Universidad de las Ciencias Informáticas

Facultad 1

Título: Módulo de gestión de control de acceso para las instituciones externas de las que se incorpora información en la cédula de identidad electrónica.

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

Autores: Eliatnee García Fernández
Wagner Díaz Lantigua

Tutores: Ing. Yoandy Rodríguez Martínez
Ing. Lityuen Amalia Porras Herrera

Ciudad de La Habana, Cuba

Curso 2011-2012

"Nosotros soñamos con otra libertad de prensa, en un país educado e informado, en un país que posea una cultura general integral y pueda comunicarse con el mundo"

Fidel Castro Ruz

Declaramos que somos los únicos autores del trabajo titulado **"**Módulo de gestión de control de acceso para las instituciones externas de las que se incorpora información en la cédula de identidad electrónica" y autorizo a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

_____ _____

Yoandy Rodríguez Martínez Lityuen Amalia Porras Herrera

_____ _____

Wagner Díaz Lantigua Eliatnee García Fernández

Opinión del Tutor

De Eliatnee:

- *A mis padres y mi hermano por estar ahí en los buenos y malos momentos y por ser tan especiales conmigo.*
- *A mi abuela Juana que siempre ha anhelado que yo pudiera llegar a este momento en el que me encuentro hoy, esta carrera sin tu apoyo no la hubiera alcanzado.*
- *A mis abuelos que hoy no están conmigo pero siempre los llevaré presente día a día.*
- *A mi prima Nélida por ser mi segunda mamá y estar pendiente siempre de mis estudios y mis cosas.*
- *A toda mi familia en sentido general por estar ahí presentes ante cada dificultad que se me presentará en esta larga carrera: a mis tías y tíos, a mis primos y primas.*
- *A todos los que han contribuido de una forma u otra ha facilitar mi estancia y permanencia hasta el final de esta ingeniería como Ailín, Vlady, Ernesto, Ramón, Ariel y Yumi.*
- *A todas las amistades con las que he compartido cada día de estos 5 maravillosos años las cuales están presentes hoy y otros que se han graduado ya, pero no puedo dejar de mencionar a Geny, Isunnis, Greybi, María, Madiela, Nairelis, Yelaysi, Wendy, Mariena, Marycary, las melli, Lianny, Giselle, Deyanira, las dos Damaris, Laura. A Jorge Fonseca, Argenis, Michel, Alexis. A todos en sentido general por compartir cada momento y cada fiesta.*
- *A mi compañero de tesis Wagner por aguantar todas mis peleas y contribuir con mis decisiones.*
- *A mis tutores y a todos los profesores que contribuyeron a mi formación en esta escuela y que me ayudaron cuando los necesite.*
- *A todos muchas gracias por existir. Los quiero muchooooo.*

De Wagner:

- *A toda mi familia por el apoyo brindado en el transcurso de la carrera, en especial a mis padres por ser mis guías y mi apoyo en la difícil tarea de convertirme en un ingeniero, a mi hermano, mi mujer Yulaine y a mi tía Miriam.*
- *A todos mis amigos(as) y compañeros(as) que me han ayudado todos estos años.*
- *A mi tutor Yoandy por estar día a día presente y apoyándonos en este trabajo.*
- *A todos los que de una forma u otra han contribuido a que sea quien hoy soy, muchas gracias.*

De Eliatnee:

- *A mi abuelita Juana que la quiero mucho, este logro alcanzado es para ti.*
- *A la memoria de mis tres abuelos Hilda, Medaldo y Félix.*
- *A mis padres, a mi hermano, a Nélida mi prima y a mi familia en general para que vean el fruto de toda la espera de este tiempo.*

De Wagner:

Yo quisiera dedicar este trabajo que es un resumen del esfuerzo llevado a cabo hace 5 años a mis padres que son los protagonistas principales de que hoy se materialice este logro, también a mi hermano Werner que ha sido un apoyo insustituible en los momentos de dificultad, a mi mujer Yulaine que sin ella mi vida hubiera sido un poco más complicada.

Resumen:

El presente trabajo se encarga de gestionar el control de acceso de las instituciones externas que desean almacenar información en la Cédula de Identidad electrónica (CIe) de la República Bolivariana de Venezuela. Entiéndase por instituciones externas aquellas con las que se relacionan el individuo, ejemplo: Salud pública, Sistema educacional, el transporte.

Una de las principales funciones de esta investigación es establecer la estructura de la información de las instituciones, logrando que el Servicio Administrativo de Identificación, Migración y Extranjería (SAIME) mantenga de forma segura la información almacenada en la cédula, siguiendo los estándares ISO/IEC 7816, específicamente el estándar de almacenamiento en las tarjetas inteligentes ISO/IEC 7816-4. A estas instituciones se le otorgará un permiso de lectura y/o escritura para acceder a la información de la cédula.

Para guiar el proceso de desarrollo de software se utiliza la metodología Extreme Programming (XP), Altova UModel como herramienta de modelado, como entorno de desarrollo MonoDevelop y como servidor web Apache.

Este sistema proveerá al SAIME una solución que permitirá a las instituciones hacer un mayor uso de la CIe y brindará diversos servicios a los ciudadanos posibilitando que las instituciones almacenen información en la cédula.

Palabras clave: Cédula de Identidad electrónica (CIe), Instituciones externas, sistema, tarjeta inteligente.

Índice de Figuras

Índice de Tablas

Introducción

En la República Bolivariana de Venezuela el SAIME es el encargado de controlar la identidad de los ciudadanos, así como el ingreso y permanencia de extranjeros que residen en el país. Este moderniza y actualiza sus servicios, debido a faltas que se venían cometiendo en el país refiriéndose a la seguridad de los ciudadanos, además del atraso tecnológico, las corrupciones existentes y un personal poco comprometido con el gobierno bolivariano ponían en riesgo la integridad de la institución.

La Cédula de Identidad Electrónica de la República Bolivariana de Venezuela (CIe) es una tarjeta inteligente o SmartCard que está conformada por un circuito integrado sin contacto (ICC) de 80 KB embebido en policarbonato, con un tamaño ID-1[1]. Tiene incorporado un sistema operativo basado en JavaCard, que permite la instalación de varias aplicaciones en una misma tarjeta, soportado por un entorno de ejecución que administra y controla su seguridad e integridad. La información relacionada con el ciudadano está impresa en láser grabado en la superficie del documento y contiene además características de impresión de seguridad como imágenes codificadas y microtextos.

Este documento de identificación es emitido por una autoridad administrativa competente para permitir la identificación personal de los ciudadanos, su posesión es obligatoria en la República Bolivariana de Venezuela así como en la mayoría de los países iberoamericanos. La misma se caracteriza por garantizar la identidad de las personas en todos los actos jurídicos, públicos y privados que estos realizan, por ejemplo: la realización de trámites, la declaración jurada de patrimonio y declaración de impuesto. Es por ello que el principal objetivo de dicha tarjeta es lograr la mayor seguridad del individuo, evitando falsificaciones o suplantaciones de identidad.

La misma usa aproximadamente 55 KB del espacio disponible por lo que en los 25 KB restantes sería posible almacenar información referente al ciudadano que no proceda del SAIME, posibilitando a éste identificarse a través de los medios informáticos ya sea en instituciones del gobierno como privadas. Este medio de identificación brinda una serie de servicios que ayuda tanto a la persona portadora guardando sus datos personales como a la institución que almacenará información en ella.

1 Tamaño de la tarjeta orientada por las especificaciones del estándar (ISO/IEC 7810). Las dimensiones nominales son: 53,98 mm ´ 85,6 mm (2,13 in ´ 3,37 in).

Los datos que se pueden introducir en la tarjeta son: en el caso de la historia clínica de un paciente por ejemplo, medicinas que toma, las enfermedades que padece, el tipo de sangre, si es donante de órganos, entre otros, posibilitando el acceso a esta información en ambientes desconectados como es el caso de una ambulancia; en el caso del sistema de transporte masivo se almacena la licencia de conducción; también en el sistema educativo información referente del estudiante y todas sus actividades docentes; en el Servicio Nacional Integrado de Administración Tributaria (SENIAT) para almacenar en la tarjeta los datos asociados al registro fiscal del ciudadano.

Actualmente en Venezuela existen diversas instituciones las cuales se relacionan con el individuo y almacenan la información de estos de forma aislada desaprovechando la posibilidad de incluir esta información en el espacio no utilizado de la cédula de identidad. El gobierno venezolano ha decidido resolver esta situación posibilitando a las instituciones almacenar los datos en la cédula. Sin embargo, hoy en día no hay forma de definir qué parte del espacio sobrante de la tarjeta corresponde a cada institución ni de lograr que el acceso a esa información sea de forma segura, de manera que solo pueda escribir y leer los datos las instituciones autorizadas.

Dada la problemática planteada se identifica como **problema científico:** ¿Cómo apoyar la gestión de control de acceso de la información almacenada por instituciones externas en la CIe?

El **objeto de estudio:** el proceso asociado a la gestión de control de acceso de la información.

Se define como **objetivo general** de esta investigación: Desarrollar una aplicación *web* que permita la gestión de control de acceso de la información almacenada por instituciones externas en la CIe.

Siendo el **campo de acción:** proceso asociado a la gestión de control de acceso de la información almacenada por instituciones externas en la CIe.

Para darle cumplimiento al objetivo antes mencionado se precisan las siguientes **tareas de investigación:**

- Realización del marco teórico – metodológico de la investigación.
- Análisis de los medios de identificación existentes.
- Investigación de posibles tecnologías, metodologías y herramientas para la solución del problema.

- Diseño de la aplicación Módulo de gestión de control de acceso para las instituciones externas de las que se incorpora información en la cédula de identidad electrónica.
- Generación de los artefactos que describen el proceso de ingeniería de *software* según la metodología seleccionada.
- Realización del Módulo de gestión de control de acceso para las instituciones externas de las que se incorpora información en la cédula de identidad electrónica.
- Validación de las pruebas realizadas a la aplicación.

Para el desarrollo de la investigación se propone la utilización de los **métodos investigativos teóricos** siguientes:

- Histórico-lógico: se usa con el objetivo de estudiar los sistemas existentes de control de acceso a la información, así como las metodologías que rigen el proceso de desarrollo del *software*, lenguajes de modelado, herramientas de desarrollo y tecnologías a utilizar.
- Analítico-Sintético: se utiliza para extraer e identificar conceptos, características y otros elementos de la bibliografía consultada que posteriormente ayudan a establecer una propuesta adecuada a las necesidades del sistema. Luego del análisis de esta información es necesario organizarla y sintetizarla para la elaboración de una proposición que contenga una estructura apropiada con los elementos fundamentales del objeto estudiado.

Como **justificación de la investigación** se plantea: El desarrollo de esta investigación proporcionará al SAIME y al gobierno establecer una comunicación entre la CIe de cada individuo y las instituciones con las que se relaciona, almacenando información proveniente de ellas.

La presente investigación está estructurada en tres capítulos los cuales se detallan a continuación:

Capítulo 1: Fundamentación Teórica

En este capítulo se reflejan los conceptos fundamentales abordados. Se realiza un estudio de los sistemas similares a la solución a implementar y se hace un análisis sobre las metodologías, tecnologías, herramientas y plataformas existentes para desarrollar el software.

Capítulo 2: Análisis y diseño

En este capítulo se describe la arquitectura de la aplicación, llevando a cabo un estudio del dominio del sistema. Se identifican las diferentes historias de usuario, se realiza el plan de iteraciones y plan de entrega rigiéndose por la metodología seleccionada.

Capítulo 3: Implementación y Prueba

En este capítulo se aborda los procedimientos relacionados con la implementación de la solución, se construye el modelo de despliegue y se realizan las pruebas que certifican el correcto funcionamiento de los requisitos funcionales.

Capítulo I: Fundamentación Teórica

1.1 Introducción

En el presente capítulo se plasma una breve explicación de los términos relacionados con el dominio del sistema y se realiza un estudio de los diferentes sistemas similares acerca del control de acceso a la información. Además se analizan las diferentes tecnologías, metodologías y herramientas en cuanto a sus aspectos más importantes para darle solución a la problemática planteada.

1.2 Cédula de Identidad

Hoy en día en diversos países se utilizan las tarjetas inteligentes como mecanismos de identificación, lo que varían de acuerdo a sus características y a la información que cada una de ellas posee, adaptadas a las necesidades de cada país. Los gobiernos a nivel mundial se preocupan por brindar a sus habitantes servicios más eficientes en cuanto a identificación personal.

La cédula de identidad es un documento que acredita la identidad de una persona. Es de carácter personal e intransferible, y constituye el documento principal de identificación para los actos civiles, mercantiles, administrativos y judiciales, y para todos aquellos casos en los cuales su presentación sea exigida por la ley. (1)

El gobierno de la República Bolivariana de Venezuela, con el fin de mejorar la eficiencia de los servicios de identificación de ciudadanos y responder a las necesidades que impone la creciente informatización del mundo moderno, concibe un nuevo documento de identificación, la Cédula de Identidad Electrónica. La cédula está compuesta por tres partes: La parte superior contiene la clave del documento, el estado u organismo expedidor, el número del documento, y elementos a discreción del estado que expide el documento; la parte intermedia contiene la fecha de nacimiento del titular, la fecha de caducidad del documento, la nacionalidad, un espacio para datos opcionales que al igual que los de la primera línea son a discreción del estado que emite el documento, la tercera y última parte contiene los nombres y apellidos del titular.

1.3 Estándar para el almacenamiento de datos en tarjetas inteligentes.

Internacionalmente se han definido estándares para lograr la interoperabilidad entre distintos fabricantes de tarjetas inteligentes y lectores de las mismas, en lo que respecta a características físicas, comunicación de datos y seguridad. La ISO/IEC 7816 (por sus siglas en inglés International Organization for Standardization/International Electrotechnical Commission) define estándares para la fabricación y uso de tarjetas inteligentes, este estándar contiene 15 partes y van desde la ISO/IEC 7816-1 hasta la ISO/IEC 7816-15.

