Universidad de las Ciencias Informáticas
Facultad 1

Título: “Sistema para la ejecución de la autenticación pasiva de los pasaportes electrónicos”.

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

Autores:
Deyanira López Torres.
Wilver Mayet Sánchez.

Tutores:
Alina Surós Vicente.
Enrique Almeida Maldonado.

La Habana, Cuba.
Junio 2012.

(...) es incuestionable que el principio de la combinación del estudio y el trabajo es la única fórmula de educación comunista. No hay otra. Nadie aprenderá a nadar sobre la tierra, y nadie caminará sobre el mar. Al hombre lo hace su medio ambiente, al hombre o hace su propia vida, su propia actividad. Y aprenderemos a respetar lo que crea el trabajo creando. Enseñaremos a respetar esos bienes enseñándolo a crear esos bienes. Y no hay otro camino.

Declaración de Autoría

Declaramos que somos los únicos autores de este trabajo y autorizamos al Centro de Identificación y Seguridad Digital (CISED) de la Universidad de las Ciencias Informáticas (UCI) los derechos patrimoniales del mismo, con carácter exclusivo.

Para que así conste firmamos la presente a los _____ días del mes de _____de 2011.

Firma del Autor: _____

Deyanira López Torres.

Firma del Autor: _____

Wilver Mayet Sánchez.

Firma del Tutor: _____

Alina Surós Vicente.

Firma del Tutor: _____

Enrique Almeida Maldonado.

Datos de Contacto

Autores:

Deyanira López Torres.

Universidad de las Ciencias Informáticas

Email: dltorres@estudiantes.uci.cu

Wilver Mayet Sánchez.

Universidad de las Ciencias Informáticas

Email: wmayet@estudiantes.uci.cu

Tutores:

Msc. Alina Surós Vicente.

Universidad de las Ciencias Informáticas

Email: asuros@uci.cu

Ing. Enrique Almeida Maldonado.

Universidad de las Ciencias Informáticas

Email: ealmeida@uci.cu

Agradecimientos

Wilver

Gracias a todas las personas que de una forma u otra me han ayudado en la culminación del trabajo de diploma, en especial a mis tutores Alina Surós Vicente y Enrique Almeida Maldonado, por su gran ayuda profesional, a mi compañera de tesis por aguantarme y trabajar en equipo, y a todos mis compañeros del laboratorio que siempre estuvieron ahí cuando surgía alguna duda.

Deyanira

Le agradezco con todo corazón a los tutores Alina Surós Vicente y Enrique Almeida Maldonado, que son los mejores tutores que se pueden tener, por ser exigentes, persistentes y sobre todas las cosas por contagiarnos con su luz larga; a todos los demás profes que una forma u otra nos dieron su ayuda incondicional como los profes Adonis, Reynier y Roberto.

A mis padres y demás familiares por siempre estar guiándome por el camino correcto.

A mi compañero Wilver por tener tanta calma conmigo y por su preocupación.

A nuestros compañeros que nos brindaron mucho apoyo Arniel, Wilfredo, Fonseca y todos los que de una forma u otra nos dieron un poquito de su sabiduría.

Dedicatoria

Wilver

..A mi madre, por ser lo que más quiero en el mundo y ser mi inspiración para lograr todos mis objetivos…

..A mi padre, por ser el ejemplo a seguir y por los consejos que siempre me dio…

..A mi hermano, por estar siempre conmigo y servirle de guía y ejemplo...

..A mi abuelo, por confiar en mí y lograr su objetivo de verme ingeniero…

..A mi tía Ángela, por apoyarme y abrirme las puertas de su casa con tanto cariño durante estos 5 años…

..A todos mis amigos, en las buenas y las malas, los que están aquí en la universidad y los que no están, Frank, Yaimi, Yamichel, Flecha, German, Noel, entre otros.

..A todas estas personas, siempre los llevo en mi corazón…

Deyanira

Yo dedico mi tesis de todo corazón a mi mamá que es mi motor impulsor, a mi abuelita que es su sueño verme terminar mis estudios, a mi abuelito que aunque ya no se encuentra en este mundo se que sería un orgullo para él, a mi papá y mi tía que siempre estuvieron al pendiente de mí, a mis hermanitas por darme ideas y apoyo. A mis tiotes que todos son como padres para mí, a mis suegros que me acogieron como una hija. A mi padrastro que aunque no lo diga yo se que esta orgulloso de mi. Y a todas las personas que contribuyeron a mi formación y culminación de estudios.

Resumen

El pasaporte electrónico (pasaporte-e) es una tecnología que desde el año 2005 ha puesto a disposición internacional la Organización Internacional de Aviación Civil (OACI). La característica fundamental de este documento es que permite almacenar elementos biométricos de su portador a través de un circuito integrado sin contacto. La posibilidad de que el pasaporte-e sea interoperable mundialmente implica el cumplimiento con las normativas establecidas por la OACI, entre las que se encuentra como única medida de seguridad de carácter obligatorio la autenticación pasiva.

La autenticación pasiva consiste en almacenar en el objeto de seguridad del documento los hash de los grupos de datos y su firma digital, permitiendo verificar la integridad de los datos y autenticidad del pasaporte-e. La comprobación de la autenticidad del documento depende de que el sistema que reciba los pasaportes pueda validar la firma digital contenida en el objeto de seguridad a partir del certificado digital del firmante del documento. Los certificados y las listas de revocación estarán disponibles de forma pública en el directorio de llaves pública de la OACI o pueden obtenerse mediante intercambio bilateral entre naciones.

En este trabajo se presenta un sistema para la ejecución de la autenticación pasiva, cuyo objetivo fundamental es verificar la firma digital almacenada en el objeto de seguridad de documento, basado en los certificados y listas de revocación. Este sistema constituye una herramienta para la verificación de la autenticidad del documento, permitiendo así el uso de los datos contenidos en el circuito integrado.

Palabras clave: autenticación pasiva, certificados, directorio de llave pública, firma digital.

Índice

Introducción.....1
Capítulo 1: Fundamentación Teórica.....6
1.1. Introducción.....6
1.2. Pasaporte electrónico.....6
1.3. Medidas de seguridad.....7
1.3.1. Opcionales.....7
1.3.2. Obligatoria.....7
1.4. Formato de certificados X.509.....9
1.5. Directorio de llaves públicas de la OACI.....10
1.6. Sistemas de verificación de autenticación pasiva.....11
1.7. Tecnologías, Metodologías y Lenguajes.....14
1.8. Herramientas de Desarrollo.....29
1.9. Conclusiones.....31
Capítulo 2: Propuesta de Solución.....33
2.1. Introducción.....33
2.2. Fase Visión y Alcance.....33
2.3. Descripción del Problema.....33
2.4. Propuesta de Solución.....34
2.5. Fase de Planificación.....36
2.6. Descripción de los escenarios.....41
2.7. Requisitos de calidad de servicio.....43
2.8. Fase de diseño.....43
2.9. Arquitectura de la solución.....44
2.10. Estilos arquitectónicos.....44
2.11. Diagrama de clases.....46
2.12. Patrones de diseño.....47
2.13. Conclusiones.....50

Capítulo 3: Implementación y pruebas 51
3.1. Introducción 51
3.2. Estándares de codificación. 51
3.3. Diagrama de componentes. 53
3.4. Diagrama de despliegue. 54
3.5. Interfaz gráfica 55
3.6. Fase de Estabilización 56
3.6.1. Pruebas unitarias 56
3.6.2. Pruebas de caja negra o validación del sistema. 59
3.7. Conclusiones 62
Conclusiones 63
Recomendaciones 64
Referencias Bibliográficas 79

Índice de tablas

Tabla 1 Módulos del sistema. 35
Tabla 2 Definición de personas. 35
Tabla 3 Prioridad de escenarios. 39
Tabla 4 Plan de iteraciones. 40
Tabla 5 Descripción del escenario "Verificar SOD" 41
Tabla 6 Descripción del escenario "Verificar integridad del pasaporte" 41
Tabla 7 Descripción del escenario "Gestionar fuente acuerdos bilaterales y del emisor" 42
Tabla 8 Descripción del escenario "Adicionar certificado CDS; CSCA al repositorio" 42
Tabla 9 Resumen de no conformidades de las pruebas unitarias. 58
Tabla 10 Descripción del caso de prueba "Gestionar repositorio de certificados". 60
Tabla 11 Resumen de no conformidades por iteraciones de las pruebas de caja negra. 61

Índice de figuras

Figura 1 Pasos para verificar autenticidad de pasaportes-e.....9
Figura 2 Esquema del directorio LDAP para la gestión de recursos.[18] 16
Figura 3 Esquema del sistema Open LDAP para la gestión de políticas.[18].....17
Figura 4 Diagrama de software. 34
Figura 5 Prioridad de escenarios por módulos. 39
Figura 6 Estilo arquitectónico en capas del módulo verificar SOD.....45
Figura 7 Estilo arquitectónico en capas del módulo repositorio. 45
Figura 8 Diagrama de clases general del sistema.....47
Figura 9 Diagrama de componentes.....54
Figura 10 Diagrama de despliegue 55
Figura 11 Interfaz gráfica del módulo repositorio.....56
Figura 12 Prueba unitaria al software.....57
Figura 13 Representación de iteraciones de pruebas unitarias.....58
Figura 14 Representación de iteraciones de pruebas de caja negra.....61

Introducción

Hoy en día el uso y desarrollo de las tecnologías pueden tener gran impacto en la sociedad, cuando se habla de los avances que se han alcanzado no se puede dejar de mencionar la gran ayuda que han brindado las computadoras en vía al progreso. En algunos campos como la educación, cultura, salud, entre otros, es evidente que las computadoras son muy necesarias, ya que pueden almacenar un gran cúmulo de información para posteriormente ser analizadas y ofrecer conclusiones que contribuyan en alguna medida a cualquier nueva inventiva, o al mejoramiento de la misma. Por ello la importancia de automatizar todos los procesos en las empresas, y hacer más dinámicas y versátiles las operaciones en las mismas. De ahí el proceso de digitalización que se está llevando a cabo en cada una de las instituciones y órganos de trabajo, con el objetivo de mejorar el proceso de análisis y manejo de la información. [1]

La Organización Internacional de Aviación Civil (OACI) no se ha quedado atrás y comienza su trabajo con los documentos de viaje de lectura mecánica (DVLM) [2] en 1968, su actividad fundamental se dirigió a recomendaciones para que la libreta o tarjeta de pasaporte normalizada fuera susceptible a la lectura mecánica, con el objetivo de acelerar el despacho de pasajeros por los puestos de control de pasaporte.

Luego continúa su labor en función de mejorar la confirmación de la identidad con los pasaportes, iniciándose en 1998 un estudio profundo para la inclusión de características biométricas en el pasaporte que permitieran validar la identidad del ciudadano, trayendo consigo la concesión de un nuevo documento de viaje: el pasaporte electrónico.

Desde ese momento la OACI ha puesto a disposición internacional el pasaporte electrónico (pasaporte-e), que se rige por las especificaciones técnicas de los Documentos de Viaje de Lectura Mecánica 9303 partes (1, 2 y 3)[3] y ha recibido la aprobación de la Organización Internacional de Normalización con carácter de normas ISO 7501-1, 7501-2, 7501-3 respectivamente[4]. Tiene como característica fundamental que cuenta con un circuito integrado sin contacto (ICC, por sus siglas en inglés) como dispositivo de almacenamiento, conforme a la norma ISO/IEC 14443.

El ICC es introducido para ampliar la capacidad de almacenamiento de los DVLM e incluir datos biométricos y biográficos del portador, lo que permite verificar la autenticidad e integridad del documento utilizando funciones resúmenes como SHA-1, MD5, entre otros[5]. La forma de organización normalizada de los datos en el ICC se denomina estructura lógica de datos (LDS, por sus siglas en inglés), y como componente más importante cuenta con un objeto de seguridad del documento (SOD, por sus siglas en

inglés), el cual es un elemento definitorio en la ejecución de la única medida de seguridad obligatoria (autenticación pasiva) con que cuentan estos documentos de viaje.

En la autenticación pasiva se verifica integridad y autenticidad del documento, se prueba que el contenido del SOD sea auténtico y no se haya modificado. Debe contarse con los certificados del firmante de documentos (CDS, por sus siglas en inglés), y de la autoridad certificadora de firmas del país (CSCA, por sus siglas en inglés). Los certificados contienen datos electrónicos que solamente se utilizarán luego de que sea efectiva la verificación del SOD, dichos certificados deben cumplir con estándares de seguridad X.509 y Norma ISO 14443. Cada país obtiene los certificados por dos vías fundamentales, mediante intercambios bilaterales entre países o por el directorio de claves públicas (PKD, por sus siglas en inglés) de la OACI.[3]

Los sistemas de inspección (SI), como son llamados internacionalmente los puntos donde se leen los DVLM, por ejemplo, los de control migratorio deben contener las funcionalidades necesarias para realizar la autenticación pasiva en los DVLM.

Para la República Bolivariana de Venezuela se desarrolla en la Universidad de las Ciencias Informáticas (UCI), específicamente en el Centro de Identificación y Seguridad Digital (CISED), el proyecto de Desarrollo de Solución Integral para la Transformación y Modernización del Sistema de Identificación, Migración y Extranjería, que cuenta con tres fases. Este proyecto en el área de software tiene como objetivo modernizar los sistemas de Identificación, Migración y Extranjería.

Situación Problémica

Partiendo de la situación problémica expresada en el proyecto técnico Puntos de Control Migratorios de la República Bolivariana de Venezuela fase III: "No existe un mecanismo en la aplicación que permita realizar lectura de documentos electrónicos, ni verificar la firma digital contenida en el objeto de seguridad del chip"[6]. Se adquiere para la solución para el sistema de control migratorio del SAIME el lector *Regula*, este cuenta en su *Software Development Kit* (SDK, por sus siglas en inglés) con funcionalidades para realizar la autenticación pasiva. Por ejemplo en el caso de la gestión de los certificados el comando *RFID_Command_Get_PassivePKD*, que se realiza a través de un número anidado de directorios donde estarán almacenados los CDS y CRL. Sin embargo la gestión de los certificados se debe realizar de forma local, tornando lento y complicado este proceso para todos los puntos de inspección donde será desplegado el sistema, de conjunto con el nivel de actualización requerido, que según recomendación de

la OACI debe ser diario. Así mismo en el caso de que se adquieran nuevos lectores, no garantizan que el SDK, en el caso de precisarlo, contenga funcionalidades para realizar la autenticación pasiva implicando que sea una condición para nuevas adquisiciones que impliquen costos mayores.

Teniendo en cuenta la situación problémica planteada se define como **problema de investigación**: ¿Cómo mejorar la ejecución de la autenticación pasiva de los pasaportes-e en los sistemas de inspección de la República Bolivariana de Venezuela?

El **objeto de estudio** de esta investigación engloba el proceso de autenticación pasiva de los pasaportes-e.

El **objetivo general** de esta investigación es desarrollar un sistema para la ejecución de la autenticación pasiva de los pasaportes-e, para los sistemas de inspección de la República Bolivariana de Venezuela.

El **campo de acción** de la investigación está específicamente referido al proceso de autenticación pasiva de los pasaportes-e, en los sistemas de inspección de la República Bolivariana de Venezuela.

Planteándose las siguientes **tareas científicas**:

- Análisis del estado del arte sobre los sistemas para la ejecución de la autenticación pasiva, los estándares, herramientas y metodologías para el desarrollo del sistema.
- Análisis y diseño de la solución.
    - Definición de los requerimientos para el desarrollo del Sistema de ejecución de la autenticación pasiva en pasaporte-e.
    - Análisis de plataformas y librerías para la ejecución de la autenticación pasiva en pasaportes-e.
    - Análisis de la arquitectura a utilizar.
    - Generación de los artefactos que describen el proceso de ingeniería de software según la metodología seleccionada.
- Implementación del módulo Repositorio de certificados.
- Implementación del módulo Verificación del SOD.
- Ejecución de pruebas al módulo Repositorio de certificados.
- Ejecución de pruebas al módulo Verificación de SOD.

Como **métodos** para la investigación se utilizarán:

Métodos Teóricos:

Analítico - Sintético: Permitirá organizar el cúmulo de información obtenidas de varias fuentes bibliográficas, seleccionando la documentación más útil para el desarrollo del tema. Se realizará un estudio de los mecanismos y estándares de seguridad existentes referidos a las normativas de los pasaportes-e, además de una profunda búsqueda de las tecnologías, herramientas y metodologías a utilizar en el desarrollo de la aplicación.

Inductivo - Deductivo: Este método apoyará el estudio del estado del arte, posibilitando la obtención de conocimientos generales, y resumir a lo particular de la solución.

Modelación: Permitirá hacer una representación, mediante la elaboración de diagramas del ambiente de la aplicación, de los componentes que serán utilizados y de los datos que se manejarán.

Métodos Empíricos:

Entrevista: Permitirá llevar a cabo reuniones con varios profesores del centro para obtener su experiencia en el tema, con el fin de aumentar el conocimiento sobre las medidas de seguridad de los pasaportes-e.

La presente investigación pretende arrojar como resultados un sistema encargado de verificar la autenticación pasiva de los pasaportes-e. Tributando a fortalecer el control de la seguridad de los pasaportes-e en el sistema de control migratorio de la República Bolivariana de Venezuela.

El trabajo estará dividido en 3 capítulos, a continuación un breve resumen de los mismos.

Capítulo 1: Fundamentación teórica.

En este capítulo se realizará la fundamentación teórica de la investigación relacionada con la ejecución de la autenticación pasiva en los pasaportes-e, profundizando en los conceptos, definiciones, metodologías y estándares que serán usados para darle solución al problema planteado. Se realizará el estudio del estado del arte para analizar las soluciones ya desarrolladas y el análisis de las herramientas más factibles para desarrollar la aplicación.

Capítulo 2: Propuesta de solución.

En este capítulo se realizará el análisis y diseño de la solución cumpliendo con los patrones de diseños y de arquitectura que se seleccionaron para el desarrollo de la aplicación. Se generarán todos los artefactos necesarios durante el ciclo de vida del proceso de desarrollo, especificando las características del sistema, desglosadas en escenarios y requisitos de calidad de servicio. Se presentarán además los diagramas de diseño que se acogen a la arquitectura y de clases, también los componentes y subsistemas especificados en el diseño.

Capítulo 3: Implementación y pruebas.

Este capítulo describirá las características de la solución propuesta, cumpliendo con los estándares de codificación para el desarrollo del sistema, se representarán además los diagramas de despliegue y de componentes de la solución. Se efectuarán las pruebas pertinentes para validar el cumplimiento de los requerimientos de la solución.

Capítulo 1: Fundamentación Teórica.

1.1. Introducción.

En este capítulo se realizará un estudio de los principales conceptos, necesarios para comprender el problema de la investigación. Se abordarán las definiciones de verificación de los SOD en pasaportes-e y de los repositorios de certificados existentes, imprescindibles para comprender el objetivo y alcance de la aplicación a desarrollar. Se realizará el estudio del estado del arte para analizar las ventajas y desventajas de las soluciones existentes. Se analizarán las principales metodologías de desarrollo de software, se seleccionará la que más se ajuste a las características del sistema y del equipo de trabajo. Se seleccionarán las herramientas y tecnologías que más se adecúen para el desarrollo de la solución.

1.2. Pasaporte electrónico.

El pasaporte electrónico también conocido como pasaporte-e, surge por la necesidad de incluir datos biométricos de los portadores, es un documento de alta tecnología de lectura mecánica que contiene un circuito integrado donde son guardados dichos datos, como lo especifica la OACI. Contiene la información biométrica necesaria para autenticar la identidad de viajeros. Los métodos actualmente estandarizados usados para estos tipos de sistemas de identificación son reconocimiento facial, reconocimiento de la huella dactilar, y reconocimiento del iris, pero de todos ellos solamente la imagen digital es la que se almacena obligatoriamente en el microprocesador. La comparación de características biométricas es realizada fuera del microprocesador de los pasaportes-e por los SI.

El ICC se encarga del almacenamiento y tratamiento de datos, que se incorpora a documentos de identidad, pasaportes y permisos de residencia biométricos. Para almacenar datos biométricos en el microprocesador sin contacto, existe un mínimo de 32 kilobytes de la memoria de almacenamiento, la interfaz cumplirá con el estándar internacional de ISO/IEC 14443, que está pensado para permitir la interoperabilidad de los pasaportes-e entre los diferentes países.[7]

Este medio electrónico puede contener datos personales (nombre, fecha de nacimiento, lugar de nacimiento) y una imagen digitalizada de la fotografía del titular. La forma en que están organizados dichos datos es llamada estructura lógica de datos, esta se compone de 16 grupos de datos algunos de

inclusión obligatoria y otros no. Para garantizar la seguridad de la información contenida en estos documentos es primordial el cumplimiento de las medidas de seguridad establecidas por la OACI, estas pueden ser implementadas de manera obligatoria u opcional, la implantación de las medidas opcionales dependen de las especificaciones internas del país o las especificaciones bilaterales convenidas entre los diferentes países que compartan dicha información.[7]

1.3. Medidas de seguridad.

1.3.1. Opcionales.

- ✓ Control de acceso básico al DVLM.

