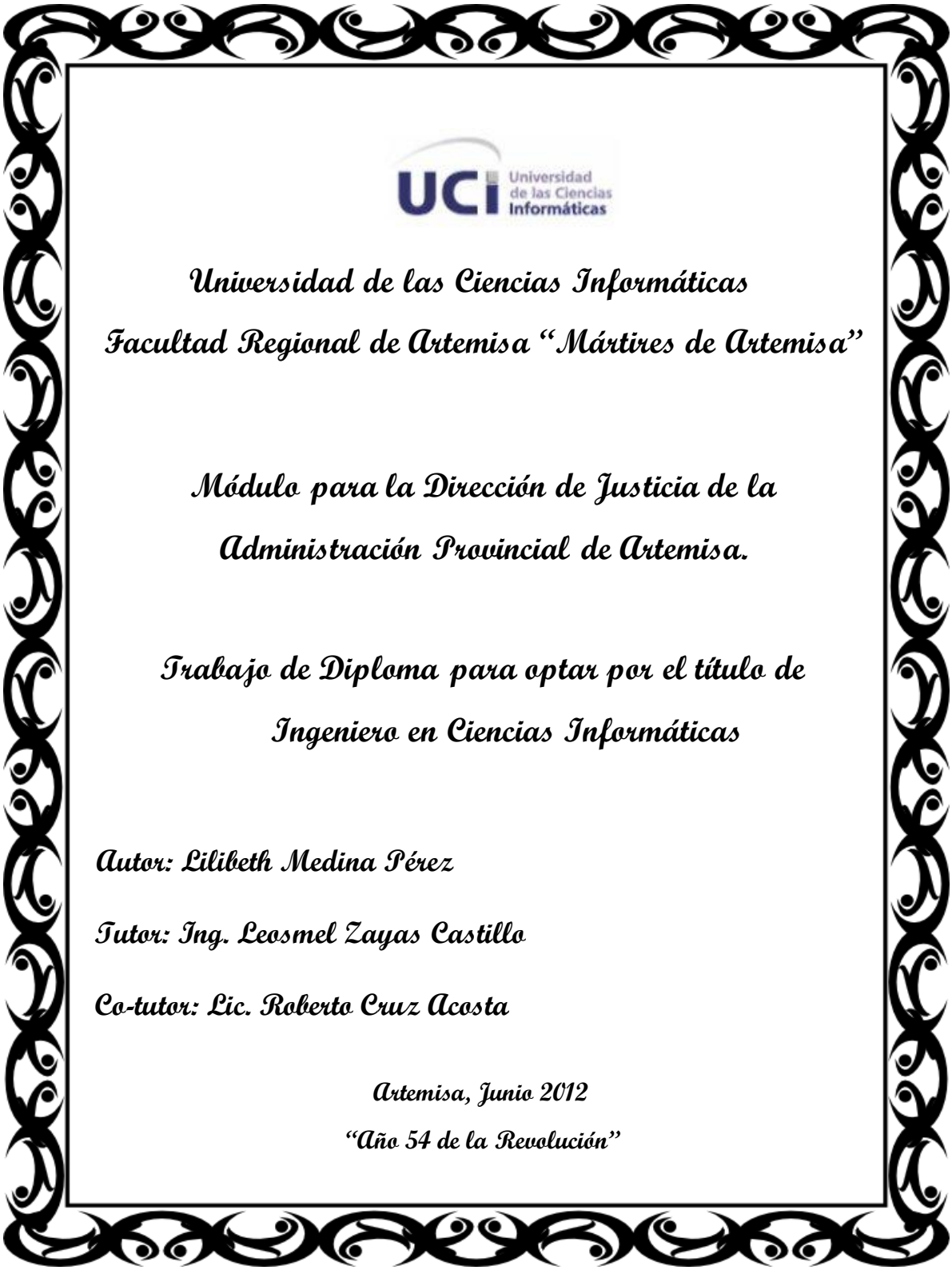


Universidad de las Ciencias Informáticas

Facultad Regional de Artemisa "Mártires de Artemisa"

# Módulo para la Dirección de Justicia de la Administración Provincial de Artemisa.

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


Autor: Lilibeth Medina Pérez

Tutor: Ing. Leosmel Zayas Castillo

Co-tutor: Lic. Roberto Cruz Acosta


Artemisa, Junio 2012

"Año 54 de la Revolución"

# Declaración de autoría

Declaro que soy el único autor de este trabajo y autorizo a la Facultad Regional "Mártires de Artemisa" de la Universidad de las Ciencias Informáticas; así como a dicho centro para que hagan el uso que estimen pertinente con este trabajo.

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

Lilibeth Medina Pérez

Ing. Leosmel Zayas Castillo

_____

Firma del Autor

_____

Firma del Tutor

"Seamos realistas y hagamos lo imposible"

Ernesto "Ché" Guevara

# Agradecimientos

Mi mayor agradecimiento es para mi familia que me ha estado apoyando a lo largo de la carrera, sobre todo a mi mamita y a mi abuela, Aya. A mi papá por exhortarme a continuar y luchar hasta el final. A mi tío por su preocupación. A mi padrastico, por estar siempre que lo necesité e indicarme el camino correcto.

Gracias a mis amigas Yailecín y Sonita que han estado a mi lado durante estos cinco años, y me han brindado su amistad incondicional. A Martica que siempre buscó la forma de sacarme de los malos ratos y estar en los buenos con una contagiosa sonrisa, le estaré eternamente agradecida. A Viviana por ser la hermana que siempre quise tener.

A Lizbety, mi compañera de tesis y prácticamente mi alma gemela, que juntas hemos pasado cada uno de los obstáculos con el mayor de los esfuerzos, para lograr nuestra meta, graduarnos. Gracias por ser una persona tan especial

Gracias a mi Sorbetero, por preocuparse y alentarme a continuar un camino que recién comienza, y porque sé que le quedará de satisfacción de verme finalmente graduada.

A mi tutor Leosmel, que siempre estaba de nuestro lado y apoyándonos hasta el cansancio, recordando sus viejos tiempos como estudiante y a todos los profesores que de una forma u otra me formaron profesionalmente, y dieron lo mejor de sí.

Gracias a todos (as) mis amigos (as) y a todas las personas que formaron parte de mi vida durante todo este tiempo y que contribuyeron a formarme como persona y como toda una profesional.

A TOD@S GRACIAS!!!

# Dedicatoria

Dedico este trabajo a mi mamita linda, a mi abuelita Aya. A Papi, mi abuelo que desafortunadamente no se encuentra conmigo, pero es mi gran ídolo, a mi tío y a mi papá. Aquí está todo el sacrificio y esfuerzo de estos cinco años. Los quiero mucho.

# Resumen

En la actualidad la gestión de la información es una de las principales tareas que se realizan en una organización, con el objetivo de llevar el control de los datos y agilizar los procesos que contribuyen a la toma de decisiones en tiempo oportuno, ejemplo de ello es en la Administración Provincial de Artemisa, que consta de varias direcciones con grandes volúmenes de información, haciendo engorroso el trabajo. Es por ello que se propone el desarrollo del proyecto Sistema Informativo de la Administración Provincial de Artemisa, con el objetivo de mejorar los procesos de gestión de la información que se manejan en dicha institución. Este proyecto está constituido por varios módulos, entre los que se encuentra el de la Dirección de Justicia.

En el presente trabajo se tienen en cuenta las diferentes etapas por las que transcurre el desarrollo del Módulo de la Dirección de Justicia de la Administración Provincial de Artemisa, entre las que se pueden mencionar: Análisis, Diseño, Implementación y Prueba las que tienen como finalidad alcanzar el desarrollo de un sistema que gestione correctamente la información y que cumpla las exigencias del cliente.

Palabras claves: Gestión de la Información, Módulo, Procesos.

