huella dactilar.

Figura 25.Resultados de una verificación exitosa.

3.5 Fase de prueba.

La fase de pruebas permite a los desarrolladores minimizar la cantidad de errores no detectados en el sistema durante la fase de implementación, permitiendo verificar y elevar la calidad del producto, dándole cierta seguridad a los desarrolladores a la hora de hacer cambios o modificaciones. Esta estrategia incluye la definición del tipo de prueba a realizar para cada iteración y sus objetivos, el nivel de cobertura y el porcentaje con el que deberían ejecutarse con un resultado específico. También permite identificar historias adicionales que no fueran obvias para el cliente o en las que este no hubiese pensado de no enfrentarse a dicha situación.

3.5.1 Tipos de pruebas

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.

Se decidió utilizar las pruebas de aceptación debido a su importancia, ya que significan la satisfacción del cliente con el producto desarrollado al final de cada iteración. Este período conocido también como período de caja negra, es donde se definen las entradas al sistema y los resultados esperados de estas entradas. El objetivo final es garantizar que los requerimientos hayan sido cumplidos y que el sistema es aceptado. Estas pruebas se desarrollaron en paralelo con el sistema, según se iban implementando las historias de usuario al final de cada iteración.

Se le realizaron casos de prueba de aceptación a la aplicación a partir de las historias de usuario. En estos se especifican los escenarios para probar que una HU ha sido implementada correctamente.

A continuación se muestran algunos casos de prueba realizados a la aplicación. Para consultar los restantes casos de prueba ver el Anexo 4.

Caso de prueba de aceptación	
Código: HU1_CP1	**HU:** 1
Nombre: Buscar usuario.	
Descripción: Se prueba la funcionalidad de buscar usuarios en el sistema. En caso de que el usuario buscado no se encuentre registrado en la base de datos, el sistema muestra un aviso informando que no se encontraron	

resultados.
Condiciones de ejecución: El usuario que realiza la acción debe estar previamente autenticado en el sistema.
Entrada/Pasos de ejecución y salidas: **Entrada:** Se muestra una interfaz que contiene un formulario con un campo para insertar alguno de los siguientes datos: usuario o nombre (también permite realizar la búsqueda con parte de estos datos), luego se oprime el botón *Buscar*. **Salida:** Se muestran los usuarios que coinciden con el criterio de búsqueda.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 11. Caso de prueba de aceptación HU1_CP1 "Buscar usuario".

Caso de prueba de aceptación	
Código: HU2_CP2	**HU:** 2
Nombre: Gestionar usuario.	
Descripción: Se debe comprobar la inserción, modificación y eliminación del usuario. Si los datos introducidos tienen errores o se dejan campos en blanco, el sistema muestra un mensaje de error. En el caso de estar correctos estos datos, se guardan en la base de datos satisfactoriamente.	
Condiciones de ejecución: El usuario que realiza la acción debe estar previamente autenticado y tener permisos administrativos.	
Entrada/Pasos de ejecución y salidas: **Entrada: "Escenario Insertar Usuario"** Se selecciona la opción *Crear usuario*, mostrándose un formulario con los campos correspondientes a los siguientes datos: nombre, usuario, contraseña, confirmación de contraseña, activo y correo, posteriormente se oprime el botón *Crear*, de esta forma queda realizada la inserción.	

Salida: "Escenario insertar usuario" Se adiciona el usuario a la base de datos. **Entrada: "Escenario Modificar datos de usuario"** Se muestra una interfaz con un listado de los usuarios, se selecciona el usuario que se desea modificar y se oprime el botón *Editar* que aparece en la nueva interfaz, mostrándose un formulario con los campos actuales del usuario seleccionado, debe permitirse modificar cualquiera de los datos pertenecientes a este, posteriormente se oprime el botón *Actualizar*. **Salida: "Escenario Modificar datos usuario"** Se actualiza el usuario en la base de datos. **Entrada: "Escenario Eliminar usuario"** Se muestra una interfaz con un listado de todos los usuarios, se selecciona el usuario que se desea eliminar y se oprime el botón *Eliminar* que aparece en la nueva interfaz. **Salida: "Escenario Eliminar usuario"** Se elimina el usuario seleccionado.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 12. Caso de prueba de aceptación HU2_CP2 "Gestionar usuario".

Como resultado de las diferentes pruebas de aceptación realizadas, fueron detectadas un conjunto de no conformidades que se han ido resolviendo gradualmente. A continuación se muestran la cantidad de no conformidades que fueron detectadas en cada iteración realizada al sistema y en el anexo 5 se describen brevemente cada una de ellas.

Caso de prueba de aceptación.	Iteración 1	Iteración 2	Iteración 3
Buscar usuario.	1	0	-
Gestionar usuario.	1	0	-
Autenticar usuario.	1	0	-
Verificar permisos.	1	0	-

Gestionar institución.	1	0	-
Gestionar licencia.	-	-	-
Configurar servicio.	-	-	-
Realizar conexión con el DGM.	-	1	0
Capturar imagen de huella dactilar.	-	0	-
Verificar identidad del individuo.	-	2	0
Mostrar datos del individuo.	-	1	0
Cambiar estado de los servicios.	-	-	-
Mostrar historial de verificaciones.	-	-	-
Mostrar reportes.	-	-	1

Tabla 13. Cantidad de no conformidades por casos de prueba de aceptación.

En la siguiente gráfica se encuentran representados los resultados de las pruebas de aceptación que se han realizado en cada una de las iteraciones hasta el momento, como se muestra, las no conformidades que fueron encontradas en cada una de las iteraciones fueron corregidas en la siguiente y en el caso de la iteración 3, donde se encontró una no conformidad, fue corregida posteriormente, lográndose de esta manera implementar todas las HU favorablemente, obteniéndose un producto aceptable.