El control de acceso básico procede a leer la información óptica o visual, es utilizada para generar llaves para autenticarse y luego establecer un canal de comunicación seguro, posee la ventaja de proteger contra lecturas no autorizadas.

- ✓ Autenticación activa.

La autenticación activa, prueba que el chip es auténtico fundamentalmente a partir de la realización positiva mediante un protocolo reto-respuesta a partir de un par de llaves pública (almacenada en el DG17) y privada (almacenada en la zona segura del ICC), previene contra la sustitución del ICC, el mecanismo se aplica para asegurar que los datos que se leen del chip sean genuinos y que éste y la página de datos se corresponden.

- ✓ Control de acceso extendido.
- ✓ Descifrado de características biométricas.

El control de acceso extendido y cifrado de datos, están asociadas a la seguridad de las características biométricas adicionales que contiene el documento, y su implementación está determinada por la decisión de la nación emisora del documento de viaje electrónico.[3]

Este trabajo hará referencia a la única medida de seguridad de ejecución obligatoria que es la autenticación pasiva.

1.3.2. Obligatoria.

- ✓ Autenticación pasiva.

La autenticación pasiva prueba que el contenido del SOD y la LDS sean auténticos y no se hayan modificado. Verificando la autenticidad e integridad del documento con acciones para asegurar que la persona en posesión de un documento de viaje es el legítimo poseedor.

Autenticidad: Capacidad de confirmar que la LDS y sus componentes fueron creados por el estado u organización expedidor.

Integridad: Capacidad de confirmar que la LDS y sus componentes, no han sido alterados con respecto a los creados por el estado u organización expedidor.

La autenticación pasiva consiste en almacenar en el SOD la firma digital de los datos contenidos en el chip del pasaporte-e. El sistema de inspección que contenga la llave pública asociada al firmante del documento, contenida en el certificado del firmante del documento (CDS, por sus siglas en inglés), podrá verificar la firma del documento, validando así la autenticidad de los datos almacenados en la LDS.[3]

Para ejecutar la autenticación pasiva en pasaportes-e deben seguirse los siguientes pasos:

1. Se obtiene el SOD del lCC.
2. Se extrae el firmante del documento del SOD.
3. Se obtiene del SOD la firma digital de los datos y es verificada utilizando la llave pública del firmante del documento obtenida del CDS. Los certificados deben estar almacenados en el sistema de inspección u obtenerse del SOD donde estará registrado opcionalmente. Estos CDS deben ser válidos (emitidos por el firmantes de documento de país) y no estar revocados (no encontrarse en las listas de revocación). De este modo se comprueba que el SOD es auténtico y no ha sido cambiado. Este paso se describe con detalles en la Figura 1.
4. El SI lee los datos a verificar del LDS.
5. El SI aplica la función resumen a los datos leídos del LDS y compara su resultado con el valor correspondiente en el SOD, asegura que los datos son íntegros y no han sido cambiados.[3]

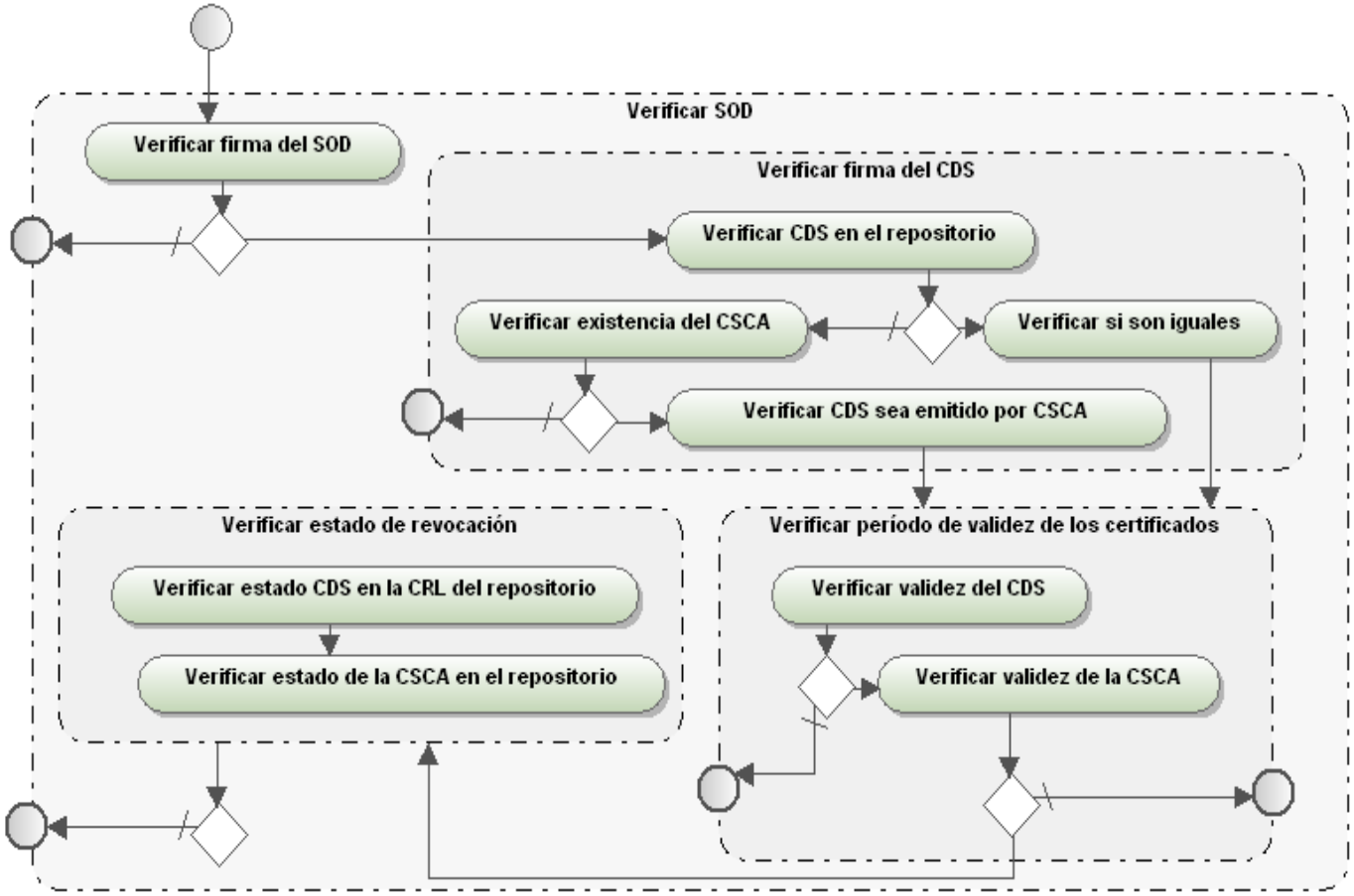

Figura 1 Pasos para verificar autenticidad de pasaportes-e.

El SOD es una estructura de datos firmada por el firmante de documento, que lleva los grupos de datos de la LDS condensado, está almacenado en el chip del DVLM y puede llevar opcionalmente el CDS.

Este CDS es el equivalente digital de un documento de identidad. Un certificado digital es un documento emitido por una autoridad certificadora a solicitud de una autoridad de registro, ésta es una organización fiable que acepta solicitudes de certificados las valida, genera certificados y mantiene la información de su estado garantizando la veracidad de los datos contenidos, referentes a una persona física o jurídica.[8].

Los CDS deben cumplir con la norma establecida X.509 para mantener un orden o estándar de los mismos.

1.4. Formato de certificados X.509

El formato de certificados X.509 es un estándar de la Unión Internacional de Telecomunicación y el ISO/IEC (*International Standards Organization* / *International Electrotechnical Commission*) que se publicó por primera vez en 1988. El formato de la versión 1 fue extendido en 1993 para incluir dos nuevos campos

que permiten soportar el control de acceso a directorios. Después se decide emplear el X.509 versión 2 para intentar desarrollar un estándar de correo electrónico seguro, el formato fue revisado para permitir la extensión con campos adicionales, dando lugar al X.509 versión 3 que es el usado actualmente, fue publicado en 1996[9].

Elementos fundamentales del formato de un certificado X.509:

- **Versión**: El campo de versión contiene el número de versión del certificado codificado, valores aceptables son 1, 2 y 3.
- **Número de serie del certificado:** Este campo es un entero asignado por la autoridad certificadora. Cada certificado emitido por una CSCA debe tener un número de serie único.
- **Identificador del algoritmo de firmado**: Este campo identifica el algoritmo empleado para firmar el certificado (como por ejemplo el RSA o el DSA).
- **Nombre del emisor**: Este campo identifica la CSCA que ha firmado y emitido el certificado.
- **Período de validez**: Este campo indica el período de tiempo durante el cual el certificado es válido. El campo contendrá la fecha inicial en la que el certificado comienza a ser válido y la fecha después de la cual el certificado deja de serlo.
- **Nombre del sujeto**: Este campo identifica la identidad cuya clave pública está certificada en el campo siguiente. El nombre debe ser único para cada entidad certificada por una CSCA dada, aunque puede emitir más de un certificado con el mismo nombre si es para la misma entidad.
- **Información de clave pública del sujeto**: Este campo contiene la clave pública, sus parámetros y el identificador del algoritmo con el que se emplea la clave.
- **Identificador único del emisor:** Este es un campo opcional que permite reutilizar nombres de emisor.
- **Identificador único del sujeto**: Este es un campo opcional que permite reutilizar nombres de sujeto. [9]

1.5. Directorio de llaves públicas de la OACI.

Para comprobar la veracidad del pasaporte-e es imprescindible la verificación del SOD, la misma solo es efectiva si el país receptor del pasaporte-e cuenta con los CDS que son utilizados para verificar la integridad del SOD, y se obtienen por intercambios bilaterales entre los países o por medio del directorio

de llaves públicas de la OACI (PKD, por sus siglas en inglés). Este se ha establecido para apoyar la interoperabilidad mundial de los pasaportes-e, y actuar como un agente central que gestiona el intercambio de certificados y listas de revocación de certificados (CRL, por sus siglas en inglés), juega un papel fundamental para minimizar el volumen de certificados que se intercambian y gestionar el cumplimiento de normas técnicas.[10]

Lo que se descarga del PKD de la OACI es un archivo de directorio de formato de intercambio ligero (LDIF, por sus siglas en inglés), es un formato que se utiliza para la importación y exportación de datos independientemente del servidor de Protocolo Ligero de Acceso a Directorios (LDAP, por sus siglas en inglés) que se esté utilizando. Cada servidor LDAP tiene una o varias maneras de almacenar físicamente sus datos en el disco rígido, es por esto que el formato LDIF provee una manera de unificar la manera de tratar los datos y así poder migrar de un servidor a otro. El formato LDIF es un formato de texto ASCII[1] para entradas LDAP[11], que contiene las CRL, los CDS válidos y las listas maestras o (Masters List).

Las CRL son un mecanismo mediante el cual la autoridad certificadora (CA, por sus siglas en inglés), publica y distribuye información acerca de los certificados anulados a las aplicaciones que los emplean. Cuando una aplicación trabaja con certificados debe obtener la última CRL de la entidad que firma el certificado que está empleando y comprobar que su número de serie no está incluido en la lista.[12]

1.6. Sistemas de verificación de autenticación pasiva.

Existen sistemas que contribuyen a mantener el control de las personas que viajan a los diferentes países, los cuales ocupan un papel importante en el progreso tecnológico en cuanto a la inspección de documentos de viajes, se encuentran a nivel internacional:

SACOM

El Sistema Automatizado de Control Migratorio (SACOM) es un novedoso sistema de procesamiento de viajeros en las fronteras que permite el uso de tarjetas inteligentes y pasaportes-e.

El SACOM consiste en un lector de tarjeta inteligente con o sin contacto y un lector sin contacto para pasaportes-e OACI. Estos lectores están conectadas a una puerta de barrera, estación de trabajo cliente

[1] **ASCII:** Código Estándar para el Intercambio de Información, se basa en un conjunto de caracteres del alfabeto con el objetivo de controlar dispositivos digitales que manipulan texto o para representar textos en pantalla.

para control fronterizo, servidor de manejo de control fronterizo y una infraestructura de red de comunicación[13].

Luego de la captura exitosa de los datos y autentificación positiva de la tarjeta o pasaporte, el sistema proveerá entrada al viajero mientras crea una entrada en la base de datos central con los detalles de salida/entrada del viajero. Durante el proceso de verificación, el sistema automáticamente ejecuta una revisión de las listas de atención biográfica y biométrica, alertas visuales y auditivas aparecerán en el sistema cliente y en la puerta de barrera.

Los datos tomados del chip serán validados contra la base de datos de la lista de atención. Si no hay una comparación exitosa contra la lista de atención, el portador del documento electrónico procederá a la captura y verificación de las biometrías relevantes. Esta verificación biométrica es una comparación uno a uno entre la biometría guardada en el pasaporte y la capturada en vivo[13].

El sistema ha sido desplegado con gran éxito en Malasia desde Agosto del 2000. Presentemente ha sido entregado e implantado el sistema SACOM en 11 puntos de entrada y salida en todo el país, no solo ha sido crucial en la reducción del personal migratorio necesario para operar las fronteras, sino que también ha reducido el tiempo promedio de procesamiento de pasaportes. Esto en conjunto con la verificación de firma digital, y la comparación rápida de las listas de atención y la verificación biométrica de identidad, se suma para hacer que los sistemas fronterizos sean más efectivos.

Se concluye que el sistema posee las características primordiales para la verificación de los pasaportes-e, pero es una de las herramientas en la actualidad que se rigen por licencias privativas de altos costos a las cuales no se tiene alcance.

D SCAN Master

El software D SCAN Master habilita a las lectoras de documentos *Authenticator* para escanear y verificar la autenticidad de las tarjetas y otras credenciales. El software provee las rutinas necesarias para procesar imágenes ópticas, acceder a información RFID[2] y funciones criptográficas, tanto como otros flujos de trabajo para inspección de documentos[14].

[2] **RFID:** (Identificación por radiofrecuencia) es un sistema de almacenamiento y recuperación de dato remoto que usa dispositivos denominados etiquetas, transponedores. El propósito fundamental de la tecnología RFID es transmitir la identidad de un objeto (similar a un número de serie único) mediante ondas de radio. Una etiqueta RFID es un dispositivo pequeño, similar a una pegatina, que puede ser adherida o incorporada a un producto, animal o persona. Contienen antenas para permitirles recibir y responder a peticiones por radiofrecuencia desde un emisor-receptor.

Características:

- El componente para el flujo de trabajo de inspección, procesamiento óptico y RFID, es controlado por una base de datos para documentos genérica y expandible, para la inspección y lectura de documentos, e identificación para viajeros con o sin chip RFID que cumple con OACI Doc. 9303, incluyendo verificación de dígitos en la zona de lectura mecánica y cheques con reacción a luz ultravioleta.
- Funcionalidad criptográfica integrada para lectura de chips empotrados en documentos electrónicos de identificación y para viajeros.
- Soporte para características especiales de seguridad, incluyendo código de barras, imágenes ocultas y otras líneas de agua digitales[15].

Funciones

- Verificación de redundancias en los datos (checksum) en la zona de lectura mecánica (MRZ, por sus siglas en inglés) de acuerdo a OACI Doc. 9303.
- Funcionalidad criptográfica para acceso a datos almacenados en chips con Autenticación Pasiva (PA, por sus siglas en inglés), Control Básico de Acceso (BAC, por sus siglas en inglés), Autenticación Activa (AA, por sus siglas en inglés), y protocolos para Control de Acceso Extendido (EAC, por sus siglas en inglés).
- Definición de documento dependiente de la extracción de datos textuales desde la Zona de Inspección Visual (VIZ, por sus siglas en inglés) de un documento[15].

Se concluye que este es un sistema líder entre las aplicaciones y servicios para la administración biométrica de identidades, posee las condiciones fundamentales para la verificación de los pasaportes-e, es creado por la compañía Cross Match Technologies de Estados Unidos por lo que posee licencia de alto costo a la cual no se tiene alcance.

Golden Reader

El *Golden Reader* es una aplicación de referencia para la lectura de documentos electrónicos de identificación. Es utilizada en todo el mundo para verificar la correcta aplicación y especificaciones técnicas durante la expedición de documentos de identificación electrónica, prueba además la interoperabilidad de los documentos de identidad de varias naciones. La base para la interoperabilidad de

los pasaportes electrónicos requiere soluciones estandarizadas, que cumplan con las necesidades de alta seguridad para cualquier país.

Los mecanismos de seguridad para documentos de viaje electrónicos que ejecuta este sistema son:

- La firma digital para verificar la autenticidad e integridad de los datos almacenados en el chip de radiofrecuencia.
- Para imágenes digitales del rostro el control de acceso básico.
- Las huellas digitales son verificadas por el control de acceso extendido.

Todas estas medidas realizadas de manera obligatoria.[16]

Es por ello que se reconoce el sistema Golden Reader como uno de los sistemas más potentes dedicados a la realización de verificaciones de los mecanismos de seguridad de los pasaportes-e, pero funciona con licencias privativa.

1.7. Tecnologías, Metodologías y Lenguajes.

1.7.1. Tecnologías.

Plataforma de Desarrollo .NET.

La plataforma .NET es el conjunto de tecnologías en las que Microsoft ha estado trabajando durante los últimos años con el objetivo de obtener una plataforma sencilla y potente para distribuir el software, que puedan ser suministrados remotamente y que puedan comunicarse y combinarse unos con otros de manera totalmente independiente de la plataforma, el lenguaje de programación y el modelo de componentes con los que hayan sido desarrollados.[17]

La plataforma .NET constituye un entorno para la construcción, desarrollo y ejecución de servicios web y otros tipos de aplicaciones. A continuación se resumen las ventajas más importantes que proporciona .NET:

- Código administrado: El CLR[3] controla los recursos del sistema para que la aplicación se ejecute correctamente a través de un control automático del código.

[3] **Common Language Runtime** (**CLR**): es el núcleo de la plataforma .NET. Es el motor encargado de gestionar la ejecución de las aplicaciones para ella desarrolladas y a las que ofrece numerosos servicios que simplifican su desarrollo y favorecen su fiabilidad y seguridad.

- Interoperabilidad multilenguaje: El código puede ser escrito en cualquier lenguaje compatible con .NET.
- Compilación JIT[4]: Compila el código intermedio a través del compilador JIT (Just in time) generando el código máquina, propio de la plataforma, aumentando así el rendimiento de la aplicación.
- Recolector de basura: Ocurre una vez que el CLR detecta cuando el programa deja de utilizar la memoria y la libera automáticamente, posibilitando que el programador no tenga que liberar la memoria de forma explícita aunque también es posible hacerlo manualmente.
- Seguridad de acceso al código: Es posible aplicar distintos niveles de seguridad al código, de forma que se puede ejecutar código procedente de la web sin tener que preocuparse si esto va a estropear el sistema, además posibilita declarar que una pieza de código tenga permisos de lectura pero no de escritura.[17]

LDAP

Lightweight Directory Access Protocol (LDAP, por sus siglas en inglés), en español Protocolo Ligero de Acceso a Directorios es un protocolo de tipo cliente-servidor para acceder a un servicio de directorio. LDAP funciona sobre TCP/IP u otros servicios de transferencia orientados a conexión y está basado en el estándar X500.[18]

Directorio LDAP

Un directorio LDAP es una base de datos optimizada para la lectura y búsqueda de información que es almacenada de manera jerárquica. Los directorios soportan opciones avanzadas de filtrado. Generalmente no soportan transacciones complejas que sí ofrecen los sistemas de gestión de bases de datos diseñadas para procesar un gran volumen de actualizaciones. Los cambios en la información almacenada en un directorio suelen ser del tipo "o todo o nada", es decir, cambios de las ramas del árbol de directorio de información (DIT[5], por sus siglas en inglés) completamente, pero, aunque no estén optimizados para ello, los directorios pueden permitir cambios muy específicos. Los directorios están preparados para dar una respuesta rápida a un gran volumen de búsquedas. Disponen de mecanismos de replicación de la

[4] **Compilación en tiempo de ejecución:** (JIT, just-in-time) La compilación JIT proporciona optimización específica para el ambiente, seguridad de tipos en tiempo de ejecución, y verificación de ensamblados. Para cumplir con esto, el compilador JIT examina los metadatos del ensamblado en busca de accesos ilegales y maneja las violaciones apropiadamente.

[5] **DIT (Directory Information Tree):** Árbol de información de directorio. Es la estructura jerárquica donde se almacena la información en el LDAP en forma de árbol.

información en varios servidores para de incrementar la disponibilidad y fiabilidad del servicio mientras se reduce el tiempo de respuesta.[18]

Un directorio es una base de datos, pero que en general contiene información más descriptiva y basada en atributos de usuarios y recursos de red. Permite hacer una preselección de las políticas mediante la selección de las ramas en función de la información de la red antes de hacer una búsqueda. Ofrece un control dinámico y coordinado de los elementos de la red ya que las decisiones se toman de manera automática basándose en reglas, peticiones de usuarios o de servicios. Esta gestión dinámica de los elementos de la red representa un cambio cualitativo, desde el punto de vista empresarial, ya que permite la gestión de los recursos de la red y de los servicios de manera más eficiente. Ver Figura 2.