# Índice

# Índice de Figuras

# Introducción

Las Tecnologías de la Información han sido conceptualizadas como la integración y convergencia de la computación, las telecomunicaciones y la técnica para el procesamiento de datos, donde sus principales componentes son: el factor humano, los contenidos de la información, el equipamiento, la infraestructura, el *software* y los mecanismos de intercambio de información.

La sociedad de estos tiempos tiene que hacer frente a sus necesidades de información. Ésta como un soporte de transmisión de conocimientos se convierte en algo casi vital, se caracteriza por ser concreta, precisa, coherente y adaptada a las exigencias que se requieren satisfacer. No se encuentra aislada ya que siempre se encuentra ligada a un entorno social y cultural.

Hoy en día se habla frecuentemente de la gestión de la información, presentándola como una tendencia de futuro imprescindible para encarar los retos de la sociedad de la información. A nivel mundial ocupa un papel importante en el desarrollo de los países, provocando un gran impacto en los cambios económicos, políticos, culturales y tecnológicos.

Con la gestión de la información se logra optimizar los procesos mediante una serie de estrategias, además de cumplir objetivos trazados por las organizaciones. Siempre teniendo en cuenta que las tecnologías son, necesariamente, un medio para trasmitir y gestionar conocimiento e información, como elemento fundamental para el desarrollo.

Actualmente muchas instituciones prescinden de gestionar la información que procesan de forma rápida y eficiente mediante el uso de sistemas informáticos, ya que se desenvuelven en una sociedad que evoluciona rápidamente y el crecimiento continuo y acelerado de la información, dificulta manipularla. El avance de las

tecnologías en el mundo se ha convertido en una herramienta al servicio de las estrategias de cada institución, como es el caso de los sistemas de gestión de la información capaces de obtener, almacenar, administrar, transmitir o recibir información organizada; permitiendo transformar datos en conocimiento de valor estratégico para facilitar la toma de decisiones en estas instituciones.

En Cuba el uso y desarrollo de las Tecnologías de la Información y las Comunicaciones (TIC) han propiciado la creación y puesta en marcha de sistemas que aportan numerosos beneficios en el desempeño, la productividad y la eficiencia de la economía cubana, además de elevar el nivel de conocimiento de la sociedad y provocar un gran impacto en todas las esferas del país.

A partir del triunfo de la Revolución cubana comienza este progreso, formando profesionales en la rama de la informática y las comunicaciones. Se inicia además el programa de informatización de la sociedad, priorizándose los sistemas de gestión de la información aplicados a la economía, la educación y la investigación para optimizar el uso, desarrollo y explotación de las mismas.

Muchos son los ejemplos del esfuerzo y avance del país, como la creación de los Joven Club de Computación, la fundación del Ministerio de Informática y las Comunicaciones (MIC), el surgimiento de los Institutos Politécnicos de Informática, el nacimiento de la Universidad de las Ciencias Informática (UCI) a raíz de la Batalla de Ideas; la cual está conformada actualmente por 7 facultades y tres facultades regionales.

Desde un inicio la UCI estuvo inmersa en la creación de varios *software* de desarrollo con el objetivo de contribuir a un mejor proceso de gestión de información en varios sectores de la sociedad. La facultad regional “Mártires de Artemisa” como parte de esta gran ciudad digital, está vinculada a diferentes proyectos productivos como es el caso del Sistema Informativo de la Administración Provincial de Artemisa, un acuerdo de cooperación con el Gobierno de la provincia,

con el propósito de informatizar los procesos gestión de la información que allí se manejan.

En el año 2011 el país sufre una nueva distribución político-administrativa surgiendo así las dos nuevas provincias Mayabeque y Artemisa, quedando en esta última, en el municipio cabecera el órgano del Gobierno Provincial, el cual está dividido en 32 órganos de direcciones y una de ellas es la de Justicia, esta dirección tiene como misión, ejecutar y asistir al Gobierno del territorio en la implementación de la política jurídica aprobada para su esfera de atención y controlar el cumplimiento de las disposiciones metodológicas que correspondan.

El proceso de gestión de información de la dirección de Justicia es defectuoso debido a que se realiza de forma manual, por vía telefónica, correo electrónico o mediante mensajeros de las entidades que se encuentran en los municipios de la provincia que entregan la información en formato duro o digital. Se dificulta también en la entidad el análisis de los reportes ya que los datos se presentan en formato “Excel”.

Por la falta de organización, existen deficiencias tales como el borrado indebido de archivos o modificaciones no autorizadas, pueden ocurrir además pérdidas y duplicados de los datos en informaciones relacionadas a las actividades del registro civil, registro de la propiedad y notarial, entre otros.

La información con la que se toman las decisiones muchas veces no se encuentra centralizada y cuando se necesita hacer una consulta de los datos se tiene que buscar de forma manual y dentro de grandes volúmenes de información, esto ocasiona problemas de eficiencia y confiabilidad a la hora de consultar los datos que se precisen en la dirección.

En la realización de las actividades diarias de la dirección, se puede decir que existen dificultades en la gestión de la información provocando la carencia de habilidades para realizar el trabajo requerido, causando grandes consecuencias afectando la toma de decisiones en la provincia y ocasionando un mal

funcionamiento en el proceso de gestión de información de la Administración Provincial de Artemisa.

Por todo lo antes descrito se plantea el **Problema Científico**: ¿Cómo contribuir al proceso de gestión de la información de la Dirección de Justicia, de forma que garantice la eficiencia y confiabilidad en la consulta de los datos en dicho proceso?

Asumiendo como **Objeto de Estudio:** La gestión de la información, limitado por el **Campo de Acción:** Los procesos de gestión de la información en la Dirección de Justicia en la Administración Provincial de Artemisa.

Como **Objetivo General:** Desarrollar un módulo para la Dirección de Justicia que garantice la eficiencia y confiabilidad en el proceso de gestión de información en la Administración Provincial de Artemisa.

Definido el objetivo general, se desglosan los siguientes **Objetivos Específicos:**

1. Establecer los fundamentos teórico-metodológicos para el desarrollo de los procesos de gestión de información.

2. Realizar el análisis y el diseño de la solución de software propuesta para la dirección de Justicia de la Administración Provincial de Artemisa.

3. Implementar el soporte para dar solución a los requerimientos de las aplicaciones clientes para la gestión de la información de la dirección de Justicia de la Administración Provincial de Artemisa.

4. Validar mediante pruebas de funcionales los resultados obtenidos con la solución.

La investigación está basada en la siguiente **Idea a Defender:** El desarrollo de un módulo para la Dirección de Justicia garantizará la eficiencia y confiabilidad en el

proceso de gestión de información en la Administración Provincial de Artemisa.

Para llevar a cabo la investigación de manera que se logre el objetivo propuesto, se trazaron las **Tareas de la Investigación** que se relacionan a continuación:

1. Establecer los fundamentos teórico-metodológicos para el desarrollo de los procesos de gestión de información.
2. Caracterizar el proceso de gestión de la información en la dirección de Justicia.
3. Desarrollar un módulo para la dirección de Justicia.
4. Validar los requisitos propuestos mediante el uso de técnicas para este fin.

Al concluir el trabajo se contará con los siguientes **resultados concretos:**

1. Módulo para la Dirección de Justicia de la Administración Provincial de Artemisa.

En el transcurso de la investigación se hace uso de varios métodos siendo de gran importancia para el desarrollo del trabajo. Los métodos que se utilizan son:

**Métodos Teóricos**

**Analítico – Sintético:** Permite el estudio y análisis de los procesos de gestión de la información, posibilitando identificar aquellos que puedan ser aplicados en el desarrollo de esta aplicación.

**Histórico – Lógico:** Permite hacer un estudio de los antecedentes de los procesos de gestión de la información en la Dirección de Justicia, para un mejor entendimiento de cuáles son las tendencias actuales y proponer soluciones acorde a las necesidades.

