

Universidad de las Ciencias Informáticas

Facultad 1



Título: *Servicio local del DGM Libre para la interacción de dispositivos de hardware con aplicaciones web en ambientes multiplataforma*

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

Autores: *Gianny Espino Hernández*

Yasmany Escofet Thompson

Tutores: *Ing. Manuel A. Quert Gómez*

Ing. Ivan Campos Cesar

La Habana

Junio 2015



"El futuro de una nación es tan prometedor como su próxima generación de ciudadanos."

Nelson Mandela

Declaración de autoría

Declaramos que somos los únicos autores del trabajo titulado:

Servicio local del DGM Libre para la interacción de dispositivos de hardware con aplicaciones web en ambientes multiplataforma, y concedemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

Autor: Gianni Espino Hernández

Autor: Yasmany Escofet Thompson

Tutor: Ing. Manuel A. Quert Gómez

Tutor: Ing. Iván Campos Cesar.

Dedicatoria

[*Dedicatoria*]

Agradecimientos

[Agradecimientos]

Opinión del tutor

[Opinión del tutor]

Firma del Tutor

Ing. Manuel A. Quert Gómez

Firma del Tutor

Ing. Ivan Campos Cesar

Resumen

Con la aparición de las nuevas tecnologías de la computación, el hombre vio la necesidad de digitalizar la información para lograr una mejor manipulación de la misma y rapidez de acceso a alguna información en específico; un ejemplo claro de esto son las aplicaciones usadas para digitalizar las características humanas a través de las cuales se pueda realizar el reconocimiento facial y escaneo de huellas dactilares. En la actualidad existen disímiles herramientas para realizar este proceso, y de esta forma hacer más fácil y rápida la captación biométrica. En Cuba, el Ministerio del Interior (MININT) es la entidad que realiza este tipo de procesos, utilizando para ello una aplicación llamada Sistema Único de Identificación Nacional (SUIN), el cual trabaja a su vez con otra aplicación denominada *Device Grid Manager* (DGM), que permite el control e interacción centralizado de dispositivos de hardware en aplicaciones web. Esta aplicación permite al SUIN, la comunicación e interacción con los dispositivos de hardware instalados en las estaciones de trabajo para obtener los datos biométricos necesarios en la identificación y personalización de los documentos de identidad de los ciudadanos. Actualmente esta aplicación se encuentra en software privativo, por lo cual el objetivo de este trabajo es migrar dicha aplicación a tecnologías libres brindando que el sistema se puede desplegar en ambiente Linux y se podrá comercializar dicha aplicación.

Para el desarrollo de la solución fueron empleadas varias herramientas y tecnologías como fueron: Visual Paradigm para el lenguaje de modelado UML, Monodevelop y C# fue el lenguaje escogido para la implementación. Se realizó un estudio del estado del arte sobre el funcionamiento de este sistema.

Palabras claves: datos biométricos, reconocimiento facial, privativo.

Índice

Introducción	1
Capítulo 1: Fundamentación Teórica	4
1.1. Introducción.....	4
1.2. Conceptos generales	4
1.3.2. Sistemas informáticos existentes en Cuba	8
1.3.3 Valoración de los sistemas analizados.....	11
1.4 Herramientas y tecnologías para el desarrollo.....	12
1.5. Conclusiones parciales.....	17
Capítulo 2 Características del Sistema.....	19
2.1 Introducción	19
2.2 Presentación de la solución	19
2.3 Modelo Conceptual	20
2.3.1 Glosario de términos del Modelo Conceptual.....	20
2.4 Descripción del sistema propuesto	21
2.5 Procesos para la extensión del Servicio Local para el manejo de dispositivos	22
2.6 Requisitos del sistema	22
2.6.1 Definición de los requisitos funcionales del sistema.....	22
2.6.2 Requisitos no funcionales.....	28
2.7 Patrones de Diseño.....	31
2.8 Conclusiones parciales.....	32
Capítulo 3 Implementación y validación de la solución propuesta.....	34
3.1 Introducción	34
3.2 Implementación.....	34
3.2.1 Plan de Releases	34
3.1.2 Diagrama de despliegue	35
3.1.3 Interfaces de usuario	36
3.2 Estándares de codificación.	37
3.3 Tarjetas CRC.....	38

3.4 Pruebas de software.....	39
3.4.1 Prueba de Caja blanca	39
3.4.2 Pruebas de Aceptación	42
3.4.3 Resultados de las pruebas	44
3.5 Conclusiones parciales.....	45
Conclusiones generales	46
Recomendaciones	47
Referencias bibliográficas.....	48
Anexos	50

Índice de figuras

Figura 2. 1 Abstracción del sistema. Fuente: Elaboración propia.....	19
Figura 2. 2 Modelo conceptual. Fuente: Elaboración propia	20
Figura 2. 3 Representación de Arquitectura n-capas. Fuente: Elaboración propia.	30
Figura 2. 4 Ejemplo de utilización del patrón subclase. Fuente: Elaboración Propia	31
Figura 2. 5 Ejemplo de utilización del patrón singleton. Fuente: Elaboración Propia.....	32
Figura 3. 1 Diagrama de despliegue. Fuente: Elaboración propia.	36
Figura 3. 2 Interfaz del panel de configuración.....	37
Figura 3. 3 Método GetPcId	41
Figura 3. 4 Grafo del flujo asociado al método adicionar asignaciones.....	41
Figura 3. 5 Grafico de las no conformidades. Fuente: Elaboración propia.	44

Índice de tablas

Tabla 1. 1 Sistemas analizados y sus características. Fuente: Elaboración propia.	12
Tabla 2. 1 HU_3 Manejar los dispositivos remotos	24
Tabla 2. 2 Estimación del esfuerzo por historia de usuario	25
Tabla 2. 3 Plan de iteraciones	27
Tabla 2. 4 Plan de entrega	28
Tabla 3. 1 Plan Release.	35
Tabla 3. 2 Tarjeta CRC de la clase PclInformation.	39
Tabla 3. 3 Caminos básicos del flujo asociado al método GetPclId.....	42
Tabla 3. 4 CP Manejar dispositivos remotos.....	44
TablaA1. 1 HU_1 Manejar los servicios web.....	50
TablaA1. 2 Aceptar nuevos controladores de dispositivos	51
TablaA1. 3 HU_4 Aceptar el manejo de dispositivos remotamente	52
TablaA1. 4 Consultar el estado del servicio remotamente.....	53
TablaA1. 5 Validar el acceso a los servicios web.....	53
TablaA1. 6 Gestionar los dispositivos a usar en el servicio local para el manejo de dispositivos.....	55
TablaA1. 7 HU_8 Gestionar los dispositivos a usar en el servicio local	57
TablaA1. 8 HU_9 Actualizar el servicio local para el manejo de dispositivos	58
TablaA2. 1 Tarjeta CRC de la clase ServiceManage	59
TablaA2. 2 Tarjeta CRC de la clase DGMConfiguration.	59
TablaA2. 3 Tarjeta CRC de la clase DGMWindowsService.	59
TablaA2. 4 Tarjeta CRC de la clase WebCam.	60
TablaA2. 5 Tarjeta CRC de la clase CanonSx110.	60
TablaA2. 6 Tarjeta CRC de la clase DactylD15Controller.	61
TablaA2. 7 Tarjeta CRC de la clase CPDIPrinter.	61
TablaA2. 8 Tarjeta CRC de la clase ScannerController.	62
TablaA3. 1 CP Manejar los servicios web.	63
TablaA3. 2 CP Aceptar nuevos controladores de dispositivos.....	64

TablaA3. 3 Aceptar el manejo de dispositivos remotamente.....	64
TablaA3. 4 CP Consultar el estado del servicio remotamente.....	65
TablaA3. 5 CP Validar el acceso a los servicios web.....	66
TablaA3. 6 CP Gestionar los servicios web a usar en el Servicio local para el manejo de dispositivos	67
TablaA3. 7 CP Gestionar los dispositivos a usar en el Servicio local para el manejo de dispositivos.....	67
TablaA3. 8 CP Actualizar el servicio local para el manejo de dispositivos.....	68

Introducción

En la humanidad desde su comienzo, las personas se han visto en la necesidad de diferenciarse unos a los otros utilizando métodos de identificación muy simples, como prendas o cualidades distintivas. Con el desarrollo de la sociedad se incorporaron procesos mucho más efectivos, surgieron los títulos de la nobleza, los apellidos y el asentamiento de nacimientos en las iglesias. Surgen los registros de identificación que le proporciona a una persona la capacidad jurídica. Es decir, será reconocido como miembro de la sociedad, por lo que tendrá una serie de derechos y obligaciones. Además, tendrá acceso a los diferentes servicios necesarios para desarrollarse y construir su vida y su porvenir, como la educación y la sanidad.

Las autoridades competentes otorgan a cada persona un documento de identidad el cual tiene suficiente valor para acreditar la identidad y los datos personales de su titular que en él se consignen. Inicialmente, estos presentaban poca información y con el paso del tiempo se les fue adicionando una mayor cantidad de datos que propiciaban mayor exactitud a la hora de llevar a cabo la identificación. Con la aparición de las tecnologías digitales se aceleraron de forma significativa los trámites en el proceso de identificación, lo que permitió que los documentos se hicieran más fáciles de emitir, pero también más propensos a la falsificación y la suplantación de identidad.

En Cuba, el proceso de identificación ciudadana lo realiza la Dirección de Identificación y Registro (DIR), perteneciente al MININT. Esta, en colaboración con la UCI, desarrolló una aplicación denominada Sistema Único de Identificación Nacional. Esta aplicación se encuentra implantada hoy en las oficinas de Carnet de Identidad a nivel nacional. El SUIN, por seguridad en los trámites, solo utiliza puestos de trabajo y dispositivos registrados en la red interna del MININT. Para la interacción con los dispositivos utiliza un servicio denominado *Device Grid Manager* (DGM) por sus siglas en inglés. Esta es una aplicación que permite el control e interacción centralizado de dispositivos de hardware con aplicaciones web. El sistema se compone por tres subsistemas fundamentales: el Servicio DGM para el manejo de dispositivos (de aquí en adelante Servicio DGM), el *framework* JavaScript DGMJS y el subsistema para el control centralizado de dispositivos. Estos subsistemas trabajan como un todo, basados en un desarrollo estandarizado que facilita la extensión de la aplicación sin grandes esfuerzos por parte de los desarrolladores.

El DGM permite al SUIN obtener los datos necesarios para llevar a cabo los procesos de captación biométrica. Actualmente esta aplicación se encuentra en software privativo por lo que se dificulta la

comercialización del producto y solo puede ser desplegada en estaciones de trabajo con sistema operativo Windows lo que imposibilita que pueda ejecutarse en cualquier otro sistema por lo que es de gran importancia desarrollar una aplicación con tecnologías libres y que se ejecute en ambientes multiplataforma.

Basado en lo anteriormente expuesto surge como **problema a resolver**: ¿Cómo garantizar la comunicación y el control de dispositivos de hardware a través de aplicaciones web en ambientes multiplataforma?

Razón por la cual se propone como **objeto de estudio**: el manejo de dispositivos de hardware a través de aplicaciones web.

El **objetivo general** se enfoca en desarrollar un servicio local para garantizar la comunicación y el control de dispositivos de hardware a través de aplicaciones web en ambientes multiplataforma.

Partiendo de la siguiente **idea a defender**, con el desarrollo del servicio local del DGM Libre, se garantizará la comunicación y el control de dispositivos de hardware a través de aplicaciones web.

Para darle cumplimiento al objetivo planteado se tendrán en cuenta los siguientes **objetivos específicos**:

- ✚ Analizar los principales conceptos asociados a los procesos de comunicación y control de dispositivos para obtener una base teórica necesaria para el desarrollo de la solución.
- ✚ Revisar la bibliografía acerca de los proveedores de hardware, SDK y soluciones existentes que permiten la comunicación con los dispositivos, para determinar fortalezas e insuficiencias en las mismas y valorar la viabilidad de su uso.
- ✚ Realizar un estudio que permita definir cuáles son las herramientas informáticas y metodologías a usar para el desarrollo del sistema.
- ✚ Especificar los requisitos del servicio local para el manejo de dispositivos, con el objetivo de determinar las funcionalidades a implementar.
- ✚ Diseñar la solución del servicio local para el manejo de dispositivos.
- ✚ Desarrollar un componente que permita la actualización de los servicios en las estaciones de trabajo.
- ✚ Desarrollar de un componente que permita la gestión de servicios por cada estación de trabajo que compone la red de dispositivos.
- ✚ Aplicación de las pruebas de software al Servicio local de manejo de dispositivos.

Esta investigación se llevó a cabo haciendo uso de **métodos científicos** de investigación.

De los **métodos teóricos**:

Analítico Sintético: Se utilizó para sintetizar los elementos más relevantes y de mayor envergadura para el desarrollo del trabajo luego de considerarse detalladamente las teorías, tendencias y todo tipo de documento relacionado con la investigación.

Análisis Histórico-Lógico: se utiliza en el estudio de aplicaciones basadas en la comunicación e interacción con dispositivos de hardware, ya sea en implementación, desarrollo, como antecedentes de la misma.

Dentro de los **métodos empíricos**:

Observación: Se utilizó para reconocer las necesidades específicas y características del Proyecto.

Entrevista: Se utilizó para conocer las características específicas del proyecto.