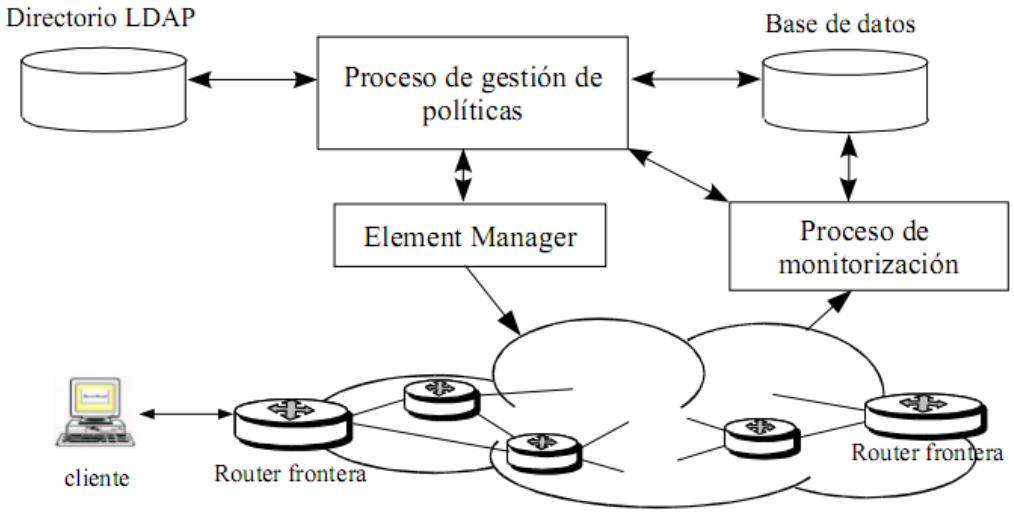

Figura 2 Esquema del directorio LDAP para la gestión de recursos.[18]

Open LDAP

Open LDAP es una aplicación de servidor LDAP, proporciona la funcionalidad de Servicios de Directorio, tales servicios incluyen gestionar las identidades y las relaciones entre los ordenadores, usuarios y grupos de ordenadores o usuarios que participan en la red, y proporcionan una forma consistente de describir, localizar y gestionar esos recursos. Ver Figura 3.

El servidor open LDAP posibilita:

- ✓ Implementación del protocolo LDAPv3 que está definido para que permita el uso de IPv4 e Ipv6.

- ✓ Autentificación y nivel de seguridad que permite servicios de autentificación mediante capa de seguridad y autenticación simple (SASL, por sus siglas en inglés). El acceso al directorio ha de ser restringido ya que contiene información de usuarios (nombres de usuarios y contraseñas), información de las políticas y parámetros. Por esta razón sólo el administrador del directorio tiene acceso a la información.
- ✓ Control de acceso: Open LDAP permite definir una serie de filtros para el control de acceso a diferentes DITs, entradas, o atributos de estas entradas.
- ✓ Es totalmente gratuito bajo licencia Open Source.
- ✓ El servidor Open LDAP hace una implementación Open Source del protocolo LDAP y requiere un motor para almacenar y hacer búsquedas de la información. [18]

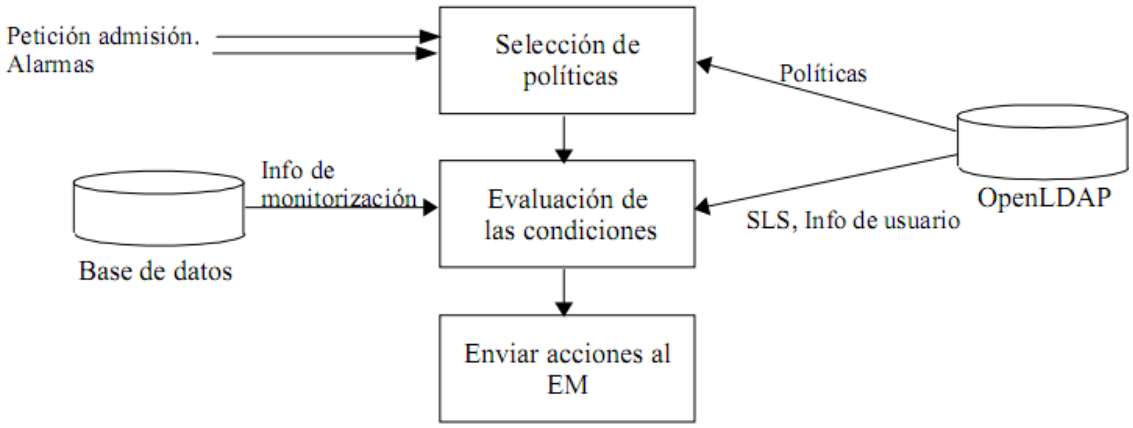

Figura 3 Esquema del sistema Open LDAP para la gestión de políticas.[18]

Active Directory

Active Directory (AD, por sus siglas en inglés) es el término que usa Microsoft para referirse a su implementación de servicio de directorio en una red distribuida de computadores. Utiliza distintos protocolos (principalmente LDAP, DNS[6], DHCP[7]) y está basado en los estándares X.500. Su estructura jerárquica permite mantener una serie de objetos relacionados con componentes de una red, como usuarios, grupos de usuarios, permisos y asignación de recursos y políticas de acceso.

6 **DNS**: Servidor de nombres de dominio.

7 **DHCP**: Protocolo de configuración dinámica de host.

AD es una implementación propietaria creada por Microsoft de los Servicios de Directorio, y proporciona una manera de compartir información entre recursos y usuarios de la red. Además de proporcionar una fuente centralizada para esa información, *Active Directory* también funciona como autoridad de seguridad centralizada de autenticación para la red. AD permite a los administradores establecer políticas a nivel de empresa, desplegar programas en muchos ordenadores y aplicar actualizaciones críticas a una organización entera. Un *Active Directory* almacena información de una organización en una base de datos central, organizada y accesible. Pueden encontrarse desde directorios con cientos de objetos para una red pequeña hasta directorios con millones de objetos. Combina capacidades que tradicionalmente se hallaban en sistemas separados y especializados de directorio, como integración simplificada, gestión y seguridad de los recursos de la red. El paquete SAMBA puede configurarse para usar los servicios de *Active Directory* desde un controlador de dominio de Windows.[19]

Su funcionamiento es similar a estructuras de LDAP, ya que este protocolo viene implementado de forma similar a una base de datos y almacena en forma centralizada toda la información relativa a un dominio de autenticación. La ventaja que presenta esto es la sincronización presente entre los distintos servidores de autenticación de todo el dominio.

La incorporación del Directorio Activo de Microsoft, como solución comercial del servidor de Directorio es hoy muy difundida en las organizaciones, sobre todo porque proporciona herramientas gráficas de administración y configuración del directorio muy fáciles de utilizar; pretendiendo aislar al administrador de la red, de los conocimientos avanzados acerca de los protocolos y estándares que implementa. Pero la sustitución del *Active Directory* por Open Ldap ofrece numerosas ventajas, no sólo por ser este un software amparado por licencias de libre distribución, sino por ser un sistema altamente configurable, que permite personalizar una organización en la medida exacta a través del directorio. Además que los dominios y subdominios se identifican con AD utilizando la misma notación de las zonas DNS, razón por la cual AD requiere uno o más servidores DNS que permitan el direccionamiento de los elementos pertenecientes a la red.[19]

Se concluye que se escoge el servidor Open LDAP por ser una herramienta libre y poseer todas las funcionalidades necesarias para la gestión de los certificados.

Librería Bouncy Castle

Bouncy Castle (BC, por sus siglas en inglés) es una implementación Java de algoritmos criptográficos. Está organizado de forma que sus interfaces de programación de aplicaciones (API, por sus siglas en inglés) sean adaptables para su uso bajo cualquier entorno de infraestructura adicional. El paquete se distribuye bajo licencia privativa.

Bouncy Castle nació como resultado del esfuerzo de dos colegas que sufrieron la necesidad de recrear librerías criptográficas en cada cambio de empleo. Un requisito de diseño inicial fue que existieran versiones de la librería para el entorno Java por lo que existen 2 juegos de librerías. La librería compatible JCE se basa en API de bajo nivel, de modo que el código fuente es un ejemplo de implementación de problemas criptográficos comunes con dichas API. Estas están optimizadas para gestionar eficientemente los algoritmos criptográficos, de forma que se puedan usar en entornos de bajos recursos (JavaME). La API de peso ligero trabaja con todo, desde el J2ME para el JDK 1.6 y también hay una API en C # que proporciona una funcionalidad equivalente. [20]

El Castillo Hinchable Criptográfico, como se le conoce a esta librería, tiene como características:

- ✓ Peso ligero de la API de criptografía de Java y C #.
- ✓ Un proveedor de Java Cryptography Extension y la arquitectura Java Cryptography.
- ✓ Biblioteca que permite leer y escribir objetos codificados ASN.1.
- ✓ Peso ligero del lado del cliente TLS API.
- ✓ Generadores para la versión 1 y versión 3 de los certificados X.509, versión 2 CRL y archivos PKCS12.
- ✓ Generadores para la versión 2 X.509 certificados de atributos.
- ✓ Generadores, transformadores de S / MIME y CMS (PKCS7/RFC 3852).
- ✓ Generadores, transformadores de OCSP (RFC 2560).
- ✓ Generadores, transformadores de TSP (RFC 3161 y RFC 5544).
- ✓ Generadores, transformadores de CMP y CRMF (RFC 4210 y RFC 4211).
- ✓ Generadores, transformadores de Open PGP (RFC 2440). [20]

1.7.2. Metodologías de desarrollo de software.

Según *Pressman*: “Una metodología de desarrollo de software es un conjunto de reglas e instrucciones relativas al proceso de desarrollo de software”. [21]

Rumbaugh, da una definición más abarcadora donde expresa: “Una metodología de ingeniería de software es un proceso para la producción organizada del software, empleando para ello una colección de técnicas predefinidas y convencionales en las notaciones. Una metodología se presenta normalmente como una serie de pasos, con técnicas y notaciones asociadas a cada uno de ellos. Los pasos de la producción del software se organizan normalmente en un ciclo de vida consistente en varias fases de desarrollo”. [22]

Las metodologías son en fin una guía para los desarrolladores de software, donde definen artefactos, roles e indicando paso a paso las actividades que se van a realizar mediante prácticas y técnicas recomendadas. Su principal objetivo es lograr un producto final de alta calidad y que cumpla con las especificaciones que pide el usuario, pero los requisitos son tan diversos y cambiantes, que han dado lugar al surgimiento de una variada cantidad de metodologías de desarrollo, que atendiendo a sus particularidades, es posible clasificarlas en dos grandes grupos: robustas y ágiles.

1.7.2.1. Metodologías robustas

Las metodologías robustas o pesadas están orientadas al control de los procesos, detallan rigurosamente tanto las tareas y actividades del equipo de desarrollo como los artefactos de software. Establecen rigurosamente las actividades a desarrollar, herramientas a utilizar y notaciones que se usarán. Son las más tradicionales y altamente recomendadas para proyectos grandes y complejos, donde se requiere de una gran organización.

Dentro de las metodologías pesadas se encuentra el Proceso Unificado de Desarrollo (RUP, por sus siglas en inglés), que por ser la más completa constituye un ejemplo académico cuando se quieren estudiar estas metodologías. RUP cuenta con cuatro fases de trabajo (Inicio, Elaboración, Construcción y Despliegue) divididas en flujos de trabajo, esta estructura hace que el desarrollo con RUP sea dirigido por casos de uso, centrado en la arquitectura e iterativo e incremental. [22]

Estas metodologías demandan de un numeroso equipo de proyecto y generan una gran cantidad de documentación, que en ocasiones resulta inconveniente para proyectos sencillos y con poco tiempo para

el desarrollo, como es el caso que se plantea. Por estas razones se decide desechar este tipo de metodología y analizar las ágiles.

1.7.2.2. Metodologías ágiles

El término “ágil” se adoptó tras una reunión desarrollada en Utah-EEUU en febrero del 2001, con la participación de 17 expertos del mundo de la industria de software. El resultado del intercambio quedó reflejado en el “Manifiesto ágil”, un documento que describe los fundamentos de las metodologías ágiles.[23]

Algunos de ellos son:

- Se encargan de valorar al individuo y las iteraciones del equipo más que a las herramientas o los procesos utilizados.
- Se hace mucho más importante crear un producto de software que funcione, que escribir mucha documentación.
- El cliente está en todo momento colaborando en el proyecto.
- Es más importante la capacidad para responder a un cambio realizado que el seguimiento estricto de un plan.[23]

Ventajas de las metodologías ágiles:

- Rápida respuesta a cambios en los requisitos a lo largo del proyecto.
- Entrega continua y en plazos cortos de software funcional.
- Trabajo conjunto entre el cliente y el equipo de desarrollo.
- Minimiza los costos frente a cambios.
- Importancia de la simplicidad, al eliminar el trabajo innecesario.
- Atención a la excelencia técnica y al buen diseño.
- Mejora continua de los procesos y el equipo de desarrollo.
- Evita malentendidos de requerimientos entre el cliente y el equipo.
- Cada componente del producto final será probado para ver si satisface los requerimientos.[23]

A continuación se valoran las características principales de las metodologías más conocidas y utilizadas en la actualidad.

1.7.2.2.1. Programación Extrema (XP por sus siglas en inglés)

XP es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. Se basa en la retroalimentación continua entre el cliente y el equipo de desarrollo, la comunicación fluida entre todos los participantes, la simplicidad en las soluciones implementadas y el coraje para enfrentar los cambios. Está concebida para proyectos con poco tiempo de desarrollo y equipos pequeños, con pocos roles, por lo que la programación se realiza en parejas. Propone que el diseño debe ser sencillo, que funcione con todas las pruebas, sin lógica duplicada y con el menor número de clases y métodos posible. [24]

1.7.2.2.2. Microsoft Solution Framework (MSF) for ASD.

Microsoft Solution Framework Agile se caracteriza por ser de planificación adaptable a cambios y orientada a las personas. Su proceso introduce ideas importantes del software ágil, junto con los principios y prácticas de MSF para CMMI como por ejemplo: admite una estrategia que utiliza múltiples iteraciones y un enfoque para la construcción de aplicaciones que se basa en escenarios. Además esta metodología incorpora prácticas para el manejo de la calidad del servicio (el rendimiento y la seguridad) y facilita la automatización y la orientación que se necesita para apoyar el equipo de trabajo, incluyendo la gestión de configuración y de proyectos.

La definición, desarrollo y prueba del producto se realizan en pequeñas iteraciones provenientes del proceso incremental del proyecto, reduciéndose así el margen de error en las estimaciones y proporcionándose información rápida acerca de la exactitud de los planes del proyecto.

Esta metodología soporta 17 flujos de trabajo básicos, en los cuales se agrupan diferentes actividades, e incluye además cinco fases para el desarrollo y seguimiento del producto, estas son: Visión y Alcance, Planificación, Desarrollo, Estabilización y Despliegue.[25]

MSF Ágil dispone de los tipos de elementos de trabajo siguientes:

- Escenario: Descripción de la necesidad o solicitud del usuario.

- Error: Defecto o desviación entre el comportamiento esperado y el comportamiento observado en el producto.
- Requisito de calidad de servicio: Material resultante esperado del producto final. El mismo puede ser un resultado, un problema resuelto o una característica, entre otros.
- Tarea: Acción independiente que debe realizar una persona o un grupo de personas.
- Riesgo: Evento o condición probable que puede dar resultados potencialmente negativos en el proyecto en el futuro. [25]

Ventajas:

- Proporciona siempre enfoque al usuario. Es decir, ayudándolo a asegurar que la solución implantada realmente es lo que el usuario necesita, que mejorará el desempeño del usuario, y no que va a ser desechada y olvidada al momento de su liberación.
- Proporciona elementos valiosos a las inevitables preguntas: ¿Cuáles elementos van en el cliente?, ¿Cuáles en el servidor?
- Permite desarrollar siguiendo una filosofía de reutilización de múltiples componentes, lo cual reduce el tiempo necesario para desarrollar nuevas aplicaciones, garantiza la uniformidad e interoperabilidad entre las mismas, y las hace mucho más flexibles para incorporar los cambios que sean necesarios en el futuro.
- Establece un equipo de trabajo balanceado, con tareas y objetivos claramente definidos, que permite no solo desarrollar buenos sistemas, sino también saber en todo momento cuál es el grado real de avance del proyecto, y cuáles son los riesgos que se corren si se decide introducir modificaciones al mismo una vez que el desarrollo ya ha sido iniciado. [25]

Argumento de metodología a utilizar:

Para decidir la metodología a seguir en el proyecto de desarrollo de software, se tuvieron en cuenta las características del mismo: complejidad, tiempo estimado de desarrollo, cantidad de personal. Una vez que estén bien definidos estos factores, se realiza un análisis preliminar para identificar cual se ajusta más al proyecto.

Se escoge la metodología MSF porque es un marco de trabajo de referencia para construir e implantar sistemas distribuidos basados en herramientas y tecnologías de Microsoft, por ejemplo la plataforma

.NET. Provee una estructura orientada a facilitar el análisis, diseño e implantación de soluciones tecnológicas efectivas. Este marco permite exponer, revelar y manejar riesgos críticos, determinar los criterios de planeación, y establecer las interdependencias necesarias para una ejecución exitosa de los proyectos. Dentro de este contexto, MSF no es una metodología en el sentido estricto, con estructuras de trabajo, tareas y productos predeterminados, sino que provee mecanismos flexibles para aplicar soluciones adecuadas a los problemas tecnológicos y de negocios. MSF no es un marco estático y evoluciona respondiendo a los cambios en la tecnología y en los requerimientos de los proyectos. Por la cantidad de personal (2) en dicho proyecto y la premura del mismo, se decidió seguir la metodología MSF ya que los miembros del equipo pueden tomar diferentes roles durante el ciclo de vida del proyecto.

1.7.3. Lenguajes.

1.7.3.1. Lenguaje de Programación.

Un lenguaje de programación es una construcción mental del ser humano para expresar programas. Está constituido por un grupo de reglas gramaticales, un grupo de símbolos utilizables, un grupo de términos con sentido único y una regla principal que resume las demás. Para que esta construcción mental sea operable en un computador debe existir otro programa que controle la validez o no de lo escrito. A este se le llama traductor.

A continuación se mencionarán los candidatos estudiados en función de seleccionar el más factible.

C Sharp o C#

C Sharp es el lenguaje de propósito general diseñado por Microsoft para su plataforma .NET. Aunque es posible escribir código para esta plataforma en muchos otros lenguajes, C# es el único que ha sido diseñado específicamente para ser utilizado en ella, por lo que usarlo es mucho más sencillo e intuitivo que hacerlo con cualquiera de los otros lenguajes ya que carece de elementos heredados innecesarios en .NET. Por esta razón, se suele decir que C# es el lenguaje nativo de .NET[26].

Características del lenguaje C#.

- **Es autocontenido:** Un programa en C# no necesita de ficheros adicionales al código fuente, como los ficheros de cabecera (.h) de C++, lo que simplifica la arquitectura de los proyectos software desarrollados con C++.

- **Es homogéneo:** El tamaño de los tipos de datos básicos es fijo e independiente del compilador, lo que facilita la portabilidad del código.
- **Es actual:** C# incorpora en el propio lenguaje elementos que se han demostrado ser muy útiles para el desarrollo de aplicaciones como el tipo básico decimal que representa valores decimales con 128 bits, lo que le hace adecuado para cálculos financieros y monetarios, incorpora la instrucción foreach, que permite una cómoda iteración por colecciones de datos, proporciona el tipo básico string, permite definir cómodamente propiedades (campos de acceso controlado), etc.
- **Está orientado a objetos:** C# soporta todas las características propias del paradigma de la programación orientada a objetos: encapsulación, herencia y polimorfismo.
    - Encapsulación: además de los modificadores de acceso convencionales: public, private y protected, añade el modificador internal, que limita el acceso al proyecto actual.
    - Herencia: sólo admite herencia simple.
    - Polimorfismo: los métodos son por defecto, sellados, y los métodos redefinibles han de marcarse obligatoriamente, con el modificador virtual.
- **Delega la gestión de memoria**. Como todo lenguaje de .NET, la gestión de la memoria se realiza automáticamente ya que tiene a su disposición el recolector de basura del CLR. Esto hace que el programador se desentienda de la gestión directa de la memoria (petición y liberación explícita) evitando que se cometan los errores habituales de este tipo de gestión.
- **Emplea un sistema de tipos unificado**. Todos los tipos de datos (incluidos los definidos por el usuario) siempre derivarán, aunque sea de manera implícita, de una clase base común llamada System.Object, por lo que dispondrán de todos los miembros definidos en ésta clase. Esto también es aplicable, lógicamente, a los tipos de datos básicos.
- **Proporciona seguridad con los tipos de datos**. C# no admite ni funciones ni variables globales sino que todo el código y datos han de definirse dentro de definiciones de tipos de datos, lo que reduce problemas por conflictos de nombres y facilita la legibilidad del código. Incluye mecanismos que permiten asegurar que los accesos a tipos de datos siempre se realicen correctamente:
    - No pueden usarse variables que no hayan sido inicializadas.
    - Sólo se admiten conversiones entre tipos compatibles
    - Siempre se comprueba que los índices empleados para acceder a los elementos de una tabla (vector o matriz) se encuentran en el rango de valores válidos.