Para la implementación de la solución se utiliza el estándar ISO/IEC 7816-4 para el almacenamiento de datos en tarjetas inteligentes debido a que ha sido definido con anterioridad por el proyecto. Además posee características que se ajustan al objetivo final de la investigación, ya que establece una estructura de la información a almacenar y crea los mecanismos de seguridad a través de las condiciones de acceso. El ISO/IEC 7816-4 contiene una serie de comandos de intercambio inter-industriales y define:

- El contenido de los pares comando-respuesta que se intercambian a nivel de interfaz.
- Estructuras para aplicaciones y datos en la tarjeta.
- La estructura y contenido de los caracteres históricos de la respuesta de *Reset*, los cuales describen las características de operación de la tarjeta inteligente.
- Métodos para extracción de objetos y elementos de datos de la tarjeta.
- Estructuras de archivos y métodos de acceso.
- Arquitectura de seguridad para derechos de acceso a los archivos y datos en la tarjeta. (2)

1.3.1 Estructura para aplicaciones y datos

Existen en la tarjeta dos categorías de estructura: los ficheros dedicados (DF siglas en inglés) y los ficheros elementales (EF siglas en inglés). Los DF están compuestos por grupos de ficheros, que pueden ser DF o EF y los EF son ficheros para almacenar datos y no contienen más ningún fichero.

Existen dos categorías de ficheros elementales:

- EF que almacena información interpretada por la tarjeta para su administración y control. Como las condiciones de acceso que posee ese fichero.

- EF que almacena información no interpretada por la tarjeta, información que se usa fuera de la tarjeta. Tal es el caso de la información de las instituciones.

Existen dos tipos de organización lógica de los ficheros dedicados y los ficheros elementales.

La figura 1 muestra la jerarquía de los DF con su correspondiente arquitectura. En la organización que se presenta, el DF raíz, es llamado fichero maestro (MF siglas en inglés). Cualquier DF puede ser un grupo de ficheros con o sin jerarquía alguna.

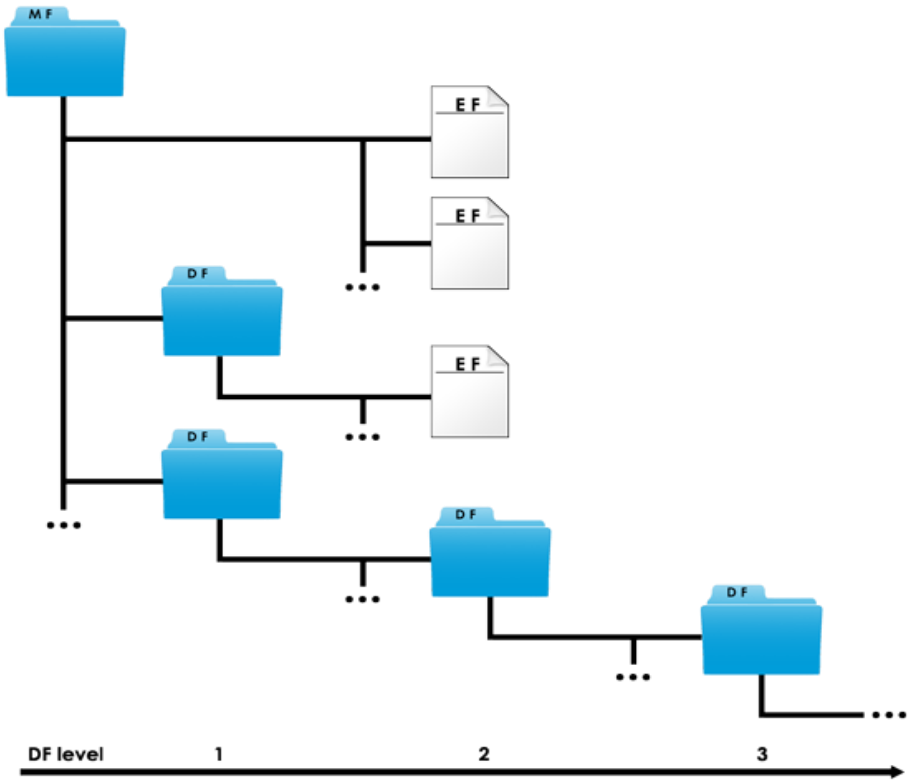


Figura 1: Jerarquía de ficheros (2)

1.3.2 Atributos de seguridad

Los atributos de seguridad de un archivo dependen de su categoría (Fichero Directorio o Fichero Elemental) y de los parámetros opcionales en su información de control y/o la información de control de su nodo padre. En particular, los atributos de seguridad deben:

- Especificar el estado de seguridad de la tarjeta actual antes de acceder a los datos.
- Restringir el acceso a datos a determinadas funciones (por ejemplo, sólo lectura) si la tarjeta tiene un estado en particular.
- Definir qué funciones de seguridad deben ser realizadas para obtener un estado de seguridad específico. (2)

1.4 Sistemas similares

Sistema para Comercios y venta al público

Este es un sistema que maneja información relativa a clientes, proveedores, cuentas corrientes, ciclo completo de documentos comerciales (presupuestos, pedidos, facturas), formas de pago, manejo de recibos, informes y cuentas bancarias. Dicho sistema proviene de Silix, una empresa dedicada a brindar soluciones informáticas que permitan a sus clientes aprovechar las ventajas del software libre. Cuenta con varios módulos entre los cuales se destacan:

- Principal
- Facturación
- Tesorería
- Informes
- Control de acceso

El módulo control de acceso brinda la posibilidad de registrar a los usuarios del sistema y otorgarles permisos específicos de acceso a la información o realización de operaciones. Entre sus principales funcionalidades resaltan:

- Gestión de usuarios y grupos de usuarios.
- Gestión de listas de control de acceso para administrar los permisos y privilegios dentro del sistema y sus módulos. (3)

TopAcesso TOPDATA

TopAcesso es un sistema de control de acceso de personas a ambientes internos dentro de establecimientos corporativos, industrias, instituciones financieras, condominios comerciales y escuelas. Permite que los lectores sean configurados de acuerdo con las propias necesidades, para ofrecer total control - desde el acceso a las porterías hasta el control de ambientes restrictos, como tesorerías y locales de almacenaje.

Facilita varias posibilidades de configuración. Se puede establecer el control deseado, con los datos que se crea más conveniente. Por ejemplo: bloqueo de doble entrada o salida, apuntamiento de mensajes para determinado usuario, control de la ruta que va a ser seguida por un determinado visitante. Se utilizan en Plantas fabriles, Centros de datos, Bancos, Tesorerías, Laboratorio.
Especificaciones Técnicas

- Control on-line de accesos a ambientes.
- Control de puertas y/o barreras.
- Banco de Datos Access y/o SQL Server.
- Cambio automático al modo off-line en caso de interrupción de la comunicación entre el servidor y los lectores. (4)

Los sistemas estudiados aunque no se enmarcan de forma específica al objetivo de la investigación, tributan a un conocimiento acerca del control de acceso a una información determinada, posibilitando que el desarrollo del sistema a implementar se lleve a cabo de una forma más abarcadora.

1.5 Metodología de desarrollo

Las metodologías de desarrollo contienen un conjunto de procedimientos, técnicas y documentos que guían el ciclo de vida del software. Se conocen las pesadas y las ágiles, la primera está basada en una buena documentación y planificación detallada de todo el software, cuenta con procesos muy controlados, con numerosas políticas/normas y sus costos son elevados en cuanto algún cambio que haya que hacerse. Las ágiles son más eficientes a la hora de realizar cambios, son flexibles, se simplifican los

costos y el software no se planifica, se adapta a las necesidades que se presenten. Entre las principales metodologías ágiles se conocen XP (eXtreme Programming), Scrum, Iconix, Cristal Methods, AUP.

Proceso Unificado de Racional (RUP)

RUP es una metodología desarrollo de software que integra los aspectos a tener en cuenta durante todo el ciclo de vida del software, con el objetivo de hacer abarcables tanto pequeños como grandes proyectos de software. Además proporciona herramientas para todos los pasos del desarrollo así como documentación en línea para sus clientes.
Las características principales de RUP son:

- Guiado/Manejado por casos de uso.
- Centrado en arquitectura.
- Iterativo e Incremental. (5)

XP

XP es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo. Se basa en la realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP es especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, donde existe un alto riesgo técnico.

Las características fundamentales de la metodología XP son:

- Diseño simple: Se debe diseñar una solución lo más simple que pueda funcionar y ser implementada en un momento determinado del proyecto. La complejidad innecesaria y el código extra debe ser removido inmediatamente.

- Pruebas: Los clientes escriben las pruebas funcionales para cada historia de usuario que deba validarse.

- Refactorización: es una actividad constante de reestructuración del código con el objetivo de remover duplicación de código, mejorar su legibilidad, simplificarlo y hacerlo más flexible para facilitar los posteriores cambios.

- Programación por parejas: se detectan muchos errores a medida que son introducidos en el código (inspecciones de código continuas), por consiguiente la tasa de errores del producto final es más baja, los diseños son mejores, el tamaño del código menor y los problemas de programación se resuelven más rápido.

- Propiedad colectiva del código: Cualquier programador puede cambiar cualquier parte del código en cualquier momento. Esta práctica motiva a todos a contribuir con nuevas ideas en todos los segmentos del sistema, evitando a la vez que algún programador sea imprescindible para realizar cambios en alguna porción de código. (6)

El ciclo de vida de XP incluye cuatro fases ver figura 2.

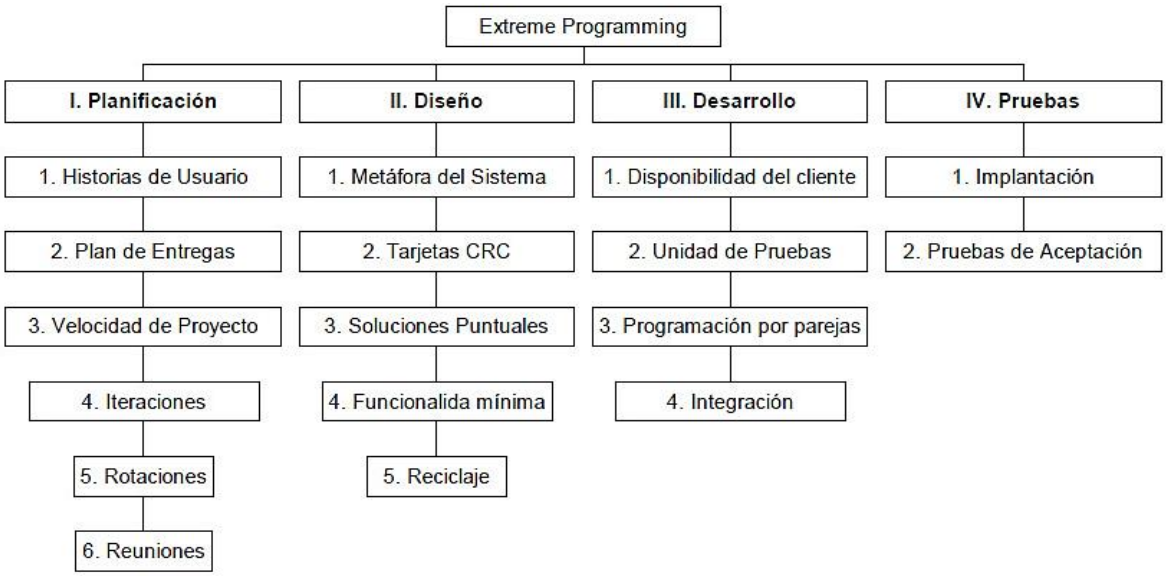


Figura 2: Fases de la metodología XP (7)

Para la guía del ciclo de vida del software después de un estudio realizado de las diferentes metodologías existentes se seleccionó XP porque se ajusta a las características del software ya que el mismo es corto, con un grupo de desarrolladores pequeño, permite realizar cambios frecuentes, se realizan iteraciones a lo

largo de todo el ciclo de vida para ver cómo va avanzando el software, se enfatiza más en la programación y no en una larga documentación.

1.6 Tecnologías, lenguajes y herramientas

A continuación se describen las tecnologías, lenguajes y herramientas que tributan a la construcción del sistema a implementar, destacando sus aspectos más importantes.

1.6.1 Entorno de desarrollo integrado

Microsoft Visual Studio es un entorno de desarrollo integrado (IDE, por sus siglas en inglés) para sistemas operativos Windows. Soporta varios lenguajes de programación tales como Visual C++, Visual C#, Visual J#, ASP.NET y Visual Basic .NET. Se utiliza para realizar tareas de desarrollo básico, simplifica la compilación y la depuración. También viene con el soporte integrado para el desarrollo con pruebas y con las herramientas de depuración que ayudan a garantizar unas soluciones de alta calidad. Visual Studio permite a los desarrolladores crear aplicaciones, sitios y aplicaciones web, así como servicios web en cualquier entorno que soporte la plataforma .NET. (8)

MonoDevelop

MonoDevelop es un IDE diseñado principalmente para C# y otros lenguajes .NET. Permite a los desarrolladores escribir rápidamente aplicaciones de escritorio y web ASP.NET en Linux, Windows y Mac OSX. Permite la creación de proyectos diversos ya sean simples o complejos.

Las características más importantes son:

- Multi-plataforma: Compatible con Linux, Windows y Mac OSX.
- Soporte para múltiples idiomas C #, Visual Basic.Net, C / C + +.
- Depurador Integrado: Por Mono depuración y aplicaciones nativas.
- ASP.NET: Crea proyectos web con soporte completo de finalización de código y pruebas de XSP.
- Otras herramientas: Control de código fuente, las pruebas unitarias, empaquetado y la implementación. (9)

Como entorno integrado de desarrollo se utiliza MonoDevelop por ser libre a diferencia del Visual Studio que es un software propietario. Este se ejecuta en diversos sistemas operativos al contrario de Visual Studio que se ejecuta solo en Windows. El proceso de instalación de MonoDevelop es mucho más rápido

que el de Visual Studio.