**Métodos Empíricos**

**Entrevista:** Se realiza para obtener las características de los procesos que tienen lugar en el área de la Dirección de Justicia y capturar los requisitos del sistema de información necesarios para el desarrollo del sistema. (Ver **Anexo1**)

**Población**: Información de la Dirección de Justicia.

Se define como población la Información de la Dirección de Justicia debido a que el módulo que se propone realizar, es para manejar el proceso de gestión de la información de dicha dirección.

**Muestra**: Reportes de Justicia.

La elaboración de los Reportes de Justicia es uno de los procesos que se realizan con el objetivo de presentar informaciones en esta dirección, es por ello que se definen estos reportes como muestra.

**Procedimientos y técnicas utilizados para seleccionar la muestra dada:** La técnica seleccionada para obtener la muestra fue la No Probabilística ya que para recopilar la información necesaria es preciso representar determinados elementos de la población. El procedimiento utilizado fue el Muestreo Intencional ya que a través de este se puede seleccionar personal que son representativos o con posibilidades de brindar mayor información dentro de la Dirección de Justicia.

**Variable Independiente**: Módulo para la Dirección de Justicia de la Administración Provincial de Artemisa.
**Variables dependientes**: Eficiencia y Confiabilidad.

La investigación está desarrollada en 3 capítulos, cuya estructura se describe a continuación:

**Capítulo 1 Fundamentación Teórica.**
Comprende los conceptos generales y básicos que permiten el entendimiento de temas relacionados con la investigación. Además, se precisa el estado del arte en

el ámbito nacional e internacional, así como se fundamentan las metodologías, tecnologías y herramientas utilizadas para el desarrollo del sistema.

**Capítulo 2 Características, análisis y diseño del sistema.**

Se realiza una propuesta del sistema, se describe cómo debe funcionar y se destaca sus características distintivas; además, se especifican sus requisitos funcionales y no funcionales. Se realiza el análisis y diseño del sistema donde se especifican los principales artefactos generados en las primeras fases.

**Capítulo 3 Adquisición y validación de los resultados del sistema.**

Incluye toda la información relacionada con el desarrollo del sistema. Además, se muestran la información adquirida a raíz de la realización de las pruebas al sistema a través de las pruebas unitarias.

# Capítulo 1: Fundamentación Teórica

**Introducción**

Se abordará en este capítulo las definiciones y conceptos fundamentales relacionados con los procesos y sistemas de gestión de la información, las principales herramientas informáticas y sus características más generales para la realización del producto y la metodología a utilizar para el correcto desarrollo del sistema. Se expone además, una perspectiva del estado del arte tanto en el ámbito nacional como internacional.

## 1.1 Conceptos fundamentales asociados al sistema

**Proceso:** Oscar Barros señala que "un proceso es un conjunto de tareas lógicamente relacionadas que existen para conseguir un resultado bien definido dentro de un negocio; por lo tanto, toman una entrada y le agregan valor para producir una salida. Los procesos tienen entonces clientes que pueden ser internos o externos, los cuales reciben a la salida, lo que puede ser un producto físico o un servicio. Éstos establecen las condiciones de satisfacción o declaran que el producto o servicio es aceptable o no". (Barros, 1994)

**Gestión:** El concepto de gestión hace referencia a la acción y al efecto de gestionar o de administrar. Gestionar es realizar diligencias conducentes al logro de un negocio o de un deseo cualquiera. (Alvero, 1976)

**Información:** Es la acción y resultado de informar o informarse. La información es un conjunto organizado de datos, que constituye un mensaje sobre un cierto fenómeno o ente. La misma permite resolver problemas y tomar decisiones, ya que su uso racional es la base del conocimiento.

**Sistema:** Conjunto de elementos que se encuentran interrelacionados y que interactúan entre sí. El concepto se utiliza tanto para definir a un conjunto de

conceptos como a objetos reales dotados de organización.

**Módulo:** En programación un módulo es una parte de un programa de ordenador. De las varias tareas que debe realizar un programa para cumplir con su función u objetivo, un módulo realiza una o varias tareas. (Lastre, 2008)

**Proceso de Gestión:** Aquel proceso de optimización de los recursos puestos a disposición de una organización, con el fin de obtener sus objetivos. Comprende las etapas de organización, planificación, ejecución y control. (De Heredia, 1995)

**Gestión de la Información:** Integra en su organización el conjunto de instancias que la constituyen en función de hacer cumplir: cómo la información se adquiere, registra, almacena, distribuye y usa, cómo el personal designado maneja y hace llegar la información a los usuarios directos, cómo las personas usan la información, desarrollan habilidades informativas y se convierten en divulgadores de la misma, cómo las tecnologías de la información se incorporan y perfeccionan los diferentes procesos de la gestión, cómo la captación y uso de la información incide en el crecimiento humano y organizacional con mejores resultados, lo que incide en los costos y beneficios de la organización.

**¿Qué son los Sistemas de Gestión de la Información?**

Definidos como conjuntos de funciones o componentes interrelacionados que forman un todo, obtienen, procesan, almacenan y distribuyen la información, manipulando los datos y consiguiendo, para una organización o empresa, la búsqueda de mejores vías para la dirección y control correspondiente de sus procesos. Un sistema de gestión es una estructura probada para la gestión y mejora continua de las políticas, los procedimientos y procesos de la organización, apoyando la toma de las decisiones en el desempeño de las funciones y ayudando a lograr los objetivos de la organización mediante una serie de estrategias, que incluyen la optimización de procesos, el enfoque centrado en la gestión y el pensamiento disciplinado.

## 1.2 Estado del Arte

Los sistemas de gestión de información juegan un papel en las empresas que ha experimentado un cambio considerable en estos últimos años, pasando de ser simples herramientas de tratamiento de datos para convertirse en la columna vertebral de cualquier organización, tanto a nivel interno como en lo referente a las relaciones con el exterior: clientes, proveedores, administración o la sociedad en general.

En la actualidad se pueden encontrar soluciones que pretenden ser globales y ofrecer soporte a todo el proceso de gestión de información en una organización. La variedad de sistemas de gestión de información en el mundo es difícil de especificar. Cada empresa o negocio tiene características y procesos determinados dentro de sus flujos de trabajo que requieren funcionalidades diferentes en cada sistema empleado.

El desarrollo de la informática y el avance tecnológico que se experimenta en el mundo, ha hecho posible la creación de *software* para la gestión de información en el sector jurídico; Cuba ha dado su aporte, como lo ha hecho en otras áreas de la informática y se empeña en continuar desarrollándose en este sentido.

A continuación se exponen algunas de las soluciones encontradas:

**En el ámbito internacional**

**Navarra, España:** Avantius es un producto de gestión integral de expedientes judiciales, que permite que los distintos órganos judiciales, fiscales, forenses o cualquier otro profesional interno o externo puedan incorporarse a un proceso con la debida seguridad, pudiendo intervenir en un expediente único para su resolución. Sus principales funciones son el registro y reparto de todos los documentos presentados ante la Administración de Justicia, la tramitación de dichos asuntos, la remisión de información a otras oficinas y servicios jurisdiccionales, las búsquedas

de información y la explotación de la información registrada. Se trata de un modelo informático, que desarrollado y actualizado por la empresa pública TRACASA. Las deficiencias en materia de estadística que se arrastra en la utilización del sistema Avantius y un mal funcionamiento del mismo traen a que en ocasiones los procedimientos no se remiten correctamente a Fiscalía, e impide que puedan tramitarse adecuadamente. La razón principal de los problemas expuestos radica en que, en un primer momento, Avantius se creó adaptado a las necesidades del Juzgado observándose que no se habían tenido en cuenta las peculiaridades propias del trabajo de Fiscalía