El presente trabajo de diploma consta de 3 capítulos, los cuales se describen brevemente a continuación:

El **capítulo 1** se refiere a la fundamentación teórica del trabajo, describiéndose todos los elementos de la teoría que sostienen el problema científico y los objetivos de la investigación. Se hace referencia al estado del arte del objeto de estudio y además se analizan y fundamentan las tecnologías, lenguajes de programación, herramientas y metodologías para validar y guiar el proceso de desarrollo del software requerido.

Como parte del **capítulo 2** se realiza la descripción de la propuesta de solución, teniendo en cuenta la especificación de los requerimientos no funcionales, historias de usuarios, diagramas de clases, del análisis y del diseño, así como la descripción de la arquitectura y patrones de diseños utilizados, con el objetivo de lograr un mayor entendimiento de la fase de diseño.

En el **capítulo 3** y último, denominado Implementación y validación de la solución propuesta, se exponen los diagramas de componentes y de despliegue, además se realiza la validación de la solución desarrollada, mediante la realización de pruebas y métricas de calidad, además de la comprobación del cumplimiento del objetivo general de la investigación.

Para cada capítulo se ofrecen sus conclusiones parciales y al final del documento se exponen las conclusiones generales y las recomendaciones.

Capítulo 1: Fundamentación Teórica

1.1. Introducción

El presente capítulo contiene la fundamentación teórica, que da respaldo a la solución propuesta. Se muestran las definiciones de dispositivos de hardware, SDK y como se realiza el control de dispositivos. Se hace además un estudio de sistemas y aplicaciones ya existentes especializadas en la comunicación, control y administración de dispositivos. Se analiza el ambiente de desarrollo propuesto caracterizando las herramientas y tecnologías que se definieron para ser utilizadas en la confección de la solución.

1.2. Conceptos generales

1.2.2. Control de Dispositivos

Regula el acceso a los dispositivos de almacenamiento externo y a los recursos de red conectados a los equipos. Los controladores de dispositivos son programas informáticos que permiten al sistema operativo interactuar con un periférico, haciendo una abstracción del hardware y proporcionando una interfaz estandarizada para usarlo. Sin la cual no se podría usar el hardware. Estos son añadidos al núcleo del sistema operativo, concebidos inicialmente para gestionar periféricos y dispositivos especiales (1).

1.2.3. Dispositivos biométricos

Los dispositivos biométricos son usados en sistemas de cómputo de seguridad, su principal funcionamiento se basa en la identificación de atributos físicos. Estos han sido diseñados para máximos estándares de seguridad y múltiples aplicaciones (2).

1.2.4. Dispositivos de hardware y sus SDK

En la actualidad los dispositivos de hardware juegan un papel crucial en el desarrollo científico técnico, pues permiten a las estaciones de trabajos y a las aplicaciones que se ejecutan en ellas, integrarse en todas las áreas de la ciencia y la técnica. Existen diversos dispositivos que posibilitan la realización de tareas diariamente, como es el caso de la impresora, escáner, micrófono, cámara digital, dispositivos de almacenamiento, entre otros que se encuentran en hogares y centros de trabajo. Coexisten con ellos otro grupo de dispositivos que son utilizados con frecuencia dentro de las soluciones empresariales y se

encargan de automatizar tareas más complejas, ejemplo de estos son los lectores de huellas dactilares, tablas electrónicas para captar firmas, relojes para controlar la entrada y salida de personal, lectores de tarjetas, entre otros (3) (4).

Existen varias empresas que fabrican y distribuyen estos equipos, tal es el caso de Hewlett-Packard, Sony, Canon, Gemalto, GreenBit (5), por solo citar algunos ejemplos. Dichas empresas además de proveer el hardware, facilitan un paquete de drivers que hacen posible la comunicación del dispositivo con la máquina y un software, comúnmente denominado SDK, el cual provee al comprador de un grupo de herramientas, librerías de clases y componentes permitiéndole interactuar desde una perspectiva de desarrollo de software con estos dispositivos.

1.3. Sistemas informáticos similares

En la actualidad existen sistemas informáticos que se encargan del monitoreo de dispositivos de hardware, facilitando de esta forma el trabajo de las empresas y garantizan la seguridad en sus instalaciones. A continuación se realizará un estudio de los mismos tanto nacionales como internacionales, en busca de un sistema que cubra la mayor parte de las necesidades del problema a resolver.

1.3.1. Sistemas informáticos internacionales.

Fingera: Fingera es un paquete biométrico para el control de acceso físico, horario y presencia. Está basado en la tecnología de reconocimiento de huellas dactilares de Innovatrics (6) con dispositivos biométricos de primera clase. Su modularidad permite implementarlo como módulo independiente en pequeñas empresas, así como integrarlo en entornos WAN/LAN a gran escala. En cuanto a la arquitectura, se puede utilizar de modo independiente en pequeñas empresas que solo necesitan un dispositivo de control de horarios y presencia. Este sistema se conecta directamente a la red local (LAN). Cuando es necesario implementar varios dispositivos en una ubicación, se requiere el uso del servidor. En el caso de que se deban instalar más terminales o dispositivos de control de acceso físico, se deben conectar mediante la red local al servidor. La solución Fingera permite realizar tanto sincronizaciones locales como por Internet. Puede conectar varios dispositivos, los cuales se sincronizan mediante el servidor local o por un servidor virtual (por Internet). Esto le permite conectar fácilmente varias oficinas distantes y disponer de un acceso centralizado a todos los datos (7).

Características

- ✚ Se pueden sincronizar varios terminales a través de la LAN o Internet para implementarlo a gran escala.
- ✚ Se puede administrar desde las instalaciones locales a través de la LAN o de forma remota por Internet.
- ✚ El acceso mediante un cliente ligero no requiere instalar ningún software para administrarlo. El acceso a los terminales se realiza a través de un navegador web habitual.
- ✚ Se puede utilizar un relé integrado para el control de acceso físico.
- ✚ Se pueden conectar escáneres adicionales a cualquier terminal que requiera el control de acceso físico en dos direcciones.

Ventajas

- ✚ Evita la suplantación de identidad, ya que requiere varios identificadores biométricos.
- ✚ Informa de manera precisa sobre el horario y la presencia de los empleados a los departamentos de Nóminas y recursos humanos para reducir los costes de procesamiento de las nóminas.
- ✚ Una de las pocas soluciones llave en mano del mercado que gestiona el control de acceso físico, horario y presencia.
- ✚ Uso sencillo e intuitivo
- ✚ Identifica rápidamente las tarjetas y las huellas dactilares.

Biometric Services Platform (BioSP™)¹

Biometric Services Platform (BioSP) de Aware es una plataforma de servidores de aplicaciones orientadas al servicio que se utiliza para habilitar funciones de procesamiento y gestión de datos biométricos en una arquitectura de bus para servicios empresariales (ESB). Es ideal para aplicaciones que necesitan la colección biométrica en toda una red distribuida y el ulterior agregado, análisis, procesamiento, distribución, comparación y uso compartido de los datos con otros componentes del sistema. Este sistema es modular, programable y ampliable; puede administrar todos los aspectos del flujo de trabajo de las transacciones,

¹ Servidor de aplicaciones biométricas orientadas al servicio

entre ellos, el sistema de mensajería, los envíos, las respuestas e inicios de sesión. BioSP hace un uso extensivo de componentes de código abierto y es compatible con J2EE (8).

Características:

- ✚ Realiza funciones automatizadas de análisis, procesamiento, formato, aseguramiento de la calidad de datos e informes
- ✚ Utiliza servicios de web en respaldo de una arquitectura orientada al servicio (SOA) ampliable y segura
- ✚ Integra funciones biométricas en otros sistemas empresarios como la gestión de la identidad, del acceso, de fichas y AFIS/ABIS
- ✚ Realiza comparaciones biométricas 1:1 y 1: varios con fines de verificación, identificación y control de duplicados.
- ✚ Permite administrar sistemas y usuarios de forma centralizada
- ✚ Aporta posibilidades avanzadas de generación de informes para la solución de problemas de captura biométrica
- ✚ Permite la configuración, distribución y gestión centralizadas de software de inscripción cliente
- ✚ Admite las modalidades de huellas dactilares, rostro y palma.

Ventajas:

- ✚ Ofrece una plataforma flexible, configurable y programable para la rápida adaptación a sistemas de legado y distintos requisitos del cliente; ideal para implementar rápidamente sistemas piloto o de demostración.
- ✚ Reduce el costo total de adquisición debido a que simplifica la distribución, la asistencia técnica y el mantenimiento del software
- ✚ Aumenta la eficacia y la confiabilidad mediante una plataforma común y unificada que admite clientes y procesos diversos

- ✚ Reduce los riesgos mediante el uso de bibliotecas de software biométrico Aware probadas y ampliamente adoptadas y componentes de infraestructura de código abierto
- ✚ Mejora el índice de éxito de las transacciones mediante la redundancia de datos, la entrega garantizada y los análisis e informes de calidad automatizados
- ✚ Reduce el impacto en los depósitos centrales mediante el agregado de conexiones cliente y envío y respuestas de transacciones biométricas
- ✚ Reduce los costos de asistencia técnica mediante la eliminación de ambigüedades respecto del estado de las transacciones enviadas, ofreciendo aseguramiento y guía a los operadores de estaciones de trabajo

1.3.2. Sistemas informáticos existentes en Cuba

Device Grid Manager (DGM): Este sistema se encuentra corriendo sobre software privativo; y por las contradicciones existentes entre nuestro país y el gobierno de los Estados Unidos es de primordial importancia su migración a tecnologías libres. Esta aplicación permite el control y comunicación de dispositivos de hardware en aplicaciones web. Está compuesto por tres subsistemas fundamentales: Servicio DGM, el *framework* JavaScript (DGMJS) y el control centralizado de dispositivos (9).

Funciones del sistema:

Servicio de DGM: Es el encargado de garantizar el manejo de los servicios asociados a los dispositivos conectados a las estaciones de trabajo. El mismo contiene los controladores de la interacción directa con los dispositivos de hardware, el mismo gestiona las peticiones que provienen del navegador web. Este cuenta de tres partes fundamentales, el servidor proxy, el motor de manejo para los servicios asociados a los dispositivos y el grupo de clientes y controladores encargados de interactuar directamente con un dispositivo de hardware. La primera parte gestiona las peticiones que provienen desde el navegador web que han sido solicitadas por el *framework* DGMJS y reenvía estas peticiones al motor de manejo de los servicios para cuando se reciba una respuesta por parte del servicio que maneja el dispositivo, enviársela al navegador web. La segunda parte gestiona las solicitudes llegadas desde el servicio proxy, estas

solicitudes traen consigo información la cual conlleva al uso de un cliente determinado para interactuar con los dispositivos. La tercera y última está conformada por el grupo de clientes y controladores, los clientes constituyen la interfaz para el uso de los dispositivos, quienes utilizan los controladores que son los que interactúan directamente con el dispositivo.

Para el funcionamiento de esta versión del servicio local y llevar a cabo su instalación es necesario tener en cuenta los siguientes requisitos:

- ✚ Microsoft Windows XP SP 3 o superior.
- ✚ Microsoft .Net Framework 4.0.
- ✚ Controladores de los dispositivos a utilizar.

Framework DGMJS: Permite la interacción con servicios publicados en la estación de trabajo donde está en ejecución el servicio DGM. De esta forma se comunican la aplicación web externa y el servicio, posibilitando así interactuar con los dispositivos.

Control Centralizado: A través una interfaz visual en forma de aplicación web permite al usuario gestionar los dispositivos de la red así como ver en tiempo real el estado de estos. Se caracteriza por su fácil uso permitiéndose usar en prácticamente cualquier escenario.

Sistema de Gestión Penitenciaria

Este sistema es adaptado a la medida del cliente y permite controlar el tránsito de los privados de libertad por cada uno de los tipos de establecimientos que conforman el sistema penitenciario. El uso de esta aplicación web contribuye con la seguridad jurídica y ciudadana, al crear un mecanismo de control para el cumplimiento de la pena a los privados de libertad. La información que se maneja a través del sistema, permite ejecutar acciones para prevenir hechos ilícitos, de violencia y manifestaciones de corrupción. Mantiene el control de la situación operativa de los penales y constituye una base para la toma de decisiones tácticas y estratégicas, a partir del análisis de las alertas y avisos que emite, así como del comportamiento histórico de los indicadores de gestión que maneja. Contribuye a la reinserción de la población penal al registrar su clasificación en régimen y su caracterización social, psicológica y criminológica. De la misma forma ayuda a mantener el control epidemiológico y evaluar el clima penitenciario a partir de los indicadores de gestión (10).

Además el sistema permite la interacción con dispositivos biométricos como son las cámaras digitales, webcams y escáner de huellas, que facilitan la identificación de privados de libertad y visitantes. La aplicación puede funcionar de manera autónoma en cada establecimiento penitenciario, pero incorpora un mecanismo de réplica que mantiene actualizada una base de datos centralizada, que sirve de soporte al funcionamiento de una sala situacional.

Para la implementación de este sistema se utilizó J2EE. El software se ejecuta en un servidor local en cada establecimiento penitenciario y utiliza el Apache Tomcat para la aplicación y el Oracle 10g para la base de datos, pudiéndose migrar a Postgret en caso de que el cliente así lo prefiera. Para acceder a ella se debe usar FireFox como navegador web.