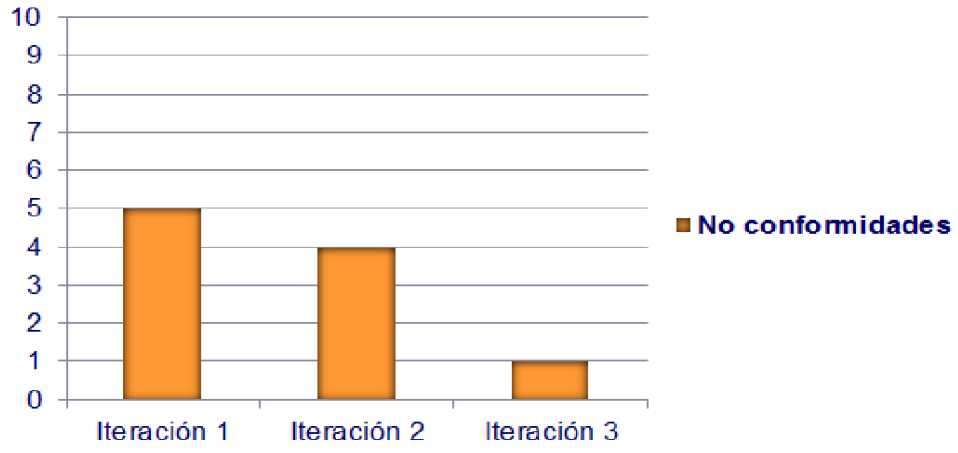

Figura 26. Resultados de las pruebas de aceptación por iteraciones.

3.6 Beneficios del sistema.

La implementación del “Sistema para la verificación de personas en línea a través de huellas dactilares” logró demostrar un gran número de beneficios los cuales se evidencian a continuación.

La **Confiabilidad** es una de las principales características de las que carecen los mecanismos de verificación de identidad comúnmente usados en las instituciones como es el caso de los documentos tradicionales. El sistema propuesto proporciona un mecanismo de verificación de identidad basado en la huella dactilar, la cual es una de las características biométricas más confiables debido a que cumple con los parámetros de universalidad (todas las personas deben poseerlo), unicidad (ningún otro individuo presenta las mismas características) y permanencia (esta característica es invariable en el tiempo).

Con respecto a la **Seguridad**, la mayoría de los sistemas web actuales solo definen usuarios y claves para acceder a ellos, o sea, todo el que conozca la forma de acceso lo podrá hacer desde cualquier estación de trabajo sin importar el cargo o rol que desempeñe. Sin embargo, el sistema propuesto emplea un mecanismo de autenticación de usuarios, que permite que solo accedan aquellos usuarios que estén registrados en la base de datos. A cada uno de los usuarios registrados se le asigna un rol de acuerdo a los recursos que tenga permiso, permitiendo que solo gestionen la información que les corresponde.

Otro de los aspectos fundamentales es la **Flexibilidad**, generalmente las aplicaciones informáticas son desarrolladas para una finalidad en específico, careciendo de esta característica, lo que impide que otros desarrolladores puedan realizar su aporte con el objetivo de obtener un producto más completo. El sistema propuesto proporciona una estructura que permite agregar nuevos servicios de verificación de identidad basados en otras características biométricas como el rostro, iris, retina, entre otras, permitiendo que pueda ser extendido fácilmente.

Conclusiones del capítulo.

En este capítulo se desarrollaron las fases de implementación y prueba definidas en la metodología XP. Mediante la representación de un diagrama de componentes se describió la estructura y organización del sistema. Se visualizaron las interfaces de usuario, ofreciendo una explicación de cada una de ellas para un mayor entendimiento de la solución. El desglose de las historias de usuario en tareas de ingeniería constituyó una buena práctica que mostró a los desarrolladores las funcionalidades específicas a implementar. La realización de las pruebas de aceptación permitió comprobar el cumplimiento de los requisitos funcionales después de cada iteración así como la aceptación del sistema.

Conclusiones generales.

Con el desarrollo de esta investigación se ha podido demostrar el cumplimiento de los objetivos principales, definidos para la creación de un sistema que permita brindar servicios de verificación de identidad en línea, con el propósito de potenciar la seguridad en las instituciones que lo implementen. El análisis de las tecnologías relacionadas con el objeto de estudio permitió una correcta elección de la metodología y herramientas necesarias para el desarrollo de la solución, mayormente basadas en licencias de software libre, encaminado a obtener un producto de alta independencia tecnológica. La definición de la arquitectura cliente/servidor y el uso del patrón MVC permitieron garantizar una mayor flexibilidad en la ejecución de la propuesta de solución. El desarrollo guiado por pruebas aseguró el cumplimiento de los objetivos trazados en las historias de usuario. El sistema desarrollado tiene la capacidad de ser multi-empresa por lo que en su base de datos puede almacenar información biométrica de individuos pertenecientes a diferentes instituciones.

Recomendaciones.

Con el objetivo de incorporar mejoras al “Sistema para la verificación de personas en línea a través de huellas dactilares” se recomienda:

- Desarrollar un componente basado en tecnologías libres que permita la comunicación con el lector de huellas dactilares.
- Confeccionar un manual de usuario para los desarrolladores con información detallada del sistema.

Referencias bibliográficas.

1. **Jain, A. k., Bolle, R. y Pankanti, S.** *Biometric: Personal Identification in Networked Society.* s.l. : Kluwer Academic Publishers., 1999.

2. Umanick. [En línea] [Citado el: 20 de Octubre de 2012.] http://www.umanick.com/index.php/tecnologia/huella-dactilar.

3. Biometría. [En línea] Agosto de 2006. [Citado el: 25 de Octubre de 2012.] http://www.biometria.gov.ar/metodos-biometricos/dactilar.aspx.

4. **Martin Rodríguez, Fernando y Suárez López, Francisco Javier.** *Identificación Dactilar basada en filtros Gabor.*

5. **Jianjiang, Feng y Jain, Anil K.** *Model Based Fingerprint Reconstruction from Minutia Template.* s.l. : Michigan State, 209.

6. **Timatá, Juan.** [En línea] [Citado el: 5 de Noviembre de 2012.] http://www.webtaller.com/maletin/articulos/paginas-web-dinamicas-estaticas.php.

7. **Campos Rodríguez, Sandy y Rodríguez Sánchez, Héctor Luis.** *Sistema para la interacción y control centralizado de dispositivos en aplicaciones web v2.0.* 2012.

8. **Cantero, Diana.** *Análisis y diseño para el componente de captura de imágenes faciales con múltiples modelos de cámaras digitales.* 2011.

9. **Navarro, Ricardo.** TOC. [En línea] [Citado el: 10 de Noviembre de 2012.] http://toc.cl.

10. **Sihit, Gustavo.** Biometrika. [En línea] [Citado el: 11 de Noviembre de 2012.] http://biometrikaclsoporte.blogspot.com/2010/09/nueva-version-de-bioportal.html.