**Islas Canarias, España:** Atlante II es un sistema de gestión procesal de la Administración de Justicia de Canarias, supone, ante su predecesor, Atlante I, un salto cualitativo en el conjunto de servicios facilitados al entorno de la justicia en Canarias. Actualmente este sistema llega a su fase final sin solventar los problemas que se detectaron desde un primer momento y sigue provocando continuas "caídas" de los servidores de los juzgados isleños, provocando consigo que el funcionario debe realizar su trabajo en el programa Microsoft Word y posteriormente, pasarlo al sistema Atlante II, cuando éste funcione. Además de se habla de vulnerabilidades de acceso remoto, la falta de seguridad del sistema ha traído consigo grandes consecuencias en las investigaciones judiciales del gobierno. (Rodríguez, 2001)

**Madrid, España:** Minerva-NOJ es una aplicación de gestión procesal, sucesora de Minerva, que soporta la tramitación de la información relativa a los procedimientos judiciales, de forma que cualquier órgano judicial implicado en la tramitación de un determinado procedimiento, pueda acceder (o debería poder) a la información asociada al mismo con las garantías de reserva, control y confidencialidad requeridas. Minerva-NOJ provoca un gran colapso en los juzgados por su mal funcionamiento, pues existe duplicación de actos, caídas constantes de dos o tres veces en un documento, presenta una limitada capacidad operativa, y esto ocurre debido a que cada día aumentan los juzgados conectados, y este colapso va en

aumento ocasionando problemas a la hora de las tramitaciones.

**En el ámbito nacional**

**Villa Clara, Cuba:** El sistema SisProP fue propuesto para abarcar las instancias Supremo y Provincial en el módulo penal, pero únicamente se desarrolló la tramitación de los procesos y apelaciones competencia del tribunal provincial. El sistema fue mal concebido desde su inicio y presenta un grupo de limitaciones que se resumirán a continuación, que tienen como causa fundamental ese error de concepción:

- No obtiene datos de la fase judicial del proceso.
- No aporta estadística, ni información alguna y al no haberse programado la introducción de los datos de la fase judicial señalados en el punto uno no es solucionable.
- El nivel de informatización que supone el sistema es mínimo y en el caso de las apelaciones es prácticamente nulo.
- El sistema no valida casi ningún dato.
- El sistema está programado en Delphi y corre sobre SQL Server, por lo que es incompatible con el *software* libre en el que se están programando los sistemas generales de cada materia judicial, fue dictaminado así por el equipo de especialistas de la UCI y fue aceptado por el propio especialista informático que hizo este sistema, Msc. Daniel E. Castro Morel, quien expresó además no estar en posibilidades de reprogramarlo si se decidiera hacerlo. (Castro, 2008)

**Ciudad de la Habana, Cuba:** En la Universidad de las Ciencias Informáticas en años anteriores se realizaron sistemas de gestión de la información relacionados con justicia, tal es el caso del sistema de gestión del proceso de Diligencias Previas

para el proyecto Tribunales Populares Cubanos (TPC) que contribuye con el mejoramiento de la gestión de la documentación oficial que transita por los TPC. Este sistema aunque fue terminado en el 2010 aún no está siendo utilizado por el cliente, y tuvo pequeños problemas cuando se concluyó.

Después de realizar un estudio de las soluciones informáticas en el ámbito nacional e internacional, se confirmó la existencia de varios *software* que se encargan de gestionar la información en el sector de justicia, la mayoría de estos, propietarios, sin embargo, hay otros que son de fácil adquisición, pero no se ajustan a las necesidades del cliente, por lo tanto no cumplen con los requisitos del sistema a implementar. Por esta razón es que se propone el desarrollo del módulo de la dirección de justicia de la Administración Provincial de Artemisa.

### 1.3 Metodología de desarrollo

Una Metodología de desarrollo de *software* es un conjunto de procedimientos, técnicas y ayudas a la documentación para el desarrollo de productos *software*. En ella se va indicando paso a paso todas las actividades a realizar para lograr el producto informático deseado, indicando además qué personas deben participar en el desarrollo de las actividades y qué papel deben tener. (Menéndez, 2011)

Muchas veces no se toma en cuenta que el utilizar una metodología adecuada, inciden en distintas dimensiones del proceso de desarrollo. Dichas metodologías detallan la información que se debe producir como resultado de una actividad y la información necesaria para comenzarla. En ocasiones el proceso de desarrollo resulta riesgoso y se convierte en una tarea difícil por lo que es necesario hallar el modo de controlar su curso de principio a fin. El problema principal radica en cómo coordinar todas las actividades que comprende el desarrollo de un proyecto, sobre todo si se trata de un proyecto de gran envergadura. De modo que se torna imprescindible contar con una forma organizada y adecuadamente estructurada para trabajar. El uso de una metodología de desarrollo acertada, permite obtener desarrolladores satisfechos con el *software* de calidad que ha sido fruto de su

trabajo.

Las metodologías para el desarrollo de *software* pueden ser clasificadas como:

- Metodologías Tradicionales o Robustas
- Metodologías Ágiles o Livianas

### 1.3.1 Metodologías tradicionales

Se caracterizan por exponer procesos basados en planeación exhaustiva. Ésta se realiza esperando que el resultado de cada proceso sea determinante y predecible. Se centran especialmente en el control del proceso, estableciendo rigurosamente las actividades involucradas, los artefactos que se deben producir, las herramientas y notaciones que se usarán. (Manios, 2009)

**Están caracterizadas por:**

- Ser rígidas y dirigidas por la documentación que se genera en cada una de las actividades desarrolladas.
- Ofrecer cierta resistencia a los cambios.
- Constituir un proceso mucho más controlado, con numerosas políticas y normas.
- Más artefactos y más roles.
- La arquitectura del *software* es esencial y se expresa mediante modelos.

El ejemplo más representativo e importante de este tipo de metodologías es el Proceso Unificado de Desarrollo *Software* o Proceso Unificado de Rational ya que los dos nombres suelen utilizarse para referirse a un mismo concepto.

**Proceso Unificado de desarrollo de software (RUP)**

Las siglas RUP (en inglés significa *Rational Unified Process*) Proceso Unificado de Rational es un producto del proceso de ingeniería de *software* que proporciona un enfoque disciplinado para asignar tareas y responsabilidades dentro de una organización del desarrollo. Su meta es asegurar la producción del *software* de alta calidad que resuelve las necesidades de los usuarios dentro de un presupuesto y tiempo establecidos. (Contreras, 2006)

En RUP se han agrupado las actividades en grupos lógicos definiéndose 9 flujos de trabajo principales. Los 6 primeros son conocidos como flujos de ingeniería y los tres últimos como de apoyo. La **Figura 1** representa el proceso en el que se grafican los flujos de trabajo, las fases y muestra la dinámica expresada en iteraciones y puntos de control.

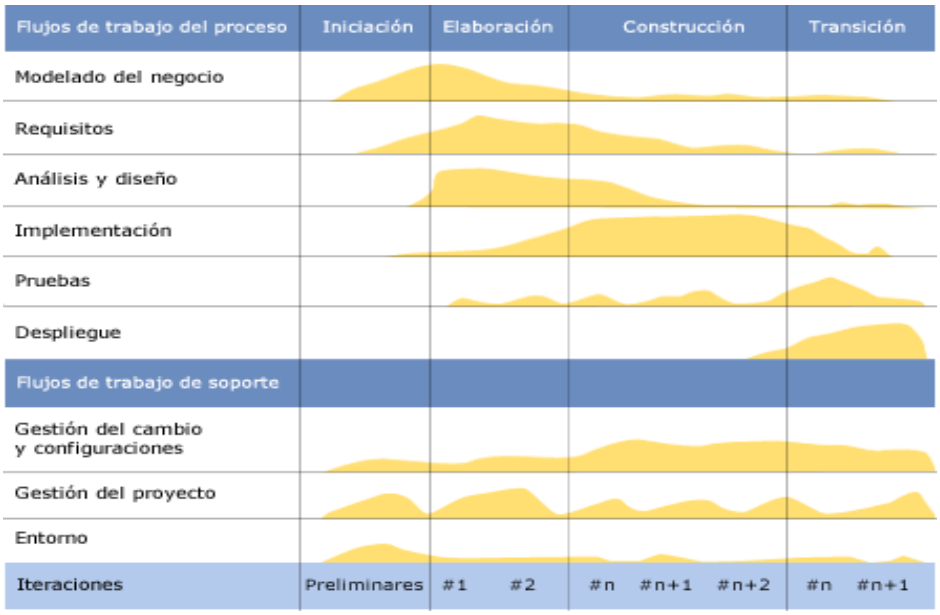



**Figura 1:** Flujos y fases del RUP

**Características de RUP:**

- Dirigido por Casos de Uso
- Centrado en la arquitectura

- Iterativo e incremental

El RUP, además, divide el proceso de desarrollo en ciclos, teniendo un producto final al culminar cada una de ellos, estos a la vez se dividen en fases que finalizan con un hito donde se debe tomar una decisión importante:

**Fases definidas por RUP:**

**Inicio:** Se describe el negocio y se delimita el proyecto describiendo sus alcances con la identificación de los casos de uso del sistema.