Biomesys Control de Asistencia

Es un sistema que aprovecha la tecnología de la identificación biométrica, encargándose de gestionar un dispositivo biométrico de huellas dactilares para registrar los eventos de asistencia en una organización por medio de la identificación de los empleados. A partir de la captura de identificaciones biométricas únicas el sistema se convierte en un generador de datos altamente confiable (11).

Características:

- ✚ No establece limitaciones de implementación asociadas al tipo de organización pudiendo ser instalado en la mayoría de los negocios públicos o privados, no importa cuál sea su tamaño.
- ✚ Se puede integrar de forma rápida con diferentes medios de autenticación como escáneres biométricos, credenciales de bandas magnéticas, tarjetas de código de barras entre otros.
- ✚ Posee interfaz de trabajo en ambiente web, para la administración y obtención de reportes estadísticos, documentos que facilitan a los decisores el control de la asistencia del personal. Es operable de forma remota.

Ventajas:

- ✚ Permite disminuir costo por utilizar la red local de la organización.
- ✚ El diseño de la estructura del sistema no está asociado a ninguna tecnología, marca o fabricante de dispositivos de autenticación.
- ✚ Extracción automática de la información de los dispositivos desconectados.

1.3.3 Valoración de los sistemas analizados.

La principal desventaja por estos sistemas en el ámbito nacional es que son soluciones propietarias, lo que implica que se realicen pagos por conceptos de licencias, capacitación y soporte, aspectos que no son factibles por las condiciones económicas de Cuba, estas están implementadas con tecnologías privativas, lo cual atenta contra el proceso de migración hacia software libre en el que está enmarcado el país. En el ámbito internacional, a pesar de existir varios sistemas que gestionan los dispositivos biométricos se pudo apreciar que a diferencia del DGM FREE, los sistemas Fingera y Biometric Services Platform (BioSP™) son ideados solamente para la captura de datos biométricos.

Finalmente se puede concluir que estos sistemas no cumplen con las necesidades del problema a resolver por lo que ninguna de las herramientas estudiadas puede ser adaptada al mismo, aunque sí sirvieron de apoyo en la concepción de algunas interfaces y funcionalidades que por la complejidad del negocio no quedaban del todo claras en los documentos que se tuvieron en cuenta para el desarrollo del componente.

Como se ha visto los sistemas presentados anteriormente poseen cada uno sus ventajas y desventajas sirviendo su análisis como base para un mejor entendimiento de las características que el nuevo sistema deberá presentar. Ver tabla 1.1.

Sistemas	Control en tiempo real de dispositivos	Administración centralizada de dispositivos	Posibilidades de uso en aplicaciones web	Extensible y modificable	Desarrollado con tecnologías libres
Fingera	<input type="checkbox"/>		<input type="checkbox"/>	<input type="checkbox"/>	
Biometric Services Platform	<input type="checkbox"/>	<input type="checkbox"/>			
Device Grid Manager (2.0)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
Biomesys	<input type="checkbox"/>				

Tabla 1. 1 Sistemas analizados y sus características. Fuente: Elaboración propia.

El estudio previamente realizado demuestra la necesidad de desarrollar una aplicación similar al sistema DGM 2.0, ya que es el que más se adapta a las necesidades reflejadas en el problema a resolver. Dicha versión debe incluir las mismas funcionalidades de la anterior, en aras de hacer del mismo un sistema integral para el manejo y control de dispositivos orientado a la arquitectura web, que provea herramientas para la administración de los dispositivos y su interacción desde aplicaciones web. Utilizando las experiencias de las aplicaciones estudiadas y adicionando nuevas funcionalidades se pretende realizar un nuevo sistema aplicable al SUIN, que permita la interacción de dispositivos en ambientes multiplataforma, la continua actualización del sistema, la configuración y control centralizado de los dispositivos, la obtención de reportes por parte del usuario, logrando reducir, con el desarrollo del mismo, el costo al país.

1.4 Herramientas y tecnologías para el desarrollo

A partir de la comprensión de los conceptos antes propuestos, y haberse analizado el objeto de estudio y el campo de acción, es de vital importancia presentar las herramientas y tecnología con la que se realizara el desarrollo de la solución. Cuando se habla de tecnología se hace alusión al conjunto de nociones técnicas aplicadas al diseño y construcción de productos y servicios de acuerdo a las necesidades humanas (12).

Metodología de Desarrollo

La metodología de Desarrollo de software en la ingeniería de software es un marco de trabajo usado para estructurar, planificar y controlar el proceso de desarrollo en sistemas de información. Es un conjunto de procedimientos, técnicas, herramientas, y un soporte documental que ayuda a los desarrolladores a diseñar un nuevo software (13).

El uso de estas metodologías, sirve de ayuda a los desarrolladores de software durante todo el ciclo de vida del mismo, sin embargo para lograr una acertada selección se deben tener en cuenta las características del software a obtener, el entorno para el que se desarrolla y el equipo de trabajo.

Metodología *Extreme Programming (XP)*

La metodología ágil XP (*eXtreme Programming* y en español Programación Extrema) es un conjunto de valores, principios y prácticas para el desarrollo vertiginoso de un software de alta calidad que proporciona el valor más alto para el cliente en el menor tiempo posible. En esta se utiliza un enfoque orientado a objetos y con su empleo se logra la confección de productos fáciles de usar, rápidos, fiables y robustos contra los fallos (14).

XP se centra en la satisfacción del cliente, que se considera como miembro del equipo. Eleva al máximo el trabajo en equipo y organiza el tiempo de los miembros del mismo para que el objetivo sea obtener una versión funcional del producto en el menor tiempo posible.

Esta metodología ofrece muchas ventajas para el proceso de desarrollo de software. En un proyecto corto se hace más eficaz el trabajo, se ahorra tiempo en documentación debido a que solo se genera la documentación imprescindible y los miembros del equipo se enfocan en poner todo sus esfuerzos en la realización del producto, que resultan más fiables y robustos contra fallos debido al diseño de las pruebas de forma previa a la codificación. Cualquier miembro del equipo de desarrollo puede desempeñar el rol de desarrollador, quedando la posibilidad de que el código siempre se mejore y se simplifique usando sistemas para la gestión de las versiones para evitar la duplicación de trabajo.

XP se ajusta a las características del sistema a implementar, ya que el equipo de desarrollo de esta conformado por dos personas y la metodología en sus principios básicos plantea la programación en equipos pequeños con pocos roles, pudiendo los miembros del equipo intercambiar responsabilidades en

un momento determinado. Otro argumento es que existe comunicación directa con el cliente por lo que va a estar en condiciones de contestar rápida y correctamente a cualquier pregunta por parte del resto del equipo de desarrollo, de forma que no se atrase la toma de decisiones. Es abierta a los cambios y genera poca documentación lo que hace la entrega del *software* menos complicada y más satisfactoria tanto para los clientes como para el equipo de entrega.

Lenguaje de modelado

El Lenguaje Unificado de Modelado (UML) es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software. Captura decisiones y conocimiento sobre los sistemas que se deben construir. Se usa para entender, diseñar, configurar, mantener y controlar la información sobre tales sistemas. Está pensado para usarse con todos los métodos de desarrollo, etapas del ciclo de vida, dominios de aplicación y medios (15).

Mediante UML se describirá el sistema que se propone en términos del mapa conceptual y diagrama de despliegue.

Herramienta para el modelado

Las herramientas para el modelado, también llamadas CASE (*Computer Aided Software Engineering*) son un conjunto de métodos, utilidades y técnicas que dan asistencia a los analistas, ingenieros de software y desarrolladores, destinadas a facilitar el desarrollo de software incrementando su productividad y disminuyendo el costo de tiempo y de dinero.

Visual Paradigm para UML 8.0

Visual Paradigm para UML es una herramienta CASE multiplataforma de modelado UML. Permite dibujar todos los tipos de diagramas de clases, genera código desde diagramas, realiza ingeniería inversa e informes en varios formatos. Esta herramienta presenta licencia gratuita y comercial (16).

Entre las ventajas por las que sobresale se encuentran:

- ✚ Navegación entre código y el modelo.

- ✚ Generador de documentación y reportes UML PDF/HTML/MS Word.
- ✚ Demanda en tiempo real y sincronización de código fuente.
- ✚ Entorno de modelado visual.
- ✚ Soporte de notaciones UML.
- ✚ Diagramas de diseño automático.
- ✚ Análisis de texto y soporte de tarjeta CRC.
- ✚ Proporciona soporte a varios lenguajes en generación de código e ingeniería inversa a través de plataformas java.

Visual Paradigm para UML 8.0 es la herramienta de modelado a utilizar, que soporta los diagramas a modelar en el ciclo de vida completo del software

Monodevelop

MonoDevelop permite a los desarrolladores escribir rápidamente las aplicaciones de escritorio y web sobre Linux, Windows y Mac OS X. También permite que sea fácil para los desarrolladores de aplicaciones .NET creados con Visual Studio trasladar a Linux y Mac OS X manteniendo una única base de código para todas las plataformas (17).

Monodevelop v2.6.8.4

- ✚ Multiplataforma: Soporta Linux, Windows y Mac OS X.
- ✚ Texto Edición avanzada: Completamiento de código para C #, plantillas de código, plegado de código.
- ✚ Banco de trabajo configurable: Diseños de ventana totalmente personalizables, atajos de teclado definidos y herramientas externas.
- ✚ Soporte para múltiples lenguajes: C #, C #, Visual Basic .NET, C / C ++, Vala.
- ✚ Depurador: Para depurar Mono y aplicaciones nativas.

- ✚ GTK # Visual Diseñador: Construir aplicaciones GTK #.
- ✚ ASP.NET: Crear proyectos web con soporte finalización de código completo y prueba en XSP, el servidor web Mono.
- ✚ Otras Herramientas: Control Fuente, la integración *makefile*, pruebas unitarias, envasado, distribución y localización.

NetBeans

NetBeans es un entorno de desarrollo integrado libre, hecho principalmente para el lenguaje de programación Java. Existe además un número importante de módulos para extenderlo. NetBeans IDE es un producto libre y gratuito sin restricciones de uso. Es un proyecto de código abierto con una gran base de usuarios y una comunidad en constante crecimiento (18).

NetBeans 8.0. Soportes para Java

Incorpora el soporte para la versión 8 del lenguaje Java. Proporciona analizadores de código, mejora el editor de perfiles, lambdas, streams, referencias a métodos, soporte para desplegar, ejecutar, depurar o perfilar aplicaciones en sistemas embebidos, tales como Raspberry PI, directamente desde el IDE.

Lenguajes de programación

C#: Es un lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft como parte de su plataforma .NET, que después fue aprobado como un estándar. C# es uno de los lenguajes de programación diseñados para la infraestructura de lenguaje común.

Aunque C# forma parte de la plataforma .NET, ésta es una API, mientras que C# es un lenguaje de programación independiente diseñado para generar programas sobre dicha plataforma. Existe un compilador implementado que provee el marco Mono - DotGNU, el cual genera programas para distintas plataformas como Windows, Unix, Android, iOS, Windows Phone, Mac OS y GNU/Linux (19).

Java: El lenguaje Java es un derivado del lenguaje C, tiene su propia estructura, reglas de sintaxis y paradigma de programación. El paradigma de programación del lenguaje Java se basa en el concepto de programación orientada a objetos (OOP), que las funciones del lenguaje soportan.

Java implementa la tecnología básica de C++ con algunas mejoras y elimina algunas cosas para mantener el objetivo de la simplicidad del lenguaje. Java trabaja con sus datos como objetos y con interfaces a esos objetos. Soporta las tres características propias del paradigma de la orientación a objetos: encapsulación, herencia y polimorfismo. Las plantillas de objetos son llamadas, como en C++, clases y sus copias, instancias. Estas instancias, como en C++, necesitan ser construidas y destruidas en espacios de memoria.

Java incorpora funcionalidades inexistentes en C++ como por ejemplo, la resolución dinámica de métodos. Esta característica deriva del lenguaje Objective C, propietario del sistema operativo Next. En C++ se suele trabajar con librerías dinámicas (DLLs) que obligan a recompilar la aplicación cuando se retocan las funciones que se encuentran en su interior. Este inconveniente es resuelto por Java mediante una interfaz específica llamada RTTI (RunTime Type Identification) que define la interacción entre objetos excluyendo variables de instancias o implementación de métodos. Las clases en Java tienen una representación en el runtime que permite a los programadores interrogar por el tipo de clase y enlazar dinámicamente la clase con el resultado de la búsqueda (20).

1.5. Conclusiones parciales

En este capítulo se mencionaron los conceptos necesarios para un mejor entendimiento de la investigación científica en cuestión. Se realizó un estudio de sistemas similares, tanto nacionales como internacionales, enfocadas a la gestión de dispositivos de identificación biométrica, concluyendo que:

- ✚ Estos sistemas son similares con respecto a algunas de las necesidades de la aplicación a desarrollar. Proporcionando un mejor entendimiento de las funcionalidades básicas del sistema.
- ✚ La mayoría de los sistemas se basan solo en los dispositivos de captación biométrica, no tienen en cuenta más ningún dispositivo de hardware.
- ✚ Están implementados con programas que son propietarios atentando contra el proceso de migración hacia software libre en el que está enmarcado el país.