11. Sitio oficial Datys. [En línea] [Citado el: 15 de Noviembre de 2012.] http://www.datys.cu/wpinfproducto.aspx?22.

12. **Solís, C. J., Figueroa, R. G. y Cabrera, A. A.** *Metodologías tradicionales vs. Metodologías ágiles.* 2011.

13. **Joskowicz, I. J.** *Reglas y prácticas en Extreme Programming.* 2008.

14. **Cortizo Pérez, J. C., Expósito Gil, Diego y Ruiz Leyva, Miguel.** *Extreme Programming.*

15. **Deemer, Pete, y otros, y otros.** *The Scrum primer.* 2009.

16. **Molpeceres, Alberto.** *Procesos de desarrollo: RUP, XP y FDD.* 2003.

17. CODEBOX. [En línea] [Citado el: 25 de Noviembre de 2012.] http://www.codebox.es/glosario.

18. Sitio oficial ASP.NET. [En línea] [Citado el: 5 de Diciembre de 2012.] http://www.asp.net.

19. **Zaninotto, Francois y Potencier, Fabien.** *Symfony, la guía definitiva.* s.l. : Apress.

20. **Bertolami, Leandro.** [En línea] [Citado el: 9 de Diciembre de 2012.] http://www.1024.com.uy/revista/index.php/galileo-galilei/110-grails-el-santo-grial-de-java.

21. Página oficial de Groovy. [En línea] [Citado el: 11 de Diciembre de 2012.] http:// groovy.codehaus.org.

22. [En línea] 15 de Febrero de 2011. [Citado el: 15 de Diciembre de 2012.] http://programaciondesarrollo.es/que-es-un-entorno-de-desarrollo-integrado-ide.

23. **Montero Garrido, Jesús Manuel.** *Plataforma Eclipse. Introducción técnica.*

24. Sitio oficial Netbeans. [En línea] [Citado el: 15 de Diciembre de 2012.] http://netbeans.org/index_es.html.

25. [En línea] [Citado el: 18 de Diciembre de 2012.] http://licenciaantivirus.blogspot.com/2012/11/netbeans-ide-721-programar-en-cc-y-java.html.

26. **Marrero, David.** [En línea] Diciembre de 2011. [Citado el: 21 de Diciembre de 2012.] http://wwwdavidmarrero.blogspot.com/2011/12/caracteristicas-del-postgresql.html.

27. PostgreSQL. [En línea] [Citado el: 5 de Enero de 2013.] http://www.postgresql.org/about/press/presskit90/es.

28. **Pecos, Daniel.** PostgreSQL vs. MySQL. [En línea] http://danielpecos.com/docs/mysql_postgres/x15.html.

29. **Sparks, Geoffrey.** *El modelo de componentes. Una introducción al UML.* Australia : Sparx systems.

30. **Menéndez Alonso.** Herramientas CASE para el proceso de desarrollo de software. [En línea] http://www.monografias.com/trabajos73/herramientas-case-proceso-desarrollo-software/herramientas-case-proceso-desarrollo-software.shtml.

31. **Foundation., The Apache Software.** Apache. [En línea] 2011. [Citado el: 20 de Enero de 2013.] http://apache.org.

32. **Larman, Craig.** *UML y patrones. Introducción al análisis y diseño orientado a objetos.* México : Prentice Hall, 1999. 970-17-0261-1.

33. **Gamma, Erich, y otros, y otros.** *Design Patterns: Elements of Reusable Object-Oriented Software.* Massachusetts : Addison Wesley Longman, 1995.

Glosario de términos.

AFIS: Sistema Automatizado de Identificación de Huellas dactilares.

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

Ant: Herramienta para compilar programas javas.

API's: Interfaces de Programación de Aplicaciones que da a los programadores los medios para desarrollar aplicaciones Java.

BeOS: Es un sistema operativo desarrollado por Be Incorporated en 1990, orientado principalmente a proveer alto rendimiento en aplicaciones multimedia.

HTML: (Hypertext Markup Language) es un lenguaje estático para el desarrollo de sitios web.

HTTP: (Protocolo de transferencia de hipertexto) protocolo que establece un esquema de comunicación cliente/servidor.

JavaFX: Plataforma que proporciona a los desarrolladores un marco de desarrollo y entorno de ejecución para crear aplicaciones empresariales y de negocios que se ejecutan en múltiples plataformas que soportan Java.

Linux: Es un sistema operativo libre, con características muy semejantes a UNIX.

ORDBMS: Sistema de Gestión de Bases de Datos objeto-relacional.

PHP: (Hypertext Preprocessor) o Preprocesador de Hipertexto, es un lenguaje de código abierto muy popular especialmente adecuado para desarrollo.

Servicio en línea: Servicio que puede ser prestado por medios electrónicos a través del portal de una institución.

SQL: (Structured Query Language) o lenguaje de consulta estructurado, es un lenguaje declarativo de acceso a bases de datos.

UNIX: Es un sistema operativo portable, multitarea y multiusuario; desarrollado por primera vez en 1969.

Windows: Microsoft Windows es el nombre de una serie de sistemas operativos desarrollados por Microsoft desde 1981.

XHTML: (eXtensible Hypertext Markup Language), es un lenguaje similar a HTML expresado como XML, por lo que es más robusto para la modelación de páginas web.

XML: (Extensible Markup Language) en español Lenguaje de Marcaje Extensible.

Anexo 1: Historias de usuario.

Historia de usuario	
Número: HU_6	**Usuario:** Administrador.
Nombre de historia: Gestionar licencia.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable: Yaricel Guerra Velázquez.	
Descripción: El administrador del sistema en el servidor tendrá la posibilidad de adicionar, modificar y desactivar las licencias.	
Observaciones: No tiene observaciones.	

Tabla 14. HU_6 Gestionar licencia.

Historia de usuario	
Número: HU_7	**Usuario:** Administrador.
Nombre de historia: Configurar servicios.	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Medio
Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable: Yaricel Guerra Velázquez.	
Descripción: El administrador del sistema en el servidor tendrá la posibilidad de configurar los servicios que brinda el sistema según las necesidades y preferencias del cliente.	
Observaciones: No tiene observaciones.	

Tabla 15. HU_7 Configurar servicios.