**Elaboración:** Se define la arquitectura del sistema y se obtiene una aplicación ejecutable que responde a los casos de uso que la comprometen.

**Construcción:** Se obtiene un producto listo para su utilización que está documentado y tiene un manual de usuario.

**Transición:** El *release* ya está listo para su instalación en las condiciones reales. Puede implicar reparación de errores.

### 1.3.2 Metodologías ágiles

Las metodologías ágiles forman parte del movimiento de desarrollo ágil de *software* conocidos anteriormente como metodologías livianas, que se basan en la adaptabilidad de cualquier cambio como medio para aumentar las posibilidades de éxito de un proyecto. Las metodologías ágiles intentan evitar los tortuosos y burocráticos caminos de las metodologías tradicionales enfocándose en la gente y los resultados. Se basan en promover iteraciones en el desarrollo a lo largo de todo el ciclo de vida del proyecto, logrando que se minimicen los riesgos desarrollando *software* en cortos tiempo. El *software* desarrollado en una unidad de tiempo es llamado una iteración, la cual debe durar poco tiempo. Cada iteración del ciclo de vida incluye: planificación, análisis de requerimientos, diseño, codificación, revisión y documentación.

**Las Metodologías ágiles se basan en los siguientes principios:**

- Realizar entregas cortas en el tiempo y continuas.
- Dar la bienvenida a los cambios.
- Entregas periódicas y frecuentes que funcionen.
- Los clientes forman parte del equipo de desarrollo.
- Pocos artefactos y pocos roles.
- Buen diseño y calidad técnica.
- Equipos auto-organizados. (Acebey, 2008)

Entre las metodologías ágiles se pueden mencionar:

**Programación Extrema (XP)**

Programación Extrema (del inglés *Extreme Programing*), es una metodología ágil para el desarrollo de *software* y consiste básicamente en ajustarse estrictamente a una serie de reglas que se centran en las necesidades del cliente para lograr un producto de buena calidad en poco tiempo. La Programación Extrema es una metodología centrada en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de *software*. Promueve el trabajo en equipo, preocupándose en todo momento del aprendizaje de los desarrolladores y estableciendo un buen clima de trabajo. Este método se basa en una realimentación continuada entre el cliente y el equipo de desarrollo con una comunicación fluida entre todos los participantes, también busca simplificar las soluciones implementadas y coraje para los múltiples cambios. Este tipo de programación es la adecuada para los proyectos con requisitos imprecisos, muy cambiantes y con un riesgo técnico excesivo. Entre los roles de XP se encuentran: Programador, Cliente, Entrenador, Rastreador y Probador. El ciclo de vida ideal de

XP consta de cuatro a seis fases, las principales son: Planificación (dentro de la que se encuentra Exploración), Desarrollo (donde se realizan tantas iteraciones como sean necesarias), Entrega y Mantenimiento y en algunos casos en que sea necesario se puede introducir una Fase Legal.

**Características de XP:**

- **Pruebas unitarias:** se basan en las pruebas realizadas a los principales procesos, de tal manera que adelante en algo hacia el futuro, se pueda hacer pruebas de las fallas que puedan ocurrir. Es como si se adelantara a obtener los posibles errores.

- **Re-fabricación:** se basa en la reutilización de código, para la cual se crean patrones o modelos estándares, siendo más flexible al cambio.

- **Programación en pares:** consiste en que dos desarrolladores participen en un proyecto en una misma estación de trabajo. Cada miembro lleva a cabo la acción que el otro no está haciendo en ese momento.

**SCRUM**

Es un proceso en el que se aplican de manera regular un conjunto de mejores prácticas para trabajar en equipo y obtener el mejor resultado posible de un proyecto. Estas prácticas se apoyan unas a otras y su selección tiene origen en un estudio de la manera de trabajar de equipos altamente productivos. Scrum, más que una metodología de desarrollo *software*, es una forma de auto-gestión de los equipos de programadores. Un grupo de programadores deciden cómo hacer sus tareas y cuánto van a tardar en ello. Scrum ayuda a que trabajen todos juntos, en la misma dirección, con un objetivo claro. Permite además seguir de forma clara el avance de las tareas a realizar, de forma que los jefes puedan ver día a día cómo progresa el trabajo. En Scrum se realizan entregas parciales y regulares del resultado final del proyecto, priorizadas por el beneficio que aportan al receptor del

proyecto. Por ello, Scrum está especialmente indicado para proyectos en entornos complejos, donde se necesita obtener resultados pronto, donde los requisitos son cambiantes o poco definidos, donde la innovación, la competitividad y la productividad son fundamentales. Esta metodología tiene algunas desventajas, como el hecho de que genera muy poca documentación en comparación con otras metodologías, no es apto para todos los proyectos y en muchas ocasiones es necesario completarlo con otros procesos de XP.

### 1.3.3 Metodología a utilizar

Es evidente que se necesita una metodología de trabajo, unas pautas a seguir que ayuden a coordinar las complejas tareas que suponen el desarrollo de *software*. Después de haber estudiado las metodologías de desarrollo anteriormente expuestas, se llega a la conclusión que las metodologías ágiles son poco orientadas al documento, exigiendo una cantidad pequeña de documentación para una tarea dada. De muchas maneras son más bien orientadas al código, siguiendo un camino que dice que la parte importante de la documentación es el código fuente.

Se propone entonces el uso de las metodologías ágiles enunciadas anteriormente, Scrum para la parte de planificación, y XP para la parte de desarrollo. Debido a las grandes ventajas que proporcionan ambas metodologías, se decide emplear para el desarrollo de este trabajo el uso de la metodología ágil SXP, que presenta características ajustables al proyecto.

**SXP**

Compuesta por las metodologías Scrum y XP ofrece una estrategia tecnológica, a partir de la introducción de procedimientos ágiles que permitan actualizar los procesos de *software* para el mejoramiento de la actividad productiva fomentando el desarrollo de la creatividad, aumentando el nivel de preocupación y responsabilidad de los miembros del equipo, ayudando al líder del proyecto a tener

un mejor control del mismo.

**Scrum** proporciona una forma de gestionar un equipo de manera que trabaje eficientemente y de tener siempre medidos los progresos, de forma que sepan por dónde andan.

**XP** más bien es una metodología encaminada para el desarrollo; que consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo, al usuario final, pues es uno de los requisitos para llegar el éxito del proyecto.


**Figura 2:** Unión de SCRUM + XP

Ambas metodologías de acoplan de la siguiente manera: Scrum puede ser visto como una interfaz entre el equipo y los clientes, esta metodología deja un agujero en el medio (ejemplo: cómo el equipo debe hacer su trabajo diario), en el cual XP encaja muy bien, como muestra la **Figura 2**. Una vez que Scrum es puesto en práctica el equipo es auto administrado y puede escoger como hacer el trabajo. El desarrollo de *software* ágil es todo sobre círculos de retroalimentación. Scrum y XP se complementan el uno al otro de manera muy positiva. Las dos metodologías dan un círculo de retroalimentación de unos pocos segundos, donde los defectos son detectados y corregidos en segundos.