Después de haber realizado el estudio de los sistemas existente se determinó la necesidad de crear una solución propia que responda a las características de las entidades cubanas y cumpla con el paradigma de independencia tecnológica por la que aboga el país. Se realizó una descripción de las tecnologías, modelo de desarrollo y herramientas con las cuales se va a desarrollar la solución para lograr un mejor entendimiento de estas, demostrando que su utilización favorece a la independencia tecnológica por el cual apuesta el país.

Capítulo 2 Características del Sistema

2.1 Introducción

Luego de hacer un estudio del arte y haber identificado una solución a la necesidad evidenciada en la problemática, se propone el siguiente capítulo en el que se describirán las características principales que posee el sistema propuesto. Se define el modelo conceptual actual, describiendo además cada una de las entidades que intervienen en este modelo en aras de lograr un mayor entendimiento del mismo, se confeccionan las Historias de usuario y las tareas de ingeniería asociadas a cada una de ellas. Se explica con detalles la propuesta del sistema, definiendo cada uno de los subsistemas que lo componen así como un diagrama que provee una vista general de este.

2.2 Presentación de la solución

La solución de software estará conformada por una aplicación que permitirá manejar todos los recursos (estaciones de trabajos y dispositivos) de manera centralizada. Su principal objetivo será proporcionar un mecanismo para poder usar los dispositivos empleados para los procesos de captación biométricos y otros dispositivos de hardware, así como que permita la interacción con la aplicación web. La figura 2.1 muestra una abstracción del sistema. Dicha solución implementará servicios de impresión, huellas, cámaras, scanner, y información de las estaciones de trabajo.



Figura 2. 1 Abstracción del sistema. Fuente: Elaboración propia.

2.3 Modelo Conceptual

El modelo conceptual es una herramienta de comunicación fundamental que obliga y permite a los desarrolladores pensar formalmente en el problema y a validar su comprensión del mismo. Mediante este modelo se establece además un vocabulario propio del problema; y junto a los requerimientos constituye la entrada más importante para el diseño (21).

El modelo conceptual, como el modelo de dominio, es una representación visual estática que se realiza con el objetivo de lograr mejor comprensión del contexto para la obtención de requisitos del sistema. Básicamente, de manera visual es un diagrama de clase UML que modela los conceptos básicos asociados al dominio del problema, sus propiedades más importantes y las relaciones que resultan imprescindibles para contextualizar dichos conceptos, a fin de poder brindar elementos que permitan la identificación de los requisitos del sistema. Además es válido para cualquier metodología de desarrollo de software.

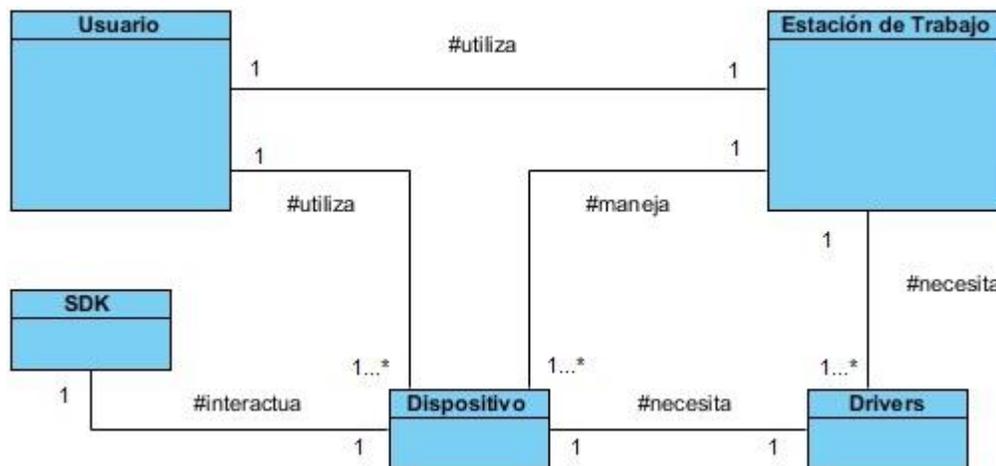


Figura 2. 2 Modelo conceptual. Fuente: Elaboración propia

2.3.1 Glosario de términos del Modelo Conceptual

En este acápite, para un mejor entendimiento se explicarán cada una de las clases que conforma el Modelo de dominio. (Ver Fig. 2.2)

Usuario: Persona encargada de interactuar con el servicio y acceder a sus funcionalidades. Posee un grupo de permisos que lo asocian al uso de un grupo de dispositivos.

Estación de trabajo: Computadora destinada a ser usada por los usuarios para interactuar con un grupo de dispositivos conectados a ella.

Dispositivos: Accesorios de hardware que se conectan a la estación de trabajo para dotarla de nuevas funcionalidades, que pueden ser utilizadas por aplicaciones, como pueden ser, capturar huellas, capturar fotos e imprimir documentos. Estos dispositivos varían en dependencia de su marca, funcionalidades y distribuidor, necesitando drivers para poder ser reconocidos en la estación de trabajo a la que se conectan.

Driver: Paquete de software que proveen los fabricantes, incluido junto con los dispositivos y que se requiere para establecer comunicación con los dispositivos en función de su correcto funcionamiento por los sistemas operativos en las estaciones de trabajos.

SDK: Conjunto de herramientas para la comunicación con determinado dispositivo, que provee el fabricante para la integración de su producto con las aplicaciones de software en general.

2.4 Descripción del sistema propuesto

El sistema propuesto es un servicio local para el manejo de dispositivos, el mismo está compuesto por tres partes fundamentales: el servicio proxy, el motor de manejo para los servicios asociados a los dispositivos y el grupo de aplicaciones clientes y controladores encargados de interactuar directamente con un dispositivo de hardware. La primera parte gestiona las peticiones que provienen desde el navegador web que han sido solicitadas por el *framework* JavaScript DGMJS y a su vez se encarga de reenviar esas peticiones al motor de manejo de los servicios. Una vez recibida una respuesta por parte del servicio que maneja el dispositivo, es enviada al navegador web. La segunda parte es la encargada de gestionar las solicitudes que llegan desde el servicio proxy, trayendo consigo información importante, la cual con lleva al uso de un cliente determinado para interactuar con los dispositivos. La última parte está conformada por el grupo de clientes y controladores. Los clientes constituyen la interfaz para el uso de los dispositivos, quienes utilizan los controladores que son los que interactúan directamente con el dispositivo.

2.5 Procesos para la extensión del Servicio Local para el manejo de dispositivos

El sistema propuesto se caracteriza por el fácil uso y extensión por parte de los desarrolladores, permitiendo usar prácticamente cualquier escenario, además de integrar otros dispositivos sin tener que emplear grandes esfuerzos por parte de los desarrolladores. En el proceso descrito a continuación se mostrará la interacción del desarrollador con el sistema, aumentando las funcionalidades del sistema para el uso de nuevos dispositivos.

De existir la necesidad de insertar un nuevo dispositivo a la red, se hace necesario llevar a cabo la extensión del sistema con el objetivo incluir el nuevo dispositivo para su manejo. Es de vital importancia la instalación de los *drivers* del dispositivo por parte del desarrollador, estos a su vez deben ser incluidos en un paquete de instalación con el dispositivo se pueden encontrar en el sitio oficial del fabricante. Seguidamente se crearía una nueva funcionalidad para interactuar con dicho dispositivo, la cual es agregada al Servicio local para llevar a cabo el manejo del nuevo dispositivo a insertar.

2.6 Requisitos del sistema

El objetivo esencial del flujo de trabajo de los requisitos dentro de un proceso de desarrollo de software es guiar de manera correcta el desarrollo del sistema. Para que esto sea posible es necesario realizar una descripción de los requisitos del sistema.

2.6.1 Definición de los requisitos funcionales del sistema

RF 1. Manejar los servicios web.

RF 2. Aceptar nuevos controladores de dispositivos.

RF 3. Manejar dispositivos remotos.

RF 3.1 Establecer comunicación con un servicio remoto.

RF 4. Aceptar el manejo de dispositivos remotamente.

RF 5. Consultar el estado de un servicio remotamente.

RF 6. Validar el acceso a los servicios web.

RF 7. Gestionar los servicios web a usar en el Servicio local para el manejo de dispositivos.

RF 8. Gestionar los dispositivos a usar en el Servicio local.

RF 9. Actualizar el Servicio local para el manejo de dispositivos.

RF 9.1 Actualizar los servicios web que se exponen.

RF 9.2 Actualizar los controladores de los dispositivos.

Las historias de usuario es una técnica utilizada por la metodología XP para especificar los requisitos del software. Estas describen brevemente las características que el sistema debe poseer. Cada historia de usuario debe ser lo suficientemente comprensible y delimitada para que los programadores puedan implementarla.

Las historias de usuario solamente proporcionaran los detalles sobre la estimación del riesgo y cuánto tiempo con llevará la implementación de dicha historia de usuario. Su nivel de detalle debe ser el mínimo posible, de manera que permita hacerse una ligera idea de cuánto costará implementar el sistema.

Durante el análisis en la fase de exploración fueron identificadas once HU, cada una de ellas corresponden a las diferentes funcionalidades, además proporcionan una idea al equipo de desarrollo de cómo debe ser su posterior implementación. A continuación en la tabla 2.1 se describe una de las historias de usuario más significativa. Para consulta las restantes HU, ver Anexos 1.

UH_3. Manejar dispositivos remotos.

Historia de Usuario	
Numero: HU_3	Usuario: Sistema
Nombre de la historia: Manejar dispositivos remotos.	
Prioridad en negocio: Muy Alta	Riesgo en desarrollo: Medio

Puntos estimados: 2	Iteración asignada: 1
Responsable: Gianni Espino Hernández	
Descripción:	
<ol style="list-style-type: none"> 1. Recibir una petición al servicio. 2. Ejecutar el método correspondiente en el controlador 3. Establece comunicación con el controlador remoto. 4. Si la conexión es satisfactoria. <ol style="list-style-type: none"> 4.1. Ejecutar el método del controlador remoto. 4.2. Generar traza de uso del dispositivo. 4.3. Retornar la información al servicio. 5. Si la conexión no es satisfactoria. <ol style="list-style-type: none"> 5.1. Mostrar mensaje de error 	
Observaciones:	
<ol style="list-style-type: none"> 1. El servicio debe estar iniciado. 2. El controlador debe estar iniciado como un servicio en la estación de trabajo remota. 3. Se debe contar con permisos para conectarse al controlador remoto. 	

Tabla 2. 1 HU_3 Manejar los dispositivos remotos

Fase de Planificación

En la fase de planificación el cliente establece la prioridad de cada historia de usuario, y correspondientemente, los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente. Una entrega debería obtenerse en no más de tres meses. Esta fase dura unos pocos días.

La planificación se puede realizar basándose en el tiempo o el alcance. La velocidad del proyecto es utilizada para establecer cuántas historias se pueden implementar antes de una fecha determinada o cuánto tiempo tomará implementar un conjunto de historias.

Al planificar según alcance del sistema, se divide la suma de puntos de las historias de usuario seleccionadas entre la velocidad del proyecto, obteniendo el número de iteraciones necesarias para su implementación.

Estimación del esfuerzo por historia de usuario

Las estimaciones de esfuerzo asociado a la implementación de las historias la establecen los programadores utilizando como medida el punto. Un punto, equivale a una semana ideal de programación. Las HU generalmente valen de 1 a 3 puntos.

Historias de Usuario	Puntos Estimados
Manejar los servicios web.	2
Aceptar nuevos controladores de dispositivos.	1
Manejar dispositivos remotos.	2
Aceptar el manejo de dispositivos remotamente.	1
Consultar el estado de un servicio remotamente.	1
Validar el acceso a los servicios web.	1
Gestionar los servicios web a usar en el Servicio local para el manejo de dispositivos.	2
Gestionar los dispositivos a usar en el Servicio local	2
Actualizar el Servicio local para el manejo de dispositivos.	1
Total	13

Tabla 2. 2 Estimación del esfuerzo por historia de usuario

Plan de Iteración

Esta fase incluye varias iteraciones sobre el sistema antes de ser entregado. El Plan de Entrega está compuesto por iteraciones de no más de cinco semanas. En la primera iteración se puede intentar establecer una arquitectura del sistema que pueda ser utilizada durante el resto del proyecto. Las HU son seleccionadas de acuerdo al orden preestablecido para cada entrega, son desarrolladas y probadas en un ciclo de iteración, esto no siempre es posible ya que es el cliente quien decide qué historias se implementarán en cada iteración para maximizar el valor de negocio. Al final de la última iteración el sistema estará listo para entrar en producción.

Iteración 1

El objetivo principal de esta iteración es la implementación de las historias de usuarios seleccionadas de mayor prioridad, de esta forma se obtienen las primeras funcionalidades tales como la manejar los servicios web, manejar los dispositivos remotos, aceptar el manejo de dispositivos remotamente y gestionar los servicios web a usar en el Servicio local para el manejo de dispositivos.

Iteración 2

En esta iteración se implementan las historias de usuario de prioridad alta correspondientes con aceptar nuevos controladores de dispositivos, gestionar los dispositivos a usar en el servidor web así como validar el acceso a los servicios web, Además se corrigen los errores de las HU de la iteración anterior. De esta forma se tiene la segunda versión del producto.