1.6.2 Lenguaje de programación

C# es un lenguaje orientado a objetos que posibilita a los desarrolladores crear una amplia gama de aplicaciones sólidas y seguras que se ejecutan en .NET Framework. Permite crear aplicaciones cliente para Windows, servicios web XML, componentes distribuidos, aplicaciones cliente-servidor, aplicaciones de base de datos.

Sus características fundamentales son:

- Lenguaje de programación orientada a objetos, simple, moderna y de propósito general.
- Inclusión de principios de ingeniería de software tales como revisión estricta de los tipos de datos, revisión de límites de vectores, detección de intentos de usar variables no inicializadas y recolección de basura automática.
- Capacidad para desarrollar componentes de software que se puedan usar en ambientes distribuidos.
- Portabilidad del código fuente.
- Fácil migración del programador al nuevo lenguaje, especialmente para programadores familiarizados con C, C++ y Java.
- Soporte para internacionalización.
- Adecuación para escribir aplicaciones de cualquier tamaño: desde las más grandes y sofisticadas como sistemas operativos hasta las más pequeñas funciones.
- Aplicaciones económicas en cuanto a memoria y procesado. (10)

JavaScript

JavaScript es una de las múltiples maneras que han surgido para extender las capacidades del lenguaje HTML. Es un lenguaje de script, interpretado y orientado a objetos, dinámico y que responde a eventos. En las aplicaciones cliente de JavaScript, este se encarga de detectar eventos dentro de una página web y dar respuesta a dichos eventos, ya sean eventos iniciados por el propio usuario como: clic en un botón, pasar sobre una imagen, eventos propios de la página como: ejecución de un evento pasado un determinado tiempo, apertura de una página.

JavaScript es un lenguaje muy utilizado y dentro de sus principales aplicaciones están:

- La validación de formularios dentro de una página.
- La personalización de la página por el usuario, que le permite tener una página web a su medida.
- La inclusión de datos del propio sistema, como son la hora y la fecha.
- El dar respuesta a eventos locales dentro de la página, como apretar un botón. (11)

Características del lenguaje.

- Simple: el desarrollador web puede crear páginas dinámicas sin necesidad de convertirse en un programador experto.
- Dinámico: Una página web se convierte en dinámica cuando responde a eventos generados en el servidor. Con documentos HTML tradicionales, las aplicaciones del extremo del servidor tenían la responsabilidad de manejar los eventos pero JavaScript transfiere esa responsabilidad al extremo del cliente, lo que hace el diseño de las páginas web más flexibles, dinámicas y con una respuesta más rápida.
- No es un lenguaje de programación: JavaScript no es un lenguaje de programación. Es un lenguaje script u orientado a documento, como pueden ser los lenguajes de macros que tienen muchos procesadores de texto. Con JavaScript se pueden mejorar las páginas web haciendo cosas sencillas como: revisión de formularios, efectos en la barra de estado y no tan sencillas como: animaciones usando HTML dinámico.
- JavaScript está basado en objetos: No es un lenguaje orientado a objetos como Java o C++ pero interacciona con los objetos. (12)

1.6.3 Gestor de base datos

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD y con su código fuente disponible libremente. Utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando. Se caracteriza por su estabilidad, potencia, robustez, facilidad de administración e implementación de estándares. Funciona muy bien con grandes cantidades de datos y una alta concurrencia de usuarios accediendo a la vez al sistema. (13)

Oracle

Oracle es un sistema de gestión de base de datos relacional fabricado por Oracle Corporation. Oracle es básicamente un herramienta cliente/servidor para la gestión de base de datos, tiene gran potencia y su elevado precio hace que solo se use en empresas muy grandes y multinacionales.

Sus características fundamentales se reflejan a continuación:

Ha sido diseñado para que las organizaciones puedan controlar y gestionar grandes volúmenes de contenidos no estructurados en un único repositorio con el objetivo de reducir los costos y los riesgos asociados a la pérdida de información. (14)

Después de analizar los gestores de base de datos se plantea utilizar PostgreSQL, ya que emplea un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Se ejecuta en casi todos los principales sistemas operativos: Linux, Unix, Mac OS, Windows. Además su costo en cuanto a soporte y uso es libre, en cambio Oracle se utiliza en empresas muy grandes y multinacionales, es un producto de elevado precio en cuanto a soporte técnico y mantenimiento, se hace necesario aplicar parches de seguridad.

1.6.4 Herramienta de modelado

Visual Paradigm

Visual Paradigm es una herramienta que utiliza UML (Lenguaje Unificado de Modelado) como lenguaje de modelado. Es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software. El software de modelado UML ayuda a una construcción más rápida de aplicaciones de calidad, y a un menor costo. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. (15)

Altova UModel

Altova UModel es una herramienta de modelado basada en UML. Permite generar código de los lenguajes Java o C# a partir de sus modelos, hacer ingeniería inversa de programas existentes a diagramas UML para abarcar rápidamente la arquitectura de software. UModel soporta la especificación de intercambio

XMI (Xml Metadata Interchange) en su versión 2.1 para abrir y editar modelos creados en herramientas. Tiene una interfaz visual y usabilidad que ayuda a disminuir la curva de aprendizaje de UML.

Algunas de las características principales de esta herramienta:

- Ayuda para los 13 tipos del diagrama de UML.
- Fuente de generación de código en Java, C # y lenguajes de VB.NET
- Modelado del esquema de XML en diagramas de UML.
- Ingeniería reversa de Java, C#, y código de VB.NET.
- Sincroniza el modelo y el código con la ingeniería del viaje de ida y vuelta.
- Distribución de los subproyectos para la colaboración o la reutilización.
- Generación de documentación adaptable del proyecto.
- Enlace hipertexto entre los diagramas, los documentos o las web pages. (16)

Aunque se lleva a cabo un estudio de estas dos herramientas de modelado y tienen facilidades similares, se determina utilizar Altova UModel, la cual ha sido seleccionada con anterioridad por el equipo de desarrollo y el cliente.

1.6.5 Lenguaje de modelado

UML es un lenguaje gráfico para visualizar, especificar, construir y documentar los artefactos de un sistema. Proporciona una forma estándar de escribir los planos de un sistema, tales como procesos del negocio, funciones del sistema, clases escritas en un lenguaje de programación específico, esquemas de bases de datos y componentes *software* reutilizables. Es la sucesión de una serie de métodos de análisis y diseño orientados a objetos. Mediante UML es posible establecer la serie de requisitos y estructuras necesarias para plasmar un sistema de *software* previo al proceso intensivo de escribir código.

Los objetivos de UML se sintetizan en las siguientes funciones:

- Visualizar: Expresar de forma gráfica un sistema de forma que otro sistema lo pueda entender.
- Especificar: Especificar cuáles son las características de un sistema antes de su construcción.
- Construir: A partir de los modelos especificados se pueden construir los sistemas diseñados.
- Documentar: Los propios elementos gráficos sirven como documentación del sistema desarrollado que pueden servir para su futura revisión. (17)

1.6.6 Servidor web

Apache es un servidor web potente y flexible que funciona en diferentes plataformas y entornos. Soporta de forma fácil y eficiente una amplia variedad de sistemas operativos. El servidor es personalizable de acuerdo a las necesidades de cada sitio web. El servidor Apache es un servidor web HTTP (por sus siglas en inglés Hyper Text Transfer Protocol) de código abierto, para plataformas Unix (BSD, GNU/Linux), Microsoft Windows, Macintosh entre otras.

Ventajas

- Código abierto.
- Multi-plataforma.
- Extensible.
- Popular (fácil conseguir ayuda/soporte). (18)

WAMP

Un servidor WAMP es una computadora con Windows que dispone de un servidor Apache, un gestor de bases de datos MySQL y el lenguaje de programación PHP. Las siglas WAMP son un acrónimo de Windows más Apache más MySQL más PHP, permite instalar aplicaciones web accesibles desde una red local. El uso de este servidor permite servir páginas html a internet, además de poder gestionar datos en ellas y proporciona lenguajes de programación para desarrollar aplicaciones web. (19)

Apache es el servidor web que por sus características antes expuesta es el seleccionado para usarlo en dicha aplicación ya que este soporta varios sistemas operativos, es una tecnología gratuita de código abierto y posee una alta configurabilidad.

1.6.7 Framework de desarrollo

Framework JQuery

Es una biblioteca de clases o Framework de Java Script, creada inicialmente por John Resig, es un *software* libre y de código abierto, se puede usar tanto en *software* libre como privativo. Este *framework* se usa para mejorar sitios web, se encarga de simplificar el tiempo a la hora de interactuar con documentos

HTML, desarrollar animaciones entre otras funcionalidades de Java Script como DOM, eventos, efectos y AJAX. (20)

Algunas de las ventajas de esta librería son:

- Tiene una amplia gama de plugins disponibles para las diversas necesidades específicas.
- Evita escribir muchas líneas de código.
- Provee un mecanismo para la captura de eventos.
- Provee un conjunto de funciones para animar el contenido de la página en forma muy sencilla.
- Por defecto integra funcionalidades para trabajar con AJAX. (21)

.NET Framework

El framework .NET facilita la programación de aplicaciones ya que encapsula operaciones complejas en instrucciones sencillas. Mediante la utilización de éste se estructura el código fuente, forzando al desarrollador a crear código más legible. Además representa una arquitectura de software que modela las relaciones generales de las entidades del dominio. Provee una estructura y una metodología de trabajo las cuales extienden o utilizan las aplicaciones del dominio.

Es usado también para el desarrollo de aplicaciones web ASP.NET entre los que se encuentran sitios web dinámicos, permitiendo a los desarrolladores escribir código ASP.NET usando cualquier lenguaje admitido en la plataforma .NET. La biblioteca de clases .NET Framework incluye ADO.NET, ASP.NET, formularios Windows Forms y Windows Presentation Foundation (WPF). Proporciona un entorno de ejecución administrado, un desarrollo e implementación simplificada y la integración con una gran variedad de lenguajes de programación. (22)

Framework ASP.NET MVC

ASP.Net MVC es un *framework* que divide la implementación de una aplicación en tres roles: modelos, vistas y controladores. Uno de los beneficios de usar MVC es que ayuda a mantener una separación limpia entre modelos, vistas y controladores en la aplicación. Manteniendo una separación clara de conceptos hace que las pruebas de las aplicaciones sean mucho más fáciles, ya que los contratos entre diferentes componentes de la aplicación están mejor definidos y articulados. Algunas de las ventajas de este Framework son:

- Es extensible.

- Incluye un potente mapeador de URL que nos permite crear aplicaciones con URLs limpias.
- El Framework MVC soporta los archivos: ASPX, ASCX y MASTER.
- Soporta todas las características de ASP.NET como ventanas de autenticación, autorización, mediante roles, mapeo de datos, administración del estado de la sesión y un sistema de configuración. (23)

NHibernate

NHibernate es la conversión de Hibernate de lenguaje Java a C# para su integración en la plataforma .NET, se puede ejecutar usando MonoDevelop. NHibernate permite al desarrollador el acceso a datos, soporta diferentes gestores de base datos como: MySQL, PostgreSQL, Oracle, MS SQL Server. Para especificar la base de datos a utilizar es necesario cambiar una línea en el fichero de configuración. NHibernate es software libre, distribuido bajo los términos de la LGPL (Licencia Pública General Menor de GNU). (24)

1.6.8 JavaScript asíncrono y XML (Ajax)

Ajax, acrónimo de Asynchronous JavaScript y XML, que se puede traducir como "JavaScript asíncrono más XML", es un conjunto de técnicas para la creación de sitios web altamente interactivos y aplicaciones web. Ajax es un término genérico para las técnicas que se utilizan para hacer que las aplicaciones web se parezcan a las aplicaciones de escritorio. Ajax no es una tecnología en sí mismo. En realidad, se trata de varias tecnologías independientes que se unen. Las tecnologías que forman AJAX son:

- XHTML y CSS, para crear una presentación basada en estándares.
- DOM, para la interacción y manipulación dinámica de la presentación.
- XML, XSLT y JSON, para el intercambio y la manipulación de información.
- XMLHttpRequest, para el intercambio asíncrono de información.
- JavaScript, para unir todas las demás tecnologías. (25)

En la figura 3 se muestran las tecnologías agrupadas que contiene Ajax.

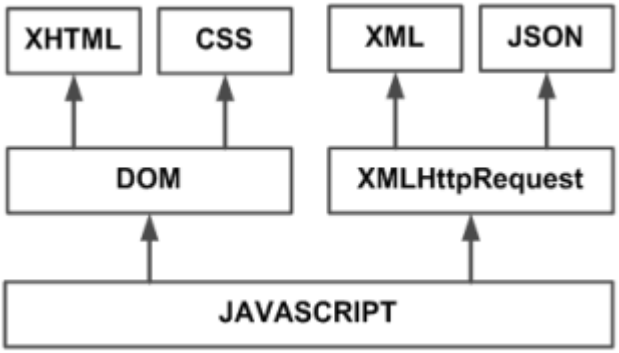


Figura 3: Tecnologías agrupadas bajo el concepto de AJAX (25)

1.6.9 Representational State Transfer (REST)

REST es el acrónimo de Transferencia de Estado Representacional, es un estilo de arquitectura desarrollado como un modelo abstracto de la arquitectura web para orientar el rediseño y la definición del protocolo HTTP y URI. REST no es un estándar, ya que es tan solo un estilo de arquitectura. Aunque no es un estándar, está basado en estándares: HTTP, URL, Representación de los recursos: XML/HTML/GIF/JPEG. (26)

Sus características más destacadas son:

Identificación de recursos:

- Todos los elementos accedidos remotamente tienen una identificación (URI).
- Se elige una representación para cada recurso (XML, JSON[2]).