Historia de usuario	
Número: HU_8	**Usuario:** Sistema.
Nombre de historia: Realizar conexión con el DGM.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alto
Puntos estimados: 1	**Iteración asignada:** 2
Programador responsable: Eliandis Matos Moreira.	
Descripción: Permite establecer la conexión con el controlador de dispositivos (DGM).	
Observaciones: En caso de que no se pueda establecer la conexión con el DGM en ese momento, el sistema muestra un aviso.	

Tabla 16. HU_8 Realizar conexión con el DGM.

Historia de usuario	
Número: HU_9	**Usuario:** Sistema.
Nombre de historia: Capturar imagen de huella dactilar.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alto
Puntos estimados: 1	**Iteración asignada:** 2
Programador responsable: Yaricel Guerra Velázquez.	
Descripción: Permite obtener la imagen de la huella dactilar capturada por el lector de huellas.	
Observaciones: En caso de que no exista ningún lector conectado al ordenador, el sistema muestra un aviso.	

Tabla 17. HU_9 Capturar imagen de huella dactilar.

Historia de usuario	
Número: HU_10	**Usuario:** Verificador.

Nombre de historia: Verificar identidad del individuo.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alto
Puntos estimados: 2	**Iteración asignada:** 2
Programador responsable: Eliandis Matos Moreira.	
Descripción: Permite verificar la identidad del individuo a partir de la huella dactilar, comparando la huella dactilar tomada por el lector con la huella de ese individuo guardada con anterioridad en la base de datos o a partir de un identificador.	
Observaciones: En caso de que no se haya realizado una buena captura de la imagen existirán problemas para verificar, por lo que el sistema muestra un aviso.	

Tabla 18. HU_10 Verificar identidad del individuo.

Historia de usuario	
Número: HU_11	**Usuario:** Sistema.
Nombre de historia: Mostrar datos del individuo.	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Medio
Puntos estimados: 1	**Iteración asignada:** 2
Programador responsable: Yaricel Guerra Velázquez.	
Descripción: Permite mostrar los datos del individuo luego de haber verificado su identidad exitosamente.	
Observaciones: En caso de no encontrarse registrado en la base de datos, el sistema muestra un aviso.	

Tabla 19. HU_11 Mostrar datos del individuo.

Historia de usuario	
Número: HU_12	**Usuario:** Administrador.
Nombre de historia: Cambiar estado de los servicios.	

Prioridad en negocio: Media	**Riesgo en desarrollo:** Medio
Puntos estimados: 1	**Iteración asignada:** 3
Programador responsable: Eliandis Matos Moreira.	
Descripción: El administrador del sistema en el servidor tiene la posibilidad de habilitar y deshabilitar los servicios que se brindan a un cliente.	
Observaciones: No tiene observaciones.	

Tabla 20. HU_12 Cambiar estado de los servicios.

Historia de usuario	
Número: HU_13	**Usuario:** Sistema.
Nombre de historia: Mostrar historial de verificaciones.	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Medio
Puntos estimados: 1	**Iteración asignada:** 3
Programador responsable: Yaricel Guerra Velázquez.	
Descripción: Permite mostrar el historial de las verificaciones realizadas en el sistema según usuario y persona.	
Observaciones: No tiene observaciones.	

Tabla 21. HU_13 Mostrar historial de verificaciones.

Historia de usuario	
Número: HU_14	**Usuario:** Sistema.
Nombre de historia: Mostrar reportes.	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Alto
Puntos estimados: 1	**Iteración asignada:** 3

Programador responsable: Eliandis Matos Moreira.
Descripción: Permite mostrar los reportes de verificación no exitosa y de vencimiento de licencia.
Observaciones: No tiene observaciones.

Tabla 22. HU_14 Mostrar reportes.

Anexo 2: Estimación de tiempo.

Historia de usuario	Estimación (semanas)
Buscar usuario.	1
Gestionar usuario.	1
Autenticar usuario.	1
Verificar permisos.	1
Gestionar institución.	1
Gestionar licencia.	1
Configurar servicios.	1
Realizar conexión con el DGM.	1
Capturar imagen de huella dactilar.	1
Verificar identidad del individuo.	2
Mostrar datos del individuo.	1
Cambiar estado de los servicios.	1
Mostrar historial de verificaciones.	1
Mostrar reportes.	1
Total.	15

Tabla 23. Estimación de tiempo.

Anexo 3: Tareas de ingeniería.

Tarea	
Número de tarea: 1	**Número de HU:** 1
Nombre de la tarea: Crear la tabla usuario en la base de datos.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.5
Fecha de inicio: 10/01/2013	**Fecha de fin:** 12/01/2013
Programador responsable: Yaricel Guerra Velázquez.	
Descripción: Se crea en la base de datos la tabla correspondiente a los usuarios registrados.	

Tabla 24. Tarea 1 "HU_1 Buscar usuario".

Tarea	
Número de tarea: 2	**Número de HU:** 1
Nombre de la tarea: Implementar el buscar usuario.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.5
Fecha de inicio: 13/01/2013	**Fecha de fin:** 16/01/2013
Programador responsable: Yaricel Guerra Velázquez.	
Descripción: Se implementa la funcionalidad que permite buscar usuarios en el sistema.	

Tabla 25. Tarea 2 "HU_1 Buscar usuario".

Tarea	
Número de tarea: 3	**Número de HU:** 2
Nombre de la tarea: Implementar el adicionar usuario.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2

Fecha de inicio: 17/01/2013	**Fecha de fin:** 18/01/2013
Programador responsable: Eliandis Matos Moreira.	
Descripción: Se implementa la funcionalidad que permite adicionar usuarios en el sistema.	

Tabla 26. Tarea 3 "HU_2 Gestionar usuario".

Tarea	
Número de tarea: 4	**Número de HU:** 2
Nombre de la tarea: Implementar el eliminar usuario.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha de inicio: 19/01/2013	**Fecha de fin:** 20/01/2013
Programador responsable: Eliandis Matos Moreira.	
Descripción: Se implementa la funcionalidad que permite eliminar usuarios en el sistema.	

Tabla 27. Tarea 4 "HU_2 Gestionar usuario".

Tarea	
Número de tarea: 5	**Número de HU:** 2
Nombre de la tarea: Implementar el modificar usuario.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.3
Fecha de inicio: 21/01/2013	**Fecha de fin:** 23/01/2013
Programador responsable: Eliandis Matos Moreira.	
Descripción: Se implementa la funcionalidad que permite modificar usuarios en el sistema.	