Iteración 3

En esta iteración se implementan las HU de prioridad media, de esta forma se obtiene la versión final del producto. A esta versión se le realizarán pruebas para la evaluación de su comportamiento y rendimiento.

Iteración	Orden de las HU	Duración
#1	Manejar los servicios web. Manejar los dispositivos remotos. Aceptar el manejo de dispositivos remotamente. Gestionar los servicios web a usar en el Servicio local para el manejo de dispositivos.	5
#2	Aceptar nuevos controladores de dispositivos. Gestionar los dispositivos a usar en el Servicio local. Validar el acceso a los servicios web.	5
#3	Actualizar el Servicio local para el manejo de dispositivos. Consultar el estado de un servicio remotamente.	3

Tabla 2. 3 Plan de iteraciones

Plan de entregas

En el plan de entrega que se plantea a continuación se hace una propuesta de la fecha aproximada en que se harán versiones al sistema, al finalizar cada iteración en la fase de implementación.

Producto a Entregar	Final 1ra Iteración 2da semana de abril	Final 2ra Iteración 2da semana de mayo	Final 3ra Iteración última semana de mayo
DGM Libre	Primera versión	Segunda Versión	Versión Final

Tabla 2. 4 Plan de entrega

2.6.2 Requisitos no funcionales

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener y resultan fundamentales en el logro de una solución acorde a las necesidades del cliente. Están vinculados a los requisitos funcionales, y determinan cómo ha de comportarse, qué cualidades debe tener o cuán rápido o grande debe ser. Existen diferentes categorías para clasificar los requisitos no funcionales, requisitos de software, hardware, restricciones del diseño, implementación, apariencia, usabilidad, seguridad y soporte son algunas de ellas. A continuación se presenta el conjunto de requisitos no funcionales del sistema.

Requisitos de apariencia

RnF 1. El sistema debe contar con una interfaz accesible e intuitiva para controlar y mostrar el estado de los servicios al usuario así como para realizar las configuraciones necesarias.

RnF 2. Los elementos del sistema contendrán claro y bien estructurados los datos, y al mismo tiempo permitirán la interpretación correcta e inequívoca de la información.

Requisitos de soporte

RnF 3. El sistema debe poseer documentos técnicos.

RnF 4. El sistema debe poseer el Manual de Usuario.

RnF 5. El sistema debe poseer el Manual de Instalación.

Requisitos de software

RnF 6. Sistema Operativo: Linux o Windows.

RnF 7. Drivers y SDK de los dispositivos a usar.

Requisitos de seguridad

RnF 8. Garantizar tratamiento de excepciones.

RnF 9. Garantizar el control de acceso a las aplicaciones.

RnF 10. Autenticación segura a partir del hardware.

Requisitos de hardware

RnF 11. Las estaciones de trabajo cliente deben tener como mínimo 128 Mb de RAM y procesador Pentium IV o superior.

Diseño

El análisis y el diseño del software son etapas importantes en su ciclo de desarrollo, toda aplicación debe estar soportada por un adecuado análisis y una correcta selección de su arquitectura.

Arquitectura

Los patrones arquitectónicos se utilizan para expresar una estructura de organización base o esquema para un software. Proporcionando un conjunto de sub-sistemas predefinidos, especificando sus responsabilidades, reglas, directrices que determinan la organización, comunicación, interacción y relaciones entre ellos.

Los patrones arquitectónicos heredan mucha de la terminología y conceptos de patrones de diseño, pero se centran en proporcionar modelos y métodos re-utilizables específicamente para la arquitectura general de los sistemas de información. Quiere decir que a diferencia de los patrones de diseño estas son plantillas incompletas y no se pueden aplicar directamente al código con modificaciones meramente contextuales (22).

Arquitectura n-capas: Está basada en una distribución jerárquica de roles y responsabilidades, propiciando una división efectiva de los problemas a resolver. Los roles indican el tipo y la forma de interacción con otras capas, y las responsabilidades la funcionalidad que implementan. Ver Fig. 2.3.

Capa Presentación: Es la que ve el usuario, presenta el sistema al mismo, le comunica la información y captura los datos que el usuario introduce en la aplicación. Esta capa se comunica únicamente con la del negocio enviando peticiones a la misma. También es conocida como interfaz gráfica y tiene la característica de ser entendible y fácil de usar para el usuario.

Capa del Negocio: Es donde residen los programas que se ejecutan durante la ejecución de la aplicación. Esta capa se comunica con la de presentación, para recibir las peticiones y presentar los resultados, y con la de acceso a datos, para interactuar con la base de datos, consultando y modificando sus informaciones. En la capa del negocio interactúan clases como DGMConfiguration y librerías como AFroge.

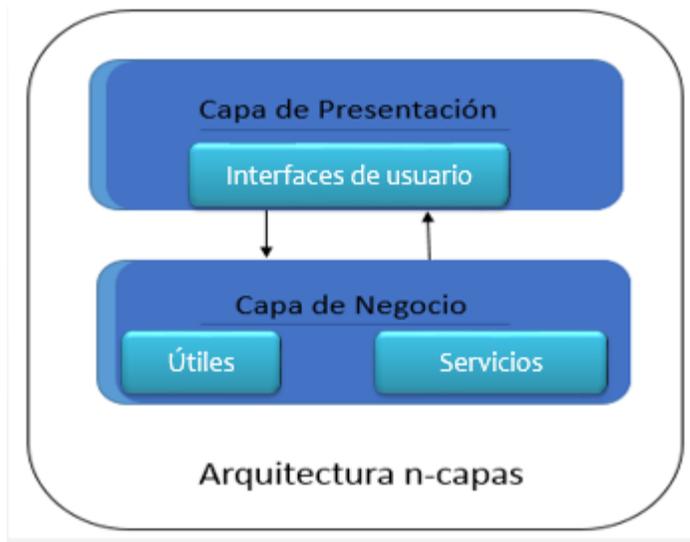


Figura 2. 3 Representación de Arquitectura n-capas. Fuente: Elaboración propia.

2.7 Patrones de Diseño

Al iniciar el desarrollo de un sistema de software se hace necesario identificar problemas comunes en el diseño del producto para aplicar soluciones existentes que se apliquen a estos. Por este motivo uno de los aspectos a tener en cuenta en la creación de una aplicación son los patrones de diseño de software que puedan ser utilizados.

Los patrones de diseño son **soluciones para problemas típicos y recurrentes** que se pueden encontrar a la hora de desarrollar una aplicación. Aunque la aplicación sea única, tendrá partes comunes con otras aplicaciones: acceso a datos, creación de objetos, operaciones entre sistemas etc. En lugar de reinventar, se puede solucionar problemas utilizando algún patrón, ya que son soluciones probadas y documentadas (23).

Subclase: propone heredar miembros por defecto de una superclase, las clases hijas tienen los atributos y métodos del padre pero tienen sus propios atributos y funcionalidades (24). Ejemplo de este patrón presenta la clase CameraClient (Ver figura 2.4).

```
public class CameraClient : BasicClient, ICameraClientService
{
    private readonly ICameraController _camaraController;

    public CameraClient()
    {
        _camaraController = (ICameraController) _controller;
    }
}
```

Figura 2. 4 Ejemplo de utilización del patrón subclase. Fuente: Elaboración Propia

Inversión de control: La inversión de control es un patrón de diseño usado para desacoplar las clases evitando de esa manera las dependencias fuertes entre ellas. Este patrón es implementado mediante la

inyección de dependencias en un componente cuando este es construido. Estas dependencias son comúnmente proveídas como interfaces (25).

Builder: abstrae el proceso de creación de un objeto complejo, centralizando dicho proceso en un único punto (26).

Prototype: crea nuevos objetos clonándolos de una instancia ya existente (26).

Singleton: Asegura que solo se cree una instancia de la clase y ofrece un punto global de acceso a esta instancia. El uso de este patrón permite que una clase pueda ser instanciada solo una vez y acceder luego a la instancia (26). Ejemplo de este patrón presenta la clase DGMNotification (Ver figura 2.5).

```
private static DGMNotification _instance;
private static ILogNotification _notifier;

private DGMNotification()
{
    _notifier = new LogsNotification();
}

public static DGMNotification Notifications()
{
    return _instance ?? (_instance = new DGMNotification());
}
```

Figura 2. 5 Ejemplo de utilización del patrón singleton. Fuente: Elaboración Propia

2.8 Conclusiones parciales

Como parte del presente capítulo se abordó todo lo referente a las primeras fases de la metodología de desarrollo de software XP, exploración y planificación, describiendo cada uno de los artefactos generados durante el transcurso de las mismas. Se hizo referencia a la etapa de diseño del sistema, describiéndose la

arquitectura a utilizar, así como los requisitos funcionales y no funcionales y los patrones de diseño que utiliza para su funcionamiento.

Capítulo 3 Implementación y validación de la solución propuesta

3.1 Introducción

Las actividades de implementación y pruebas abarcan gran parte de las acciones para el desarrollo de un producto de software. Primeramente la implementación en sí, con todo el proceso de formación de la solución, que incluye el establecimiento de los estándares de codificación, el tratamiento de los errores, la elaboración de los componentes y sus relaciones, el desarrollo de cada una de las interfaces según las necesidades del cliente, hasta el diseño del futuro despliegue del sistema. En segundo lugar las pruebas encargadas de obtener todos los errores cometidos durante el desarrollo del producto. El presente capítulo refleja cada una de las etapas de las mencionadas actividades.

3.2 Implementación

Una vez definidas las historias de usuario y concluido el diseño se pasa a la etapa de implementación de la solución propuesta cuyos objetivos van destinados a desarrollar de forma iterativa e incremental un producto completo, consiguiendo versiones del producto de forma rápida y práctica, que paulatinamente completen la planeación, diseño, desarrollo y prueba de toda la funcionalidad necesaria. Durante la codificación la programación debe ser por pares, con el propósito de obtener el código de cada una de las clases y servicios así como las interfaces en correspondencia con el Plan de *Release* realizado.

En los epígrafes que se muestran a continuación se expone el Plan de *Release*, el diagrama de despliegue y una muestra de una de las interfaces del sistema que respaldan la propuesta de solución.

3.2.1 Plan de Releases

Release	Descripción de la Iteración	Orden de la HU a implementar	Duración total
1ra Iteración	Historias de usuario de prioridad Alta y Muy Alta.	HU_1 Manejar los servicios web HU_2 Aceptar nuevos controladores de dispositivos HU_3 Manejar los dispositivos Remotos	10 semanas

		HU_6 Validar el acceso a los servicios web HU_4 Aceptar el manejo de dispositivos remotamente HU_9 Actualizar el servicio local para el manejo de dispositivos	
2da Iteración	Historias de usuario de prioridad Media y Baja.	HU_5 Consultar el estado del servicio remotamente HU_7 Gestionar los servicios web a usar en el servicio local HU_8 Gestionar los dispositivos a usar en el servicio local	5 semanas

Tabla 3. 1 Plan Release.

3.1.2 Diagrama de despliegue

El diagrama de despliegue define la arquitectura física del sistema por medio de nodos interconectados. Estos nodos son elementos de hardware sobre los cuales pueden ejecutarse los elementos de software. (27) Para la propuesta de solución se estableció el diagrama de despliegue representado en la Fig. 3.1, el cual está compuesto por una estación de trabajo que será el nodo cliente que tendrá por vía USB los dispositivos de hardware conectados a ellas, el nodo cliente tendrán comunicación con el Servidor de Aplicaciones a través del protocolo HTTPS que contendrá el sistema para el control centralizado de dispositivos.

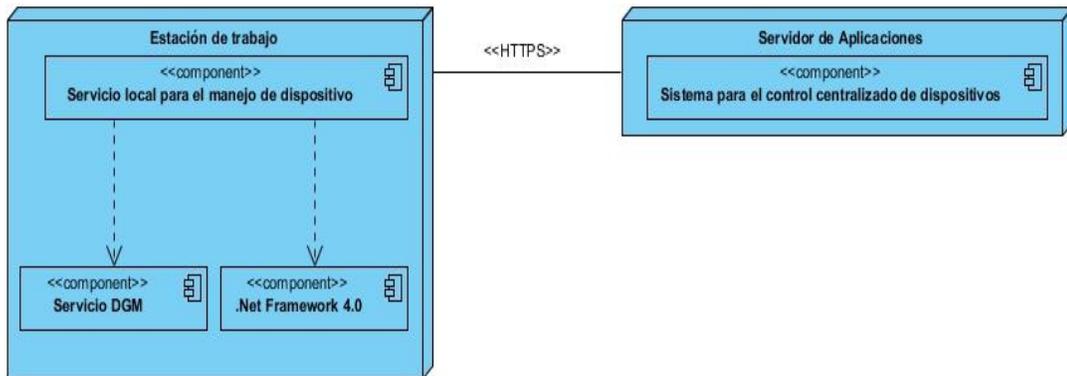


Figura 3. 1 Diagrama de despliegue. Fuente: Elaboración propia.

3.1.3 Interfaces de usuario

A continuación se muestra la Fig.3.2 con las interfaces obtenidas en la nueva versión funcional del servicio DGM Libre.



Figura 3.2 Interfaz del panel de configuración.

3.2 Estándares de codificación.

Los estándares de codificación son generalmente utilizados para garantizar la uniformidad de los elementos que se diseñan e implementan cuando se desarrolla un software. Su uso facilita una mejor comprensión y el entendimiento del equipo de desarrollo y mejora la interpretación del código de programación. A continuación se reflejan los estándares definidos para la creación del DGM Libre.