Manipulación de recursos:

- Uso de los verbos HTTP para implementar CRUD.
- Las peticiones y respuestas incluyen metadatos (cache, validez).
- Incluidos en los *headers* de HTTP.

Mensajes auto-descriptivos:

- Los metadatos permiten tratar el mensaje. (27)

Los objetivos de este estilo de arquitectura se listan a continuación:

2 Es un acrónimo de JavaScript Object Notation.

- Escalabilidad de la interacción con los componentes. La web ha crecido exponencialmente sin degradar su rendimiento. Una prueba de ellos es la variedad de clientes que pueden acceder a través de la web: estaciones de trabajo, sistemas industriales, dispositivos móviles.
- Generalidad de interfaces. Gracias al protocolo HTTP, cualquier cliente puede interactuar con cualquier servidor HTTP sin ninguna configuración especial. (28)

1.6.10 ActiveRecord

Castle ActiveRecord es un *framework* libre desarrollado por el proyecto *Castle* que se encarga de la configuración de la persistencia a través del *framework* NHibernate. ActiveRecord permite la creación de componentes persistentes mediante el uso de atributos en clases de C# con la configuración mínima. El *framework* incluye además facilidades para la validación de datos. Active Record es un mecanismo de Mapeo Objeto-Relacional (Object Relational Mapping - ORM).

Ayuda a realizar todas las tareas de las bases de datos sin tener que escribir una sentencia SQL, representa a las tablas de las base de datos a través de clases e instancias. Una clase que trabaja en relación a una tabla de base de datos es llamada modelo, las instancias de estas clases son instancias de modelo. Los modelos encapsulan la lógica de negocios. También soporta la mayoría de las bases de datos tales como Oracle, SqlServer, Postgresql. (29)

1.6.11 Backbone

Backbone.js es una librería que provee una API para la creación de aplicaciones RIA basadas en el patrón arquitectónico MVC. Incluye un motor de plantillas, componentes para persistencia de datos a través de REST y un modelo de eventos para el DOM. (30)

1.7 Conclusiones parciales

En este capítulo para el desarrollo del Módulo de gestión de control de acceso para las instituciones externas de las que se incorpora información en la cédula de identidad electrónica se llegaron a las siguientes conclusiones:

- Se realizó un estudio de los sistemas similares existentes proporcionando una amplia visión de sus funcionalidades para darle solución al problema existente.

- Se analizó el estándar de almacenamiento utilizado en las tarjetas inteligentes ISO/IEC 7816-4 para usarlo en el almacenamiento de la información proveniente de las instituciones externas proporcionando una estructura más detallada de la información.
- Se seleccionaron las herramientas, metodologías, tecnologías y lenguajes a utilizar en la implementación de este módulo permitiendo el desarrollo del mismo.

Capítulo II: Análisis y diseño

2.1 Introducción

En el presente capítulo se describe detalladamente la propuesta de solución del Módulo de gestión de control de acceso para las instituciones externas de las que se incorpora información en la cédula de identidad electrónica, para ello se identifican los procesos de negocio relacionados con el objeto de estudio. Se plantean los principales conceptos relacionados entre sí a través del modelo de dominio. Se determinan los requisitos funcionales y no funcionales con los que debe cumplir la solución. Se generan los artefactos, los cuales serán las premisas para la entrega final del producto.

2.2 Modelo de dominio

Después de un análisis se llega a la conclusión de que no se identifican procesos del negocio que enmarcan el problema investigativo sino elementos conceptuales que definen correctamente todo el proceso de solución, por lo que se confecciona el Modelo de domino; el cual representa conceptos relacionados entre sí. Se utiliza para lograr un mejor entendimiento del negocio tomándose como punto de partida para el diseño del mismo. Ver figura 4.

2.2.1 Descripción de las clases del Modelo de dominio

Usuario: Interactúa con los servicios que brinda la aplicación.

Aplicación: es el portal *web* que brinda servicios, a través de él las instituciones externas guardan la información en la cédula de identidad, es la encargada de todos los procesos a realizar.

Administrador: es el encargado de gestionar varios procesos en la aplicación así como llevar el control del sistema.

Operador: es el encargado de establecer la estructura de la información a guardar.

Institución: es la entidad que solicita los servicios brindados para ofrecerlos a las personas para que almacenen información relacionada con dicha institución en la cédula de identidad.

Estructura: es la forma en que van a quedar los datos almacenados.

UsuariosInst: son los usuarios que van a pertenecer a cada institución.

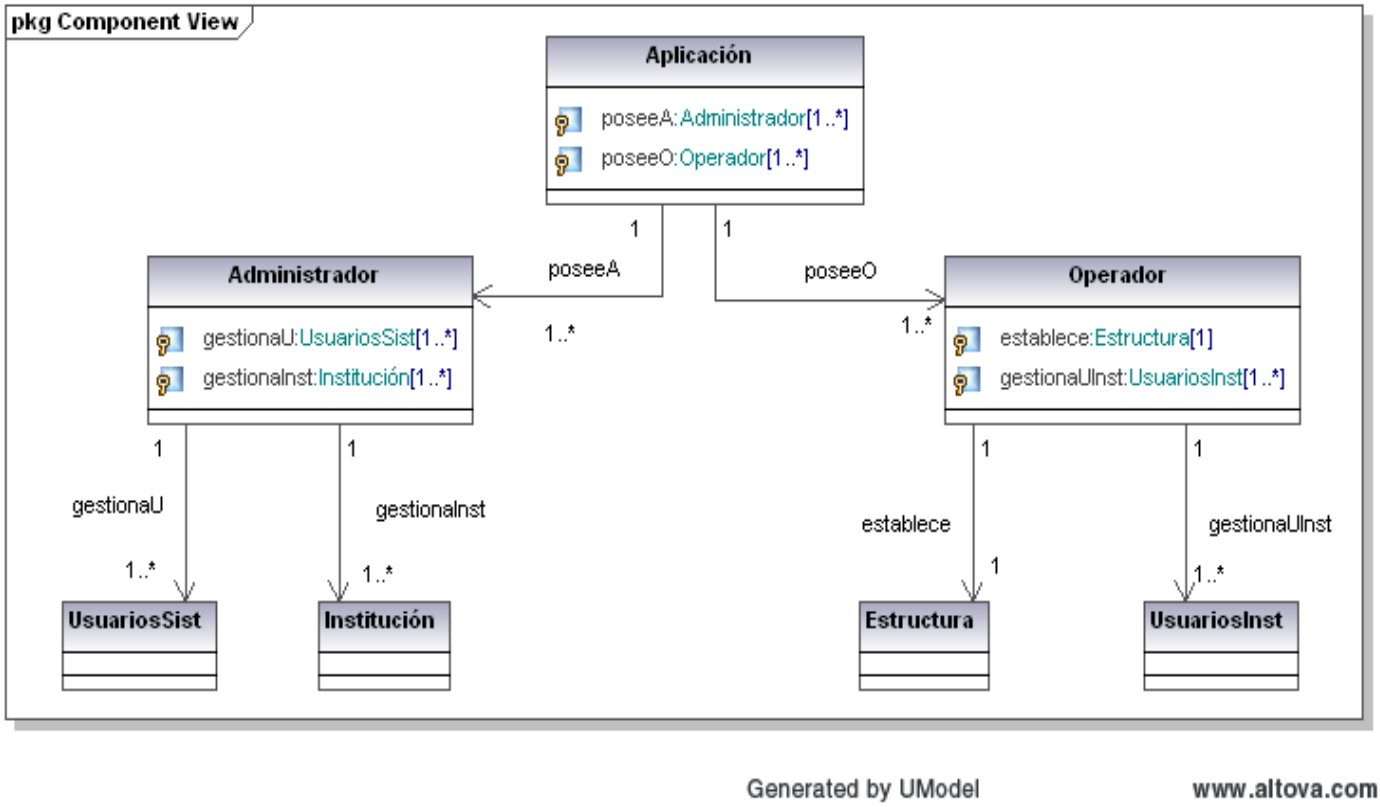


Figura 4: Modelo de dominio

2.3 Requisitos de la aplicación

A continuación se describen los requisitos funcionales, los cuales son características que el sistema debe cumplir, es decir, las funcionalidades del sistema que se expresan en forma de declaración verbal.

Requisitos funcionales

RFGE1 Gestionar información acerca de las instituciones que tienen acceso a la memoria de la CIe.

RFGE1.1 Registrar nuevas instituciones dados los datos de contacto:

- Nombre de la institución (obligatorio).
- RIF (Registro de Identificación Fiscal) (obligatorio).

- Correo electrónico de la institución (obligatorio).
- Dirección de contacto. (obligatorio).

RFGE1.2 Modificar información de instituciones.

RFGE1.3 Eliminar información de instituciones.

RFGE1.4 Gestionar usuarios por institución dados los datos:

- Nombre. (obligatorio).
- Número de cédula. (obligatorio).
- Dirección de correo electrónico (obligatorio).
- Número de teléfono (obligatorio).

RFGE1.4.1 Modificar información de usuarios.

RFGE1.4.2 Eliminar información de usuarios.

RFGE2 Gestionar la estructura de archivos de la institución.

RFGE2.1 Diseñar la estructura de archivo correspondiente a cada institución.

RFGE2.1.1 Especificar Archivo Maestro (MF) (obligatorio).

RFGE2.1.2 Especificar Archivos de Directorio (DF) (opcional).

RFGE2.1.3 Especificar Archivo Elemental (EF) (obligatorio).

RFGE3 Gestionar control de acceso al sistema.

RFGE3.1 Registrar usuarios en el sistema dados los datos:

- Nombre completo (obligatorio).
- Correo electrónico (obligatorio).

- Teléfono de contacto (opcional).

RFGE3.2 Modificar usuarios del sistema.

RFGE3.3 Eliminar usuarios del sistema.

RFGE3.4 Asignar rol de Administrador a usuarios del sistema.

RFGE3.5 Eliminar rol de Administrador a usuarios del sistema.

RFGE3.6 Adicionar rol de Operador a usuarios del sistema.

RFGE3.7 Eliminar rol de Operador a usuarios del sistema.

Requisitos no funcionales

Los requisitos no funcionales son características requeridas del sistema del proceso de desarrollo, del servicio prestado o de cualquier otro aspecto que señala una restricción del mismo.

- Software

Ambiente del Componente RIA:

- Sistema operativo con soporte para comunicación en redes.
- Navegador compatible con HTML 4.0, CSS 2.0 y EcmaScript 4.0.

Ambiente del Componente MVC:

- Sistema Operativo Linux con kernel 2.6 o superior.
- Mono versión 2.10 o superior.
- PostgreSQL 8.4 o superior.

- Usabilidad

Enfoque de los manuales de ayuda a tres tipos de usuarios.

- Los administradores y los operadores son los usuarios que accederán a las funcionalidades del módulo de gestión de control de acceso para instituciones externas de las que se incorpora información en la cédula de identidad electrónica con todos los privilegios y por tanto se encargarán de administrar la asignación del espacio y el acceso de las instituciones a la CIe.

- Desarrolladores de las instituciones externas que se encargan de implementar las funcionalidades para almacenar y acceder a la información específica asignada a la institución correspondiente.
- El ciudadano podrá acceder a la información almacenada en la CIe según las condiciones de acceso que tenga.

- Apariencia o interfaz externa

El sistema deberá tener una interfaz gráfica uniforme a través del mismo incluyendo pantallas, menús y opciones. Las pautas de diseño serán definidas por el equipo de diseño gráfico y se realizarán siguiendo los lineamientos de la arquitectura de información.

Tanto los títulos de los componentes de la interfaz, como los mensajes para interactuar con los usuarios, así como los mensajes de error, deben ser en idioma español y tener una apariencia uniforme en todo el sistema. Los mensajes de error deberán ser lo suficientemente informativos para dar a conocer la severidad del error.

- Fiabilidad

El sistema deberá prever contingencias que pueden afectar la prestación estable y permanente del servicio. La siguiente lista de contingencias son las que se deben tener en cuenta y se pueden considerar críticas:

- Sobrecarga del sistema por cantidad de lotes procesados en paralelo.
- Caída del sistema por sobrecarga de transacciones.
- Caída del sistema por volumen de datos excedido en la base de datos.

- Eficiencia

El sistema debe ser capaz de dar respuestas a las peticiones con un nivel aceptable de desempeño. Teniendo en cuenta el nivel de concurrencia que pueda existir, debe ser capaz de prestar servicio sin que se deterioren los tiempos de respuestas. En ningún caso la respuesta del sistema debe superar los 2 segundos.

- Restricciones de diseño

- Plataforma de desarrollo .NET 4.0.
- Framework ASP.NET MVC 2.0.
- Biblioteca JQuery.
- Entorno de desarrollo MonoDevelop.
- Framework NHibernate.

➢ Seguridad

- Disponibilidad

La aplicación debe estar disponible solo desde las oficinas centrales del SAIME por personal autorizado todo el día.

- Confiabilidad

Se debe asegurar un canal seguro de comunicación entre el cliente y el servidor de aplicaciones debido a la confidencialidad de los datos de las instituciones externas que se manejan en la aplicación. La comunicación entre el servidor de aplicaciones y el cliente sea de forma segura utilizando el protocolo HTTPs.

2.4 Historias de Usuario

Las Historias de Usuario (HU) son la técnica utilizada en XP para especificar los requisitos del *software*, en las cuales el cliente describe brevemente las características que el sistema debe poseer. El tratamiento de las HU es muy dinámico y flexible. Cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas. (31)

Tabla 1: Gestionar información acerca de las instituciones que tienen acceso a la memoria de la Cle.