    - Siempre se comprueba que los valores que se pasan en una llamada a métodos que pueden admitir un número indefinido de parámetros (de un cierto tipo) sean del tipo apropiado.
- **Proporciona instrucciones seguras**. En C# se han impuesto una serie de restricciones para usar las instrucciones de control más comunes. Por ejemplo, toda condición está controlada por una expresión condicional, los casos de una instrucción condicional múltiple (switch) han de terminar con una instrucción *break* o *goto*, etc.
- **Facilita la extensibilidad de los operadores**. C# permite redefinir el significado de la mayoría de los operadores, incluidos los de conversión, tanto para conversiones implícitas como explícitas cuando se aplican a diferentes tipos de objetos.
- **Apuesta por la compatibilidad**. C# mantiene una sintaxis muy similar a C++ o Java que permite, bajo ciertas condiciones, incluir directamente en código escrito en C# fragmentos de código escrito en estos lenguajes. [27]

En resumen, podemos concluir que:

- Es un lenguaje orientado al desarrollo de componentes ya que los componentes son objetos que se caracterizan por sus propiedades, métodos y eventos y estos aspectos de los componentes están presentes de manera natural en C#.
- En C# todo son objetos, desaparece la distinción entre tipos primitivos y objetos de lenguajes como Java o C++.
- El software es robusto y duradero, el mecanismo automático de recolección de basura, la gestión de excepciones, la comprobación de tipos, la imposibilidad de usar variables sin inicializar y hacer conversiones de tipo (castings) no seguras, ayudan a desarrollar software poco propenso a errores.
- Además, no hay que olvidar el aspecto económico, la posibilidad de utilizar C++ puro (código no gestionado o inseguro), la facilidad de interoperabilidad (XML, SOAP, COM, DLL) junto con un aprendizaje relativamente sencillo (para los que ya conocen otros lenguajes de programación) hace que el dominio y uso del lenguaje junto a otras tecnologías sea muy apreciado[28].

JAVA

Es un lenguaje orientado a objeto, de una plataforma independiente. Fue desarrollado por la compañía Sun Microsystems, con la idea original de usarlo para la creación de páginas WEB. Esta programación Java tiene muchas similitudes con el lenguaje C y C++, así que si se tiene conocimiento de este lenguaje, el aprendizaje de la programación Java será de fácil comprensión por un programador que haya realizado programas en estos lenguajes. Con la programación en Java, se pueden realizar distintos aplicativos, como son applets, que son aplicaciones especiales, que se ejecutan dentro de un navegador al ser cargada una página HTML en un servidor WEB, por lo general los applets son programas pequeños y de propósitos específicos. Otra de las utilidades de la programación en Java es el desarrollo de aplicaciones, que son programas que se ejecutan en forma independiente, es decir con la programación Java, se pueden realizar aplicaciones como un procesador de palabras, una hoja que sirva para cálculos, una aplicación gráfica, etc.

En resumen cualquier tipo de aplicación se puede realizar con ella. La programación en Java, permite el desarrollo de aplicaciones bajo el esquema de Cliente-Servidor, como de aplicaciones distribuidas, lo que lo hace capaz de conectar dos o más computadoras, ejecutando tareas simultáneamente, y de esta forma logra distribuir el trabajo a realizar. [29]

Características principales:

1.- Lenguaje Simple: Se lo conoce como lenguaje simple porque viene de la misma estructura de C y C++; ya que C++ fue un referente para la creación de java por eso utiliza determinadas características de C++ y se han eliminado otras.

2.- Orientado a Objeto: Toda la programación en java en su mayoría está orientada a objeto, ya que al estar agrupados en estructuras encapsuladas es más fácil su manipulación.

3.- Distribuido: Permite abrir sockets, establecer y aceptar conexiones con los servidores o clientes remotos; facilita la creación de aplicaciones distribuidas ya que proporciona una colección de clases para aplicaciones en red.

4.- Robusto: Es altamente fiable en comparación con C, se han eliminado muchas características como la aritmética de punteros, proporciona numerosas comprobaciones en compilación y en tiempo de ejecución.

5.- Seguro: La seguridad es una característica muy importante en java ya que se han implementado barreras de seguridad en el lenguaje y en el sistema de ejecución de tiempo real.

6.- Indiferente a la arquitectura: Java es compatible con los más variados entornos de red, cualesquiera sean éstos, desde *Windows 95, Unix a Windows Nt y Mac*, para poder trabajar con diferentes sistemas operativos. Java es muy versátil ya que utiliza byte-codes que es un formato intermedio que sirve para transportar el código eficientemente o de diferentes plataformas (Hardware - Software).
7.- Portable: Por ser indiferente a la arquitectura sobre la cual está trabajando, esto hace que su portabilidad sea muy eficiente, sus programas son iguales en cualquiera de las plataformas, ya que java especifica tamaños básicos.
8.- Interpretado y compilado a la vez: Java puede ser compilado e interpretado en tiempo real, ya que cuando se construye el código fuente este se transforma en una especie de código de máquina.
9.- Multihebra o Multihilos: Java tiene una facilidad de cumplir varias funciones al mismo tiempo, gracias a su función de multihilos ya que por cada hilo que el programa tenga se ejecutaran en tiempo real muchas funciones al mismo tiempo.
10.- Dinámico: El lenguaje java es muy dinámico en la fase de enlazado, sus clases solamente actuaran en medida en que sean requeridas o necesitadas con esto permitirá que los enlaces se puedan incluir incluso desde fuentes muy variadas o desde la red.
12.- Alto rendimiento: Java es considerado de alto rendimiento por ser veloz en el momento de correr los programas y por ahorrase muchas líneas de código. [30]

Argumento de lenguaje a utilizar:

Se selecciona el lenguaje C Sharp porque es nativo de la plataforma. NET, es un lenguaje robusto, orientado a objetos, que posee gestión de excepciones y comprobación de tipos, estos ayudan a tener un software menos propenso a errores, ya que muestra donde está el error para que sea corregido, es muy moderno y soporta el trabajo con varios lenguajes. Además porque será un sistema que se integrará con una aplicación ya realizada, se recomienda por cuestión de estandarización y política del proyecto utilizar el mismo lenguaje en que se desarrolló la anterior.

1.7.3.2. Lenguaje de Modelado.

UML.

Lenguaje Unificado de Modelado (UML): Lenguaje gráfico que especifica, construye, visualiza y documenta las partes o artefactos que constituyen la información utilizada y originada mediante un proceso de software, además es un lenguaje orientado a objetos.[31]

Principales beneficios de UML:

Modelar sistemas utilizando conceptos orientados a objetos.

- Establece conceptos y artefactos ejecutables.
- Permite encaminar el desarrollo del escalamiento en sistemas complejos de misión crítica.
- Establece soporte a la planeación y al control de proyectos.
- Permite alta reutilización y minimización de costos.[32]

Se puede emplear de diferentes formas para dar soporte a una metodología de desarrollo de software pero no especifica en sí que metodología o proceso utilizar. UML no se puede comparar con la programación estructurada, pues no es una programación, solo diagrama la realidad de un requerimiento. Mientras que, la programación estructurada es una forma de programar como lo es la orientada a objetos, sin embargo, la orientada a objetos es un complemento perfecto de UML, pero esto no quiere decir que UML se toma únicamente para lenguajes orientados a objetos.

1.8. Herramientas de Desarrollo.

1.8.1. Entorno de desarrollo.

Un Entorno de Desarrollo Integrado (IDE por sus siglas en inglés), es un programa donde es posible escribir código fuente, compilarlo y ejecutarlo sin tener que cambiar de aplicación. Algunos incluyen una herramienta para, hacer debug o depurar gráficamente. Permiten desarrollar las aplicaciones de forma mucho más rápida, incorporando en muchos casos librerías con componentes previamente implementados. [33]

A continuación se comparan diferentes entornos de desarrollo para escoger el que más se identifique con el trabajo.

Visual Studio 2010 Ultimate.

Visual Studio 2010 Ultimate añade nuevas características mejoradas que hacen que el proceso de desarrollo, desde el diseño hasta la implementación, sea más sencillo. Además, posibilita que los equipos puedan observar un mayor ahorro de costes y mayor productividad al hacer uso de características de colaboración avanzadas, así como herramientas de pruebas y depuración integradas que de una manera u otra ayudan a crear un código de gran calidad.

A continuación se muestran algunas características importantes:

- Administración del ciclo de vida de las aplicaciones (ALM)[8]: Contribuye a que las organizaciones colaboren y se comuniquen de forma efectiva en todos los niveles, y a que tengan una idea precisa del estado real del proyecto, garantizando que se ofrezcan soluciones de gran calidad al tiempo que se reducen los costos.
- Depuración y diagnóstico: Se pueden archivar errores enriquecidos y modificables. Además incluye análisis de código estático, métricas de código y creación de perfiles.
- Arquitectura y modelado: La arquitectura ayuda a entender los activos de código existentes y otras interdependencias. Los diagramas por capas ayudan a garantizar el cumplimiento de la arquitectura y permiten validar artefactos de código con respecto al diagrama. Además, *Visual Studio 2010 Ultimate* admite los cinco diagramas de UML más comunes que conviven junto con su código.
- Desarrollo de bases de datos: Visual Studio 2010 Ultimate proporciona potentes herramientas de implementación y administración de cambios que garantizan que la base de datos y la aplicación estén siempre sincronizadas.[33]

Fundamentación del IDE a utilizar:

Se escogió el IDE Visual Studio 2010 Ultimate para cumplir con el patrón que se viene desarrollando desde el inicio, ya que se utiliza la plataforma .NET y el lenguaje de programación C Sharp. Además por ser un software potente, seguro y sus características hacen más fácil el proceso de desarrollo.

---

[8] **ALM:** Application Lifecycle Management.

1.8.2. Herramienta de Modelado.

Altova UModel 2010.

Altova UModel 2010 genera códigos C#, se utiliza para diseñar modelos de aplicaciones en UML y documenta el proyecto. Con esta herramienta se puede realizar una ingeniería inversa a los programas existentes pasándolos a diagramas UML, luego afina los diseños y termina generando el código a partir de estos. Además es la manera simple de dibujar diagramas en UML, pues incluye funcionalidades para proveer a los usuarios de ventaja en el desarrollo de software y combina una interfaz visual agradable con funcionalidades de usabilidad superiores para ayudar a suavizar la curva de aprendizaje de UML.

Características principales:

- Soporta 14 tipos de diagramas UML.
- Modela esquemas XML en diagramas UML.
- Diagrama el proceso de negocio.
- Genera un código fuente en lenguajes Java, C#, y VB.NET.
- Ingeniería inversa de código fuente y ficheros binarios Java, C# y VB.NET.
- Sincroniza el modelo y el código a través de ingeniería de ida y vuelta.
- Crea diagramas de secuencia desde el código fuente de la ingeniería inversa.
- Genera documentación personalizable del proyecto.
- Se integra con sistemas de control de versiones.
- Estrecha integración con Visual Studio y Eclipse. [34]

1.9. Conclusiones.

- El estudio de los conceptos relacionados con el proceso de verificación de autenticación pasiva y el análisis de los sistemas similares existentes, demostró que existen soluciones que cumplen con la realización de dicho proceso pero pertenecen a compañías privativas y el costo de adquisición del producto es alto, de ahí la propuesta de realizar un sistema que dé solución a los problemas del sistema de control migratorio para la República Bolivariana de Venezuela.
- El análisis de las diferentes metodologías, tecnologías y lenguajes según las necesidades de la solución, determinó la utilización de la metodología MSF Ágil, UML como lenguaje de modelado y

la herramienta Altova UModel como herramienta de modelado. Como tecnología la plataforma .NET y el lenguaje de programación C Sharp, usando el IDE de desarrollo Visual Studio 2010 Ultimate.

Capítulo 2: Propuesta de Solución.

2.1. Introducción.

En el presente capítulo se realizará la descripción del problema a resolver, y partiendo del exhaustivo análisis de la problemática se dará la propuesta de solución al mismo, obteniendo así una visión precisa de cómo será el funcionamiento de la aplicación. Se procederá luego al levantamiento y descripción de los requerimientos funcionales, en calidad de escenarios, y los no funcionales o de calidad de servicios como propone MSF para el desarrollo de un software ágil. Se generarán además las diferentes vistas propuestas por la metodología enfocándose mayormente en las de integración e infraestructura.

2.2. Fase Visión y Alcance.

En el presente epígrafe se documentará la fase Visión y Alcance, explicando profundamente los objetivos que se persiguen con el desarrollo del sistema y proporcionando además una idea de cómo sería la lógica del negocio en dicho sistema.

2.3. Descripción del Problema.

Como parte del proyecto Sistema de Control Migratorio para la República Bolivariana de Venezuela se desarrolló una aplicación por el CISED que no cuenta con funcionalidades para realizar la lectura de documentos electrónicos, ni verificar la firma digital contenida en el objeto de seguridad del chip, siendo esta la medida de seguridad más importante y la única de obligatorio cumplimiento para que sea efectiva la entrada al país del viajero. Se adquiere para la solución el lector *Regula*, este cuenta en su *Software Development Kit* (SDK, por sus siglas en inglés) con funcionalidades para realizar la autenticación pasiva. Pero en el caso de la gestión de los certificados se realizar a través de un número anidado de directorios donde estarán almacenados los CDS y CLR, esto trae como inconveniente realizar la gestión de los certificados de forma local, tornando lento y complicado el proceso para todos los SI donde será desplegado el sistema, de conjunto con el nivel de actualización requerido que según recomendación de la OACI debe ser diario. Así mismo en el caso de que se adquieran nuevos lectores, no garantizan que el SDK en el caso de precisarlo, contenga funcionalidades para realizar la autenticación pasiva, pero si implica adquisiciones de costos mayores.

2.4. Propuesta de Solución.

Se propone desarrollar dicha funcionalidad como un servicio, utilizando la nueva tecnología *Windows Comunication Fundation* (WCF, por sus siglas en inglés), este servicio se integraría a la aplicación ya desarrollada. Este tipo de solución es muy factible ya que no se suspenden las prestaciones de la aplicación solo se le adjuntaría este servicio para que cumpla su cometido. La solución contará de dos módulos fundamentales, verificar SOD y Repositorio. El primero se encargará de todo lo relacionado con comprobar que todos los datos del SOD estén íntegros y auténticos, el segundo gestionará todo lo relacionado con los certificados digitales, guardándolos en un repositorio local. Ver Figura 4.

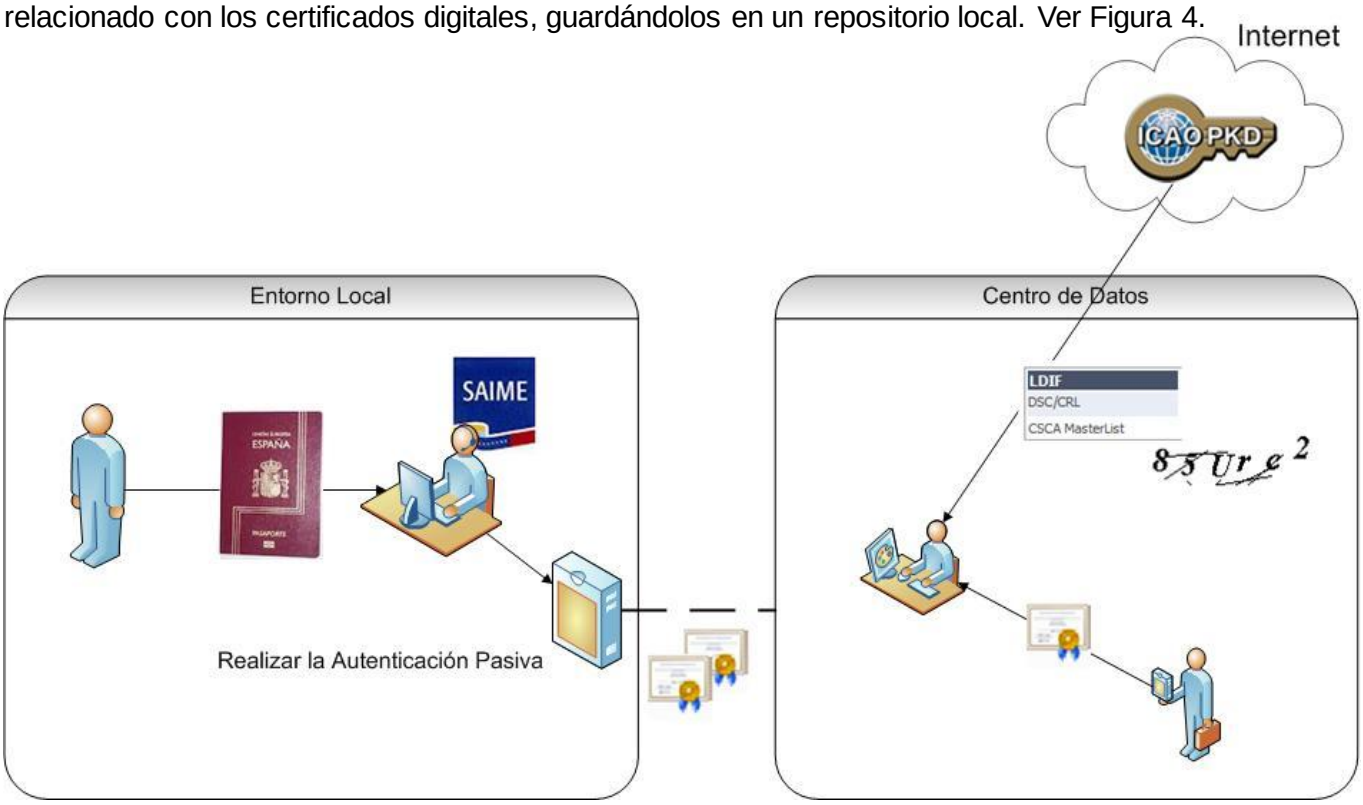

Figura 4 Diagrama de software.

El proceso comienza cuando llega un viajero al punto de inspección y entrega su pasaporte para que sea verificado, en este punto de inspección estarán instaladas varias computadoras, estas tendrán implementado un servicio (WCF), conectado a un servidor Open LDAP local de certificados. El servicio se encargará de verificar la autenticación pasiva en los pasaportes mediante el SOD obtenido, esta verificación está formada por dos partes fundamentales, la verificación de integridad del pasaporte-e que

se realiza buscando en el repositorio local un CDS igual al que trae el SOD del pasaporte a verificar, y la verificación de autenticidad que prueba que el viajero es quien dice ser, es decir, comprobar si los datos del CDS del SOD a comprobar y los del CDS del repositorio coinciden. Es por ello que se necesita una aplicación desktop encargada de gestionar los certificados de las dos formas posibles, ya sea por acuerdos bilaterales o directamente descargados del PKD de la OACI, esta aplicación tiene como objetivo fundamental mantener actualizado el repositorio central de certificados y replicar a todos los servidores locales que existan, ya que si hubiera demora en las actualizaciones se pudiera dar el caso de estar dejando entrar al país a personas que no están autorizadas.

La Tabla 1 muestra la descripción de los módulos en los que se dividió la implementación del sistema.

Módulos del sistema	**Descripción**
Verificar SOD	Realizan todas las verificaciones pertinentes para comprobar la veracidad del pasaporte-e.
Repositorio	Realiza el proceso de gestión de los certificados digitales y las listas de revocación.

Tabla 1 Módulos del sistema.

La Tabla 2 describe quien interactuará con el sistema.

Personas	**Descripción**
Sistema de control migratorio	Es el encargado de tomarle el pasaporte al viajero y proceder a hacer las verificaciones. Podrá realizar operaciones de verificación de integridad y autenticidad del documento.
Administrador del sistema	Es el encargado de gestionar lo relacionado con los certificados así como mantener actualizados los repositorios de certificados.

Tabla 2 Definición de personas.

2.5. Fase de Planificación.

Esta fase estará dirigida a planificar y estructurar el proyecto, se centrará en los escenarios y los requerimientos de calidad de servicios.

2.5.1. Listado de Escenarios.

Los escenarios definen la interacción entre las personas y el sistema, estos registran los pasos específicos a seguir para lograr el cumplimiento efectivo de una funcionalidad. Una descripción detallada de los escenarios es de gran ayuda a la hora de implementar las funcionalidades, es por ello que se llevará a cabo en este momento su definición.

Módulo Verificar SOD:

RF 1. Verificar SOD.

RF 1.1. Verificar firma del SOD.

RF 1.2. Verificar firma del DS.

RF 1.2.1. Si el SOD se verificó con el CDS contenido en el SOD debe verificarse si este CDS está contenido en el repositorio.

RF 1.2.1.1. Si el CDS no está contenido en el repositorio, se verifica con el apropiado CSCA (Extraer del DS el AuthorityKeyIdentifier y buscar en el CSCA por el SubjectKeyIdentifier).

RF 1.3. Verificar período de validez de los certificados.

RF 1.3.1. Verificar período de validez del CDS.

RF 1.3.2. Verificar período de validez de CSCA.

RF 1.4. Verificar estado de revocación del certificado.

RF 1.5. Verificar integridad del SOD.