Estilos

Pascal: La primera letra en el identificador y la primera letra de cada subsiguiente palabra concatenada se capitalizan. Puede utilizar los identificadores de Pascal case en caso de tres o más caracteres. Por ejemplo: CustomerName.

Camello: La primera letra en el identificador está en minúscula y la primera letra de cada subsiguiente palabra concatenada es mayúscula. Por ejemplo: costumerName.

Mayúscula: Todas las letras en el identificador se capitalizan. Esta convención se utilizará sólo para los identificadores que constan de dos o menos letras. Por ejemplo: System.IO.

Tratamiento de errores

Para garantizar una mayor integridad y confiabilidad de la información que se procesa en el sistema, se realizan diferentes acciones, las cuales se mencionan a continuación:

- ✚ Implementación de clases para el lanzamiento y la captura de excepciones.
- ✚ Los mensajes de error mostrados en el sistema deben estar escritos en un lenguaje comprensible para los usuarios.
- ✚ Luego de introducir información errónea o dejar campos vacíos en los formularios se mostrará un mensaje y se resaltará el campo indicando el error.

3.3 Tarjetas CRC

Las tarjetas CRC (Clase - Responsabilidad - Colaboración), constituyen uno de los artefactos de la metodología XP que guía el proceso de desarrollo de la solución propuesta. Se dividen en tres secciones que contienen la información del nombre de la clase, sus responsabilidades y sus colaboradores.

Una clase es la definición de las características concretas de un determinado tipo de objetos. Las responsabilidades de una clase son las funcionalidades que realiza y sus atributos. Los colaboradores son las demás clases con las que interactúa con el fin de cumplir sus responsabilidades.

A continuación se define como ejemplo una de las tarjetas CRC de la solución propuesta, las demás se encuentran descritas en el Anexo 2:

PcInformation	
Responsabilidades	Colaboradores
Devuelve todas las características físicas de la estación de trabajo en cuanto a bios, board, disco duro, red, memoria ram, y procesador.	<ul style="list-style-type: none"> -BiosManage -BoardManager -HDDManage -NetworkManage -PhysicalMemoryManage -ProcessorManage

Tabla 3. 2 Tarjeta CRC de la clase PcInformation.

3.4 Pruebas de software

El único instrumento adecuado para determinar el status de la calidad de un producto software es el proceso de pruebas. En este proceso se ejecutan pruebas dirigidas a componentes del software o al sistema de software en su totalidad, con el objetivo de medir el grado en que el software cumple con los requerimientos (28).

3.4.1 Prueba de Caja blanca

Las pruebas de caja blanca, también conocidas como pruebas de caja transparente o pruebas estructurales, se centran en los detalles procedimentales del software, por lo que su diseño está ligado al código fuente. Este método intenta garantizar que se ejecutan al menos una vez todos los caminos independientes que presente el módulo y que todas las estructuras de datos internas serán usadas (29).

Para el sistema desarrollado es necesario evidenciar si es un producto de alta calidad, por tanto es preciso valorar que tan certera ha sido la implementación. Por tanto se debe aplicar una de las técnicas que comprende las pruebas de caja blanca, en este caso la del camino básico o ruta básica. Para ello es necesario conocer el número de caminos independientes, por lo que se construirá un grafo asociado y se calculará de la complejidad ciclomática.

Un Grafo de Flujo está formado por 3 componentes fundamentales que ayudan a su elaboración, comprensión y brinda información para corroborar que el trabajo se está haciendo adecuadamente. Los componentes son:

- ✚ **Nodo:** Cada círculo representado se denomina nodo del Grafo de Flujo, el cual representa una o más secuencias procedimentales. Un solo nodo puede corresponder a una secuencia de procesos o a una sentencia de decisión. Puede ser también que hallan nodos que no se asocien, se utilizan principalmente al inicio y final del grafo.
- ✚ **Aristas:** Las aristas representan el flujo de control. Una arista debe terminar en un nodo, incluso aunque el nodo no represente ninguna sentencia procedimental.
- ✚ **Regiones:** Las regiones son las áreas delimitadas por las aristas y nodos. También se incluye el área exterior del grafo, contando como una región más. Las regiones se enumeran y la cantidad de regiones es equivalente a la cantidad de caminos independientes del conjunto básico de un programa.

A continuación se analiza el código y se enumeran las sentencias de código del procedimiento realizado sobre el método **GetPcId** encargado de devolver el token con la información de una estación de trabajo, ver figura 3.2 y el grafo de flujo asociado al mismo figura 3.3:

```

95 public string GetPcId()
96 {
97     var token = DGMConfiguration.DGMConfiguration.Configuration().PcInformation.PcId;
98     var pcInfoData = DateTime.Now + "\n*****\n" + DGMConfiguration.DGMConfiguration.Configuration().PcInformation;
99     var fileName = Path.Combine(AppDomain.CurrentDomain.BaseDirectory, token + ".dgm");
100     try
101     {
102         if (!File.Exists(fileName)) → 2
103         {
104             var streamWriter = new StreamWriter(fileName);
105             streamWriter.WriteLine(pcInfoData);
106             streamWriter.Flush();
107             streamWriter.Close(); } 3
108         }
109     }
110     catch
111     {
112         return token; } 4
113     }
114     return token;
115 } → 5
116

```

Figura 3.3 Método GetPcId

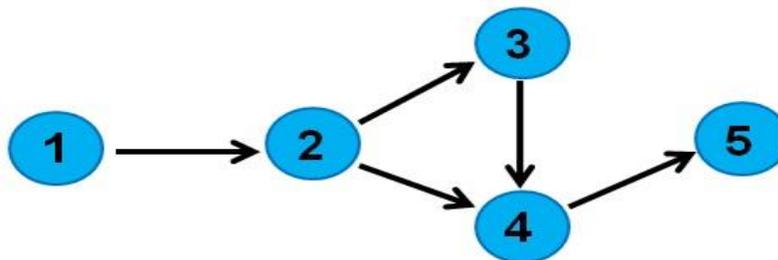


Figura 3. 4 Grafo del flujo asociado al método adicionar asignaciones

Fórmulas para calcular la complejidad ciclomática:

1. $V(G) = (A - N) + 2$

Donde "A" es la cantidad de aristas y "N" es la cantidad de nodos.

$$V(G) = (5 - 5) + 2$$

$$V(G) = 2$$

2. $V(G) = P + 1$

Siendo "P" la cantidad de nodos predicados (son los nodos de los cuales parten dos o más aristas).

$$V(G) = 1 + 1$$

$$V(G) = 2$$

3. $V(G) = R$

Donde "R" representa la cantidad de regiones en el grafo.

$$V(G) = 2$$

El cálculo efectuado mediante las fórmulas ha dado el mismo valor, por lo que se puede decir que la complejidad ciclomática del código es 2, lo que significa que existen dos posibles caminos por donde el flujo puede circular, este valor representa el límite mínimo del número total de casos de pruebas para el procedimiento tratado.

Número	Caminos básicos
1	1-2-3-4-5
2	1-2-4-5

Tabla 3. 3 Caminos básicos del flujo asociado al método GetPcId

3.4.2 Pruebas de Aceptación

El uso de cualquier producto de software tiene que estar justificado por las ventajas que ofrece. Sin embargo, antes de empezar a usarlo es necesario determinar si sus ventajas realmente justifican su uso. El mejor instrumento para esta determinación son las pruebas de aceptación. En estas pruebas se evalúa el grado de calidad del software con relación a todos los aspectos relevantes para que el uso del producto se justifique (30).

Caso de Prueba de Aceptación	
Código: CP3_HU3	HU_3:Manejar dispositivos remoto
Responsable de la prueba: Gianni Espino Hernández	
Descripción: Manejar un dispositivo ubicado en una estación de trabajo remota.	
Condiciones de ejecución: <ol style="list-style-type: none"> 1. El servicio debe estar iniciado. 2. El controlador debe estar iniciado como un servicio en la estación de trabajo remota. 3. Se debe contar con permisos para conectarse al controlador remoto. 	
Entrada / Pasos de ejecución: <ol style="list-style-type: none"> 1. Recibir una petición al servicio. 2. Ejecutar el método correspondiente en el controlador 3. Establece comunicación con el controlador remoto. 4. Si la conexión es satisfactoria. <ol style="list-style-type: none"> 4.1. Ejecutar el método del controlador remoto. 4.2. Generar traza de uso del dispositivo. 4.3. Retornar la información al servicio. 5. Si la conexión no es satisfactoria. <ol style="list-style-type: none"> 5.1. Mostrar mensaje de error. 	

Resultado esperado: Queda retornado un valor como respuesta a la petición recibida al servicio.
--

Evaluación de la prueba: Satisfactoria

Tabla 3. 4 CP Manejar dispositivos remotos.

3.4.3 Resultados de las pruebas

Después de aplicadas las pruebas a partir de los casos de pruebas propuestos, al concluir la primera iteración según el plan de iteraciones y de acuerdo a las funcionalidades implementadas hasta la fecha se detectaron 14 no conformidades, que fueron resueltas en un período de tiempo de 13 días. Se realizó una segunda iteración y se detectaron 8 no conformidades siendo solucionadas en 6 días, se realizó otra corrida detectándose 3 no conformidad mitigándose en 2 días, y luego se probó otra vez con 100% de los resultados esperados. Las no conformidades fueron resueltas luego de identificadas. Lo anterior se evidencia en la figura 3.5.



Figura 3. 5 Grafico de las no conformidades. Fuente: Elaboración propia.

3.5 Conclusiones parciales

En este capítulo se abordó a través de la codificación y prueba del sistema. Durante la etapa de implementación el uso de los estándares de codificación definidos permitió desarrollar un código reutilizable, comprendido por todos los integrantes del equipo de desarrollo. La representación del diagrama de despliegue y las pruebas realizadas, visualizaron la estructura y funcionamiento de la herramienta, en el cual tras la ejecución de varias iteraciones cuyas entregas eran inmediatamente probadas se detectaron varias no conformidades, que luego de ser corregidas en su totalidad, validaron su correcto funcionamiento en relación a los requisitos definidos por el cliente.

Conclusiones generales

La investigación realizada permite a los autores concluir que:

- ✚ A través del estudio de los principales conceptos y sistemas homólogos a nivel mundial se demostró que estas herramientas, aunque poseen características requeridas por el cliente son propietarias o no cumplen con las especificaciones del proyecto. En el caso del DGM 2.9.4 se determinó tomarlo como base para el desarrollo.
- ✚ A partir del análisis de las tecnologías y los mecanismos para acceder a recursos locales de una estación de trabajo desde la web confirmó que es posible el manejo de los dispositivos en ambiente multiplataforma.
- ✚ La especificación de la metodología XP permitió la descripción de las historias de usuario y se determinaron las funcionalidades a implementar.
- ✚ Se realizó el plan de iteraciones para una mejor organización del tiempo con el objetivo de culminar el sistema en el tiempo requerido. La realización de cuatro iteraciones de pruebas durante el proceso de desarrollo, permitió obtener una versión estable final del producto.

Recomendaciones

Los objetivos de esta investigación fueron logrados satisfactoriamente, sin embargo es necesario tener en cuenta algunas recomendaciones:

- ✚ Añadir una nueva funcionalidad que permita compartir más de un dispositivo de un mismo tipo.
- ✚ Utilizar la documentación del presente trabajo como material de estudio en futuras investigaciones para garantizar el perfeccionamiento del sistema.

Referencias bibliográficas

1. Trend, Corporation. Trend Micro Incorporated. *Control de Dispositivos*. [En línea] 2011. [Citado el: 12 de Noviembre de 2014.] http://docs.trendmicro.com/all/ent/officescan/v10.6/es-es/osce_10.6_olhsrv/OHelp/Device/dctrl.htm.
2. Ecured. Ecured. *Dispositivos Biométricos*. [En línea] 25 de febrero de 2013. http://www.ecured.cu/index.php/Dispositivos_biom%C3%A9tricos.
3. About.com. *WebTrends*. [En línea] The New York Time. http://webtrends.about.com/od/webapplications/a/web_application.htm..
4. Mora, Sergio Luján. Programación de aplicaciones web: historia, principios básicos y clientes web. s.l. s.l. : Club Universitario, 2002.
5. Neurotechnology. Corporation. Neurotechnology.com. *GreenBit*. [En línea] 1998. [Citado el: 1 de Enero de 2015.] <http://www.neurotechnology.com/fingerprint-scanner-green-bit-dactyscan-84n.html>.
6. Innovatics. Incorporated. Innovatics. [En línea] [Citado el: 6 de Enero de 2015.] <http://www.innovatics.com/es/>.
7. Innovatics. Fingera Incorporated . Innovatics. *Fingera*. [En línea] 2014. <http://www.innovatics.com/es/products/fingera>.
8. Aware. biometrics Corporation. Aware.com. *Biometrics*. [En línea] 2014. <http://www.aware.com/es/biometrics/biosp.html>.
9. Campos Rodriguez, Sandy y Rodríguez Sánchez, Hector Luis. *Sistema para la interacción y control centralizado de dispositivos en aplicaciones*. La Habana : s.n., 2001.
10. Albanet. Albanet. *Sistema de Gestion Penitenciaria*. [En línea] 2013. <http://www.albet.cu/lineas/gobierno/sistema-de-gestion-penitenciaria>.
11. Datys. Datys. *Biomesys*. [En línea] 2014. http://www.datys.cu/spa/docs/Documentaci%C3%B3n%20Productos/Biomesys%20Control%20de%20Asistencia/Biomesys%20Control%20de%20Asistencia%20V2.0_ET_Es.pdf.
12. México, Universidad Autónoma de México. Fcausa. *Adminastración para la toma de decisiones*. [En línea] 28 de Enero de 2012. <http://fcasua.contad.unam.mx/apuntes/interiores/docs/2005/administracion/5/1553.pdf..>
13. SCRIB. Metodologías de desarrollo de software. Scribd. [En línea] [Citado el: 9 de Diciembre de 2014.] <http://www.scribd.com//doc/2050925/metodologias-de-desarrollo-software..>
14. Méndez, A. V. Metodología de desarrollo de software. *Institución Tecnológico Superior de Apatzingán*. [En línea] 2010.