**SXP consta de 4 fases principales:**

- **Planificación-Definición:** donde se establece la visión, se fijan las expectativas y se realiza el aseguramiento del financiamiento del proyecto;
- **Desarrollo:** es donde se realiza la implementación del sistema hasta que esté listo para ser entregado;
- **Entrega:** es la puesta en marcha; y por último
- **Mantenimiento:** es la fase donde se realiza el soporte para el cliente.

De cada una de estas fases se realizan numerosas actividades tales como el levantamiento de requisitos, la priorización de la Lista de Reserva del Producto, definición de las Historias de Usuario, Diseño, Implementación, Pruebas, entre otras; de donde se generan artefactos para documentar todo el proceso.

SXP está especialmente indicada para proyectos de pequeños equipos de trabajo, rápido cambio de requisitos o requisitos imprecisos, muy cambiantes, donde existe un alto riesgo técnico y se orienta a una entrega rápida de resultados y una alta flexibilidad. Ayuda a que trabajen todos juntos, en la misma dirección, con un objetivo claro, permitiendo además seguir de forma clara el avance de las tareas a realizar, de forma que se pueda ver cómo progresa el trabajo.

## 1.4 Herramientas y tecnologías asociadas al desarrollo del *software*

### 1.4.1 Lenguaje de modelado

El lenguaje de modelado de objetos es un conjunto estandarizado de símbolos y de modos de disponerlos para modelar un diseño de *software* orientado a objetos o parte de este.

**Lenguaje de Modelado Unificado (UML) 2.0**

Es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de *software*. Captura decisiones y

conocimientos sobre los sistemas que se deben construir. Se usa para entender, diseñar, configurar, mantener, y controlar la información sobre tales sistemas. Está pensado para emplearse con todos los métodos de desarrollo, etapas del ciclo de vida, dominios de aplicación y medios. El lenguaje de modelado pretende unificar la experiencia pasada sobre técnicas de modelado e incorporar las mejores prácticas actuales en un acercamiento estándar. UML incluye conceptos semánticos, notación, y principios generales. Tiene partes estáticas, dinámicas, de entorno y organizativas. Puede ser utilizado en herramientas interactivas de modelado visual que tengan generadores de código así como generadores de informes. La especificación de UML no define un proceso estándar pero se ideó para ser útil en un proceso de desarrollo iterativo. Pretende dar apoyo a la mayoría de los procesos de desarrollo orientados a objetos. (Rumbaugh, 2000)

### 1.4.2 Herramienta de ingeniería del software asistida por computadoras (CASE)

Una herramienta CASE es un conjunto de aplicaciones informáticas que tienen el objetivo de aumentar la productividad en el desarrollo de un *software* mitigando los costes en términos de tiempo. (Pressman, 2005)

La utilidad de estas herramientas consiste principalmente en realizar un buen diseño del proyecto y a partir de este, implementar parte del código automáticamente, documentar o detectar errores, entre otras que hacen de ellas una fuente importante para la creación de un producto con calidad.

**Visual Paradigm 6.4**

Es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de *software*: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El *software* de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código

desde diagramas y generar documentación. La herramienta UML CASE también proporciona abundantes tutoriales, demostraciones interactivas y proyectos UML.

Se decide emplear como herramienta CASE Visual Paradigm debido a que esta cuenta con la ventaja de ser multiplataforma a diferencia de las otras herramientas; soporta el ciclo de vida completo del desarrollo de *software*. Esta herramienta es de gran utilidad para el proyecto por su nivel de integración con el entorno de desarrollo y por la posibilidad de que ambos se ejecuten sobre la plataforma libre GNU/Linux, sistema operativo sobre el cual está montado el entorno de desarrollo del proyecto.

### 1.4.3 Entornos de desarrollo integrados (IDE)

Un entorno de desarrollo integrado (en inglés *Integrated Development Environment* o IDE) es un programa compuesto por una serie de herramientas que utilizan los programadores para desarrollar código. Esta herramienta puede estar ideada para su utilización con un único lenguaje de programación o bien puede dar cabida a varios de estos.

**NetBeans 7.0.1**

Es un entorno de desarrollo, hecho principalmente para el lenguaje de programación Java. La plataforma NetBeans es una base modular y extensible usada como una estructura de integración para crear aplicaciones de escritorio grandes. Empresas independientes asociadas, especializadas en desarrollo de *software*, proporcionan extensiones adicionales que se integran fácilmente en la plataforma y que pueden también utilizarse para desarrollar sus propias herramientas y soluciones. Es una herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas. NetBeans IDE es un producto libre y gratuito sin restricciones de uso. El NetBeans es un IDE de código abierto de gran éxito con una gran base de usuarios.

**Eclipse**

Es un entorno de desarrollo integrado, de código abierto y multiplataforma. Mayoritariamente se utiliza para desarrollar lo que se conoce como "Aplicaciones de Cliente Enriquecido", opuesto a las aplicaciones "Cliente-liviano" basadas en navegadores. Es una potente y completa plataforma de Programación, desarrollo y compilación de elementos tan variados como sitios Web, programas en C++ o aplicaciones Java. No es más que un entorno de desarrollo integrado en el que se encuentren todas las herramientas y funciones necesarias para trabajar, recogidas además en una atractiva interfaz que lo hace fácil y agradable de usar.

Eclipse es un entorno de desarrollo integrado que, aunque pretende ser un entorno versátil tolerando varios lenguajes de programación, con el que mejor se integra es con el lenguaje Java. Es también gratuito pero mucho más complejo.

**Selección del IDE**

NetBeans es un entorno de desarrollo que tiene una interfaz amigable y fácil de comprender aun cuando los usuarios son inexpertos, sencillo, libre y de código abierto, característica fundamental en un IDE. La arquitectura de *plugins* de Netbeans permite, además de integrar diversos lenguajes sobre un mismo IDE, introducir otras aplicaciones accesorias que pueden resultar útiles durante el proceso de desarrollo como: herramientas UML, editores visuales de interfaces y ayuda en línea para librerías. Por todas estas características que presenta NetBeans IDE y por ser además una plataforma flexible es que se determinó el uso de este IDE para el desarrollo del sistema.

### 1.4.4 Lenguajes de programación

Un lenguaje de programación, es un lenguaje que puede ser utilizado para controlar el comportamiento de una máquina, particularmente una computadora. Consiste en un conjunto de reglas sintácticas y semánticas que definen su estructura y el

significado de sus elementos, respectivamente.

**Java**

En la actualidad es un lenguaje muy extendido y cada vez cobra más importancia tanto en el ámbito de Internet como en la informática en general. Es una tecnología orientada al desarrollo de *software* con la cual se puede realizar cualquier tipo de programa.

En un primer nivel, Java es un lenguaje de programación de propósito general, orientado a objetos, que fue introducido por Sun Microsystems en 1995, y diseñado en principio para el ambiente distribuido de Internet. Pero lo que hace de Java un concepto diferente es que, en un segundo nivel, es también un entorno para la ejecución de programas, englobado en la llamada máquina virtual de Java. Este entorno es un *software* que permite que las aplicaciones escritas en Java se ejecuten en cualquier ordenador, independientemente del sistema operativo y de la configuración de *hardware* utilizados. (Enjoiras, 1998)

Algunas de sus características son:

- Robustez.
- Gestión automática de la memoria.
- Permite programación multihilos.
- Soporta la arquitectura cliente-servidor.
- Mecanismos de seguridad incorporados.
- Herramientas de documentación incorporadas.