Tabla 28. Tarea 5 "HU_2 Gestionar usuario".

Tarea	
Número de tarea: 6	**Número de HU:** 2
Nombre de la tarea: Implementar el asignar rol al usuario.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.3
Fecha de inicio: 24/01/2013	**Fecha de fin:** 26/01/2013
Programador responsable: Eliandis Matos Moreira.	
Descripción: Se implementa la funcionalidad que permite al administrador asignarle roles a los usuarios en el sistema.	

Tabla 29. Tarea 6 "HU_2 Gestionar usuario".

Tarea	
Número de tarea: 7	**Número de HU:** 3
Nombre de la tarea: Implementar la interfaz de autenticación.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha de inicio: 27/01/2013	**Fecha de fin:** 03/02/2013
Programador responsable: Yaricel Guerra Velázquez.	
Descripción: Se crea una interfaz para que el usuario inserte sus datos (usuario y contraseña).	

Tabla 30. Tarea 7 "HU_3 Autenticar usuario".

Tarea	
Número de tarea: 8	**Número de HU:** 4
Nombre de la tarea: Implementar el verificar permisos.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1

Fecha de inicio: 04/02/2013	**Fecha de fin:** 10/02/2013
Programador responsable: Eliandis Matos Moreira.	
Descripción: Se implementa la funcionalidad que permite verificar los permisos de los usuarios dependiendo del rol que desempeñen.	

Tabla 31. Tarea 8 "HU_4 Verificar permisos".

Tarea	
Número de tarea: 9	**Número de HU:** 5
Nombre de la tarea: Implementar el adicionar institución.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.3
Fecha de inicio: 11/02/2013	**Fecha de fin:** 13/02/2013
Programador responsable: Eliandis Matos Moreira.	
Descripción: Se implementa la funcionalidad que permite adicionar instituciones en el sistema.	

Tabla 32. Tarea 9 "HU_5 Gestionar institución".

Tarea	
Número de tarea: 10	**Número de HU:** 5
Nombre de la tarea: Implementar el eliminar institución.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.3
Fecha de inicio: 14/02/2013	**Fecha de fin:** 16/02/2013
Programador responsable: Eliandis Matos Moreira.	
Descripción: Se implementa la funcionalidad que permite eliminar instituciones en el sistema.	

Tabla 33. Tarea 10 "HU_5 Gestionar institución".

Tarea	
Número de tarea: 11	**Número de HU:** 5
Nombre de la tarea: Implementar el modificar institución.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.3
Fecha de inicio: 17/02/2013	**Fecha de fin:** 19/02/2013
Programador responsable: Eliandis Matos Moreira.	
Descripción: Se implementa la funcionalidad que permite modificar instituciones en el sistema.	

Tabla 34. Tarea 11 "HU_5 Gestionar institución".

Tarea	
Número de tarea: 12	**Número de HU:** 6
Nombre de la tarea: Implementar el adicionar licencia.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.3
Fecha de inicio: 20/02/2013	**Fecha de fin:** 22/02/2013
Programador responsable: Yaricel Guerra Velázquez.	
Descripción: Se implementa la funcionalidad que permite adicionar licencias en el sistema.	

Tabla 35. Tarea 12 "HU_6 Gestionar licencia".

Tarea	
Número de tarea: 13	**Número de HU:** 6
Nombre de la tarea: Implementar el eliminar licencia.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.3
Fecha de inicio: 23/02/2013	**Fecha de fin:** 25/02/2013

Programador responsable: Yaricel Guerra Velázquez.
Descripción: Se implementa la funcionalidad que permite eliminar licencias en el sistema.

Tabla 36. Tarea 13 "HU_6 Gestionar licencia".

Tarea	
Número de tarea: 14	**Número de HU:** 6
Nombre de la tarea: Implementar el modificar licencia.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.3
Fecha de inicio: 26/02/2013	**Fecha de fin:** 27/02/2013
Programador responsable: Yaricel Guerra Velázquez.	
Descripción: Se implementa la funcionalidad que permite modificar licencias en el sistema.	

Tabla 37. Tarea 14 "HU_6 Gestionar licencia".

Tarea	
Número de tarea: 15	**Número de HU:** 7
Nombre de la tarea: Implementar la configuración de los servicios.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha de inicio: 28/02/2013	**Fecha de fin:** 04/03/2013
Programador responsable: Yaricel Guerra Velázquez.	
Descripción: Se implementa la funcionalidad que permite configurar los servicios que brinda el sistema.	

Tabla 38. Tarea 15 "HU_7 Configurar servicios".

Tarea

Número de tarea: 16	**Número de HU:** 8
Nombre de la tarea: Integrar el componente DGM.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.5
Fecha de inicio: 05/03/2013	**Fecha de fin:** 07/02/2013
Programador responsable: Eliandis Matos Moreira.	
Descripción: Se integra el componente DGM al sistema.	

Tabla 39. Tarea 16 "HU_8 Realizar conexión con el DGM".

Tarea	
Número de tarea: 17	**Número de HU:** 8
Nombre de la tarea: Realizar conexión con el DGM.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.5
Fecha de inicio: 09/03/2013	**Fecha de fin:** 11/03/2013
Programador responsable: Eliandis Matos Moreira.	
Descripción: Se implementa la funcionalidad que permite realizar la conexión con el controlador de dispositivos.	

Tabla 40. Tarea 16 "HU_8 Realizar conexión con el DGM".

Tarea	
Número de tarea: 18	**Número de HU:** 9
Nombre de la tarea: Capturar imagen de huella dactilar.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha de inicio: 12/03/2013	**Fecha de fin:** 18/03/2013
Programador responsable: Yaricel Guerra Velázquez.	

Descripción: Se implementa la funcionalidad que permite obtener la imagen de la huella dactilar capturada por el lector.

Tabla 41. Tarea 18 "HU_9 Capturar imagen de huella dactilar".

Tarea	
Número de tarea: 19	**Número de HU:** 10
Nombre de la tarea: Integrar el componente de extracción de características.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha de inicio: 19/03/2013	**Fecha de fin:** 25/03/2013
Programador responsable: Eliandis Matos Moreira.	
Descripción: Se implementa la funcionalidad que permite integrar el componente de extracción de características de la huella dactilar.	