RF 1.5.1. Se comparan los hash de los DGs almacenados en el SOD con los DGs almacenados en el SOD con los contenidos en el documento.

RF 2.

RF 2.1. El sistema obtiene el SOD que es enviado por el SI el cuál es leído del pasaporte-e con la tecnología específica desplegada en el SI.

RF 2.2. Procesar SOD, se obtienen del certificado los datos:

RF 2.2.1. Tipo de función resumen o hash empleada que pueden ser: SHA-1, SHA-224, SHA-256, SHA-384 and SHA-512.

RF 2.2.2. Tipo de llaves utilizadas, que pueden ser para CSCA tamaño mínimo del módulo de n RSA 3072 bits, tamaño mínimo del módulo de p y q DSA 2048 y 224 bits, ECDSA tamaño mínimo 256 bits y para los CDS tamaño mínimo del módulo de n RSA 2048 bits, tamaño mínimo del módulo de p y q DSA 1024 y 160 bits, ECDSA tamaño mínimo de la orden de punto de base de 224 bits.

RF 2.2.3. Nación emisora para verificar con los CSCA, según código de país especificado en ISO/IEC 3166. (Extraer del DS el AuthorityKeyIdentifier y buscar en el CSCA por el SubjectKeyIdentifier)

RF 2.2.3.1. En el caso que no coincidan las llaves se buscará el CSCA por el keyidentifiers.

RF 2.2.4. Se obtiene el número de serie del CDS.

RF 3. Obtener DG

RF 4. Verificar validez del SOD

RF 4.1. Verificar autenticidad del SOD

RF 4.1.1. Se verifica si se cuenta con el CSCA en el repositorio del país emisor.

4.1.1.1. Debe verificar el período de validez del CSCA y del CDS

4.1.1.2. Se verifica que el CDS contenido en el SOD sea emitido por la CSCA del emisor.

RF 4.1.2. Se obtiene del CDS del repositorio por el número de serie y se comparan los certificados.

RF 4.1.3. Se verifica en la CLR el estado de revocación del certificado

RF 4.1.4. Se verifica la firma del SOD con el CDS válido.

RF 4.2. Verificar integridad del SOD

RF 4.2.1. Se comparan los hash de los DGs almacenados en el SOD con los DGs almacenados en el SOD con los contenidos en el documento.

RF 5. Notificar resultado de las acciones de verificación.

Módulo Repositorio:

Fuente Acuerdos bilaterales y del emisor.

RF 6. Adicionar certificado CDS, CSCA al repositorio.

RF 7. Eliminar certificado.

RF 8. Adicionar CLR.

Fuente ICAO PKD.

RF 9. Procesar fichero ldif con CDS.

RF 10. Procesar fichero ldif con CLR.

RF 11. Procesar fichero ldif con Master List.

RF 12. Listar certificados.

RF 13. Buscar certificado.

RF 14. Ver detalles del certificado.

RF 15. Control de acceso básico, solo para usuarios administrador.

2.5.1.1. Priorizar la lista de escenarios.

Es de gran importancia la priorización de los escenarios, porque permite identificar los escenarios más relevantes para darle un tratamiento diferenciado cuando llegue el momento de implementarlos, por ejemplo los escenarios que sea de prioridad alta se implementarán en las primeras iteraciones. Los escenarios que se clasificarán en prioridad baja se simbolizarán con el número 3, los de prioridad media con 4 y los de prioridad alta con 5. Ver Tabla 3.

Escenario	Prioridad
Verificar SOD.	5
Verificar Integridad del SOD.	5
Verificar Autenticidad del SOD.	5
Notificar resultado de las acciones de verificación.	5
Gestionar fuente de acuerdos bilaterales y del emisor.	5
Gestionar Fuente OACI PKD.	5
Procesar fichero ldif con CDS.	4
Verificar integridad de los ldif.	5
Procesar fichero ldif con CRL.	4

Procesar Master List.	4
Listar certificados.	4
Buscar certificado.	3
Ver detalles del certificado.	5

Tabla 3 Prioridad de escenarios.

A continuación se presenta la Figura 5, en ella se puede apreciar el cúmulo de escenarios por módulos que se definieron para la solución del problema y sus respectivas prioridades, éstas conceptualizadas según el peso que tendrían los escenarios en el sistema.

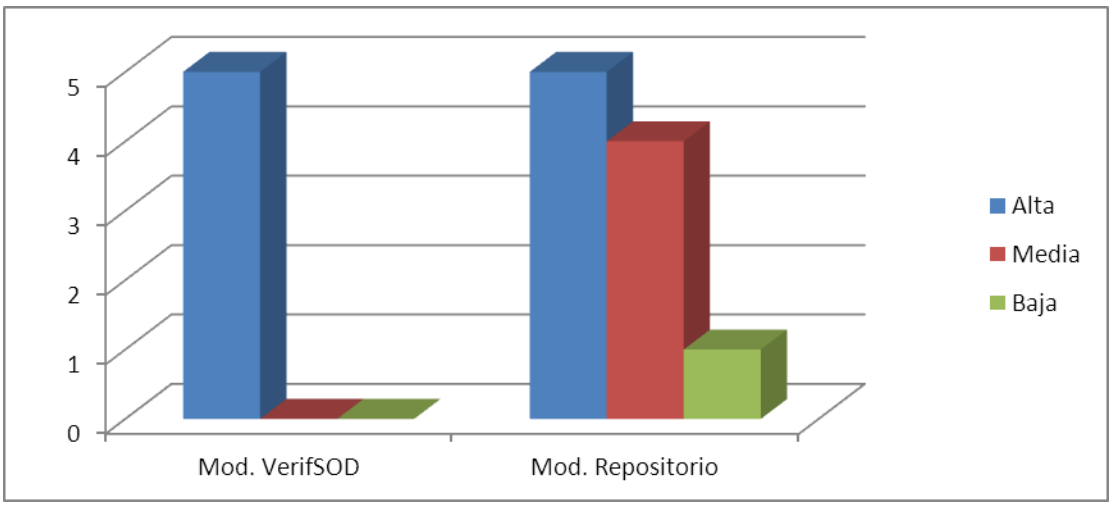

Figura 5 Prioridad de escenarios por módulos.

2.5.1.2. Plan de Iteraciones.

Para lograr la implementación en tiempo y forma del sistema de ejecución de autenticación pasiva en pasaportes-e, es necesario hacer una estimación del tiempo que tomará ejecutar la codificación de cada uno de los escenarios. En relación con la prioridad que tenga cada escenario se decide cuáles de ellos se desarrollarán en las primeras iteraciones, pues las funcionalidades críticas del sistema deben ser codificadas en las iteraciones más tempranas de su ciclo de vida. Ver Tabla 4.

La implementación de los escenarios estará dividida en las siguientes iteraciones:

- Iteración 1: Se propone codificar los escenarios que proveen la prioridad alta de los módulos Verificar SOD y Repositorio.
- Iteración 2: Se codificarán los escenarios que proveen la prioridad media y baja de los módulos Verificar SOD y Repositorio.

Iteración	Escenario	Prioridad	Riesgo	Esfuerzo(Días)
1	Verificar Integridad del SOD.	5	Alto	7
1	Verificar Autenticidad del SOD.	5	Alto	7
1	Notificar resultado de las acciones de verificación.	5	Alto	1
1	Gestionar fuente de acuerdos bilaterales y del emisor.	5	Alto	3
1	Gestionar Fuente OACI PKD.	5	Alto	4
2	Procesar fichero ldif con CDS.	4	Alto	6
1	Verificar integridad de los ldif.	5	Alto	7
2	Procesar fichero ldif con CRL.	4	Alto	5
2	Procesar Master List.	4	Alto	5
2	Listar certificados.	4	Medio	7
2	Buscar certificado.	3	Medio	1
1	Ver detalles del certificado.	5	Alto	5

Tabla 4 Plan de iteraciones.

2.6. Descripción de los escenarios.

A continuación se presentan dos descripciones de escenario por módulos, las restantes estarán en el Anexo 1.

2.6.1. Módulo Verificar SOD.

Verificar estado de revocación del CDS.

Nombre del Escenario: Verificar estado de revocación del CDS.		***Identificador:*** *1*
Objetivo del Escenario: Verificar estado de revocación del certificado.		
Persona: Sistema de control migratorio.		
Iteración: 1ra	***Prioridad:*** 1	***Complejidad:*** 3
Descripción: El escenario comienza cuando el sistema de control migratorio desea verificar si el certificado se encuentra revocado, para realizar esta verificación se comparan el rango de fechas de inicio y fin del certificado con la fecha actual, además se busca por el número de serie este certificado en la lista de revocación almacenada en el repositorio, si no se encuentra el certificado es válido.		

Tabla 5 Descripción del escenario "Verificar SOD"

Verificar Integridad del pasaporte.

Nombre del Escenario: Verificar integridad del pasaporte.		***Identificador:*** *1*
Objetivo del Escenario: Verificar la integridad del documento.		
Persona: Sistema de control migratorio.		
Iteración: 1ra	***Prioridad:*** 1	***Complejidad:*** 3
Descripción: El escenario inicia cuando se invoca la funcionalidad de verificar la autenticación pasiva del SOD del pasaporte, para verificar este paso el sistema obtiene la LDS del pasaporte y le aplica una función resumen, luego se compara el resultado con los datos que vienen en el SOD que es la LDS con función resumen aplicada, si son iguales el documento no ha sido alterado.		

Tabla 6 Descripción del escenario "Verificar integridad del pasaporte"

2.6.2. Módulo Repositorio.

Ver detalles del certificado.

RF 16. ***Nombre del Escenario:*** Ver detalles del certificado.		***Identificador:*** *1*
Objetivo del Escenario: Ver detalles del certificado.		
Persona: Administrador del sistema.		
Iteración: 1ra	***Prioridad:*** 1	***Complejidad:*** 3
Descripción: El escenario comienza cuando se selecciona el certificado que se quiere mostrar, se muestran de dicho certificado en un listboxt los datos: número serie, país emisor, fecha inicio, fecha fin, la llave con que se firmó el certificado, entre otros.		

Tabla 7 Descripción del escenario "Gestionar fuente acuerdos bilaterales y del emisor"

Adicionar certificado CDS, CSCA al repositorio.

Nombre del Escenario: Adicionar certificado CDS, CSCA al repositorio.		***Identificador:*** *1*
Objetivo del Escenario: Adicionar los certificados al repositorio.		
Persona: Administrador del sistema.		
Iteración: 1ra	***Prioridad:*** 1	***Complejidad:*** 3
Descripción: El escenario comienza cuando se obtiene un fichero ldif descargado del PKD de la OACI, se verifica que el formato sea válido y se procede a adicionar a las listas del repositorio los CDS y las CSCA, es muy importante contar con estas listas actualizadas para llevar a cabo la verificación de la autenticación pasiva en los pasaportes electrónicos.		

Tabla 8 Descripción del escenario "Adicionar certificado CDS; CSCA al repositorio"

2.7. Requisitos de calidad de servicio.

Los requerimientos de calidad de servicio, como son conocidos por la metodología MSF, son las especificaciones que el sistema debe cumplir, se clasifican en diferentes tipos como de soporte, usabilidad, rendimiento, fiabilidad, eficiencia, entre otros. A continuación se mostrará la lista de los requerimientos que se identificaron para el desarrollo del sistema.

2.7.1. Usabilidad.

1. Será aplicable a cualquier SI que precise la verificación de la autenticación pasiva.
2. La aplicación será distribuida en idioma español.
3. La aplicación poseerá una estructura y diseño homogéneos en todas sus ventanas, que facilite su utilización.

2.7.2. Fiabilidad.

1. La aplicación será accesible las 24 horas del día los 365 días del año.
2. No se afectará el funcionamiento de la aplicación cuando se esté actualizado el repositorio.

2.7.3. Eficiencia.

1. El sistema, en el módulo de verificación del SOD debe emitir respuestas en el orden menor de 1 segundo.
2. El repositorio debe ser capaz de soportar más de 3000 certificados (teniendo en cuenta que se emitan 4 certificados anuales, por 120 países, durante 3 años).

2.7.4. Interfaces de usuario.

1. Las interfaces de usuario del módulo repositorio funcionarán en una resolución de 1024x768.
2. Se utilizarán menús para acceder a las funcionalidades.

2.8. Fase de diseño.

En el diseño es donde se modela el sistema para que soporte todos los requisitos, incluyendo los de calidad de servicio. El diseño permite traducir los requisitos definidos, en aras de construir un software

capaz de cumplir con todas las expectativas del cliente, lo que garantiza que durante la construcción del sistema haya un bajo nivel de cambios.

2.9. Arquitectura de la solución.

La arquitectura de software en una solución es fundamental ya que por ella se regirá toda la implementación, esta agilizará y organizará el trabajo porque está enmarcada en sus componentes, las relaciones entre ellos, el ambiente y los principios que orientan su diseño y evolución. Además es una vista estructural de alto nivel, que ocurre tempranamente en el ciclo de vida y define los estilos o grupos de estilos adecuados para cumplir con los requerimientos de calidad del servicio.

2.10. Estilos arquitectónicos.

Los estilos arquitectónicos son utilizados para describir la estructura de las soluciones, escogiendo de los tantos que existen el estilo que más se adecúe a la ocasión, para hacer esta selección se deben tener en cuenta las ventajas, desventajas y características de cada uno de estos estilos.

Después de un análisis se selecciona el estilo en capas, debido a que entre sus principales particulares se pueden mencionar su gran flexibilidad a la hora de confeccionar las estructuras de las clases y las relaciones entre ellas, ya que no es un estilo en el que las capas sean de estricto cumplimiento, mayormente se usan 3 capas presentación, controlador, y acceso a datos pero la cantidad de capas varían de acuerdo a la necesidad, el flujo de datos en este estilo está concebido unidireccionalmente, es decir los datos viajarán en una dirección y desde la capa presentación no se puede acceder a la base de datos, y viceversa, lo que asegura la integridad de los datos. Facilita la modularidad, reusabilidad, el cambio y la portabilidad. Posibilita realizar grandes cambios en una de las capas y las demás no se verán afectadas, permitiendo además trabajar de manera transparente una vez establecidas las conexiones entre las capas. En las Figuras 6 y 7, se especifica cómo se aplica el estilo en capas en el sistema.

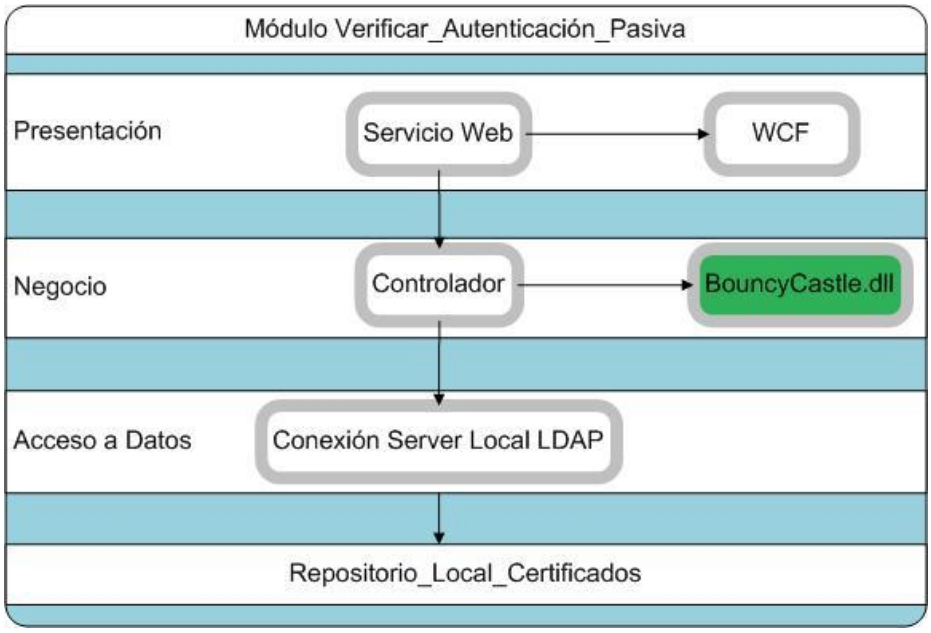

Figura 6 Estilo arquitectónico en capas del módulo verificar SOD.

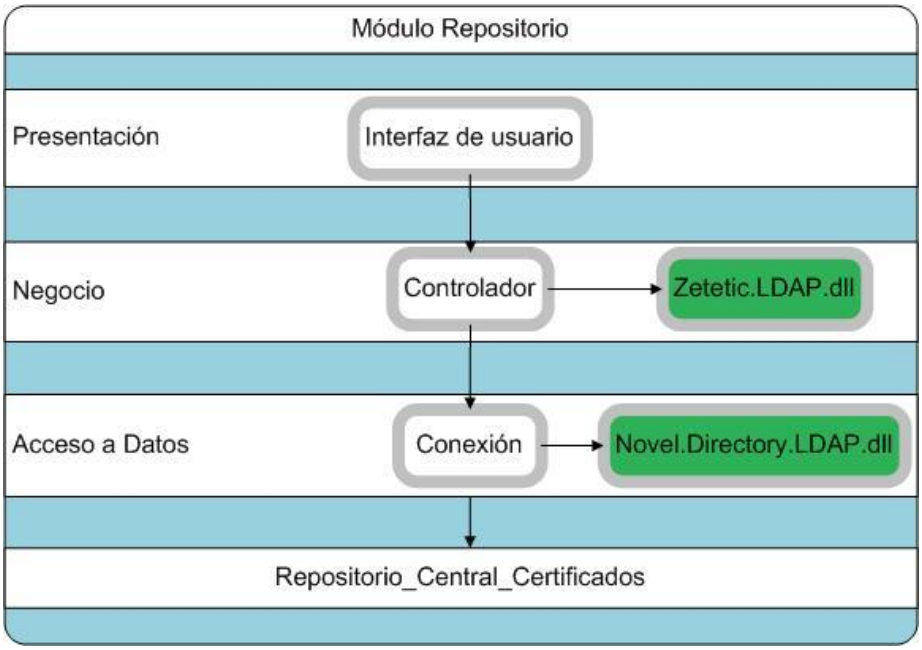

Figura 7 Estilo arquitectónico en capas del módulo repositorio.

Presentación: este estilo está pensado para proyectos en el que se usen una sola vista. En esta capa se encuentran todas las vistas que presentan el sistema al usuario, Esta capa está representada por un servicio web utilizando WCF y tiene una interacción directa con la capa de negocio. Para el desarrollo del mismo se hace uso del *framework ASP.NET*.

Negocio: en esta capa se contendrá la o las clases necesarias para realizar de manera correcta todas las funcionalidades que dan solución a los requerimientos del sistema, son las que manejan todas las operaciones sobre las entidades del negocio definidas, o comúnmente llamadas como clases controladoras.

Acceso a datos: esta capa se relaciona con las funcionalidades definidas en la capa de negocio, posibilitando de esta forma que se puedan hacer cambios en esta capa sin afectar a las demás. Su función principal es trabajar directamente con la fuente de datos establecida. En esta capa se incluye el uso de repositorios de certificados utilizando LDAP.

2.11. Diagrama de clases.

El diagrama de clases es un diagrama de tipo estático, que describe la estructura de un sistema representando las clases que serán utilizadas, sus atributos y las relaciones que existen entre ellas. Las clases del sistema están organizadas de una forma estructural adecuada según lo necesario en cada una de ellas, lo que las hace fáciles de manipular y permite que se interrelacionen siempre que se necesite.
A continuación se representan las clases del sistema con sus atributos, métodos y relaciones. Ver Figura 8.