Java surge como el lenguaje de programación para ambiente de red por excelencia, ya que es orientado a objetos, portable, distribuido, robusto, seguro,

dinámico, interpretado, de arquitectura neutral, de alto rendimiento y sencillo. Se convirtió en un lenguaje potente, seguro y universal gracias a que lo puede utilizar todo el mundo y es gratuito. Estas características hacen que Java sea muy usado en el desarrollo de aplicaciones de gestión. Por lo antes descrito, se decidió utilizar este lenguaje para el desarrollo de la solución propuesta.

### 1.4.5 Marcos de trabajos que soportarán el desarrollo (*Framework* de desarrollo)

El término *framework* se refiriere a una estructura *software* compuesta de componentes personalizables e intercambiables para el desarrollo de una aplicación. En otras palabras, un *framework* se puede considerar como una aplicación genérica incompleta y configurable a la que puede añadirse las últimas piezas para construir una aplicación concreta. (Gutierres, 2008)

Los objetivos principales que persigue un *framework* son: acelerar el proceso de desarrollo, reutilizar código ya existente y promover buenas prácticas de desarrollo como el uso de patrones.

**Jwebsocket 1.0**

Es una tecnología orientada al desarrollo de aplicaciones basadas en *websockets* que gozan de altos niveles de velocidad, escalabilidad y seguridad. Es una solución *Open-Source* en Java y JavaScript para el protocolo WebSocket de HTML5. Con Jwebsocket se puede crear fácilmente aplicaciones de noticias, monitorización, chat, redes y aplicaciones sociales, juegos en línea o suites de colaboración en línea. Ofrece una amplia gama de funcionalidad de desde el intercambio de bajo nivel basado en *tokens*, hasta la sincronización de interfaz gráfica de usuario, las llamadas a procedimiento remoto y mucho más. Con esta tecnología se puede programar rápidamente una solución independiente con un mínimo de instalación y configuración o puede fácilmente integrar en sus aplicaciones existentes. Reutilizando todo el vasto cúmulo de tecnologías, librerías y aplicaciones existentes

para la tecnología Java, puede decirse que Jwebsocket es una sólida base para la creación de las más diversas aplicaciones web basadas en websockets.

**Spring**

Es un *framework* de código abierto de desarrollo de aplicaciones para la plataforma Java. Por su diseño el *framework* ofrece mucha libertad a los desarrolladores en Java y soluciones muy bien documentadas y fáciles de usar.

Está diseñado desde el principio para ayudar a escribir código fácil de probar. Presenta una potente gestión de configuración basada en JavaBeans, aplicando los principios de Inversión de Control. El uso de un contenedor de Inversión de Control reduce grandemente la complejidad del código a interfaces, más que a clases. El objetivo central de Spring es permitir que objetos de negocio y de acceso a datos sean reutilizables.

Spring es poderoso sin dejar de ser flexible y simple. Facilita la manipulación de objetos, elimina la necesidad de usar distintos y variados tipos de ficheros de configuración, mejora la práctica de programación y suaviza la curva de aprendizaje favorablemente para el desarrollador.

**Selección del *Framework* de desarrollo**

Se escogió Jwebsocket como *framework* de desarrollo pues está basado en el protocolo de comunicación Websocket, tecnología que proporciona un canal de comunicación bidireccional y full-dúplex sobre un único socket TCP. Facilita el desarrollo y funcionamiento de la aplicación, contando con una gran variedad de utilidades tanto para la conectividad como para las propias funcionalidades y sistemas de seguridad por el lado del servidor.

### 1.4.6 Sistemas de control de versiones (CVS)

**Subversion 1.6.12**

Es un sistema de control de versiones de *software* libre bajo una licencia de tipo Apache/BSD y se le conoce también como svn por ser el nombre de la herramienta utilizada en la línea de órdenes. Una característica importante de Subversion es que, a diferencia de CVS, los archivos versionados no tienen cada uno un número de revisión independiente, en cambio, todo el repositorio tiene un único número de versión que identifica un estado común de todos los archivos del repositorio en un instante determinado.

Subversion puede acceder al repositorio a través de redes, lo que le permite ser usado por personas que se encuentran en distintas computadoras. A cierto nivel, la posibilidad de que varias personas puedan modificar y administrar el mismo conjunto de datos desde sus respectivas ubicaciones fomenta la colaboración. Se puede progresar más rápidamente sin un único conducto por el cual deban pasar todas las modificaciones. Y puesto que el trabajo se encuentra bajo el control de versiones, no hay razón para temer porque la calidad del mismo vaya a verse afectada si se ha hecho un cambio incorrecto a los datos, simplemente se deshace ese cambio.

**RapidSVN 0.12.0**

RapidSNV es un cliente de Subversion multiplataforma, *software* libre que se utiliza para gestionar los datos del repositorio. Se distribuye bajo la licencia pública general de GNU y permite la gestión los documentos e información a partir de la asignación de permisos según el rol que desempeñe el cliente.

### 1.4.7 Herramienta para la gestión de proyectos Java

**Maven 2.2.1**

Es una herramienta de *software* para la gestión y construcción de proyectos Java creada por Jason van Zyl, de Sonatype, en 2002. Es similar en funcionalidad a Apache Ant (y en menor medida a PEAR de PHP y CPAN de Perl), pero tiene un

modelo de configuración de construcción más simple, basado en un formato XML. Estuvo integrado inicialmente dentro del proyecto Jakarta pero ahora ya es un proyecto de nivel superior de la *Apache Software Foundation*.

Maven utiliza un *Project Object Model* (POM) para describir el proyecto de *software* a construir, sus dependencias de otros módulos y componentes externos, y el orden de construcción de los elementos. Viene con objetivos predefinidos para realizar ciertas tareas claramente definidas, como la compilación del código y su empaquetado.

Una característica clave de Maven es que está listo para usar en red. El motor incluido en su núcleo puede dinámicamente descargar *plugins* de un repositorio, el mismo repositorio que provee acceso a muchas versiones de diferentes proyectos *Open Source* en Java, de Apache y otras organizaciones y desarrolladores. Este repositorio y su sucesor reorganizado, el repositorio Maven 2, pugnan por ser el mecanismo de facto de distribución de aplicaciones en Java, pero su adopción ha sido muy lenta. Maven provee soporte no sólo para obtener archivos de su repositorio, sino también para subir artefactos al repositorio al final de la construcción de la aplicación, dejándola al acceso de todos los usuarios. Una caché local de artefactos actúa como la primera fuente para sincronizar la salida de los proyectos a un sistema local.

Maven está construido usando una arquitectura basada en *plugins* que permite que utilice cualquier aplicación controlable a través de la entrada estándar. En teoría, esto podría permitir a cualquiera escribir *plugins* para su interfaz con herramientas como compiladores, herramientas de pruebas unitarias, etcétera, para cualquier otro lenguaje. En realidad, el soporte y uso de lenguajes distintos de Java es mínimo.

**Conclusiones Parciales**

El capítulo finalizado, contiene en sus inicios un estudio de los principales conceptos sobre los procesos y sistemas de gestión de la información. Se tuvo en

cuenta además, un conocimiento previo de las metodologías, con el objetivo de seleccionar la indicada para minimizar el tiempo de desarrollo del sistema a realizar. Se abordó sobre las herramientas y tecnologías a emplear para el desarrollo del *software*, exponiendo algunas de las características y ventajas de las mismas así como dando una valoración del porqué de la selección realizada de cada una, considerando que es la mejor propuesta para el producto a desarrollar. Se realizó un estudio profundo del estado del arte, donde se verificó la existencia de varios *software* a nivel internacional y nacional, resultando que estas soluciones informáticas en el sector Jurídico, no se ajustan al sistema que se implementará.