Tabla 42. Tarea 19 "HU_10 Verificar identidad del individuo".

Tarea	
Número de tarea: 20	**Número de HU:** 10
Nombre de la tarea: Integrar el componente de comparación de características.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha de inicio: 26/03/2013	**Fecha de fin:** 01/04/2013
Programador responsable: Eliandis Matos Moreira.	
Descripción: Se implementa la funcionalidad que permite integrar el componente de comparación de características.	

Tabla 43. Tarea 20 "HU_10 Verificar identidad del individuo".

Tarea	
Número de tarea: 21	**Número de HU:** 11
Nombre de la tarea: Mostrar datos del individuo.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha de inicio: 02/03/2013	**Fecha de fin:** 08/04/2013
Programador responsable: Yaricel Guerra Velázquez.	
Descripción: Se implementa la funcionalidad que permite mostrar los datos del individuo a verificar en el sistema.	

Tabla 44. Tarea 21 "HU_11 Mostrar datos del individuo".

Tarea	
Número de tarea: 22	**Número de HU:** 12
Nombre de la tarea: Habilitar servicios.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.5
Fecha de inicio: 09/04/2013	**Fecha de fin:** 12/04/2013
Programador responsable: Eliandis Matos Moreira.	
Descripción: Se implementan las funcionalidades que permiten habilitar los servicios que brinda el sistema.	

Tabla 45. Tarea 22 "HU_13 Cambiar estado de los servicios".

Tarea	
Número de tarea: 23	**Número de HU:** 12
Nombre de la tarea: Deshabilitar servicios.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.5
Fecha de inicio: 13/04/2013	**Fecha de fin:** 16/04/2013

Programador responsable: Eliandis Matos Moreira.
Descripción: Se implementa la funcionalidad que permite deshabilitar los servicios en el sistema.

Tabla 46. Tarea 23 "HU_13 Cambiar estado de los servicios".

Tarea	
Número de tarea: 24	**Número de HU:** 13
Nombre de la tarea: Mostrar historial de verificaciones según usuario.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha de inicio: 17/04/2013	**Fecha de fin:** 18/04/2013
Programador responsable: Yaricel Guerra Velázquez.	
Descripción: Se implementa la funcionalidad que permite mostrar el historial de verificaciones según usuario.	

Tabla 47. Tarea 24 "HU_13 Mostrar historial de verificaciones".

Tarea	
Número de tarea: 25	**Número de HU:** 13
Nombre de la tarea: Mostrar historial de verificaciones según persona.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha de inicio: 19/04/2013	**Fecha de fin:** 20/04/2013
Programador responsable: Yaricel Guerra Velázquez.	
Descripción: Se implementa la funcionalidad que permite mostrar el historial de verificaciones según persona.	

Tabla 48. Tarea 25 "HU_13 Mostrar historial de verificaciones".

Tarea

Número de tarea: 26	**Número de HU:** 14
Nombre de la tarea: Mostrar reporte de vencimiento de licencia.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha de inicio: 23/04/2013	**Fecha de fin:** 24/04/2013
Programador responsable: Eliandis Matos Moreira.	
Descripción: Se implementa la funcionalidad que permite reportar el vencimiento de la licencia.	

Tabla 49. Tarea 27 "HU_14 Mostrar reporte".

Tarea	
Número de tarea: 27	**Número de HU:** 14
Nombre de la tarea: Mostrar reporte de verificación no exitosa.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha de inicio: 25/04/2013	**Fecha de fin:** 02/05/2013
Programador responsable: Eliandis Matos Moreira.	
Descripción: Se implementa la funcionalidad que permite reportar las verificaciones no exitosas.	

Tabla 50. Tarea 27 "HU_14 Mostrar reporte".

Anexo 4: Casos de prueba de aceptación.

Caso de prueba de aceptación	
Código: HU3_CP3	**HU:** 3
Nombre: Autenticar usuario.	
Descripción: Se prueba la funcionalidad de autenticar usuarios en el sistema.	
Condiciones de ejecución: El usuario que realiza la acción debe estar registrado en el sistema. La dirección URL debe contener el identificador de la institución a la que pertenece y en caso de ser el súper administrador el identificador es admin.	
Entrada/Pasos de ejecución y salidas: **Entrada:** Se muestra una interfaz que contiene un formulario con los campos correspondientes a los siguientes datos: usuario y contraseña, luego de introducir estos datos se oprime el botón *Aceptar*. **Salida:** Se autentica correctamente al usuario.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 51. Caso de prueba de aceptación HU3_CP3 "Autenticar usuario".

Caso de prueba de aceptación	
Código: HU4_CP4	**HU:** 4
Nombre: Verificar permisos.	
Descripción: Se prueba la funcionalidad de verificar permisos para el uso de la aplicación. Solo si el usuario posee los permisos necesarios podrá acceder al recurso que desea.	
Condiciones de ejecución: El usuario que realiza la acción debe estar previamente autenticado en el sistema.	
Entrada/Pasos de ejecución y salidas:	

Entrada: El usuario que desea ejecutar la acción solo debe intentar acceder al recurso que desee. **Salida:** El usuario accede a los recursos que desea solo si tiene permisos.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 52. Caso de prueba de aceptación HU4_CP4 "Verificar permisos".

Caso de prueba de aceptación	
Código: HU5_CP5	**HU:** 5
Nombre: Gestionar institución.	
Descripción: Se debe comprobar la inserción, modificación y eliminación de las instituciones en el sistema. Si los datos introducidos tienen errores o se dejan campos en blanco, el sistema muestra un mensaje de error. En el caso de estar correctos estos datos, se guardan en la base de datos satisfactoriamente.	
Condiciones de ejecución: El usuario que realiza la acción debe estar previamente autenticado y tener permisos administrativos.	
Entrada/Pasos de ejecución y salidas: **Entrada: "Escenario Insertar Institución"** Se selecciona la opción *Crear entidad*, mostrándose un formulario con los campos correspondientes a los siguientes datos: identificador, nombre, dirección, ip, posteriormente se selecciona el botón *Crear*, de esta forma queda realizada la inserción. **Salida: "Escenario insertar Institución"** Se adiciona la institución en la base de datos. **Entrada: "Escenario Modificar datos de la Institución"** Se muestra una interfaz con un listado de las instituciones, se selecciona la institución que se desea modificar y se oprime el botón *Editar* en la nueva interfaz, mostrándose un formulario con los campos actuales de la institución seleccionada, debe permitirse modificar cualquiera de los datos pertenecientes a esta, posteriormente	

se oprime el botón *Actualizar*. **Salida: "Escenario Modificar datos de la Institución"** Se actualiza la institución en la base de datos. **Entrada: "Escenario Eliminar Institución"** Se muestra una interfaz con un listado de todas las instituciones, se selecciona la institución que se desea eliminar y se oprime el botón *Eliminar* en la nueva interfaz. **Salida: "Escenario Eliminar Institución"** Se elimina la institución seleccionada.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 53. Caso de prueba de aceptación HU5_CP5 "Gestionar institución".