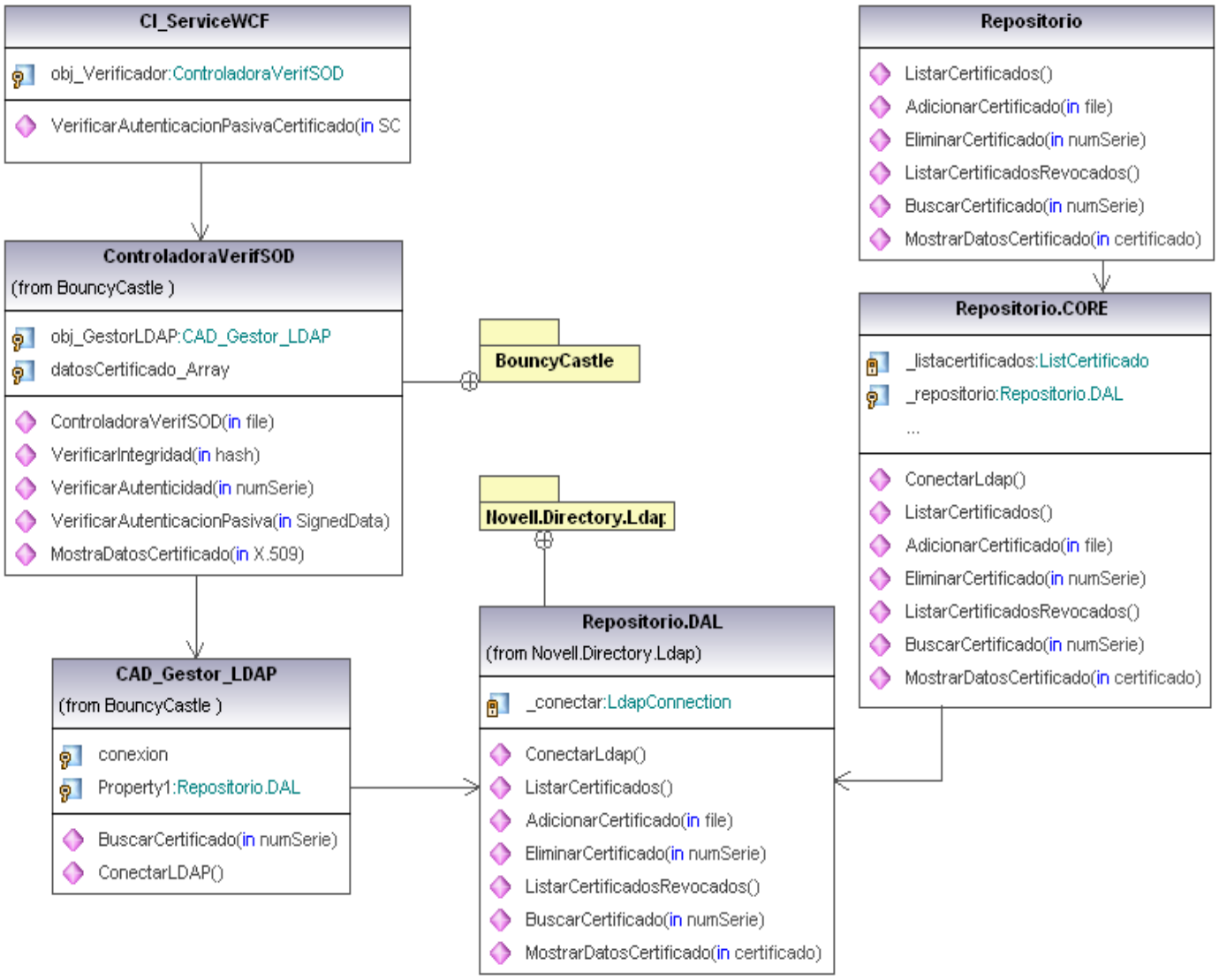

Figura 8 Diagrama de clases general del sistema.

2.12. Patrones de diseño.

Los patrones de diseño imponen reglas sobre la arquitectura y es usado junto a un estilo arquitectónico para determinar la forma de la estructura general de un sistema. Un patrón de diseño es una descripción de clases y objetos comunicándose entre sí. Es una solución repetible a un problema recurrente en el diseño de software. Esta solución no es un diseño terminado que puede traducirse directamente a código, sino más bien una descripción sobre cómo resolver el problema, la cual puede ser utilizada en diversas situaciones.

Para la realización de un diseño más eficiente es conveniente utilizar varios patrones, ya que constituyen experiencias de diseñadores expertos en orientación a objetos, los cuales permiten dar solución a problemas a través de la codificación del conocimiento y principios existentes. Cada patrón describe un problema que ocurre una y otra vez en el sistema, y luego describe el núcleo de la solución a ese problema.

2.12.1. Patrones GRASP.

Los patrones GRASP son patrones generales de software para asignación de responsabilidades. Aunque se considera que más que patrones, son una serie de buenas prácticas de aplicación recomendable en el diseño de software. Estos patrones constituyen un proceso que sirve para determinar quién hace qué cosa en cada momento. Se dividen en varias categorías, y las que se usan para desarrollar la aplicación son las siguientes: [35]

Patrón Experto: Asignar responsabilidades de la forma más eficiente.

Se le asignarán las principales responsabilidades al experto en la información o la clase que cuenta con la información necesaria para cumplir dichas responsabilidades. Se hace uso de este patrón para que las clases controladoras le asignen responsabilidades a sus respectivas clases de presentación y de acceso a datos, según las peticiones recibidas. Por ejemplo cuando se quiere obtener un certificado, la clase *ControlarRepositorio* del módulo Repositorio, le asigna la responsabilidad a la clase *AdRepositorio* de buscar un certificado por su número de serie.

Patrón Creador: Responsable de crear una nueva instancia de alguna clase.

El patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. Se hace uso de este patrón cuando es necesario que las clases creen instancias con el nivel necesario de información para acceder a los datos almacenados, de acuerdo a la ejecución de una determinada acción. Por ejemplo en la clase *AdRepositorio* se crea el objeto de la clase *LdapConecction* para poder hacer la conexión al servidor Openldap para la gestión de los certificados.

Patrón bajo acoplamiento: Soportar bajas dependencias.

El Bajo Acoplamiento es un principio que se debe recordar durante las decisiones de diseño, porque es el encargado de disminuir las dependencias de una clase con las demás. Soporta el diseño de clases más independientes y esto reduce el impacto de los cambios. En el diseño propuesto el acceso a datos sólo depende de la clase controladora que la usa, así se pueden realizar cambios en cada clase de forma independiente. Por ejemplo, para hacer la conexión al servidor Openldap, los atributos (*port, host, user, password*) fueron definidos en la clase *app.config*, por si es necesario cambiar la configuración de algunos de estos atributos se cambie en dicha clase y no afecte la conexión al Ldap.

Patrón alta cohesión: Mantiene la complejidad manejable.

Como el patrón Bajo Acoplamiento, también Alta Cohesión es un principio que se debe tener presente en todas las decisiones de diseño, esta es la meta principal que ha de buscarse en todo momento. Este patrón evita asignar demasiadas responsabilidades a las clases. Con el uso de este patrón se tienen que las clases controladoras se encargan de ejecutar acciones de acuerdo a las peticiones que le llegan y las de acceso a datos interactúan con las tablas de la base de datos, de forma tal que se elimina la sobrecarga de funcionalidades en las clases controladoras. Una clase de alta cohesión posee un número relativamente pequeño de responsabilidades, con una importante funcionalidad relacionada y poco trabajo por hacer. Colabora con otros objetos para compartir el esfuerzo si la tarea es grande. Este patrón mejora la calidad y facilidad con que se puede entender el diseño, se genera un bajo acoplamiento y permite fomentar la reutilización. Se puede evidenciar en la clase *Sección* que solo implementa la autenticación del administrador a la aplicación Gestionar Repositorio.

Patrón controlador: Responsable de gestionar un evento de entrada al sistema.

El controlador es un intermediario entre la capa de presentación y el núcleo de las clases donde reside la lógica de la aplicación. El controlador no realiza mucho trabajo por sí mismo; más bien coordina la actividad de otros objetos. Se hace uso del patrón controlador definiendo clases controladoras que dirigen todo el flujo de información. Por ejemplo, las clases *Controladora* del módulo *VerificarSOD* y la *ControlarRepositorio* del módulo Repositorio gestionan toda la información necesaria para verificar la autenticación pasiva de los pasaportes-e.

2.13. Conclusiones.

- La utilización de la metodología MSF for ASD permitió obtener cada uno de los artefactos necesarios, para desarrollar una aplicación informática ajustada a los principios definidos y concluir satisfactoriamente la primera iteración.
- La especificación de los escenarios y los requisitos de calidad de servicio permitió describir las funcionalidades que el cliente necesita y que la aplicación cumplirá.
- Se definió la arquitectura en capas porque cuando es necesario hacer algún cambio en una de las capas esto no afecta a las demás, siendo una gran ventaja para la evolución de la aplicación.

Capítulo 3: Implementación y pruebas.

3.1. Introducción.

Este capítulo estará referido a la construcción del sistema de acuerdo con la fase de Desarrollo de la metodología MSF for ASD. Se describirán y modelarán las clases y funcionalidades que darán solución al problema de la investigación, así como la realización de las pruebas pertinentes para asegurar que el software tenga la mayor calidad posible.

3.2. Estándares de codificación.

Los estándares de codificación comprenden muchos aspectos para la generación de código. Los programadores deben implementar un estándar práctico, se debe reflejar un estilo armonioso como si un único programador hubiese escrito el código. Es bueno que sea establecido el estándar desde el comienzo del software para asegurarse que todos los programadores trabajen de forma coordinada. Un código escrito con buen estilo en cualquier lenguaje sigue las siguientes propiedades:

- Organización
- Fácil de leer.
- Fácil de mantener.
- Fácil de detectar errores.
- Eficiente

La legibilidad del código fuente repercute directamente en lo bien que un programador comprende un sistema de software. Usar técnicas de codificación sólidas y realizar buenas prácticas de programación con vistas a generar un código de alta calidad es de gran importancia para la calidad del software y para obtener un buen rendimiento. Además, si se aplica de forma continuada un estándar de codificación bien definido, se utilizan técnicas de programación apropiadas, y, posteriormente, se efectúan revisiones del código de rutinas, caben muchas posibilidades de que un proyecto de software se convierta en un sistema de software fácil de comprender y de mantener.

Aunque el propósito principal para llevar a cabo revisiones del código a lo largo de todo el desarrollo es localizar defectos en el mismo, las revisiones también pueden afianzar los estándares de codificación de manera uniforme. La adopción de un estándar de codificación sólo es viable si se sigue desde el principio hasta el final del proyecto de software. No es práctico, ni prudente, imponer un estándar de codificación

una vez iniciado el trabajo. A continuación se define el estándar que se utilizará en la implementación del sistema.

Nombres

- El esquema de nombres es una de las ayudas más importantes para entender el flujo lógico de una aplicación. Un nombre debe más bien expresar el "qué" que el "cómo". El interés de poner un nombre es que para que indique que se necesita analizar o definir con mayor precisión el propósito de un elemento. Los nombres expresivos funcionan como ayuda para el lector, por eso, es lógico dar nombres que sean fáciles de comprender.

Por ejemplo el sistema aplicará la utilización de los nombres de la clase con la primera letra en mayúsculas, si estuviera compuesta por más de una palabra el comienzo de cada palabra sería con mayúsculas y sin espacios entre las palabras, ejemplo la clase acceso a datos del módulo Repositorio se nombra *ADRepositorio*.

Variables

Para la declaración de las variables se tomarán en cuenta las siguientes especificaciones:

- Dado que la mayoría de nombres de variables se construyen concatenando varias palabras, se empleará una mezcla de mayúsculas y minúsculas para simplificar la lectura. Poniendo la primera letra de cada palabra en mayúscula, exceptuando la primera.
- Para el nombre de variables de una sola palabra, antes de poner el nombre se utilizará un guión bajo y luego con letra minúscula el nombre de la variable.
- Para el caso de una variable de poco uso, que deba aparecer sólo en unas cuantas líneas de código, se empleará un nombre descriptivo.
- Las variables de una sola letra, como i o j sólo se utilizarán para índices cortos.
- Se minimizará el uso de abreviaturas; pero si se emplean, serán de forma coherente. Una abreviatura sólo debe tener un significado, y del mismo modo a cada palabra abreviada sólo debe corresponder una abreviatura.

Por ejemplo una variable compuesta de dos palabras se denominará *numSerie*, y una variable de una sola palabra sería *_conectar*.

Comentarios

- Cuando se modifique el código, se mantendrán actualizados los comentarios circundantes.

- Al principio de cada método se harán comentarios estándar, repetitivos, que indiquen el propósito de la rutina. Un comentario repetitivo podría consistir en una breve introducción que explicara por qué existe y qué puede hacer.
- Se evitarán los comentarios recargados, como las líneas enteras de asteriscos.
- Se usarán frases completas cuando se escriban comentarios. Los comentarios deben aclarar el código, no añadirle ambigüedad.
- Se irá comentando al mismo tiempo que programa, porque probablemente no tenga tiempo de hacerlo más tarde. Por otro lado, aunque tuviera oportunidad de revisar el código que ha escrito, lo que parece obvio hoy es posible que después no lo sea.
- Se comentará cualquier cosa que no sea legible de forma obvia en el código.
- Se realice los comentarios en un estilo uniforme, respetando una puntuación y estructura coherentes a lo largo de toda la aplicación.

Formato

- Se establecerá un tamaño estándar de sangría y se alinearán las secciones de código mediante la sangría predeterminada.
- Se alinearán verticalmente las llaves de apertura y cierre donde los pares de llaves se alinean.
- Se aplicarán una sangría al código en todas las líneas de construcción lógica. Si no aplica la sangría, el código resulta difícil de seguir.
- Se utilizarán espacios antes y después de los operadores siempre que eso no altere la sangría aplicada al código.

Nombre de carpetas

- Los archivos y los nombres de carpetas describirán claramente su finalidad por ejemplo la carpeta de la capa de acceso a datos se denominará *Repositorio.DAL*.

3.3. Diagrama de componentes.

El diagrama de componentes representa la parte física y reemplazable de un sistema, posee características similares a la de las clases por ejemplo: nombre, relaciones, instancias, puede componer interfaces y realizarlas. La diferencia es que los componentes representan un elemento físico y las clases son abstracciones lógicas, los componentes se pueden representar en nodos físicos, las clases no. Las

operaciones de un componente solo se alcanzan a través de interfaces, las de una clase podrían ser accesibles directamente. [36]

Un componente es una parte del sistema que representa un bloque de construcción fundamental sobre el cual se puede diseñar y construir sistemas. Un sistema puede ser solo un componente en un nivel de abstracción mayor, compuesto por varios componentes como por ejemplo: librerías, ejecutables, tablas de bases de datos, archivos y documentos.

A continuación se muestra en la Figura 9, el diagrama de componentes del sistema.

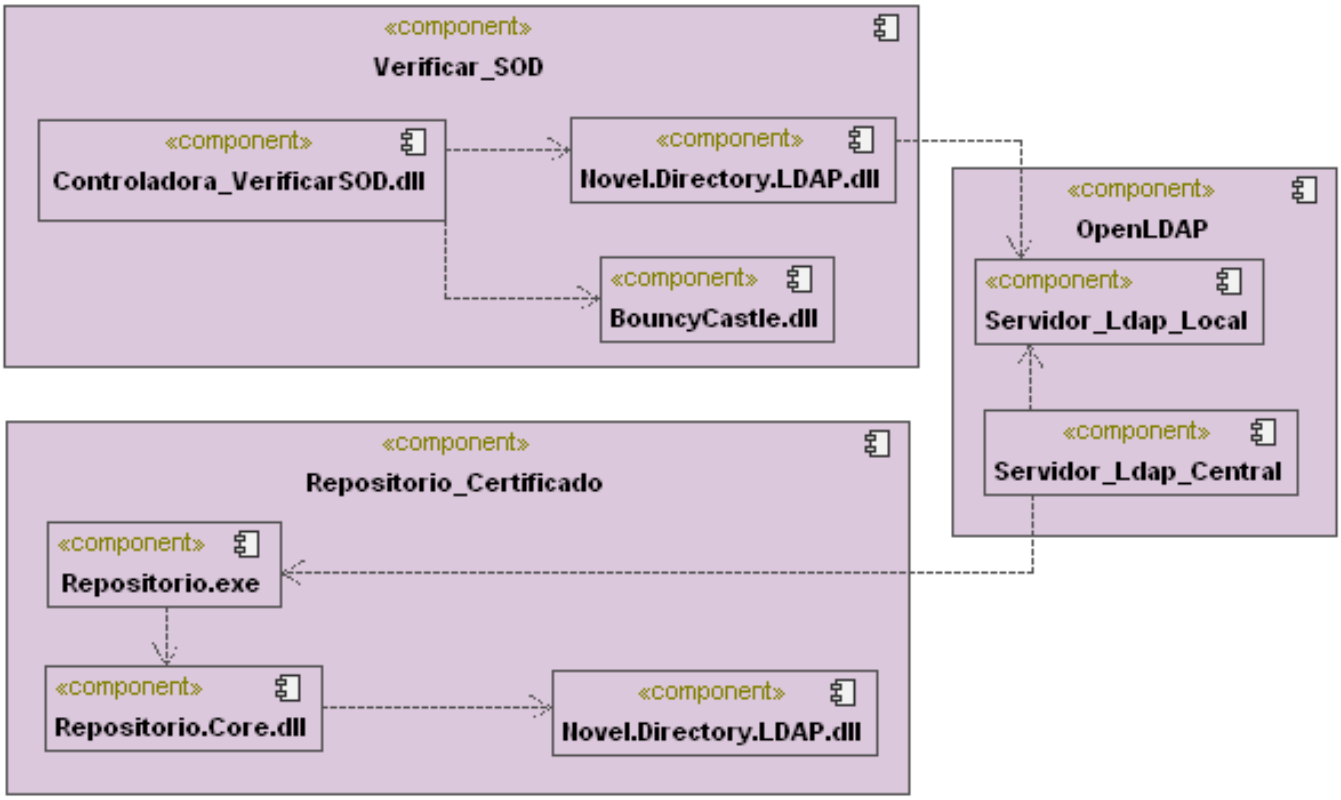

Figura 9 Diagrama de componentes

3.4. Diagrama de despliegue.

El diagrama de despliegue esquematiza los aspectos físicos de un sistema, representados gráficamente mediante nodos y como elemento fundamental las relaciones de dependencia, generalización, asociación y realización que lo componen.

Tiene como meta principal:

- Modelar la vista de despliegue estática de un sistema.
- Modelar la configuración de los nodos y los componentes que residen en ellos.
- Modelar la topología del hardware donde se ejecuta el sistema. [36]

A continuación se muestra en la Figura 10, el diagrama de despliegue del sistema.

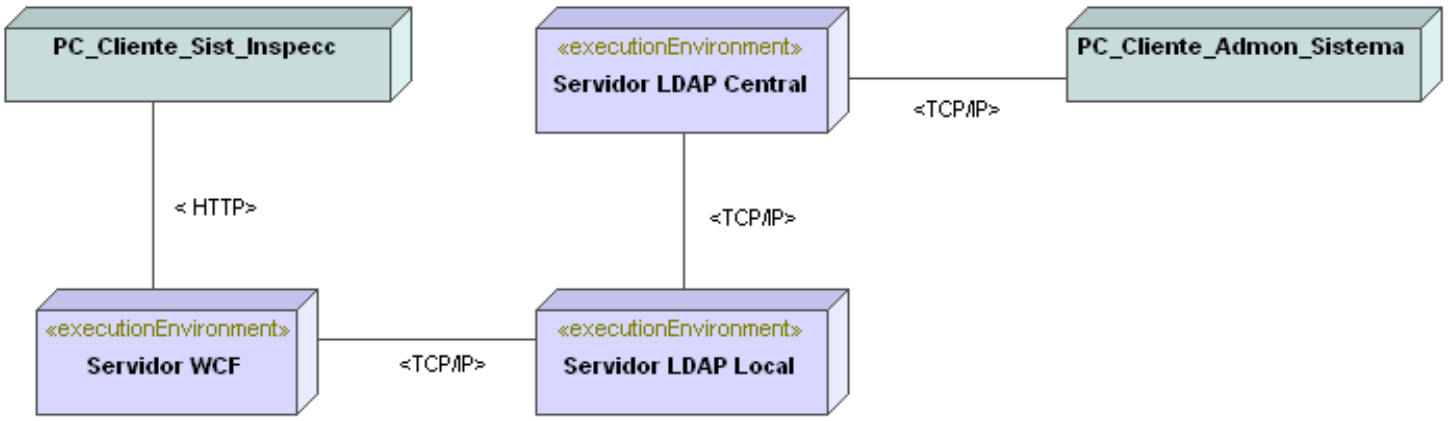

Figura 10 Diagrama de despliegue

3.5. Interfaz gráfica.

La interfaz gráfica del módulo Repositorio permite administrar todas las acciones que se realizan con los certificados digitales. Cuenta con las opciones adicionar certificado, eliminar certificado, listar certificados y ver datos de un certificado, en la Figura 11 se muestran los detalles.

Figura 11 Interfaz gráfica del módulo repositorio.

3.6. Fase de Estabilización.

En esta fase se le aplican las pruebas al software desarrollado, comprobando así si éste cumple con los requisitos necesarios para llevar a cabo un despliegue satisfactorio del sistema. Las pruebas son un conjunto de actividades que se lleva a cabo sistemáticamente, estas pueden planificarse por adelantado y ejecutarse una vez construido el código. Dentro de cada una de las etapas de desarrollo de un software las pruebas son fundamentales ya que a partir de ellas es posible controlar que los productos cumplan requisitos mínimos de operatividad además de garantizar la calidad de estos productos. Las pruebas constituyen una actividad en la cual un sistema o componente es ejecutado bajo condiciones específicas, se observan o almacenan los resultados y se realiza una evaluación de algún aspecto del sistema o componente.

3.6.1. Pruebas unitarias.

Las pruebas unitarias tienen como objetivo fundamental, comprobar los caminos lógicos del software

proponiendo casos de prueba que se ejerciten en conjuntos específicos de condiciones y/o bucles, estas pruebas son realizadas al código implementado para examinar el estado del programa en varios puntos, y determinar si el estado real coincide con el esperado, llamando directamente métodos que pasándole los parámetros apropiados simulen el proceso.

Estas pruebas se le realizaron a las funcionalidades más críticas, utilizando la herramienta *Visual Studio Team System 2010*.

A continuación se muestra en la Figura 12 una de las pruebas que se le realizaron al software, y el resultado que arrojó la misma, el cual indica que las funcionalidades probadas están correctamente implementadas. El resto de las pruebas unitarias se encuentran en el Anexo 2.