Historia de Usuario	
Número: RFGE 1	**Usuario:** Desarrollador
Nombre historia: Gestionar información acerca de las instituciones que tienen acceso a la memoria de la Cle.	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Media
Puntos estimados: 2	**Iteración asignada:** 1

Programador responsable: Wagner Díaz Lantigua
Descripción: La interfaz del sistema permitirá insertar, modificar y eliminar instituciones en la aplicación. Para cada institución es obligatorio incluir todos sus datos de contacto (Nombre de la entidad, RIF (Registro de Identificación Fiscal), Correo electrónico del enlace en la entidad, Dirección) sin dejar ningún campo vacío.
Observaciones:

Tabla 2: Gestionar usuarios por institución dados los datos.

Historia de Usuario	
Número: RFGE 1.4	**Usuario:** Desarrollador
Nombre historia: Gestionar usuarios por institución dados los datos.	
Prioridad en negocio: Baja	**Riesgo en desarrollo:** Baja
Puntos estimados: 2	**Iteración asignada:** 1
Programador responsable: Wagner Díaz Lantigua	
Descripción: La interfaz del sistema deberá permitir al operador insertar, modificar, eliminar usuarios por cada una de las instituciones que se encuentran en la aplicación. Por cada institución adicionada es obligatorio incluir todos sus datos del usuario perteneciente a la misma: Nombre, Número de cédula, Dirección de correo electrónico, Número de teléfono sin dejar ningún campo vacío.	
Observaciones:	

Tabla 3: Gestionar la estructura de archivos de la institución.

Historia de Usuario	
Número: RFGE 2	**Usuario:** Desarrollador
Nombre historia: Gestionar la estructura de archivos de la institución.	

Prioridad en negocio: Media	**Riesgo en desarrollo:** Media
Puntos estimados: 3	**Iteración asignada:** 2
Programador responsable: Wagner Díaz Lantigua	
Descripción: La interfaz del sistema debe permitir al operador diseñar la estructura de almacenamiento de archivos de cada entidad mediante el estándar ISO 7816-4, en el cual se deben especificar ciertos datos como (Archivo Maestro (MF), Archivos de Directorio (DF), Archivo Elemental (EF). Brinda las opciones de crear los DF y los EF así como eliminarlos. Además permite especificar el espacio que va a tener cada institución en la tarjeta y las condiciones de acceso a la información que va a poseer cada institución los cuales pueden ser de lectura, escritura o ambos.	
Observaciones:	

Tabla 4: Gestionar control de acceso al sistema.

Historia de Usuario	
Número: RFGE 3	**Usuario:** Desarrollador
Nombre historia: Gestionar control de acceso al sistema.	
Prioridad en negocio: Baja	**Riesgo en desarrollo:** Baja
Puntos estimados: 2	**Iteración asignada:** 2
Programador responsable: Wagner Díaz Lantigua	
Descripción: El sistema debe permitir al administrador registrar usuarios recogiendo los siguientes datos (Nombre completo, Correo electrónico, Teléfono de contacto). También modificar y eliminar dichos usuarios. Así como asignar y eliminar los roles de Administrador y Operador a los usuarios del sistema que se estimen convenientes. Una vez que se inserta un usuario al sistema se le especifica el rol a desempeñar y cuando no se desee que este usuario ocupe ese rol se eliminará dicho usuario.	
Observaciones:	

2.5 Propuesta de solución.

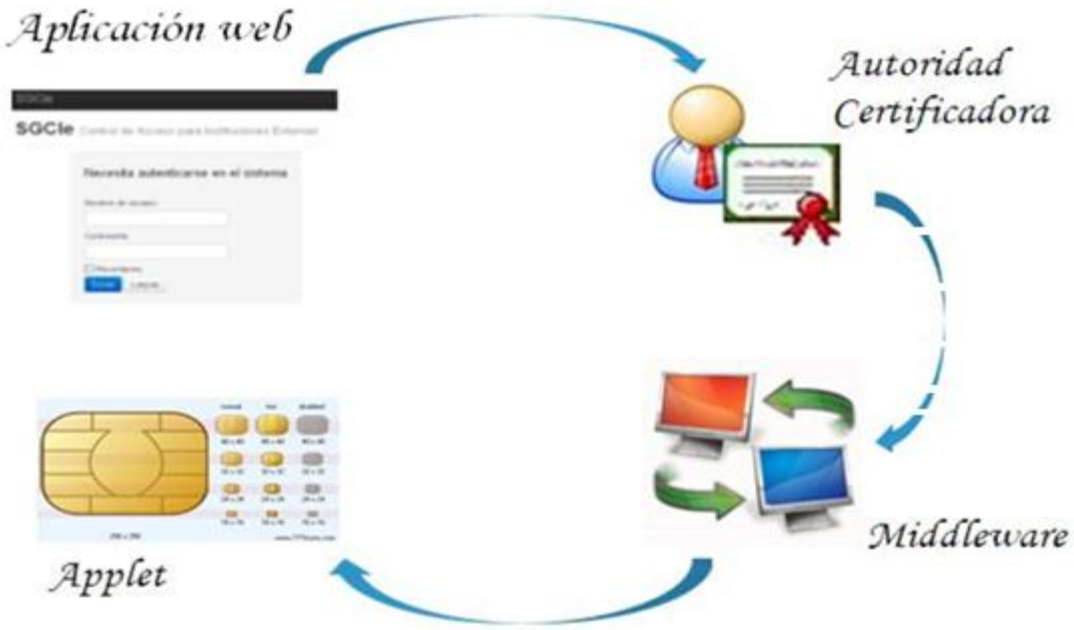


Figura 5: Propuesta general

La solución general es una aplicación web que lleva la gestión de control de acceso de las instituciones externas para poder almacenar información en la tarjeta. Dicha aplicación es la encargada de crear la estructura de la información siguiendo el estándar de almacenamiento en las tarjetas inteligentes ISO 7816-4, así como otorgarle permisos a dichas instituciones para que puedan acceder a determinada información. Una vez creada toda la información es enviada a la entidad certificadora la cual crea el certificado digital conteniendo el *token* de acceso que va dentro del certificado, lo envía al *middleware* donde su función principal es verificar dicho certificado y sacar dicho *token*. A su vez establece una comunicación con el *applet* verificando el *token* y muestra la estructura con los permisos otorgados. Ver figura 5.

Aplicación web

La aplicación está compuesta por un componente RIA (del inglés Rich Internet Aplications) que es el encargado de capturar los datos del usuario y un componente MVC (Modelo-Vista-Controlador) que

gestiona el proceso de negocio, que interactúan entre sí a través de una arquitectura cliente – servidor utilizando Ajax.

RIA es una Aplicación Web que tiene características y funcionalidades de una aplicación de escritorio común, con la gran diferencia de que RIA no necesita instalar la aplicación en la maquina local del usuario, sino que es accesible desde un navegador web (Firefox, Internet Explorer, Opera), haciéndola *"cross-platform"*, es decir, que no importa realmente la plataforma que el usuario utilice para acceder a esta aplicación, siempre funcionará y se ve igual. (32)

MVC es el encargado de realizar todas las acciones de lógica de negocio en el lado del servidor, expone la lógica de negocio a través de varias interfaces conforme al estilo arquitectónico REST, lo cual divide completamente el funcionamiento del cliente que se utilice. A este componente se integra el componente RIA, el cual actúa como cliente enviando los datos necesarios para las operaciones. En la Figura 6 se muestra esta interacción.

Los permisos que se le otorgan a las instituciones de lectura y/o escritura están establecidos sobre las siguientes condiciones de acceso: Autenticación Match on Card (MoC), Autenticación mutua, Mensajería segura, Autenticación por PIN (Personal Identification Number).

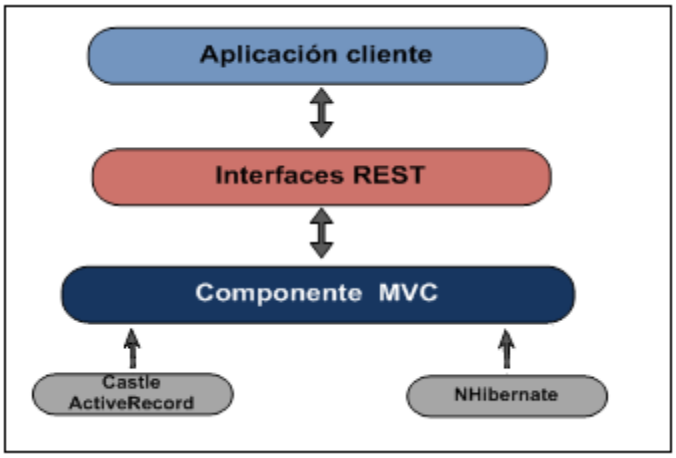


Figura 6: Interacción del componente MVC con el resto de los componentes (33)

RIA actúa como cliente capturando los datos del usuario y enviándolos para su procesamiento al componente MVC. En la figura 7 se muestra esta interacción.

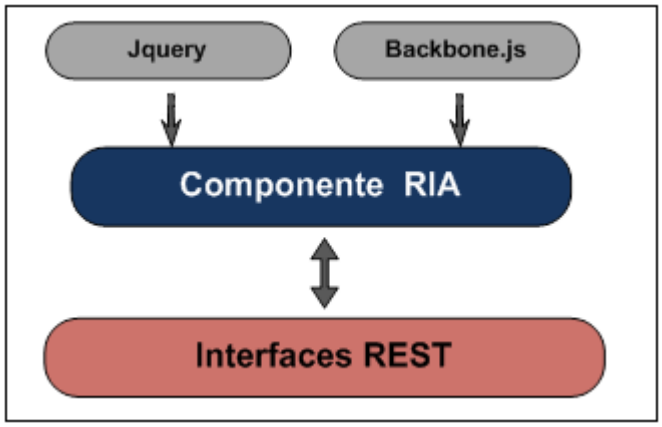


Figura 7: Interacción del componente RIA con el servidor (33)

La comunicación entre los componentes se efectúa mediante llamadas sobre el protocolo HTTP sin que medie ningún otro proceso que la comunicación entre el navegador *web* donde se ejecuta el componente RIA y el servidor de aplicaciones que ejecuta el componente MVC. (33)

2.6 Arquitectura del sistema

La arquitectura Cliente/Servidor permite a los usuarios finales obtener acceso a la información en forma transparente aún en entornos multiplataforma. El cliente envía un mensaje solicitando un determinado servicio a un servidor (hace una petición), y este envía uno o varios mensajes con la respuesta (provee el servicio). En un sistema distribuido, cada máquina puede cumplir el rol de servidor para algunas tareas y el rol de cliente para otras. El cliente es el proceso que permite al usuario formular los requerimientos y pasarlos al servidor, se le conoce con el término front-end. El servidor es el encargado de atender a múltiples clientes que hacen peticiones de algún recurso administrado por él. Al proceso servidor se le conoce con el término *back-end*. (34)

La arquitectura general del sistema está compuesto por tres partes principales, el cliente, el servidor web y la base de datos. El cliente son las interfaces de comunicación con el servidor web. El servidor web emplea el patrón arquitectónico Modelo-Vista-Controlador donde la vista es la página HTML y el

código que provee de datos dinámicos a la página aspx de ASP.Net. El controlador es el responsable de recibir los eventos de entrada desde la vista y el modelo es la lógica de negocio que administra el comportamiento y los datos del dominio de aplicación, responde a requerimientos de información sobre su estado y es el encargado del acceso a la base de datos a través del framework NHibernate. Ver figura 8.

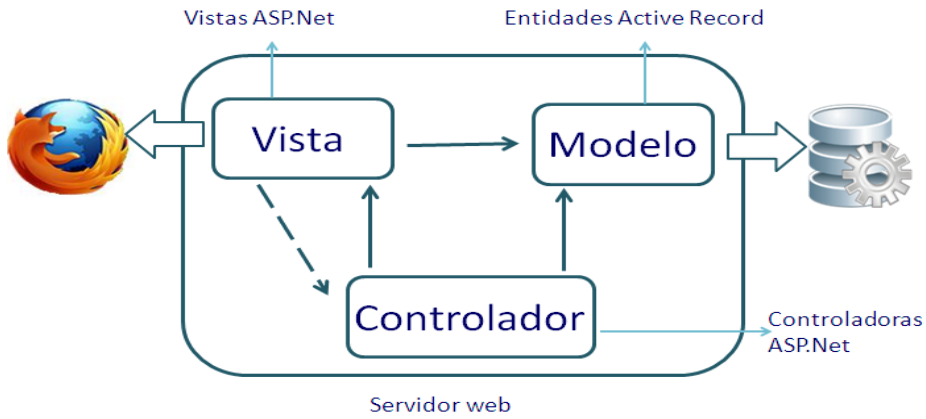


Figura 8: Arquitectura del sistema

2.7 Estimación de tiempo

Los programadores estiman el tiempo que necesitan para desarrollar cada HU, este valor se expresa en semanas. Una HU no debe desarrollarse en menos de una, ni en más de tres semanas, en otro caso debe dividirse la historia de usuario.

Tabla 5: Estimación de tiempo

No.HU	Historia de usuario	Estimación (semanas)
HU1	Gestionar información acerca de las instituciones que tienen acceso a la memoria de la Cédula de identidad electrónica.	2
HU2	Gestionar usuarios por institución dados los datos.	2
HU3	Gestionar la estructura de archivos de la institución.	3
HU4	Gestionar control de acceso al sistema.	2

Total		9

2.8 Plan de entrega

El plan de entrega se realiza con el objetivo de que el equipo de desarrollo y el cliente tengan una planificación del tiempo que los programadores tardan en implementar las historias de usuario. Este artefacto cumple con el principio de las "liberaciones frecuentes", generalmente asociadas al fin de un grupo de iteraciones.

Tabla 6: Plan de entrega

Entregable	Iteración 1 (marzo 2012)	Iteración 2 (mayo 2012)
Producto "Módulo de gestión de control de acceso para las instituciones externas de las que se incorpora información en la cédula de identidad electrónica"	Versión 0.1	Versión 1.0

2.9 Plan de iteraciones

XP propone confeccionar el plan de iteraciones en el cual se especifican que HU se van a realizar primero según su prioridad y el tiempo de duración de cada una de ellas en cada iteración, siempre asignando las HU de mayor prioridad en la 1ra iteración. En la solución propuesta se han identificado 4 historias de HU y se definieron dos iteraciones; con una duración de cuatro semanas para la primera y cinco semanas la segunda iteración.