15. Rumbaugh , J., y otros. *El Lenguaje unificado de modelado: manual de referencia*. s.l. : Pearson Educación, 2007. ISBN 9788478290871.
16. Visual Paradigm International. Visual Paradigm. *Visual Paradigm Internacional*. [En línea] 2010. <http://www.visual-paradigm.com/product/vpuml/>.
17. Monodevelop.com. Monodevelop. [En línea] [Citado el: 15 de Abril de 2015.] <http://www.monodevelop.com/>.
18. Netbeans. Incorporated. Netbeans. *Características del IDE*. [En línea] [Citado el: 21 de Enero de 2015.] <https://netbeans.org/features/index.html>.
19. Microsoft Corporation. Microsoft.com. *Visual C#*. [En línea] [Citado el: 6 de Febrero de 2015.] <https://msdn.microsoft.com/es-es/library/kx37x362.aspx>.
20. IBM Corporation. IBM. *Developersworks.java*. [En línea] [Citado el: 11 de Febrero de 2015.] <http://www.ibm.com/developerworks/ssa/java/tutorials/j-introjava1/>.
21. Fernández, Adelaida Ramírez. CLASIFICACIONES DE TIPOS DE REQUISITOS PARA LA MEJORA DEL PROCESO DE DESARROLLO DEL SOFTWARE. 2012.
22. Roger S, Presman. *Ingeniería de software. Enfoque practico.Arquitectura de Software* . s.l. : 5 Edición .
23. Ganbetadev.com. Ganbetadev. *Metodologías-de-programacion: patrones-de-diseno-que-son-y-por-que-debes-usarlos*. [En línea] [Citado el: 6 de Enero de 2015.] <http://www.genbetadev.com/metodologias-de-programacion/patrones-de-diseno-que-son-y-por-que-debes-usarlos>.
24. Servicios, Líderes del grupo de Tecnología SOA de Accenture. *Arquitectura Orientada a. Centro de Alto Rendimiento de Accenture (CAR)*. 2008.
25. Shalloway, Alan y Trott, James. *Desing Patterns Explained*. s.l. : Addison Wesley, 2004. ISBN: 9780321247148.
26. Pressman, Roger S. *Ingeniería del Software. Un enfoque Practico. 5ta Edición* . s.l. : McGraw-hil Interamericana, 2001. ISBN: 8448132149.
27. Larman, Craig. *UML y Patrones. Introducción al análisis y diseño orientado a objetos*. La Habana. : Felix Varela, 2004.
28. Globetesting.com. Globetesting. *Pruebas del sistema*. [En línea] [Citado el: 20 de Febrero de 2015.] <http://www.globetesting.com/pruebas-funcionales/>.
29. Pressman, Roger. *Ingeniería del Software. Un enfoque práctico*. España : Quinta Edicion, 2001.
30. Pruebas de software. *Prebas de aceptación*. [En línea] [Citado el: 14 de Mayo de 2015.] <http://pruebasdesoftware.com/pruebadeaceptacion.htm>.

Anexos

Anexo 1: Historias de usuario

HU_1. Manejar los servicios web.

Historia de Usuario	
Numero: HU_1	Usuario: Sistema
Nombre de la historia: Manejar los servicios web.	
Prioridad en negocio: Muy Alta	Riesgo en desarrollo: Medio
Puntos estimados: 2	Iteración asignada: 1
Responsable: Gianni Espino Hernández	
Descripción: <ol style="list-style-type: none">1. Obtener de la configuración del servicio local y extraer de cada servicio la siguiente información:<ol style="list-style-type: none">a) Nombre del ensamblado del cliente.b) Nombre del ensamblado del controlador.c) Si está activo o no el servicio.2. Conformar los servicios usando los clientes y los controladores respectivamente de cada uno de los servicios activos.3. Iniciar cada uno de los servicios conformados.4. Detener cada uno de los servicios conformados.5. Detener los servicios tras un tiempo de inactividad.	
Observaciones: 1. Debe existir el fichero de configuración del sistema.	

TablaA1. 1 HU_1 Manejar los servicios web

HU_2. Aceptar nuevos controladores de dispositivos.

Historia de Usuario	
Numero: HU_2	Usuario: Sistema
Nombre de la historia: Aceptar nuevos controladores de dispositivos.	
Prioridad en negocio: Alta	Riesgo en desarrollo: Alta
Puntos estimados: 1	Iteración asignada: 1
Responsable: Gianni Espino Hernández	
Descripción: 1. Revisar la configuración del sistema. 2. Obtener la configuración del servicio. 3. Sí existe cambio en el controlador. 3.1. Se debe detener el servicio. 4. Conformar un nuevo servicio con el controlador nuevo. 5. Iniciar el servicio.	
Observaciones:1. Debe existir un cliente que lo utilice.	

TablaA1. 2 Aceptar nuevos controladores de dispositivos

UH_4. Aceptar el manejo de dispositivos remotamente.

Historia de Usuario	
Numero: HU_4	Usuario: Sistema
Nombre de la historia: Aceptar el manejo de dispositivos remotamente.	

Prioridad en negocio: Muy Alta	Riesgo en desarrollo: Media
Puntos estimados: 1	Iteración asignada: 1
Responsable: Gianni Espino Hernández	
Descripción: 1. Recibir una petición de manejo de dispositivo. 2. Verificar que el dispositivo no se encuentre en uso. 3. Si se encuentra en uso. 3.1. Rechazar la solicitud. 4. Si no se encuentra en uso. 4.1. Ejecutar el método correspondiente a la petición. 4.2. Devolver el valor retornado a la estación de trabajo remoto.	
Observaciones: 1. El controlador local del dispositivo debe estar expuesto como un servicio.	

TablaA1. 3 HU_4 Aceptar el manejo de dispositivos remotamente

HU_5. Consultar el estado de un servicio remotamente.

Historia de Usuario	
Numero: HU_5	Usuario: Sistema
Nombre de la historia: Consultar el estado de un servicio remotamente.	
Prioridad en negocio: Media	Riesgo en desarrollo: Bajo
Puntos estimados: 1	Iteración asignada: 1
Responsable: Gianni Espino Hernández	
Descripción: 1. Recibir una petición al servicio de administración. 2. Retornar el estado del servicio con la siguiente información: a) Iniciado. b) Detenido. c) No existe. d) Controlador activo.	

Observaciones: 1. El servicio de administración debe estar iniciado.

TablaA1. 4 Consultar el estado del servicio remotamente

HU_6. Validar el acceso a los servicios web.

Historia de Usuario	
Numero: HU_6	Usuario: Sistema
Nombre de la historia: Validar el acceso a los servicios web.	
Prioridad en negocio: Alta	Riesgo en desarrollo: Alto
Puntos estimados: 1	Iteración asignada: 1
Responsable: Gianni Espino Hernández	
Descripción: 1. Recibir una petición al servicio web. 2. Extraer la siguiente información de la petición: a) Vía de acceso. b) Usuario c Dispositivo. 3. Validar que la vía de acceso esté permitida y que el usuario tiene permisos sobre el dispositivo. 4. Si el acceso es permitido. 4.1. Procesar la petición. 4.2. Retornar la respuesta de la petición.	
Observaciones: 1. El servicio web debe estar iniciado en el Servicio local para el manejo de dispositivos.	

TablaA1. 5 Validar el acceso a los servicios web

HU_7. Gestionar los servicios web a usar en el Servicio local para el manejo de dispositivos.

Historia de Usuario	
Numero: HU_7	Usuario: Administrador
Nombre de la historia: Gestionar los servicios web a usar en el Servicio local para el manejo de dispositivos.	
Prioridad en negocio: media	Riesgo en desarrollo: Media
Puntos estimados: 2	Iteración asignada: 1
Responsable: Gianni Espino Hernández	
Descripción: 1. Seleccionar el servicio que se desea gestionar. 2. Definir si el servicio estará habilitado o no con la siguiente información: a) Sí. b) No. Cambiarle el estado al servicio seleccionado	
Observaciones:	

Prototipo de interfaz:

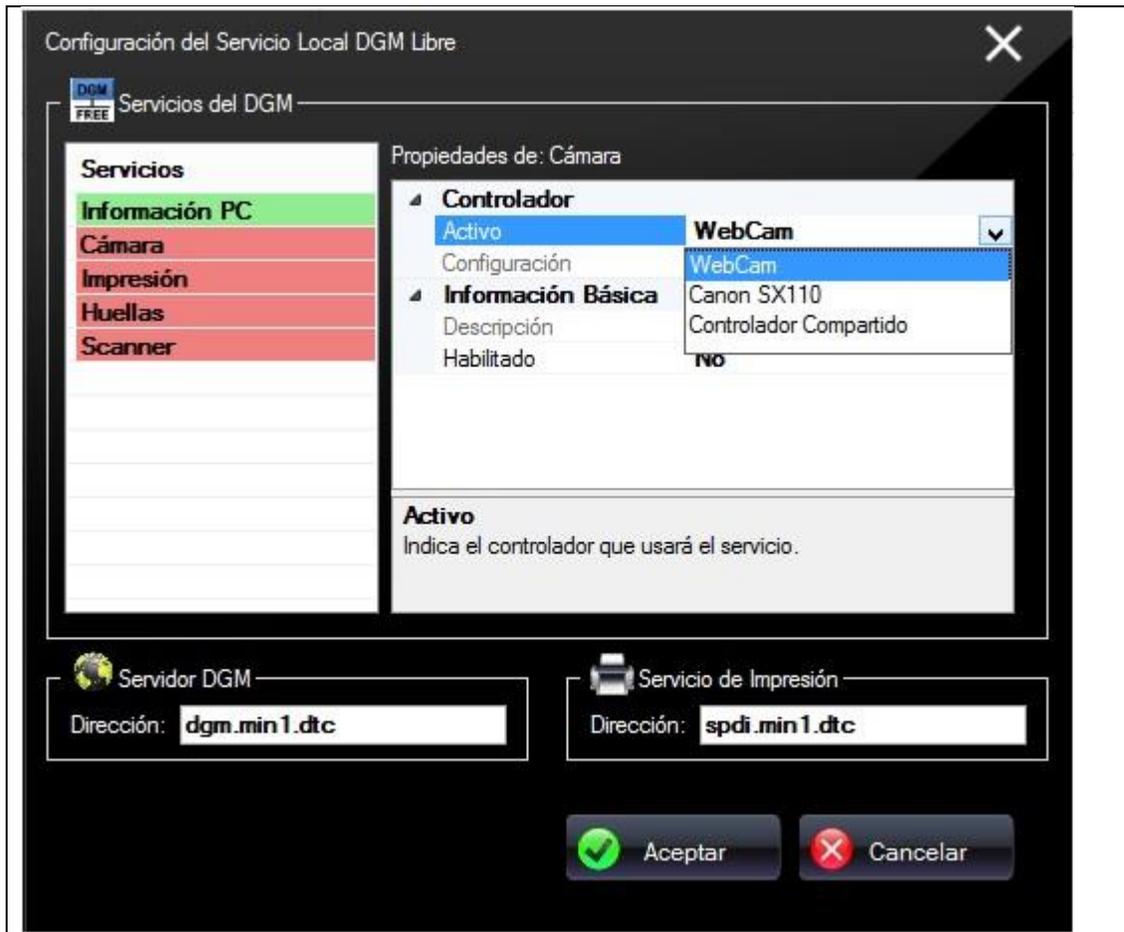


TablaA1. 6 Gestionar los dispositivos a usar en el servicio local para el manejo de dispositivos

HU_8 Gestionar los dispositivos a usar en el Servicio local.

Historia de Usuario	
Numero: HU_8	Usuario: Administrador

Nombre de la historia: Gestionar los dispositivos a usar en el Servicio local	
Prioridad en negocio: media	Riesgo en desarrollo: Medio
Puntos estimados: 2	Iteración asignada: 1
Responsable: Gianni Espino Hernández	
Descripción:	
<ol style="list-style-type: none"> 1. Seleccionar el servicio al cual se desea cambiar el dispositivo. 2. Seleccionar el dispositivo que se desea usar para el servicio. 3. Cambiar el dispositivo del servicio. 	
Observaciones:	
Prototipo de interfaz:	



TablaA1. 7 HU_8 Gestionar los dispositivos a usar en el servicio local

HU_9 Actualizar el Servicio local para el manejo de dispositivos.