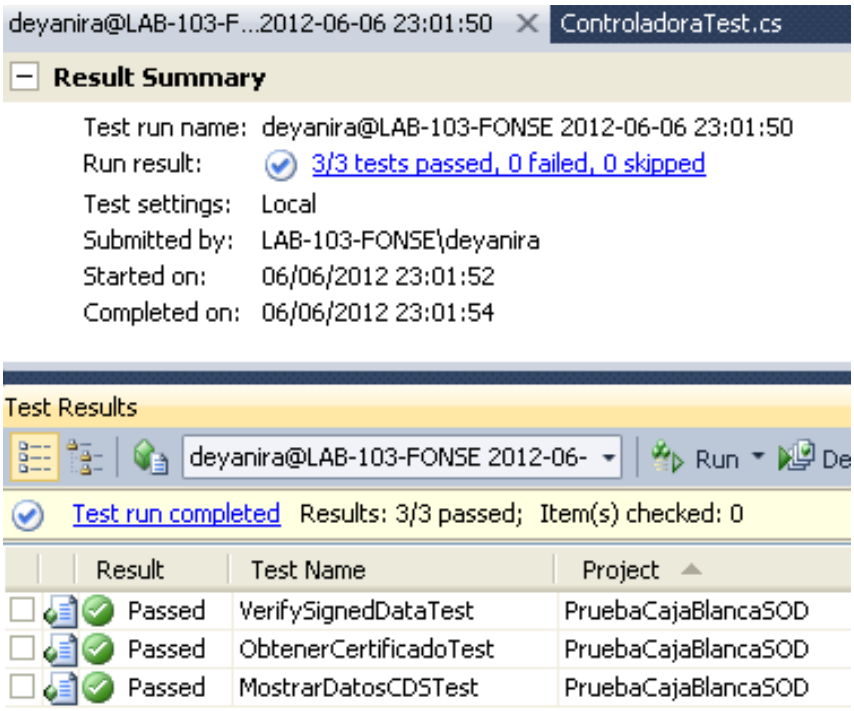

Figura 12 Prueba unitaria al software.

3.6.1.1. Resultado de las pruebas

Fueron aplicadas 6 pruebas unitarias a las funcionalidades más importantes del sistema, arrojando los resultados que se muestran en la Tabla 9.

Iteración	No	Funcionalidades	Aspecto correspondiente	Resultado
1	1	AdicionarCertificadoAcuerdosBilaterales()	Gestionar fuente de acuerdos bilaterales y del emisor.	No pasó
1	2	VerifySignedData()	Verificar autenticación pasiva	Pasó
1	3	ListarCertificados()	Sistema completo	Pasó
1	4	MostrarDatosCDS()	Fuente ICAO PKD.	Pasó
1	5	ObtenerCertificado()	Fuente ICAO PKD.	No pasó
1	6	ListarCertificadosRevocados()	Fuente ICAO PKD.	Pasó
2	7	AdicionarCertificadoAcuerdosBilaterales()	Gestionar fuente de acuerdos bilaterales y del emisor.	Pasó
2	8	ObtenerCertificado()	Fuente ICAO PKD.	Pasó

Tabla 9 Resumen de no conformidades de las pruebas unitarias.

A continuación se muestra en la Figura 13 una representación de las pruebas realizadas a las funcionalidades más importantes del sistema, quedando en la 1ra iteración (3) no conformidades que fueron resueltas en la 2da iteración.

Figura 13 Representación de iteraciones de pruebas unitarias.

3.6.2. Pruebas de caja negra o validación del sistema.

Dentro de las pruebas más conocidas están las de caja negra o de validación del sistema que son aplicadas a las interfaces del sistema, y las pruebas unitarias utilizando la técnica de caja blanca que se aplican directamente al código fuente.

Se refieren a pruebas que se llevan a cabo en las interfaces del software, mediante casos de pruebas que pretenden demostrar que las funciones del producto son operativas, que la entrada de los datos se realiza de manera adecuada, que se produce un resultado correcto y que la integridad de la información externa se mantiene.

Las pruebas de caja negra se centran principalmente en los requisitos funcionales del software y permiten obtener un conjunto de condiciones de entrada que ejerciten completamente estos requisitos.

A continuación se presenta en la tabla 10, el caso de prueba realizado al escenario Gestionar repositorio de certificado, los restantes se podrán ver en el Anexo 3.

Escenario: Gestionar Repositorio de certificado.

Escenario	Descripción	Variable 1	Variable 2	Variable n	Respuesta del sistema	Flujo central
EC 2.1.1 Adicionar certificado (CDS)	*Se procede a adicionar un certificado seleccionado, obtenido mediante acuerdos bilaterales.*	*V certificado*	*NA*	*NA*	*El sistema adiciona el certificado en el repositorio central.*	*Opción Gestionar Repositorio de Certificados / Adicionar certificado.*
		I certificado			*El sistema muestra un mensaje de error "El certificado no es válido".*	
EC 2.1.2 Eliminar certificado	*Se procede a eliminar un certificado una vez seleccionado de la lista de certificados.*	*V certificado*	*NA*	*NA*	*El sistema elimina el certificado seleccionado del repositorio y actualiza la lista de los certificados válidos.*	*Opción Gestionar Repositorio de Certificados / Eliminar certificado.*

					El sistema muestra un mensaje para que señale el certificado a eliminar.	
EC 2.1.3 Adicionar CRL.	*Se procede a adicionar una lista de revocación.*	*V CRL* *I CRL*	*NA*	*NA*	*El sistema adiciona a la lista de certificados revocados la nueva lista.* *El sistema muestra un mensaje de error "No se pudo adicionar la CRL"*	*Opción Gestionar Repositorio de Certificados / Adicionar CRL.*

Tabla 10 Descripción del caso de prueba "Gestionar repositorio de certificados".

3.6.2.1. Resultado de las pruebas

Fueron aplicados 6 casos de pruebas de caja negra al software que arrojaron las no conformidades que se muestran en la Tabla 11. En cada iteración de prueba se genera un resumen de todas las no conformidades existentes, para entregárselas a los desarrolladores que son los encargados de erradicarlas.

Iteración	No	No Conformidad	Aspecto correspondiente	Etapa de detección del error	Importancia
1	1	Mostrar la notificación del resultado de la verificación en la interfaz.	Notificar resultados.	Al verificar la autenticación pasiva.	No significativa
1	2	Mostrar la interfaz con el tamaño 1024x768.	Gestionar Repositorio de certificado.	Luego de autenticarse el administrador	No Significativa
1	3	Adicionar atributos de los certificados de acuerdos	Gestionar fuente de acuerdos bilaterales	Luego de adicionar un certificado por	Significativa

		bilaterales.	y del emisor.	acuerdos bilaterales.	
1	4	Mostrar mensaje de que se adicionó correctamente el CDS o el CSCA.	Adicionar certificado CDS, CSCA al repositorio.	Luego de adicionar un CDS o una CSCA.	No Significativa
2	5	Mostrar mensaje de que se eliminó correctamente el CDS.	Eliminar certificado.	Al eliminar un certificado.	No significativa
2	6	Mostrar mensaje de que se adicionó correctamente la CRL.	Adicionar CRL	Al adicionar una CRL.	No significativa
2	7	Mostrar un mensaje de que se encontró el certificado.	Buscar certificado.	Al buscar un certificado.	No Significativo
2	8	Mostrar en un listboxt todos los atributos de los certificados.	Ver detalles del certificado	Al seleccionar un certificado.	Significativo

Tabla 11 Resumen de no conformidades por iteraciones de las pruebas de caja negra.

A continuación se muestra en la Figura 14 una representación de los casos de pruebas realizados a las interfaces del sistema, quedando en la 1ra iteración (2) no conformidades que fueron resueltas en la 2da iteración.

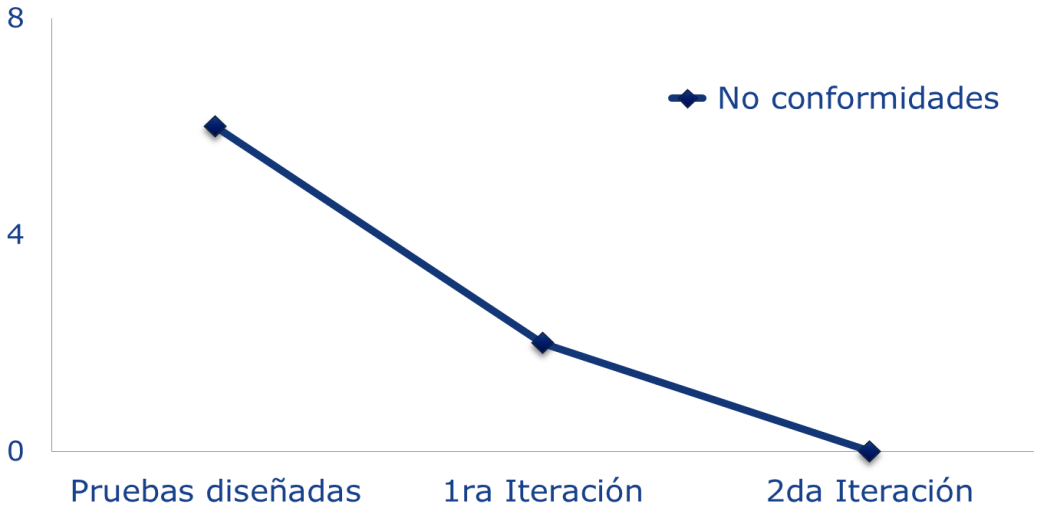

Figura 14 Representación de iteraciones de pruebas de caja negra.

3.7. Conclusiones

- La utilización de los patrones Grasp permitieron realizar una programación más organizada.
- La realización de los diferentes diagramas permitió obtener una visión detallada del sistema. Los diagramas de despliegue y componentes permitieron una mejor visibilidad de la distribución de los nodos físicos que serán necesarios para el despliegue del sistema.
- Las pruebas unitarias posibilitaron evaluar las respuestas de cada funcionalidad implementada, mientras que las pruebas de caja negra brindaron las opciones de comprobar la correcta ejecución de los requerimientos funcionales previamente definidos, posibilitando localizar y mitigar la presencia de posibles no conformidades en la aplicación.

Conclusiones

Después de haber completado el trabajo de diploma “Sistema para la ejecución de la autenticación pasiva de los pasaportes electrónicos”, se concluye que:

- El análisis de las diferentes metodologías, herramientas y tecnologías hizo posible su selección adecuada para el desarrollo del sistema.
- El cumplimiento de estándares de seguridad aplicables a los DVLM y el estudio de los sistemas similares, posibilitaron complementar funcionalidades que aporta mejoras al sistema de control migratorio de la República Bolivariana de Venezuela.
- Se desarrolló un sistema que cumple con las funcionalidades necesarias para realizar la autenticación pasiva, apoyándose en la especificación de escenarios, el correcto diseño de la arquitectura y uso de los patrones de diseño.
- La realización de pruebas demostraron que se cumplieron las funcionalidades especificadas en los requerimientos del sistema.

Por lo antes expuesto se concluye que la presente investigación cumplió su objetivo general, logrando un sistema para la ejecución de la autenticación pasiva de los pasaportes-e.

Recomendaciones

- Adicionar al sistema nuevas medidas de seguridad para pasaportes-e como por ejemplo control de acceso extendido.
- Adicionarle al sistema el uso del protocolo seguro HTTPS en la conexión del sistema de inspección con el servidor WCF donde estará público el servicio.

Anexo 1

1.1. Descripción de Escenarios.

1.1.1. Módulo Verificar SOD.

Verificar autenticidad del DVLM.

Nombre del Escenario: Verificar autenticidad del DVLM.		***Identificador:*** *1*
Objetivo del Escenario: Verificar la autenticidad del documento.		
Persona: Sistema de control migratorio.		
Iteración: 1ra	***Prioridad:*** 1	***Complejidad:*** 3
Descripción: El escenario inicia cuando el sistema de control migratorio procede a leer el DVLM del viajero. El sistema verifica la autenticidad del DVLM comparando si los datos del LDS obtenidos se corresponden con los del cliente.		
***Validaciones*:** El sistema debe verificar la autenticidad del DVLM.		

Tabla 12 Descripción del escenario "Verificar integridad del DVLM"

Notificar resultados.

Nombre del Escenario: Notificar resultados.		***Identificador:*** *1*
Objetivo del Escenario: Notificar resultados de las acciones de verificación.		
Persona: Sistema de verificación de SOD.		
Iteración: 1ra	***Prioridad:*** 1	***Complejidad:*** 3
Descripción: El escenario inicia cuando el sistema termina las verificaciones de integridad y autenticidad del DVLM. El sistema lanza un mensaje de notificación de verificación “satisfactoria” o “no satisfactoria”.		
***Validaciones*:**		

Tabla 13 Descripción del escenario "Notificar resultados"

1.1.2. Módulo Repositorio

Eliminar Certificado

Nombre del Escenario: Eliminar certificado.		***Identificador:*** *1*
Objetivo del Escenario: Eliminar certificado seleccionado		
Persona: Sistema para la verificación del SOD.		
Iteración: 1ra	***Prioridad:*** 1	***Complejidad:*** 3
Descripción: El escenario comienza cuando se selecciona el certificado que desea eliminar.		
***Validaciones*:** El sistema debe verificar que exista el certificado a eliminar en el repositorio.		
Prototipo de Interfaz de Usuario:		

Tabla 14 Descripción del escenario "Eliminar certificado"

Adicionar CRL

Nombre del Escenario: Adicionar CRL.		***Identificador:*** *1*
Objetivo del Escenario: Tener en el repositorio la lista de los certificados revocados.		
Persona: Sistema para la verificación del SOD.		
Iteración: 1ra	***Prioridad:*** 1	***Complejidad:*** 3
Descripción: El escenario comienza cuando se obtiene mediante el intercambio bilateral entre países o el PKD de la OACI, el número de serie de los certificados que están revocados y se adicionan a una lista de certificados revocados en el repositorio.		
***Validaciones*:** El sistema debe verificar la veracidad del CDS.		
Prototipo de Interfaz de Usuario:		

Tabla 15 Descripción del escenario "Adicionar CRL"

Fuente OACI PKD

Nombre del Escenario: Fuente OACI PKD.		***Identificador:*** *1*
Objetivo del Escenario: Almacenar los certificados válidos.		
Persona: Sistema para la verificación del SOD.		
Iteración: 1ra	***Prioridad:*** 1	***Complejidad:*** 3
Descripción: El escenario comienza cuando el sistema accede al OACI PKD centralizado y se descargan los certificados válidos para poder realizar la verificación de pasaportes electrónicos.		
***Validaciones*:**		
Prototipo de Interfaz de Usuario:		

Tabla 16 Descripción del escenario "Fuente ICAO PKD"

Procesar Master List

Nombre del Escenario: Procesar Master List		***Identificador:*** *1*
Objetivo del Escenario:		
Persona: Sistema para la verificación del SOD.		
Iteración: 1ra	***Prioridad:*** 1	***Complejidad:*** 3
Descripción: El escenario comienza cuando el sistema accede al OACI PKD centralizado y descarga la lista de maestra firmada de certificados de la autoridad certificadora obtenidos mediante acuerdos bilaterales con otros estados.		
***Validaciones*:**		
Prototipo de Interfaz de Usuario:		

Tabla 17 Descripción del escenario "Procesar Master List"

Verificar integridad de los ldif

Nombre del Escenario: Verificar integridad de los ldif.		***Identificador:*** *1*
Objetivo del Escenario: Verificar la correlación de los datos.		
Persona: Sistema para la verificación del SOD.		
Iteración: 1ra	***Prioridad:*** 1	***Complejidad:*** 3
Descripción: El escenario comienza cuando se verifica que los datos entrados para el intercambio de información están correctos.		
***Validaciones*:** El sistema debe verificar la integridad de los ldif.		
Prototipo de Interfaz de Usuario:		

Tabla 18 Descripción del escenario "Verificar integridad de los ldif"

Listar certificados

Nombre del Escenario: Listar certificados.		***Identificador:*** *1*
Objetivo del Escenario: Mostrar la lista de todos los certificados del repositorio.		
Persona: Sistema para la verificación del SOD.		
Iteración: 1ra	***Prioridad:*** 1	***Complejidad:*** 3
Descripción: El escenario comienza cuando se selecciona la opción listar certificados para ver los certificados existentes en el repositorio.		
***Validaciones*:** El sistema debe verificar que la petición sea válida.		
Prototipo de Interfaz de Usuario:		

Tabla 19 Descripción del escenario "Listar certificados"

Buscar certificados

Nombre del Escenario: Buscar certificado.		***Identificador:*** *1*
Objetivo del Escenario: Buscar un certificado.		
Persona: Sistema para la verificación del SOD.		
Iteración: 1ra	***Prioridad:*** 1	***Complejidad:*** 3
Descripción: El escenario comienza cuando se introduce en el sistema el dato del certificado que quiere buscar en el repositorio.		
***Validaciones*:** El sistema debe verificar que la petición sea válida.		
Prototipo de Interfaz de Usuario:		

Tabla 20 Descripción del escenario "Buscar certificados"

Anexo 2

2.1. Pruebas de caja blanca.

2.1.1. Módulo Verificar SOD.

2.1.1.1. Funcionalidad: VerifySignedData.

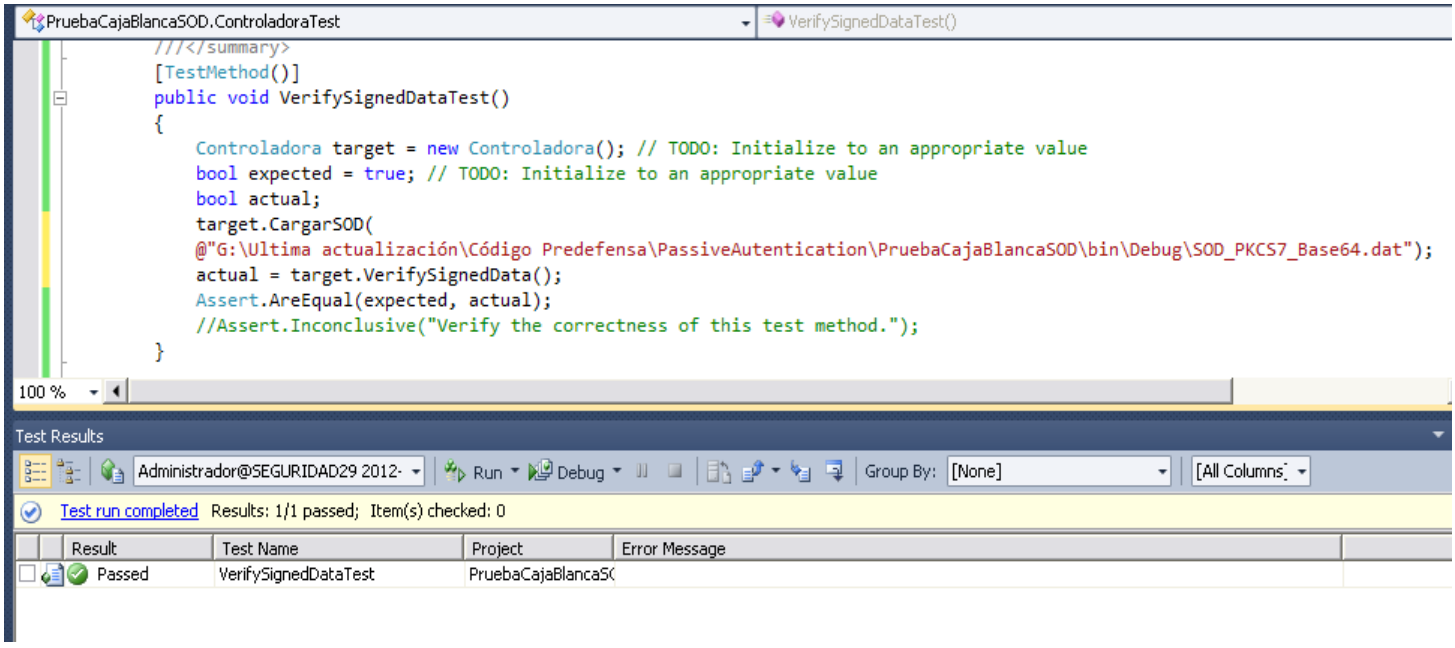

Figura 15 Prueba Unitaria. Funcionalidad: VerifySignedData.