Tabla 7: Plan de iteraciones

Iteración	No.HU	Historia de usuario	Duración Estimada
Iteración 1	HU 1	Gestionar información acerca de las instituciones que tienen acceso a la memoria de la Cédula de identidad electrónica.	4 semanas

	HU 2	Gestionar usuarios por institución dados los datos.	
Iteración 2	HU 3	Gestionar la estructura de archivos de la institución.	5 semanas
	HU 4	Gestionar control de acceso al sistema.	

2.10 Conclusiones parciales

Una vez finalizado este capítulo se llegaron a las siguientes conclusiones:

- Se identificaron los principales conceptos relacionados entre sí y su representación en el modelo de dominio, mediante el cual se pudo diseñar la propuesta de solución.
- Se obtuvo la descripción de los requisitos funcionales y no funcionales lo que permitió dar cumplimiento a los objetivos de la investigación para facilitar la posterior implementación de la aplicación.
- Se hizo una descripción de la arquitectura utilizada logrando un mejor entendimiento a la hora de programar.

Capítulo III: Implementación y Prueba

3.1 Introducción

En el presente capítulo se modelan artefactos de la metodología XP como el diagrama de despliegue, así como la descripción de las Tarjetas Contenido, Responsabilidad y Colaboración (CRC) y las tareas de ingeniería desglosadas por cada HU. Se describen los patrones de diseño para la creación de la solución y se realizan las pruebas de aceptación.

3.2 Diseño de la solución

La metodología XP propone realizar diseños sencillos y simples superando con éxito todas las pruebas, no tiene lógica duplicada, refleja claramente la intención de implementación de los programadores y tiene el menor número posible de clases y métodos. Si alguna parte de la implementación resulta especialmente compleja, se replantea. XP se ajusta al plan de iteraciones y siempre realizando las tareas correspondientes a cada iteración permitiendo una fácil evolución para el equipo de desarrollo para conseguir el producto deseado.

La metodología XP no requiere la representación del sistema mediante diagramas de clases utilizando notación UML, en su lugar se usan las tarjetas CRC (Contenido, Responsabilidad y Colaboración), se trata de simples tarjetas de papel para fichado que remplazan a los diagramas en la representación de modelos. Esta técnica que utiliza XP está basada en objetos, brindan la posibilidad que el equipo completo de desarrolladores contribuya en la tarea de diseño, generalmente cada tarjeta representa una clase diferente en la codificación y tienen como ventaja que todo el equipo contribuye a la elaboración del diseño de la solución. Seguidamente se describen cada una de estas tarjetas, las demás se pueden ver en el Anexo 1.

Tabla 8: Tarjeta CRC de la clase HomeController

Clase HomeController	
Funcionalidades	Colaboración
Login Logout ValidateLogin	 SystemUser SystemUser

Tabla 9: Tarjeta CRC de la clase InstitutionsController

Clase InstitutionsController	
Funcionalidades	Colaboración
Create Edit Update Delete	Institution Institution Institution Institution

Tabla 10: Tarjeta CRC de la clase SecurityConditionController

Clase SecurityConditionController	
Funcionalidades	Colaboración
Create Edit Update Delete	SecurityCondition SecurityCondition SecurityCondition SecurityCondition

3.3 Patrones de diseño

Los patrones de diseño proponen una forma de reutilizar la experiencia de los desarrolladores, para ello clasifican y describen formas de solucionar problemas que ocurren de forma frecuente en el desarrollo del software.

Patrones GRASP

GRASP, nombre elegido para indicar la importancia de captar estos principios, si se quiere diseñar eficazmente el software orientado a objetos. Para la realización de la solución se utilizaron los patrones: Experto, Creador, Alta Cohesión y Controlador, todos incluidos dentro del grupo de patrones GRASP. A continuación se dará una breve descripción de estos patrones.

- Alta Cohesión: La cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme. Este patrón mejora la calidad y facilidad del diseño, genera un bajo acoplamiento y promueve la reutilización. Se ve evidenciado el uso de este patrón en la clase *RestfulControllerBase*.

- Experto: Permite asignar una responsabilidad al experto en información, es decir, la clase que cuenta con la información necesaria para cumplir la responsabilidad. Este patrón se ve evidenciado en la clase *BaseModel*.
- Creador: Tiene como propósito fundamental encontrar un creador que se conecte con el objeto producido en cualquier evento. Guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. Este patrón se pone de manifiesto en las clases *InstitutionsController* que se hace llamado a la clase *Institution* así como en la clase *UsersController* que hace referencia a la clase *User*.
- Controlador: La mayor parte de los sistemas reciben eventos de entrada externa. En estos casos hay que elegir controladores que manejen esos eventos de entrada. Este patrón ofrece una guía para tomar decisiones apropiadas en la elección de los controladores de eventos. Su utilización propicia que las operaciones del sistema se manejen en la capa de dominio de los objetos, y no en la de presentación. Este patrón se utiliza en las clases controladoras como: *FilesystemController, HomeController, InstitutionsController, SecurityConditionController, SystemUsersController* y *UsersController.*

Patrones GoF

Los patrones de diseño GoF deben su nombre a sus creadores: Erich Gamma, Richard Helm, Ralph Johnson y John Vlisides. Ellos recopilaron y documentaron 23 patrones de diseño aplicados usualmente por expertos diseñadores de software orientado a objetos. El grupo GoF se clasifica en tres grandes categorías basadas en su propósito: creacionales, estructurales y de comportamiento. A continuación se explica el patrón Singleton del grupo creación utilizado en este trabajo:

- Creación: crea instancias de objetos. El objetivo de estos patrones es de abstraer el proceso de instanciación y ocultar los detalles de cómo los objetos son creados o inicializados.
    - Singleton: Asegura que sólo se pueda crear una instancia de la clase y ofrece un punto global de acceso a esta instancia. El uso de este patrón permite que los servicios creen una instancia de los objetos sólo una vez. Ver ejemplo de este patrón en la figura 9. (35)

```
class SaltedHash
{
  private static SaltedHash instance;

  public static SaltedHash Default {
     get
     {
        if (instance == null)
        {
           instance = new SaltedHash();
        }
        return(instance);
     }
   }
}
```

Figura 9: Patrón Singleton

3.4 Implementación del sistema

En cada una de las iteraciones se implementarán las HU correspondientes, siempre revisando nuevamente el plan de iteraciones y modificándolas en caso de ser necesario. Cada HU se desglosa en tareas de ingeniería, asignándolas a un grupo de desarrollo (o una persona) responsable de su implementación. Como estas tareas son para el uso estricto de los programadores pueden escribirse en un lenguaje técnico, al contrario de las HU que son escritas en el lenguaje del cliente.

XP propone que la implementación sea realizada mediante iteraciones, una vez concluida cada iteración es necesario entregar una versión del producto, el cual debe ser probado y mostrado al cliente sirviendo de retroalimentación para el equipo de trabajo. Seguidamente se explica las dos iteraciones planteadas en el capítulo anterior con las tareas de ingenierías definidas para cada una de las historias de usuarios.

3.4.1 Iteración 1

En esta primera iteración se desarrollan las historias de usuario de mayor prioridad en el sistema, con el objetivo de obtener una primera versión del producto con las principales características o funcionalidades para ser mostrado al cliente. Las tareas de ingeniería restantes se pueden ver en el Anexo 2.

Tabla 11: Historias de Usuario abordadas en la primera iteración.

Iteración	No. HU	Historia de Usuario	Estimación
Iteración 1	HU 1	Gestionar información acerca de las instituciones que tienen acceso a la memoria de la Cédula de identidad electrónica.	2
	HU 2	Gestionar usuarios por institución dados los datos.	2
	Total		4

Tabla 12: Tareas de ingeniería de la iteración 1

Iteración	No. HU	Historia de Usuario	Tareas
Iteración 1	HU 1	Gestionar información acerca de las instituciones que tienen acceso a la memoria de la Cédula de identidad electrónica.	➢ Insertar entidad ➢ Modificar entidad ➢ Eliminar entidad
	HU 2	Gestionar usuarios por institución dados los datos.	➢ Insertar usuario ➢ Modificar usuario ➢ Eliminar usuario

Tareas de ingeniería detalladas de la Iteración 1

Tabla 13: Tarea de ingeniería HU1_T1

Tarea	
Número de la Tarea: HU1_T1	**HU:** Gestionar información acerca de las instituciones que tienen acceso a la memoria de la Cédula de identidad electrónica.
Nombre: Insertar institución	
Tipo: Desarrollo	**Puntos Estimados:** 2
Fecha Inicio: 15-2-2012	**Fecha Fin:** 19-2-2012

Responsable: Wagner Díaz Lantigua
Descripción: El sistema debe permitir insertar entidades e incluir los datos de cada uno de ellos (Nombre de la entidad, RIF, Correo electrónico del enlace en la entidad, Dirección) sin dejar campos vacíos.

Tabla 14: Tarea de ingeniería HU1_T2

Tarea	
Número de la Tarea: HU1_T2	**HU:** Gestionar información acerca de las instituciones que tienen acceso a la memoria de la Cédula de identidad electrónica.
Nombre: Modificar institución	
Tipo: Desarrollo	**Puntos Estimados:** 2
Fecha Inicio: 20-2-2012	**Fecha Fin:** 23-2-2012
Responsable: Wagner Díaz Lantigua	
Descripción: El sistema debe permitir modificar instituciones y cambiar los datos de cada uno de ellos (Nombre de la entidad, RIF, Correo electrónico del enlace en la entidad, Dirección) en cada campo correspondiente sin dejar campos vacíos.	

Tabla 15: Tarea de ingeniería HU1_T3

Tarea	
Número de la Tarea: HU1_T3	**HU:** Gestionar información acerca de las instituciones que tienen acceso a la memoria de la Cédula de identidad electrónica.
Nombre: Eliminar institución	
Tipo: Desarrollo	**Puntos Estimados:** 2
Fecha Inicio: 25-2-2012	**Fecha Fin:** 28-2-2012
Responsable: Wagner Díaz Lantigua	
Descripción: El sistema selecciona que institución desea eliminar y después la elimina.	

3.4.2 Iteración 2

En esta iteración se le dará conclusión a las HU que faltan por realizarse, con las que se le proporcionará un seguimiento a las ya implementadas en la iteración 1.

Tabla 16: Historias de Usuario abordadas en la segunda iteración

Iteración	No. HU	Historia de Usuario	Estimación
Iteración 2	HU 3	Gestionar la estructura de archivos de la institución.	3
	HU 4	Gestionar control de acceso al sistema.	2
	Total		5

Tabla 17: Tareas de la ingeniería iteración 2

Iteración	No. HU	Historia de Usuario	Tareas
Iteración 2	HU 3	Gestionar la estructura de archivos de la institución.	➢ Gestionar la estructura de almacenamiento de archivos de la institución.
	HU 4	Gestionar control de acceso al sistema.	➢ Insertar usuario ➢ Modificar usuario ➢ Eliminar usuario ➢ Asignar rol de Administrador y Operador ➢ Eliminar roles de Administrador y Operador

Tareas de la ingeniería detalladas de la segunda iteración

Tabla 18: Tarea de ingeniería HU3_T1

Tarea	
Número de la Tarea: HU3_T1	**HU:** Gestionar la estructura de archivos de la institución.
Nombre: Gestionar la estructura de archivos de la institución.	
Tipo: Desarrollo	**Puntos Estimados:** 1
Fecha Inicio: 2-4-2012	**Fecha Fin:** 13-4-2012
Responsable: Wagner Díaz Lantigua	
Descripción: Diseñar la estructura de cada institución mediante el estándar ISO 7816-4 y crear y/o eliminar los (Archivo Maestro (MF), Archivos de Directorio (DF), Archivo Elemental (EF)), especifica el espacio que va a ocupar en la tarjeta dicha institución así como otorgar los permisos de acceso a la información para las entidades existentes.	

Tabla 19: Tarea de ingeniería HU4_T1

Tarea	
Número de la Tarea: HU4_T1	**HU:** Gestionar control de acceso al sistema.
Nombre: Insertar usuario	
Tipo: Desarrollo	**Puntos Estimados:** 1
Fecha Inicio: 15-4-2012	**Fecha Fin:** 18-4-2012
Responsable: Wagner Díaz Lantigua	
Descripción: El sistema debe permitir al administrador insertar usuario e incluir sus datos (Nombre completo, Correo electrónico, Teléfono de contacto) sin dejar campos vacíos.	

Tabla 20: Tarea de ingeniería HU4_T2

Tarea

Número de la Tarea: HU4_T2	**HU:** Gestionar control de acceso al sistema.
Nombre: Modificar usuario	
Tipo: Desarrollo	**Puntos Estimados:** 1
Fecha Inicio: 19-4-2012	**Fecha Fin:** 25-4-2012
Responsable: Wagner Díaz Lantigua	
Descripción: El sistema debe permitir al administrador modificar usuario y cambiar sus datos (Nombre completo, Correo electrónico, Teléfono de contacto) en cada campo correspondiente y sin dejar ninguno vacío.	

Tabla 21: Tarea de ingeniería HU4_T3

Tarea	
Número de la Tarea: HU4_T3	**HU:** Gestionar control de acceso al sistema.
Nombre: Eliminar usuario	
Tipo: Desarrollo	**Puntos Estimados:** 1
Fecha Inicio: 25-4-2012	**Fecha Fin:** 30-4-2012
Responsable: Wagner Díaz Lantigua	
Descripción: El sistema debe permitir al administrador eliminar usuario.	