Historia de Usuario	
Numero: HU_9	Usuario: Administrador
Nombre de la historia: Actualizar el Servicio local para el manejo de dispositivos.	

Prioridad en negocio: Media	Riesgo en desarrollo: Alto																		
Puntos estimados: 1	Iteración asignada: 1																		
Responsable: Yasmany Escofe Thomson																			
Descripción: <ol style="list-style-type: none"> 1. Se selecciona la opción “Actualizar” en el Panel de Control del DGM. 2. Conectar el actualizador al servicio de actualización. 3. Descargar la información necesaria para llevar a cabo el proceso de actualización. 4. Detener todos los servicios web. 5. Descargar todos los ficheros necesarios para la actualización. 6. Reemplazar todos los ficheros antiguos. Iniciar todos los servicios web ya actualizados. 																			
Observaciones: El servicio de actualización y la aplicación de actualización deben estar en ejecución y correctamente configuradas.																			
Prototipo de interfaz:  <p>The image shows two screenshots. The top one is a Windows Services console window for 'Servicios' with the 'Actualizar' option highlighted. The bottom one is a log window titled 'Configuración del Servicio Local DGM Libre' with a list of messages:</p> <table border="1"> <thead> <tr> <th>No.</th> <th>Mensaje</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>Iniciando actualización.</td> </tr> <tr> <td>2</td> <td>Conectando al servicio de actualización del DGM.</td> </tr> <tr> <td>3</td> <td>Obteniendo información desde el servicio de actualización ...</td> </tr> <tr> <td>4</td> <td>Descargando archivos desde el servidor de actualización.</td> </tr> <tr> <td>5</td> <td>Todos los archivos han sido descargados satisfactoriamente.</td> </tr> <tr> <td>6</td> <td>Cerrando las aplicaciones del DGM.</td> </tr> <tr> <td>7</td> <td>Todas las aplicaciones han cerrado satisfactoriamente.</td> </tr> <tr> <td>8</td> <td>Iniciando copia de seguridad de los archivos.</td> </tr> </tbody> </table>		No.	Mensaje	1	Iniciando actualización.	2	Conectando al servicio de actualización del DGM.	3	Obteniendo información desde el servicio de actualización ...	4	Descargando archivos desde el servidor de actualización.	5	Todos los archivos han sido descargados satisfactoriamente.	6	Cerrando las aplicaciones del DGM.	7	Todas las aplicaciones han cerrado satisfactoriamente.	8	Iniciando copia de seguridad de los archivos.
No.	Mensaje																		
1	Iniciando actualización.																		
2	Conectando al servicio de actualización del DGM.																		
3	Obteniendo información desde el servicio de actualización ...																		
4	Descargando archivos desde el servidor de actualización.																		
5	Todos los archivos han sido descargados satisfactoriamente.																		
6	Cerrando las aplicaciones del DGM.																		
7	Todas las aplicaciones han cerrado satisfactoriamente.																		
8	Iniciando copia de seguridad de los archivos.																		

TablaA1. 8 HU_9 Actualizar el servicio local para el manejo de dispositivos

Anexo 2: Descripción de las tarjetas CRC

ProxyService	
Responsabilidades	Colaboradores
Establece la comunicación con el sistema para el control centralizado de dispositivos.	-

TablaA2. 1 Tarjeta CRC de la clase ProxyService.

ServiceManage	
Responsabilidades	Colaboradores
Gestiona el servicio ejemplo carga la lista de clientes, retorna una lista de servicios activados.	DGMConfiguration

TablaA2. 2 Tarjeta CRC de la clase ServiceManage

DGMConfiguration	
Responsabilidades	Colaboradores
Carga la última configuración del servicio a través de un archivo de configuración.	PcInformation

TablaA2. 3 Tarjeta CRC de la clase DGMConfiguration.

DGMWindowsService	
Responsabilidades	Colaboradores
Inicia el servicio y genera el archivo de configuración	-

TablaA2. 4 Tarjeta CRC de la clase DGMWindowsService.

WebCam	
Responsabilidades	Colaboradores
<p>Clase que controla la web cam es la encargada de todas las responsabilidades de la misma.</p> <p>-Captured.</p> <p>-Connect.</p> <p>-Status.</p> <p>-Disconnect.</p>	<p>- BasicController</p> <p>- ICameraController</p>

TablaA2. 5 Tarjeta CRC de la clase WebCam.

CanonSx110	
Responsabilidades	Colaboradores
<p>Clase que controla la canonSx110 es la encargada de todas las responsabilidades de la misma.</p> <p>-Captured.</p> <p>-Connect.</p> <p>-Status.</p> <p>-Disconnect.</p>	<p>- BasicController</p> <p>- ICameraController</p>

TablaA2. 6 Tarjeta CRC de la clase CanonSx110.

DactyID15Controller	
Responsabilidades	Colaboradores
<p>Clase que controla el dispositivo de captación biométrica un lector de huellas Dacty ID 15 es la encargada de todas las responsabilidades de la misma.</p> <p>-Captured.</p> <p>-Connect.</p> <p>-Finger.</p> <p>-Image.</p> <p>-Disconnect.</p>	<p>-BasicController</p> <p>-IFingerController</p>

TablaA2. 7 Tarjeta CRC de la clase DactyID15Controller.

CPDIPrinter	
Responsabilidades	Colaboradores
<p>Clase que controla el dispositivos de impresión es la encargada de todas las responsabilidades de la misma.</p> <p>-Print.</p> <p>-Connect.</p> <p>-Status.</p> <p>-Disconnect.</p>	<p>-BasicController</p> <p>-ICPDIPrinterController</p>

TablaA2. 8 Tarjeta CRC de la clase CPDIPrinter.

ScannerController	
Responsabilidades	Colaboradores
<p>Clase que controla el scanner es la encargada de todas las responsabilidades del mismo.</p> <p>-CompressImage.</p> <p>-Scan.</p> <p>-Image.</p> <p>-Connect.</p> <p>-Status.</p> <p>-CloseConnection.</p>	<p>-BasicController</p> <p>-IScannerController</p>

TablaA2. 9 Tarjeta CRC de la clase ScannerController.

Anexo 3: Casos de prueba de aceptación

Caso de Prueba de Aceptación	
Código: CP1_HU1	HU_1:Manejar los servicios web
Responsable de la prueba: Gianni Espino Hernández	
Descripción: Obtener de la configuración del servicio local y conformar los servicios usando los clientes y los controladores respectivamente de cada uno de los servicios activos.	

Condiciones de ejecución: Debe existir el fichero de configuración del sistema.
Entrada / Pasos de ejecución: 1-Ejecutar el servicio y acceder al panel de configuración. 2-Configurar el servicio e iniciar cada uno de los servicios conformados. 3-Detener los servicios tras un tiempo de inactividad
Resultado esperado: La configuración se guarda satisfactoriamente y los servicios se activan o se desactivan a petición del usuario.
Evaluación de la prueba: Satisfactoria

TablaA3. 1 CP Manejar los servicios web.

Caso de Prueba de Aceptación	
Código: CP2_HU2	HU_2: Aceptar nuevos Controladores de dispositivos
Responsable de la prueba: Gianni Espino Hernández	
Descripción: Actualizar un controlador existente.	
Condiciones de ejecución: Debe existir un cliente que lo utilice.	
Entrada / Pasos de ejecución: 1. Revisar la configuración del sistema. 2. Obtener la configuración del servicio. 3. Sí existe cambio en el controlador. 3.1. Se debe detener el servicio. 4. Conformar un nuevo servicio con el controlador nuevo. 5. Iniciar el servicio.	

Resultado esperado: El controlador del servicio web quedó disponible.
Evaluación de la prueba: Satisfactoria

TablaA3. 2 CP Aceptar nuevos controladores de dispositivos.

Caso de Prueba de Aceptación	
Código: CP4_HU4	HU_4: Aceptar el manejo de dispositivos remotamente
Responsable de la prueba: Yasmany Escofet Thompson	
Descripción: Interactuar con un dispositivo remotamente.	
Condiciones de ejecución: El controlador local del dispositivo debe estar expuesto como un servicio.	
Entrada / Pasos de ejecución:	
<ol style="list-style-type: none"> 1. Recibir una petición de manejo de dispositivo. 2. Verificar que el dispositivo no se encuentre en uso. 3. Si se encuentra en uso. <ol style="list-style-type: none"> 3.1. Rechazar la solicitud. 4. Si no se encuentra en uso. <ol style="list-style-type: none"> 4.1. Ejecutar el método correspondiente a la petición. 4.2. Devolver el valor retornado a la estación de trabajo remoto. 	
Resultado esperado: Se interactuó con el dispositivo remoto.	
Evaluación de la prueba: Satisfactoria	

TablaA3. 3 Aceptar el manejo de dispositivos remotamente.

Caso de Prueba de Aceptación	
Código: CP5_HU5	HU_5:Consultar el estado del servicio remotamente
Responsable de la prueba: Yasmany Escofet Thompson	
Descripción: Ofrecer información sobre el funcionamiento del servicio remotamente.	
Condiciones de ejecución: El servicio de administración debe estar iniciado.	
Entrada / Pasos de ejecución:	
<ol style="list-style-type: none"> 1. Recibir una petición al servicio de administración. 2. Retornar el estado del servicio con la siguiente información: <ol style="list-style-type: none"> a Iniciado. b Detenido. c No existe. d Controlador activo. 	
Resultado esperado: Se obtuvo información del servicio.	
Evaluación de la prueba: Satisfactoria	

TablaA3. 4 CP Consultar el estado del servicio remotamente.

Caso de Prueba de Aceptación	
Código: CP6_HU6	HU_6:Validar el acceso a los servicios web
Responsable de la prueba: Yasmany Escofet Thompson	
Descripción: Validar el acceso a un servicio web expuesto en el Servicio local para el manejo de dispositivos.	

Condiciones de ejecución: El servicio web debe estar iniciado en el Servicio local para el manejo de dispositivos.
Entrada / Pasos de ejecución: 1. Recibir una petición al servicio web. 2. Extraer la siguiente información de la petición: a Vía de acceso. b Usuario. c Dispositivo. 3. Validar que la vía de acceso esté permitida y que el usuario tiene permisos sobre el dispositivo. 4. Si el acceso es permitido. 4.1. Procesar la petición. 4.2. Retornar la respuesta de la petición.
Resultado esperado: Se concede o deniega el acceso al servicio web.
Evaluación de la prueba: Satisfactoria

TablaA3. 5 CP Validar el acceso a los servicios web

Caso de Prueba de Aceptación	
Código: CP7_HU7	HU_7:Gestionar los servicios web a usar en el Servicio local para el manejo de dispositivos
Responsable de la prueba: Gianni Espino Hernández	
Descripción: Poder configurar que servicios web estarán activos y cuáles no.	
Condiciones de ejecución:	-
Entrada / Pasos de ejecución:	

<p>1. Seleccionar el servicio que se desea gestionar.</p> <p>2. Definir si el servicio estará habilitado o no con la siguiente información:</p> <p>a Sí.</p> <p>b No.</p> <p>3.Cambiarle el estado al servicio seleccionado</p>
Resultado esperado: El servicio especificado cambia de estado.
Evaluación de la prueba: Satisfactoria

TablaA3. 6 CP Gestionar los servicios web a usar en el Servicio local para el manejo de dispositivos

Caso de Prueba de Aceptación	
Código: CP8_HU8	HU_8: Gestionar los dispositivos a usar en el Servicio local para el manejo de dispositivos.
Responsable de la prueba: Gianni Espino Hernández	
Descripción: Poder configurar que dispositivo usar para cada servicio web.	
Condiciones de ejecución: -	
Entrada / Pasos de ejecución:	
<p>1. Seleccionar el servicio al cual se desea cambiar el dispositivo.</p> <p>2. Seleccionar el dispositivo que se desea usar para el servicio.</p> <p>3. Cambiar el dispositivo del servicio.</p>	
Resultado esperado: Cambia el dispositivo para el servicio especificado.	
Evaluación de la prueba: Satisfactoria	

TablaA3. 7 CP Gestionar los dispositivos a usar en el Servicio local para el manejo de dispositivos

Caso de Prueba de Aceptación	
Código: CP9_HU9	HU_9: Actualizar el Servicio local para el manejo de dispositivos.
Responsable de la prueba: Gianni Espino Hernández	
Descripción: Mantener el Servicio local para el manejo de dispositivos, todos los servicios web, y todos los controladores y sus dependencias actualizadas.	
Condiciones de ejecución: El servicio de actualización y la aplicación de actualización deben estar en ejecución y correctamente configuradas.	
Entrada / Pasos de ejecución: <ol style="list-style-type: none"> 1. Se selecciona la opción “Actualizar” en el Panel de Control del DGM Libre. 2. Conectar el actualizador al servicio de actualización. 3. Descargar la información necesaria para llevar a cabo el proceso de actualización. 4. Detener todos los servicios web. 5. Descargar todos los ficheros necesarios para la actualización. 6. Reemplazar todos los ficheros antiguos. Iniciar todos los servicios web ya actualizados. 	
Resultado esperado: El Servicio local para el manejo de dispositivos con todos los servicios web se encuentran actualizados.	
Evaluación de la prueba: Satisfactoria	

TablaA3. 8 CP Actualizar el servicio local para el manejo de dispositivos