Caso de prueba de aceptación	
Código: HU6_CP6	**HU:** 6
Nombre: Gestionar licencia.	
Descripción: Se debe comprobar la inserción, modificación y eliminación de las licencias en el sistema. Si los datos introducidos tienen errores o se dejan campos en blanco, el sistema muestra un mensaje de error. En el caso de estar correctos estos datos, se guardan en la base de datos satisfactoriamente.	
Condiciones de ejecución: El usuario que realiza la acción debe estar previamente autenticado y tener permisos administrativos.	
Entrada/Pasos de ejecución y salidas: **Entrada: "Escenario Insertar licencia"** Se selecciona la opción *Crear licencia*, mostrándose un formulario con los campos correspondientes a los siguientes datos: cantidad máxima, cantidad real, fecha activación, entidad y servicio, posteriormente se selecciona el botón *Crear*, de esta forma queda realizada la inserción. **Salida: "Escenario insertar licencia"**	

Se adiciona la licencia en la base de datos. **Entrada: "Escenario Modificar datos de la licencia"** Se muestra una interfaz con un listado de las licencias, se selecciona la licencia que se desea modificar y se oprime el botón *Editar* en la nueva interfaz, mostrándose un formulario con los campos actuales de la institución seleccionada, debe permitirse modificar cualquiera de los datos pertenecientes a esta, posteriormente se oprime el botón *Actualizar*. **Salida: "Escenario Modificar datos de la licencia"** Se actualiza la licencia en la base de datos. **Entrada: "Escenario Eliminar licencia"** Se muestra una interfaz con un listado de todas las licencias, se selecciona la licencia que se desea eliminar y se oprime el botón Eliminar en la nueva interfaz. **Salida: "Escenario Eliminar Institución"** Se elimina la licencia seleccionada.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 54. Caso de prueba de aceptación HU6_CP6 "Gestionar licencia".

Caso de prueba de aceptación	
Código: HU7_CP7	HU: 7
Nombre: Configurar servicio.	
Descripción: Se prueba la funcionalidad de configurar servicios en el sistema.	
Condiciones de ejecución: El usuario que realiza la acción debe estar previamente autenticado y tener permisos administrativos.	
Entrada/Pasos de ejecución y salidas: **Entrada:** Se selecciona la opción *Crear servicio*, luego se muestra una interfaz con los campos correspondientes a los	

siguientes datos: activo, nombre, entidad, verificar huella, posteriormente se oprime el botón *Crear*. **Salida:** Se configura correctamente el servicio.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 55. Caso de prueba de aceptación HU7_CP7 "Configurar servicio".

Caso de prueba de aceptación	
Código: HU8_CP8	**HU:** 8
Nombre: Realizar conexión con el DGM.	
Descripción: Se prueba la funcionalidad de realizar la conexión con el DGM. Al cargarse la página que requiera del lector de huella debe mostrarse un mensaje informando que el lector está listo.	
Condiciones de ejecución: El usuario que realiza la acción debe estar previamente autenticado. Debe haber conectado al menos un lector de huellas.	
Entrada/Pasos de ejecución y salidas: **Entrada:** Se selecciona la opción verificar persona, editar o crear huella dactilar. **Salida:** Se conecta correctamente el lector de huellas.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 56. Caso de prueba de aceptación HU8_CP8 "Realizar conexión con el DGM".

Caso de prueba de aceptación	
Código: HU9_CP9	**HU:** 9
Nombre: Capturar la imagen de la huella dactilar.	
Descripción: Se prueba la funcionalidad de capturar la imagen de la huella dactilar. Se captura la huella dactilar	

en vivo del individuo a verificar (muestra actual).
Condiciones de ejecución: El usuario que realiza la acción debe estar previamente autenticado y tiene que tener permisos para ello, además en caso de realizar la verificación la entidad a la que pertenece debe tener contratado y activo dicho servicio. Debe haber conectado al menos un lector de huellas.
Entrada/Pasos de ejecución y salidas: **Entrada:** Se muestra una interfaz donde se oprime el botón *Capturar***.** **Salida:** Se captura correctamente la huella dactilar.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 57. Caso de prueba de aceptación HU9_CP9 "Capturar la imagen de la huella dactilar".

Caso de prueba de aceptación	
Código: HU10_CP10	**HU:** 10
Nombre: Verificar identidad del individuo.	
Descripción: Se prueba la funcionalidad de verificar identidad del individuo que puede realizarse a través de un identificador o de la huella dactilar.	
Condiciones de ejecución: El usuario que realiza la acción debe estar previamente autenticado y tiene que tener permisos para ello, además en caso de realizar la verificación la entidad a la que pertenece debe tener contratado y activo dicho servicio. Debe haber conectado al menos un lector de huellas.	
Entrada/Pasos de ejecución y salidas: **Entrada: "Escenario Verificar identidad a través de identificador"** Se muestra una interfaz que contiene un formulario con el campo correspondiente al siguiente dato: identificador, luego de introducir este dato se oprime el botón *Aceptar*. **Salida: "Escenario Verificar identidad a través de identificador"**	

Se verifica correctamente al individuo. **Entrada: "Escenario Verificar identidad a través de la huella dactilar"** Se muestra una interfaz que contiene la huella dactilar del individuo (muestra actual) donde se oprime el botón *Procesar.* **Salida: "Escenario Verificar identidad a través de la huella dactilar"** Se verifica correctamente al individuo.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 58. Caso de prueba de aceptación HU10_CP10 "Verificar identidad del individuo".