3.5 Diagrama de Despliegue

Los Diagramas de Despliegue muestran las relaciones físicas de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos. La vista de despliegue representa la disposición de las instancias de componentes de ejecución en instancias de nodos conectados por enlaces de comunicación. Un nodo es un recurso de ejecución tal como un computador, un dispositivo o memoria. En la figura 10 se muestra el modelo de despliegue de la solución. (36)

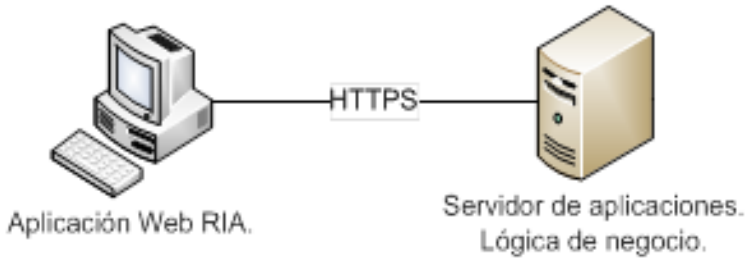


Figura 10: Modelo de despliegue de la solución

3.6 Pruebas

El proceso de pruebas es una de las piedras angulares de la metodología XP. Una vez implementado el *software* es necesario realizarle dichas pruebas para así validar su correcto funcionamiento. Estas permiten aumentar la calidad del sistema implementado, disminuyendo los errores que puedan aparecer. XP realiza las pruebas de aceptación o pruebas funcionales, destinadas a evaluar si al final de una iteración se alcanzó la funcionalidad requerida. (37)

3.6.1 Pruebas de integración

Las pruebas de integración son pruebas ordenadas que van desde los componentes o módulos y abarcan el funcionamiento de todo el sistema. Existen dos tipos de pruebas de integración: la integración incremental que combina el siguiente módulo que se debe probar con el conjunto de módulos que ya han sido probados y la integración no incremental que prueba cada módulo por separado y luego se integran todos de una vez y se prueba el programa completo. Se puede probar de dos formas ascendente que comienza por los módulos hoja y descendente que comienza por el módulo raíz. (38)

3.6.2 Pruebas de aceptación

Las pruebas funcionales se caracterizan por probar la aplicación detalladamente para cada una de sus funcionalidades, estas prueban todas las capas de la aplicación: el sistema de enrutamiento, el modelo, las acciones y las plantillas.

Las pruebas de aceptación son creadas en base a las historias de usuarios, en cada ciclo de la iteración del desarrollo. El cliente debe especificar uno o diversos escenarios para comprobar que una historia de

usuario ha sido correctamente implementada. Una historia de usuario no se puede considerar terminada hasta tanto pase correctamente todas las pruebas de aceptación. (39)

El objetivo de las pruebas de aceptación es validar que un sistema cumple con el funcionamiento esperado y permitir al usuario de dicho sistema que determine su aceptación, desde el punto de vista de su funcionalidad y rendimiento. Estas son definidas por el usuario del sistema y preparadas por el equipo de desarrollo, aunque es responsabilidad del cliente verificar la corrección de las pruebas y tomar decisiones acerca de las mismas. Los restantes casos de prueba de aceptación se encuentran en el Anexo 4.

Tabla 22: Caso de prueba de aceptación de la HU1_P1.

Caso de Prueba de Aceptación	
Código: HU1_P1	**HU:** Gestionar información acerca de las instituciones que tienen acceso a la memoria de la Cédula de identidad electrónica.
Nombre: Insertar instituciones	
Descripción: Prueba de funcionalidad para insertar una institución.	
Condiciones de ejecución:	
Entrada / Pasos de Ejecución: 1- El administrador selecciona el icono que representa insertar una institución. 2- Se introducen todos los datos referentes a la institución sin dejar ningún campo vacío. 3- Se inserta la institución.	
Resultado esperado: En el sistema se insertó la institución correspondiente.	
No conformidades:	
Evaluación de la prueba: Satisfactoria	

Tabla 23: Caso de prueba de aceptación de la HU1_P2.

Caso de Prueba de Aceptación	
Código: HU1_P2	**HU:** Gestionar información acerca de las instituciones que tienen acceso a la memoria de la Cédula de identidad electrónica.

Nombre: Modificar institución
Descripción: Prueba de funcionalidad para modificar una institución.
Condiciones de ejecución:
Entrada / Pasos de Ejecución: 1- El administrador selecciona el icono que representa que institución desea modificar. 2- Se introducen los nuevos datos que se desean modificar de dicha institución. 3- Se modificó la institución.
Resultado esperado: En el sistema se modificó la institución seleccionada.
No conformidades:
Evaluación de la prueba: Satisfactoria

Tabla 24: Caso de prueba de aceptación de la HU1_P3.

Caso de Prueba de Aceptación	
Código: HU1_P3	**HU:** Gestionar información acerca de las instituciones que tienen acceso a la memoria de la Cédula de identidad electrónica.
Nombre: Eliminar institución	
Descripción: Prueba de funcionalidad para eliminar una institución.	
Condiciones de ejecución:	
Entrada / Pasos de Ejecución: 1- El administrador selecciona que institución desea eliminar. 2- Selecciona el icono que representa eliminar la institución. 3- Se eliminó la institución.	
Resultado esperado: En el sistema se eliminó la institución.	
No conformidades:	
Evaluación de la prueba: Satisfactoria	

Tabla 25: Caso de prueba de aceptación de la HU2_P1

Caso de Prueba de Aceptación	
Código: HU2_P1	**HU:** Gestionar usuarios por institución dados los datos.
Nombre: Insertar usuario	
Descripción: Prueba de funcionalidad para insertar un usuario.	
Condiciones de ejecución:	
Entrada / Pasos de Ejecución: 1- El administrador selecciona el icono que representa insertar usuario. 2- Se introducen todos los datos referentes al usuario sin dejar campos vacíos. 3- Se insertó el usuario.	
Resultado esperado: En el sistema se insertó el usuario referente a esa institución.	
No conformidades:	
Evaluación de la prueba: Satisfactoria	

Tabla 26: Caso de prueba de aceptación de la HU2_P2

Caso de Prueba de Aceptación	
Código: HU2_P2	**HU:** Gestionar usuarios por institución dados los datos.
Nombre: Modificar usuario	
Descripción: Prueba de funcionalidad para modificar un usuario.	
Condiciones de ejecución:	
Entrada / Pasos de Ejecución: 1- El administrador selecciona el campo que desea modificar de un determinado usuario. 2- Se introducen los nuevos datos referentes al usuario sin dejar campos vacíos. 3- Se modificó el usuario.	
Resultado esperado: En el sistema se modificó el usuario referente a esa institución.	
No conformidades:	
Evaluación de la prueba: Satisfactoria	

Tabla 27: Caso de prueba de aceptación de la HU2_P3

Caso de Prueba de Aceptación	
Código: HU2_P3	**HU:** Gestionar usuarios por institución dados los datos.
Nombre: Eliminar usuario	
Descripción: Prueba de funcionalidad para eliminar un usuario.	
Condiciones de ejecución:	
Entrada / Pasos de Ejecución: 1- El administrador selecciona que usuario desea eliminar. 2- Selecciona el icono que representa eliminar el usuario. 3- Se eliminó el usuario.	
Resultado esperado: En el sistema se eliminó el usuario determinado.	
No conformidades:	
Evaluación de la prueba: Satisfactoria	

3.7 Resultados de las pruebas

Elemento	No	No conformidad	Aspecto correspondiente	Etapa de detección del error	Importancia
1	1	No se valida que no existieran campos vacíos	Gestionar institución	Al insertar una institución	Significativa
1.1	2	No se muestran mensajes de error cuando los datos sean incorrectos	Gestionar institución	Al insertar una institución	No significativa
1.2	3	No se valida que no existieran campos vacíos	Gestionar usuarios por institución	Al insertar un usuario	Significativa
1.3	4	No se muestran mensajes de error cuando los datos sean	Gestionar usuarios por	Al insertar un usuario	No significativa

		incorrectos	institución		
2	5	No se indica la opción recordar contraseña	Gestionar control de acceso al sistema	Al autenticarse a la aplicación	No significativa
2.1	6	No se valida que no existieran campos vacíos	Gestionar estructura de información	Al insertar los datos referentes con la estructura de la información (campo tamaño).	Significativa

En la figura 11 se muestra un resumen de las no conformidades por cada iteración realizada, la figura está confeccionada en: no significativa, significativa y resuelta.

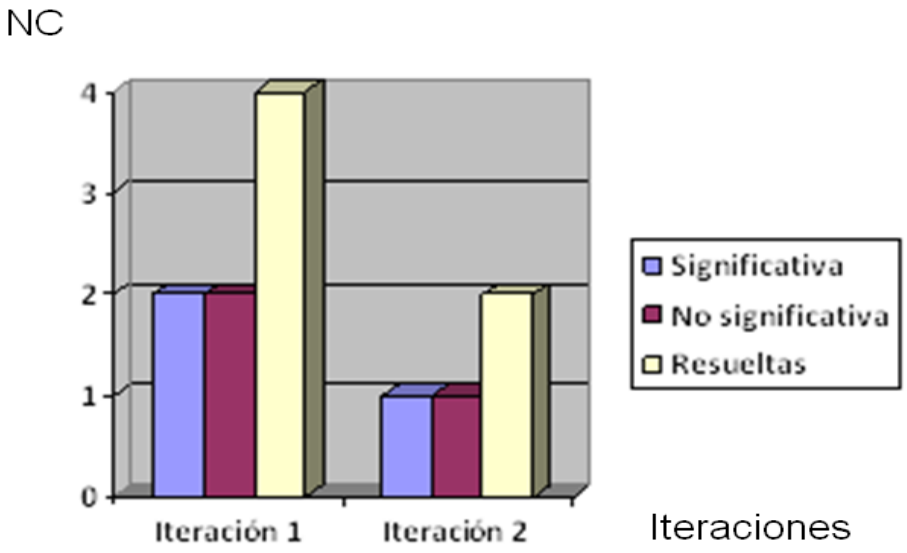


Figura 11: Resultado de las pruebas

3.8 Beneficios de la aplicación

SAIME:

- Proveer una solución tecnológica que permitirá a las instituciones, hacer un mayor uso de la CIe.

Instituciones y Gobierno:

- Brindar diversos servicios a los ciudadanos, con alto nivel tecnológico y de seguridad.

Ciudadano:

- Utilizar su documento de identificación para los servicios de instituciones del gobierno y privadas.

3.9 Conclusiones parciales

Concluido este capítulo se completó el ciclo de desarrollo del Módulo de gestión de control de acceso para las instituciones externas de las que se incorpora información en la cédula de identidad electrónica llegando a las siguientes conclusiones:

- Con el uso de varias prácticas para darle cumplimiento al objetivo inicial, como las tarjetas CRC se logró una vista de las relaciones y funcionalidades entre las diferentes clases, así como las tareas de ingeniería seleccionadas por cada historia de usuario correspondiente mostrando a los programadores detalladamente las funcionalidades específicas a implementar.
- Con el proceso de prueba se valida el correcto funcionamiento de la aplicación, realizando las pruebas de integración y de aceptación, concluyendo éstas de forma satisfactoria.

Conclusiones

Con el desarrollo de esta investigación se cumplieron los objetivos trazados inicialmente para la creación del Módulo de gestión de control de acceso para las instituciones externas de las que se incorpora información en la cédula de identidad electrónica permitiendo la gestión de las instituciones y la información proveniente de ellas. Luego de finalizado el producto es posible llegar a las siguientes conclusiones:

- La selección de las diferentes metodologías, herramientas y tecnologías hizo posible la creación del Módulo de gestión de control de acceso para las instituciones externas de las que se incorpora información en la cédula de identidad electrónica cumpliendo con las expectativas y necesidades identificadas inicialmente.
- El desarrollo de la aplicación permitió guardar información referente de ellas, estableciendo una estructura que se guía por el estándar de almacenamiento utilizado en las tarjetas inteligentes ISO 7816-4, donde los ficheros están definidos por los dedicados y los elementales.
- La definición de la arquitectura cliente/servidor y el uso del patrón MVC permitieron garantizar una mayor flexibilidad en la ejecución de la propuesta de solución.
- Las pruebas realizadas al sistema, las de integración y de aceptación demostraron el cumplimiento de los requisitos propuestos por el cliente.

Recomendaciones

Al finalizar la presente investigación se plantean las siguientes recomendaciones para las siguientes versiones de la solución:

- Integrar este módulo con la Autoridad Certificadora para permitir la creación y emisión de certificados digitales para cada institución.
- La continuidad de versiones del producto donde se le añadan nuevas funcionalidades como: importar archivos de texto plano que contengan datos de instituciones o usuarios que serán ingresados al sistema, dando la posibilidad que el sistema sea más dinámico.

Referencias Bibliográficas

1. [En línea] [Citado el: 4 de noviembre de 2011.] http://www.oas.org/juridico/mla/sp/ven/sp_ven-mla-law-id.html.

2. **Standard, International.** *Identification cards- Integrated circuit. .* 2005.

3. Silix Soluciones informáticas libres. [En línea] [Citado el: 12 de noviembre de 2011.] http://www.silix.com.ar/servicios/sistemas/comercios.

4. [En línea] [Citado el: 10 de noviembre de 2011.] http://www.directindustry.es/fabricante-industrial/software-control-acceso-74886.html.

5. **Martínez, Alejandro Martínez y Raúl.** *Guía a Rational Unified Process .*

6. **Penadés, Patricio Letelier y Mª Carmen.** *Metodologias Agiles para el desarrollo de software XP.*

7. **Fernández., Gerardo Escribano.** *Introducción a Extreme Programming.* 2002.

8. Microsoft Visual Studio. [En línea] http://www.microsoft.com/visualstudio/en-us/products .

9. MonoDevelop. [En línea] [Citado el: 12 de noviembre de 2011.] http://www.monodevelop.com..

10. Microsoft. [En línea] Noviembre de 2005. http://msdn.microsoft.com/es-es/library/z1zx9t92%28v=VS.80%29.aspx..

11. **Vilà., Fermí.** *JavaScript .*

12. **Danny Goodman, Michael Morrison.** *JavaScript bible 5th edition. .*

13. [En línea] [Citado el: 20 de noviembre de 2011.] www.postgree.org.es..