2.1.1.2. Funcionalidad: MostrarDatosCDS.

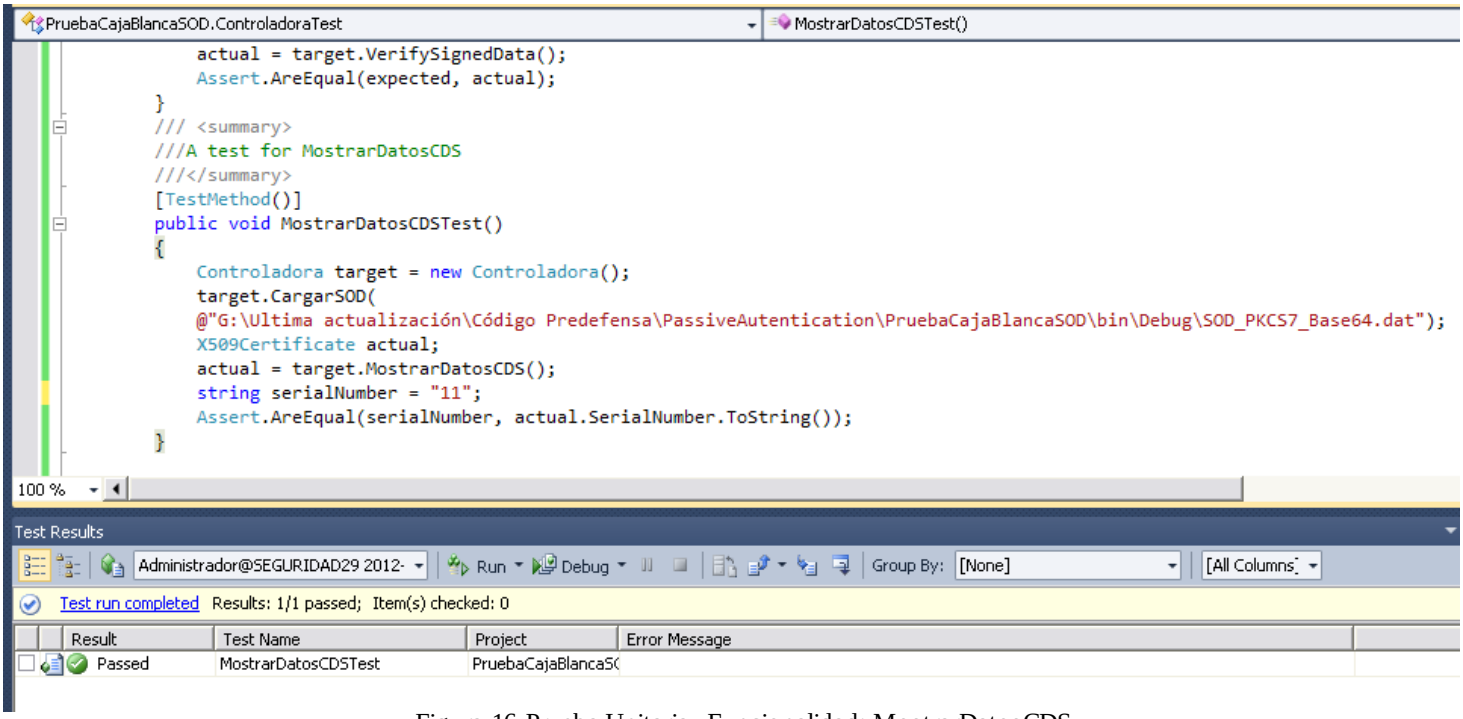

Figura 16 Prueba Unitaria. Funcionalidad: MostrarDatosCDS.

2.1.1.3. Funcionalidad: ObtenerCertificado.

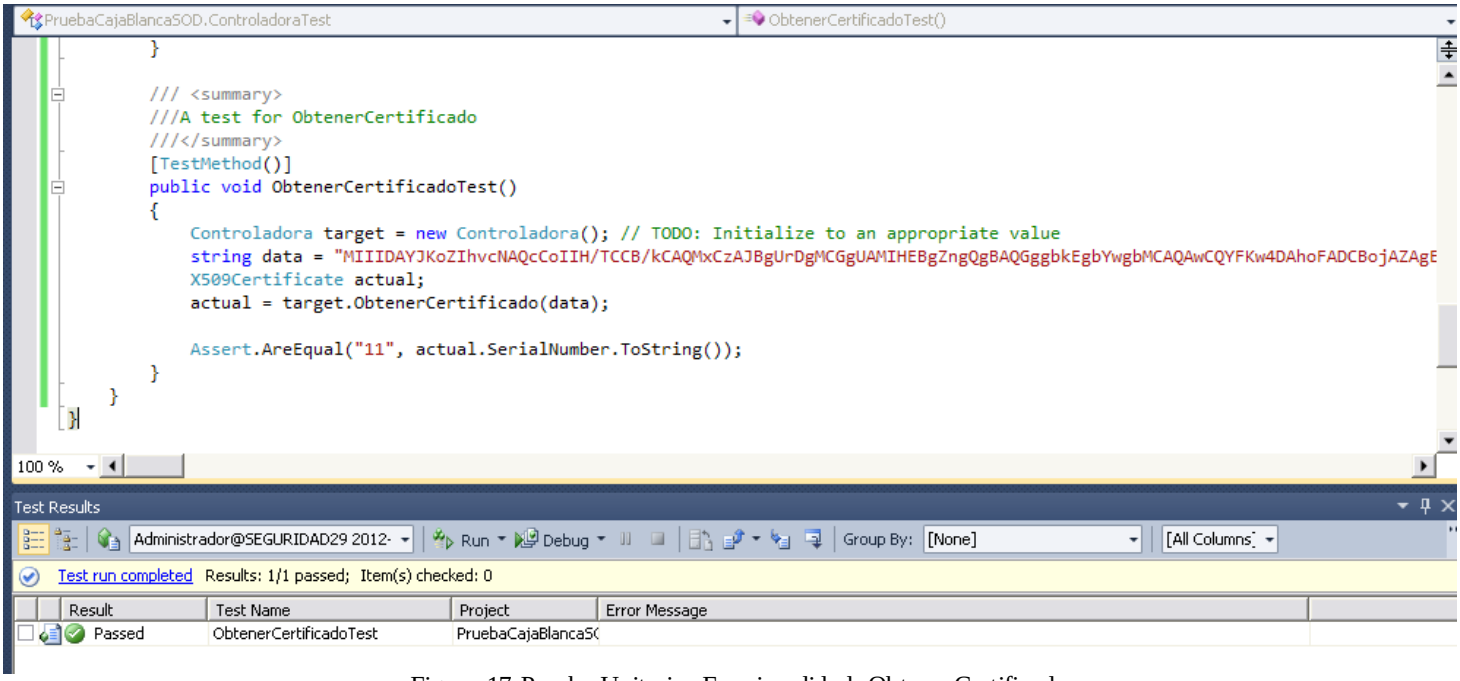

Figura 17 Prueba Unitaria. Funcionalidad: ObtenerCertificado.

2.1.2. Módulo Repositorio.

2.1.2.1. Funcionalidad: ListarCertificadosRevocados.

```
        [TestMethod()]
        public void ListarCertificadosRevocadosTest()
        {
            AdRepositorio target = new AdRepositorio();
            target.Puerto = 389;
            target.User = "cn=Manager,dc=maxcrc,dc=com";
            target.Pass = "123";
            target.Host = "192.168.103.3";
            // TODO: Initialize to an appropriate value
            List<string> expected = new List<string>();
            expected.Add("cn=9DEF8DAD");

            List<string> actual = new List<string>();
            actual = target.ListarCertificadosRevocados();
            Assert.AreEqual(expected[0], actual[0]);
        }
    }
}
```

Test Results

deyanira@LAB-103-FONSE 2012-05- | Run | Debug

Test run completed Results: 1/1 passed; Item(s) checked: 0

Result	Test Name	Project
Passed	ListarCertificadosRevocadosTest	PruebaCajaBlancaRepositorio

Figura 18 Prueba Unitaria. Funcionalidad: ListarCertificadosRevocados.

2.1.2.2. Funcionalidad: *ListarCertificado.*

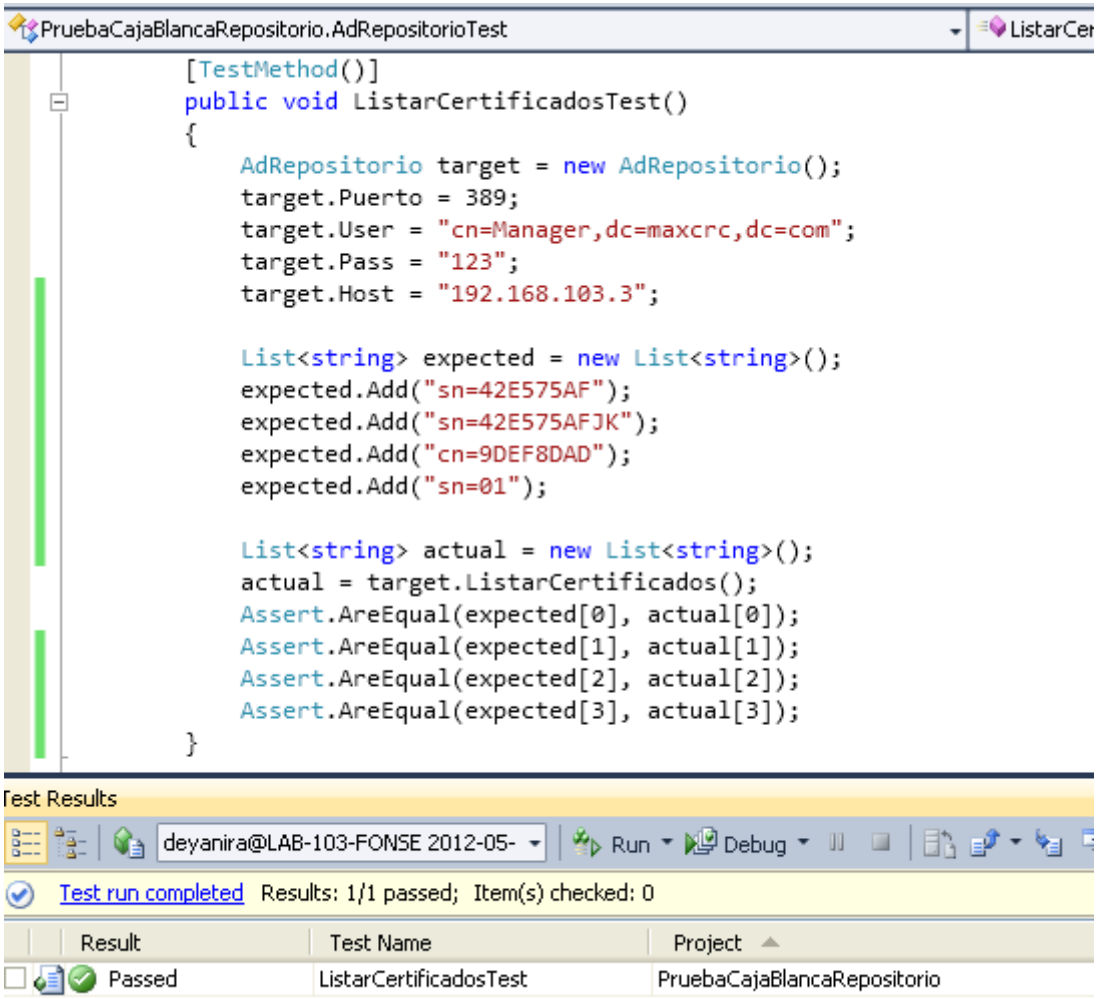

Figura 19 Prueba Unitaria. Funcionalidad: ListarCertificados.

Anexo 3

3.1. Pruebas de caja negra.

3.1.1.Módulo Verificar SOD.

Escenario: Verificar SOD.

Escenario	Descripción	Variable 1	Variable 2	Variable n	Respuesta del sistema	Flujo central
EC 1.1 Verificar Integridad	*Se procede a verificar la integridad de un objeto de seguridad de pasaporte-e una vez obtenido el objeto de seguridad del pasaporte-e.*	*V Número Serie*	*V País*	*V CSCA*	*El sistema muestra un mensaje de error "No se encontró el certificado"*	*Se selecciona el botón verificar autenticación pasiva.*
		I Número Serie	*I País*	*I CSCA*		
EC 1.2 Verificar Autenticidad	*Se procede a verificar la autenticidad de un objeto de seguridad una vez obtenida la estructura lógica de datos del pasaporte-e.*	*NA*	*NA*	*NA*		*Se selecciona el botón verificar autenticación pasiva.*
					El sistema muestra un mensaje de error "La autenticidad del pasaporte-e no ha sido verificada"	
EC.1.3 Notificar resultados de las acciones de	*Se procede a notificar los resultados luego de realizadas las acciones de verificación del pasaporte-e*	*NA*	*NA*	*NA*		*Se selecciona el botón verificar autenticación pasiva.*

verificación					*El sistema muestra un mensaje e información "Ha sido satisfactoria la verificación de la autenticación pasiva en el pasaporte-e".*	

Tabla 21 Descripción de casos de prueba al escenario "Verificar SOD"

3.1.2. Módulo Repositorio.

Escenario: Gestionar Fuente OACI PKD.

Escenario	Descripción	Variable 1	Variable 2	Variable n	Respuesta del sistema	Flujo central
EC 2.2.1 Procesar fichero ldif con CDS.	*Procede a procesar un fichero ldif con CDS para ser almacenado en el repositorio una vez descargados del PDK de la OACI.*	*V Fichero ldif*	*NA*	*NA*	*El sistema muestra un mensaje de información "El fichero se procesó correctamente"*	*Opción Gestionar Fuente OACI PKD / Procesar fichero ldif con CDS.*
		I Fichero ldif	*NA*	*NA*	*El sistema muestra un mensaje de información "El fichero no se procesó correctamente"*	

EC 2.2.2. Verificar integridad de los ldif.	*Se procede a verificar la integridad de un fichero ldif una vez descargados del PDK de la OACI.*	*V Número Serie*	*NA*	*NA*	*El sistema muestra un mensaje de información "La verificación de integridad del fichero ldif fue satisfactoria."*	*Opción Gestionar Fuente OACI PKD / Verificar integridad de los ldif.*
		I Número Serie			*El sistema muestra un mensaje de información "La verificación de integridad del fichero ldif no fue satisfactoria."*	
EC 2.2.3. Procesar Master List	*Se procede a procesar la Master List una vez descargada del PDK de la OACI.*	*V Master List*	*NA*	*NA*	*El sistema muestra un mensaje de información "Master List procesada correctamente".*	*Opción Gestionar Fuente OACI PKD / Procesar Master List.*
		I Master List	*NA*	*NA*	*El sistema muestra un mensaje de información "Master List no se procesó correctamente".*	

Tabla 22 Descripción de casos de prueba al escenario "Gestionar fuente OACI PKD"

Escenario: Listar certificados, Buscar certificados, Mostrar detalles de CDS.

Escenario	Descripción	Variable 1	Variable 2	Variable n	Respuesta del sistema	Flujo central
EC 2.3 Listar certificados	*Se procede a listar los certificados del repositorio una vez seleccionada la opción Listar certificado.*	*NA*	*NA*	*NA*	*El sistema muestra un listado con todos los certificados existentes en el repositorio.*	*Opción Gestionar Fuente OACI PKD / Listar certificados.*
EC 2.4 Buscar certificado.	*Se procede a buscar un certificado en el repositorio una vez seleccionada la opción Buscar certificado.*	*V Número Serie*	*NA*	*NA*	*El sistema muestra un mensaje de información "Se encontró el certificado en el repositorio."*	*Opción Gestionar Fuente OACI PKD / Buscar certificado*
		I Número Serie	*NA*	*NA*	*El sistema muestra un mensaje de error "No se encontró el certificado en el repositorio."*	
EC 2.5 Ver detalles de certificado.	*Se procede a ver detalles de un certificado una vez seleccionada la opción Ver detalles de un certificado.*	*V Número Serie*	*NA*	*NA*	*El sistema muestra un mensaje de información con los datos del certificado seleccionado en el repositorio.*	*Opción Gestionar Fuente OACI PKD / Ver detalles del certificado.*
		I Número Serie	*NA*	*NA*	*El sistema muestra un mensaje de error "No se encontró un certificado con ese número de serie en el repositorio"*	

Tabla 23 Descripción de casos de prueba al escenario "Listar certificados", "Buscar certificados", "Mostrar detalles de CDS"

Referencias Bibliográficas

1. *Foro TIC y Educación Metas Educativas 2021. [cited; Available from: http://www.oei.es/metas2021/foroticsyeducacion.htm.*
2. *OIM - Sistemas sobre Pasaportes y Visas. [cited; Available from: http://www.iom.int/jahia/Jahia/about-migration/managing-migration/passport-visa-systems/lang/es.*
3. *Home - OACI Home. OACI. DVLM Pasaporte-e. Volumen 1.2. 2006. [cited; Available from: http://www2.OACI.int/EN/HOME/default.aspx.*
4. *ISO/IEC 7501-1:2005 - Identification cards -- Machine readable travel documents -- Part 1: Machine readable passport. [cited; Available from: http://www.iso.org/iso/catalogue_detail.htm?csnumber=42766&utm_source=ISO&utm_medium=RSS&utm_campaign=Catalogue.*
5. *Firmas electrónicas | Kioskea.net. [cited; Available from: http://es.kioskea.net/contents/crypto/signature.php3.*
6. *Cuevas L.Y.P. Contrato de Desarrollo de Solución Integral para la Transformación y Modernización del Sistema de Identificación, Migración y Extranjería (Fase 2). 2008.*
7. *Pasaporte biométrico. [cited; Available from: http://www.commercialsecuritydevices.com/es/pasaporte_biom%C3%A9trico.html.*
8. *Seguridad en JAVA: Certificados digitales: Certificados Digitales. [cited; Available from: http://www.uv.es/sto/cursos/seguridad.java/html/sjava-17.html.*
9. *Seguridad en JAVA: Certificados digitales: Certificados X.509. [cited; Available from: http://www.uv.es/~sto/cursos/seguridad.java/html/sjava-18.html.*
10. *OACI, Regulations for the ICAO Public Key Directory. 2009: p. 8.*
11. *Configuración de la sincronización de perfiles mediante un archivo de Formato ligero de intercambio de directorios (LDIF) (SharePoint Server 2010). [cited; Available from: http://technet.microsoft.com/es-es/library/ff959234.aspx.*
12. *Seguridad en JAVA: Certificados digitales: Listas de Anulación de Certificados (CRLs). [cited; Available from: http://www.uv.es/sto/cursos/seguridad.java/html/sjava-19.html.*

13. *AGSETEC. [cited; Available from: http://www.agsetec.com/controlmigratorio.htm.*
14. *D SCAN Master biometric identity management software - Cross Match Technologies. [cited; Available from: http://www.crossmatch.com/spanish/d-scan-master.php.*
15. *D SCAN Authenticator CF – Lectora Autenticadora de Documentos, Cross Match Technologies. [cited; Available from: http://www.crossmatch.com/spanish/authenticator-cf.php.*
16. *BSI: Projects and developments. [cited; Available from: https://www.bsi.bund.de/EN/Topics/ElectrIDDocuments/Projects/projectsGRT/GRT_node.html.*
17. *Ventajas de .Net. [cited; Available from: http://www.desarrolloweb.com/articulos/1329.php.*
18. *David Morón Ruano, A.B.M., LDAP. p. 4.*
19. *Gabriela Rebeca Aguilar Baquero, O.C.A.T. Análisis e implementación de un sistema automatizado de digitalización de documentos (SADO) para soluciones inteligentes, en CIENCIAS DE LA COMPUTACIÓN. 2011, ESCUELA POLITÉCNICA DEL EJÉRCITO. Active Directory.*
20. *bouncycastle.org. [cited; Available from: http://www.bouncycastle.org/.*
21. *Ingeniería de Software Un Enfoque Práctico (Pressman 5th Ed). [cited; Available from: http://es.scribd.com/doc/7978336/Ingenieria-de-Software-Un-Enfoque-Practico-Pressman-5th-Ed.*
22. *Microsoft Colombia. [cited; Available from: http://www.microsoft.com/colombia/portafolio/msf.htm.*
23. *Metodologías ágiles (En línea). [cited; Available from: http://www.agile-spain.com*
24. *Programación Extrema o XP - EcuRed. [cited; Available from: http://www.ecured.cu/index.php/Programación_Extrema_o_XP.*
25. *Corporation, Microsoft. Process Guidance. 2006.*
26. *HI-04: Glosarios Buscar. [cited; Available from: http://aprendeenlinea.udea.edu.co/lms/moodle/mod/glossary/showentry.php?courseid=297&concept=lenguaje+de+programación.*
27. *DPA - Introducción al lenguaje de programación C#. [cited; Available from: http://decsai.ugr.es/~cb/CSharp/languaje/intro.xml.*
28. *Una sinfonía en C#: Introducción al lenguaje de programación C#. [cited; Available from: http://lomejorecsharp.blogspot.com/2011/08/introduccion-al-lenguaje-de.html.*
29. *Intel Engage: PROGRAMACIÓN JAVA. [cited; Available from: http://engage.intel.com/docs/DOC-23717.*

30. *Java. [cited; Available from: http://es.scribd.com/doc/85116599/Java.*
31. *¿Qué es UML? [cited; Available from: http://profesores.fi-b.unam.mx/carlos/aydoo/uml.html.*
32. *Grupo UML: abril 2006. [cited; Available from: http://umlpanamericana.blogspot.com/2006_04_01_archive.html.*
33. *Visual Studio 2010 Ultimate - Microsoft Visual Studio. [cited; Available from: http://www.microsoft.com/spain/visualstudio/products/2010-editions/ultimate/features.*
34. *UModel Altova 2011 Enterprise Edition Release 2 SP 1. [cited; Available from: http://descargardownloads.com/umodel-altova-2011-enterprise-edition-release-2-sp1.*
35. *Patrones Grasp « El Mundo Informático. [cited; Available from:* http://jorgesaavedra.wordpress.com/category/patrones-grasp/*.*
36. *Daniele, I.M., UML (Unified Modeling Language). 2007.*