Caso de prueba de aceptación	
Código: HU11_CP11	**HU:** 11
Nombre: Mostrar datos del individuo.	
Descripción: Se prueba la funcionalidad de mostrar datos del individuo.	
Condiciones de ejecución: El usuario que realiza la acción debe estar previamente autenticado.	
Entrada/Pasos de ejecución y salidas: **Entrada:** Se muestra una interfaz con el listado de las personas almacenadas en la base de datos, donde se selecciona la persona que se desea mostrar. **Salida:** Se muestran los datos del individuo.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 59. Caso de prueba de aceptación HU11_CP11 "Mostrar datos del individuo".

Caso de prueba de aceptación

Código: HU12_CP12	**HU:** 12
Nombre: Cambiar estado de los servicios.	
Descripción: Se prueba la funcionalidad de cambiar el estado de los servicios de habilitado a deshabilitado y viceversa.	
Condiciones de ejecución: El usuario que realiza la acción debe estar previamente autenticado y tener permisos administrativos para ejecutar esta acción.	
Entrada/Pasos de ejecución y salidas: **Entrada: "Escenario Deshabilitar servicio"** Se muestra una interfaz con el listado de los servicios, se selecciona el servicio que se desea deshabilitar donde se oprime el botón *Editar* y se desmarca la opción Activo. **Salida: "Escenario Deshabilitar servicio"** Se deshabilita correctamente el servicio. **Entrada: "Escenario Habilitar servicio"** Se muestra una interfaz con el listado de los servicios, se selecciona el servicio que se desea habilitar donde se oprime el botón *Editar* y se marca la opción Activo. **Salida: "Escenario Habilitar servicio"** Se habilita correctamente el servicio.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 60. Caso de prueba de aceptación HU12_CP12 "Cambiar estado de los servicios".

Caso de prueba de aceptación	
Código: HU13_CP13	**HU:** 13
Nombre: Mostrar historial de verificaciones.	
Descripción: Se prueba la funcionalidad de mostrar historial de verificaciones que puede ser según usuario y persona.	

Condiciones de ejecución: El usuario que realiza la acción debe estar previamente autenticado.
Entrada/Pasos de ejecución y salidas: **Entrada: "Escenario Mostrar historial de verificaciones según usuario"** Se muestra una interfaz con el listado de todos los usuarios, se selecciona el usuario que se desea conocer el historial de verificaciones. **Salida: "Escenario Mostrar historial de verificaciones según usuario"** Se muestra el historial de verificaciones según usuario. **Entrada: "Escenario Mostrar historial de verificaciones según persona"** Se muestra una interfaz con el listado de todas las personas, se selecciona la persona que se desea conocer el historial de verificaciones. **Salida: "Escenario Mostrar historial de verificaciones según persona"** Se muestra el historial de verificaciones según persona.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 61. Caso de prueba de aceptación HU13_CP13 "Mostrar historial de conexiones".

Caso de prueba de aceptación	
Código: HU14_CP14	**HU:** 14
Nombre: Mostrar reportes.	
Descripción: Se prueba la funcionalidad de mostrar reportes de vencimiento de licencia y de las verificaciones no exitosas.	
Condiciones de ejecución: El usuario que realiza la acción debe estar previamente autenticado y tener permisos administrativos.	
Entrada/Pasos de ejecución y salidas: **Entrada: "Escenario Mostrar reporte de vencimiento de licencia"** Se muestra la página principal del administrador de la institución con los últimos 10 reportes. Se debe haber	

agotado la cantidad de verificaciones asignadas a la institución. **Salida: "Escenario Mostrar cantidad de vencimiento de licencia"** Se muestra correctamente el reporte de vencimiento de licencia. **Entrada: "Escenario Mostrar reporte de verificación no exitosa"** Se muestra la página principal del administrador de la institución con los últimos 10 reportes. Se debe intentar verificar a una persona no almacenada en la base de datos o con una identidad falsa. **Salida: "Escenario Mostrar reporte de verificación no exitosa"** Se muestra correctamente el reporte de verificación no exitosa.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 62. Caso de prueba de aceptación HU14_CP14 "Mostrar cantidad de verificaciones".

Anexo 5: No conformidades encontradas en las 3 iteraciones.

Iteración 1					
Elemento	**No.**	**No conformidad**	**Aspectos correspondientes**	**Etapa de detección del error**	**Importancia**
HU_1	1	Realizar la implementación del botón “Buscar”.	Buscar usuario.	Al presionar el botón “Buscar”.	Significativa.
HU_2	2	Cambiar el término “User” por “Usuario”.	Corregir datos.	Al mostrar la interfaz.	Significativa.
HU_3	3	Cambiar el término “User” por “Usuario”.	Corregir datos.	Al mostrar la interfaz.	Significativa.
HU_4	4	Denegar permisos para la gestión de instituciones a usuarios distintos del administrador.	Denegar permisos.	Al intentar acceder a la gestión de instituciones.	Significativa.
HU_5	5	Cambiar el término “LicenciaLista” por “Lista de Licencias”.	Corregir datos.	Al mostrar la interfaz.	Significativa.
HU_6	6	-	-	-	-
HU_7	7	-	-	-	-

Tabla 63. No conformidades detectadas en la iteración 1.

Iteración 2					
Elemento	**No.**	**No conformidad**	**Aspectos correspondientes**	**Etapa de detección del error**	**Importancia**
HU_8	8	Establecer conexión con el lector de huellas.	Establecer conexión.	Al conectar el lector de huellas.	Significativa.
HU_9	9	-	-	-	-
HU_10	10	Cambiar el término “Procesa” por “Procesar”.	Corregir datos.	Al mostrar la interfaz.	Significativa.
HU_10	11	Realizar la implementación del botón “Procesar”.	Procesar huella dactilar.	Al presionar el botón “Procesar”.	Significativa.
HU_11	12	Cambiar el término “Mostra	Corregir datos.	Al mostrar la interfaz.	Significativa.

		datos” por “Mostrar datos”.			

Tabla 64. No conformidades detectadas en la iteración 2.

Iteración 3					
Elemento	**No.**	**No conformidad**	**Aspectos correspondientes**	**Etapa de detección del error**	**Importancia**
HU_12	13	-	-	-	-
HU_13	14	-	-	-	-
HU_14	15	Mostrar reporte de verificación no exitosa.	Mostrar reporte.	Al mostrar la interfaz.	Significativa.

Tabla 65. No conformidades detectadas en la iteración 3.