14. [En línea] http://www.iessanvicente.com/colaboraciones/oracle.pdf .

15. **García., Yanio Vidal.** *Arquitectura de Software metodología SXP .* Habana : s.n., 2011.

16. Altova. [En línea] http://www.altova.com/umodel.html.

17. **Booch, Grady, Rumbaugh, James y Jacobson, Ivar.** *El Lenguaje Unificado de Modelado. Addison-Wesley Object Technology Series . .* 2005.

18. **Sánchez., M.Sc. Javier.** *SISTEMAS OPERATIVOS SERVIDORES WEB PARA LINUX . .*

19. [En línea] http://recursostic.educacion.es/observatorio/web/es/software/servidores/800-monografico-servidores-wamp?showall=1 .

20. **Chaffer, Karl Swedberg Jonathan.** *jQuery: Learning Better Interaction Design and Web Development with Simple JavaScript. .*

21. [En línea] http://www.nohaylimites.com/?p=141..

22. [En línea] http://msdn.microsoft.com/es-es/library/w0x726c2.aspx..

23. [En línea] http://es.downv.com/download-Microsoft-ASP.NET-MVC-10357448.htm..

24. Nhibernate. [En línea] www.fincher.org/tips/.../NHibernate.shtml..

25. **Eguíluz., Javier Pérez.** *Introducción a ajax .*

26. **Ruiz, Diego Sevilla.** *Sistemas Distribuidos Facultad de Informática Murcia.* 2012.

27. [En línea] http://ditec.um.es/ssdd/sesion5.pdf..

28. **Marset., Rafael Navarro.** *Modelado, Diseño e Implementación de Servicios Web.* 2006-2007.

29. Active Record. [En línea] 27 de Marzo de 2012. http://es.scribd.com/doc/86931478/Active-Record.

30. *Documento de Arquitectura v1.0 Sistema de gestión de Servicios para la CIe.* Habana : s.n., 2011.

31. **Larman, Craig.** *UML y Patrones Introducción al análisis y diseño orientado a objetos.* La Habana : s.n., 2004.

32. [En línea] [Citado el: 23 de abril de 2012.] http://www.riactive.com/2006/12/06/rich-internet-application-definicion/ .

33. Documento Arquitectura de Software v1.0 Subproyecto Sistema de Gestión de Servicios para la CIe. 2011.

34. [En línea] http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/marquez_a_bm/capitulo5.pdf ..

35. **Erich Gamma, Richard Helm,Ralph Johnson,John Vlisides.** *Design Patterns. Elements of Reusable Object-Oriented Software.*

36. **Hugo Michael, Quisbert Limachi Nancy Susana.** [En línea] virtual.usalesiana.edu.bo/web/practica/archiv/despliegue.doc..

37. **J. J. Gutiérrez, M. J. Escalona, M. Mejías, J. Torres.** [En línea] http://www.lsi.us.es/~javierj/investigacion_ficheros/PSISEXTREMA.pdf..

38. [En línea] http://alarcos.inf-cr.uclm.es/doc/ISOFTWAREI/Tema09.pdf...

39. **José Joskowicz.** [En línea] 10 de febrero de 2008. [Citado el: 5 de mayo de 2012.] http://iie.fing.edu.uy/~josej/docs/XP%20-%20Jose%20Joskowicz.pdf.

40. **José H. Canós, Patricio Letelier yMª Carmen Penadés.** *Métodologías Ágiles en el Desarrollo de Software.* Valencia : s.n.

Glosario de términos

API: Interfaz de programación de aplicaciones (del inglés *Application Programming Interface*).

CISED: Centro de Identificación y Seguridad Digital.

CIe: Cédula de Identidad Electrónica de la República Bolivariana de Venezuela.

DF: Archivo dedicado (del inglés *Dedicated File*).

EF: Archivo elemental (del inglés *Elementary File*).

IDE: Entorno de desarrollo integrado (del inglés *Integrated Development Environment*).

ICC: Circuito integrado sin contacto

MF: Archivo maestro (del inglés *Master File*).

ORM: Mapeo Objeto Relacional.

PKI: Infraestructura de Clave Pública (del inglés *Public Key Infrastructure*).

RIA: Aplicaciones *web* que simulan la metáfora de escritorio para una mejor experiencia de usuario.

REST: Transferencia de estado representacional. Técnica de arquitectura de *software* utilizada en la *web*.

SAIME: Servicio Administrativo de Identificación, Migración y Extranjería.

SmartCard: Tarjeta inteligente.

SENIAT: Servicio Nacional Integrado de Administración Tributaria.

Anexo 1: Tarjetas CRC

Tabla 28: Tarjeta CRC de la clase *FilesystemController*

Clase FilesystemController	
Funcionalidades	Colaboración
Create Edit Update Delete	Structure Structure Structure Structure

Tabla 29: Tarjeta CRC de la clase *SystemUsersController*

Clase SystemUsersController	
Funcionalidades	Colaboración
Create Edit Update Delete	SystemUser SystemUser SystemUser SystemUser

Tabla 30: Tarjeta CRC de la clase *UserController*

Clase UserController	
Funcionalidades	Colaboración
Create Edit Update Delete	Institution Institution Institution Institution

Anexo 2: Tareas de ingeniería

Tabla 31: Tarea de ingeniería HU2_T1

Tarea	
Número de la Tarea: HU2_T1	**HU:** Gestionar usuarios por institución dados los datos.
Nombre: Insertar usuario	
Tipo: Desarrollo	**Puntos Estimados:** 2
Fecha Inicio: 3-3-2012	**Fecha Fin:** 7-2-2012

Responsable: Wagner Díaz Lantigua
Descripción: El sistema debe permitir insertar un usuario de una entidad e incluir sus datos (Nombre, Número de cédula, Dirección de correo electrónico, Número de teléfono.)

Tabla 32: Tarea de ingeniería HU2_T2

Tarea	
Número de la Tarea: HU2_T2	**HU:** Gestionar usuarios por institución dados los datos.
Nombre: Modificar usuario	
Tipo: Desarrollo	**Puntos Estimados:** 2
Fecha Inicio: 12-2-2012	**Fecha Fin:** 8-3-2012
Responsable: Wagner Díaz Lantigua	
Descripción: El sistema debe permitir modificar un usuario de una institución y cambiar sus datos (Nombre, Número de cédula, Dirección de correo electrónico, Número de teléfono) en cada campo específicamente.	

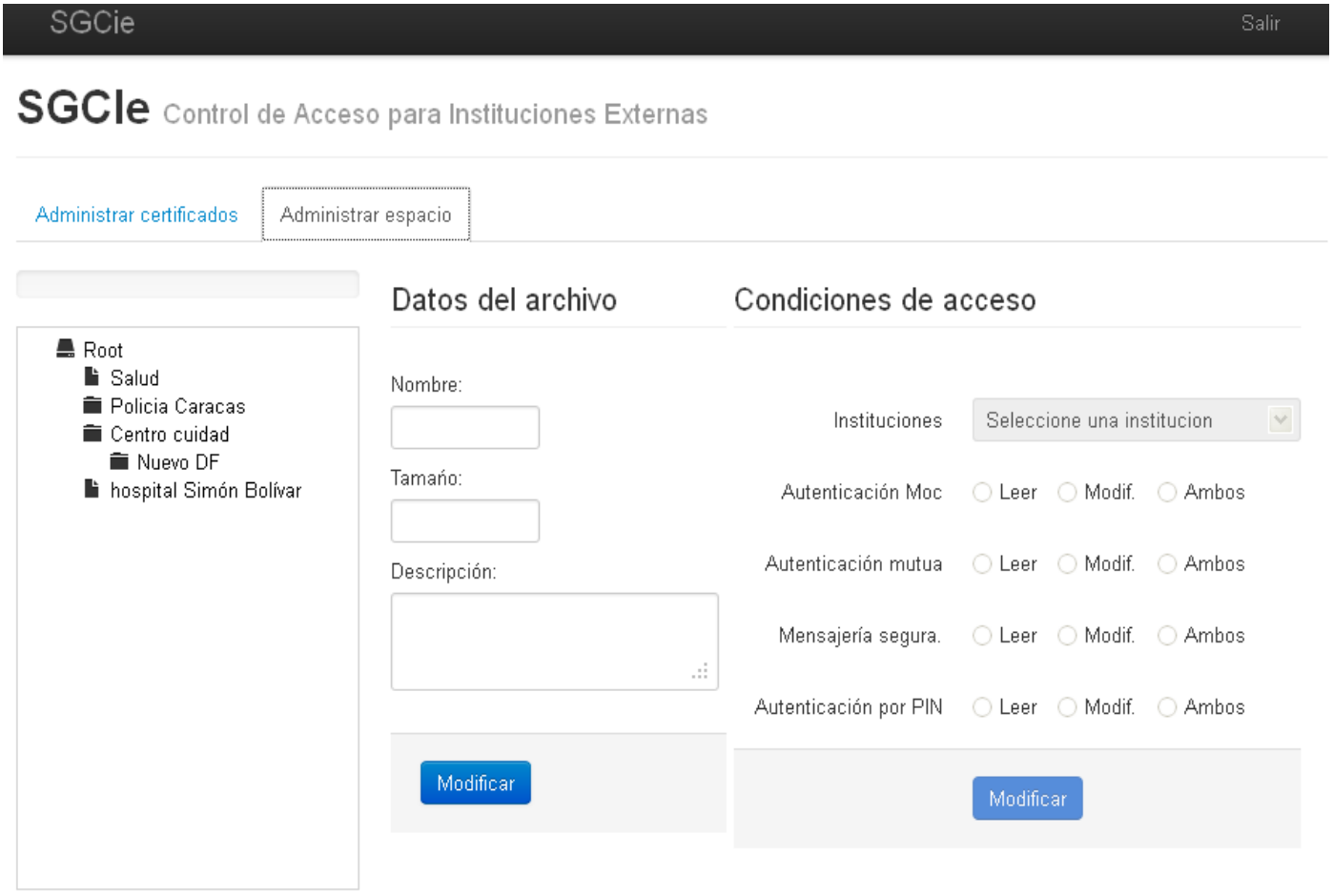
Tabla 33: Tarea de ingeniería HU2_T3

Tarea	
Número de la Tarea: HU2_T3	**HU:** Gestionar usuarios por institución dados los datos.
Nombre: Eliminar usuario	
Tipo: Desarrollo	**Puntos Estimados:** 2
Fecha Inicio: 13-3-2012	**Fecha Fin:** 20-3-2012
Responsable: Wagner Díaz Lantigua	
Descripción: El sistema debe permitir eliminar un usuario determinado, una vez que este sea seleccionado se elimina dando click en el símbolo que representa eliminar.	

Anexo 3: Interfaces de la aplicación

Prototipo de interfaz Administrar usuarios

Prototipo de interfaz Insertar institución



Prototipo de interfaz Administrar espacio

Anexo 4: Caso de Prueba de Aceptación

Tabla 34: Caso de prueba de aceptación de la HU3_P1.

Caso de Prueba de Aceptación	
Código: HU3_P1	**HU:** Gestionar la estructura de archivos de la institución.
Nombre: Gestionar la estructura de archivos de la institución.	
Descripción: Prueba de funcionalidad para la estructura de archivos de la institución.	
Condiciones de ejecución:	

Entrada / Pasos de Ejecución: 1- El operador crea los ficheros ya sean los dedicados y/o elementales referente a cada institución. 2- Se llenan los campos especificando el espacio que va a ocupar dicha institución en la tarjeta. 3- Se especifican las condiciones de acceso (lectura, escritura, ambos) para las instituciones existentes.
Resultado esperado: Se mostraron la estructura de archivo de la institución seleccionada.
No conformidades:
Evaluación de la prueba: Satisfactoria

Tabla 35: Caso de prueba de aceptación de la HU4_P1.

Caso de Prueba de Aceptación	
Código: HU4_P1	**HU:** Gestionar control de acceso al sistema.
Nombre: Insertar usuario	
Descripción: Prueba de funcionalidad para insertar un usuario.	
Condiciones de ejecución:	
Entrada / Pasos de Ejecución: 1- El administrador selecciona el icono que representa insertar usuario. 2- Se introducen todos los datos referentes al usuario sin dejar campos vacíos. 3- Se insertó el usuario.	
Resultado esperado: En el sistema se insertó usuario.	
No conformidades:	
Evaluación de la prueba: Satisfactoria	

Tabla 36: Caso de prueba de aceptación de la HU4_P2.

Caso de Prueba de Aceptación	
Código: HU4_P2	**HU:** Gestionar control de acceso al sistema.

Nombre: Modificar usuario
Descripción: Prueba de funcionalidad para modificar un usuario.
Condiciones de ejecución:
Entrada / Pasos de Ejecución: 1- El administrador selecciona el usuario que desea modificar. 2- Se introducen los nuevos datos referentes al usuario que se deseen cambiar sin dejar campos vacíos. 3- Se modificó el usuario.
Resultado esperado: En el sistema se modificó usuario.
No conformidades:
Evaluación de la prueba: Satisfactoria

Tabla 37: Caso de prueba de aceptación de la HU4_P3.

Caso de Prueba de Aceptación	
Código: HU4_P3	**HU:** Gestionar control de acceso al sistema.
Nombre: Eliminar usuario	
Descripción: Prueba de funcionalidad para eliminar un usuario.	
Condiciones de ejecución:	
Entrada / Pasos de Ejecución: 1- El administrador selecciona el usuario que desea eliminar. 2- Se selecciona el icono que representa eliminar el usuario. 3- Se eliminó el usuario.	
Resultado esperado: En el sistema se eliminó el usuario.	
No conformidades:	
Evaluación de la prueba: Satisfactoria	