# Capítulo 2: Características, análisis y diseño del sistema

**Introducción**

En este capítulo se abordan elementos relacionados con el objeto de estudio que sirven de base fundamental para el desarrollo del sistema, guiado por la metodología ágil SXP. Se han expuesto los aspectos que tributan a la descripción de la propuesta de la solución, se especifican los principales artefactos generados durante el transcurso de las fases, donde se capturan los requisitos funcionales y no funcionales, se realiza el diseño y se tienen en cuenta las tareas a desarrollar, así como la planificación del tiempo y el esfuerzo para cada una de estas. Todas estas acciones se consideran de gran importancia ya que permiten alcanzar un rápido entendimiento del sistema a implementar.

## 2.1 Descripción de los procesos del negocio.

El proceso de gestión de la información de la dirección de Justicia de la AP de Artemisa, se opera sin la confiabilidad y la organización requerida, ya que está compuesta por varios departamentos, donde cada uno de ellos maneja su información independientemente, imposibilitando el orden, la calidad y la claridad de la información, afectando a su vez la toma de decisiones en tiempo oportuno; provocando así pérdida de la información por falta de eficiencia y control de los datos que se manejan en esta dirección, debido a que el formato en el cual se hace la entrega de la información o se envía no es el idóneo.

El sistema a implementar consiste en la informatización del proceso de gestión de la información del órgano de dirección de Justicia, que cuenta con un número determinado de funcionalidades analizadas previamente en la fase de requisito, tales como insertar, eliminar, modificar y buscar los datos de los modelos de cada

departamento y de esta dirección en general, también podrá generar reportes que se originen a partir de la información de dichos modelos, todas estas funcionalidades permitirán lograr los objetivos propuestos, y a partir de su desarrollo y puesta en marcha. Se propone seguir para un en futuro, perfeccionar la calidad de este sistema, con el propósito de contribuir a la mejora de la confiabilidad y eficiencia de los datos generados u obtenidos de la Dirección de Justicia.

### 2.2 Planificación de proyecto por roles

El desarrollo de *software* con metodologías ágiles exige de la creación de pequeños grupos de trabajo, donde los roles son pocos, pero están bien definidas sus actividades. El principal aspecto antes de comenzar el proceso de documentación es distribuir las tareas por cada uno de los roles existentes, lo que garantiza un trabajo organizado, de ahí la necesidad de tenerlos bien definidos. En la siguiente tabla de muestra la planificación del proyecto por roles.

| Rol | Responsabilidad | Nombre |
|---|---|---|
| Gerente (Management) | Dirigir y controlar las tareas del equipo. Tomar las decisiones finales. Participar en la selección de objetivos y requerimientos. Controlar el progreso y da seguimiento a cada iteración. Evaluar si los objetivos son alcanzables con las restricciones de tiempo y recursos presentes. | Ing. Dania Fernández |
| Cliente (Customer) | Contribuir a definir las historias de usuario y los casos de prueba de aceptación, para validar su implementación.<br>Asignar la prioridad a las historias de usuario y | Administración Provincial de Artemisa |

| | | |
|---|---|---|
| | decidir cuáles se implementan en cada iteración centrándose en aportar mayor valor al negocio. Además de participar en la concepción inicial del sistema y contribuir a definir las tareas que involucra la Lista de reserva del producto además de definir la meta de la iteración y aprobar modificaciones. | |
| Líder de Proyecto (Scrum Master) | Encargado de asegurar que el proyecto se está llevando a cabo de acuerdo con las prácticas y que todo funciona según lo planeado. Remueve los impedimentos que pudiera presentar el proyecto. Define y reduce los riesgos del producto. Además, coordina y facilita las reuniones. | Ing. Leosmel Zayas Castillo |
| Analista (Analyst) | Asignar la prioridad a las historias de usuario y decidir cuáles se implementan en cada iteración centrándose en aportar mayor valor al negocio, todo esto lo realiza junto con el cliente. Escribir la concepción del sistema y las historias de usuario. Crear el Modelo de historia de usuario del negocio y la LRP. | Lilibeth Medina Pérez<br>Lizbety Lache Macías |
| Programadores (Programmers) | Elaborar el código de las nuevas funcionalidades a implementar. Escribe las pruebas unitarias. Debe existir una comunicación y coordinación adecuada entre los programadores y el resto del equipo. | Lilibeth Medina Pérez<br>Lizbety Lache Macías |

| | | |
|---|---|---|
| Diseñadores (Designers) | Encargados del diseño del sistema; así como el de los prototipos de interfaces, máximos responsables de la realización del diseño de las metáforas y supervisan el proceso de construcción. | Lilibeth Medina Pérez<br>Lizbety Lache Macías |
| Arquitecto (Architect) | Se vincula directamente con el analista y el diseñador debido a que su trabajo tiene que ver con la estructura y el diseño en grande del sistema. Ayuda en el diseño de las metáforas. | Lilibeth Medina Pérez<br>Lizbety Lache Macías |
| Encargado de Pruebas (Tester) | Encargado de escribir los casos de prueba de aceptación y ejecutar las pruebas regularmente. Difundir a su vez los resultados en el equipo. Responsable de las herramientas de soporte para pruebas. | Lilibeth Medina Pérez<br>Lizbety Lache Macías |

### 2.3 Modelo de Dominio

Disímiles son las actividades definidas en la metodología SXP, dentro de las más importantes se encuentra la definición de la plantilla del Modelo Historias de Usuario del Negocio, donde se definen las características específicas del negocio, así como la forma en que interactúa el sistema con los clientes y viceversa, es un artefacto que se genera en la fase de Planificación-Definición, en el cual se hace una detallada descripción del negocio en cuestión y se hace mucho más fácil comprender. Cuando el negocio que se está desarrollando no tiene claramente definidas las actividades ni las personas que responden ante estos procesos, se realiza el modelo del dominio que permite mostrar de manera visual los principales conceptos que se manejan, posibilitando a usuarios, clientes y desarrolladores a utilizar un vocabulario común para poder entender el contexto en que se desarrolla

el sistema. Puede ser tomado como el punto de partida para el diseño del mismo. A continuación en la **Figura 3** se muestra el Diagrama del Modelo de Domino:

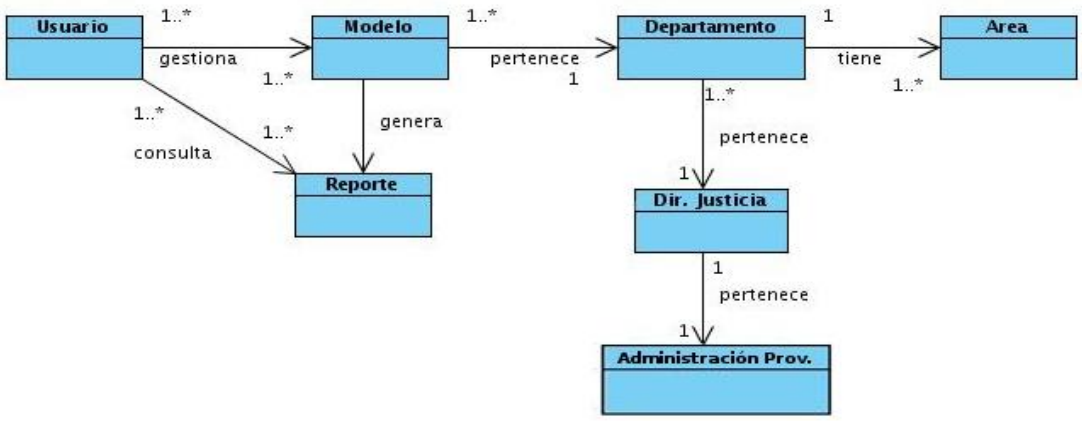



**Figura.3:** Diagrama del Modelo del Dominio

**Conceptos del modelo de dominio:**

**Usuario:** Persona interesada en gestionar la información referente a la dirección de Justicia.

**Modelo:** Plantilla que se llena con la información que se maneja en los departamentos para realizar su trabajo.

**Reporte:** Información en formato duro o digital con el resultado de los datos pedidos por alguna dirección.

**Departamento:** Es la representación de los departamentos que conforman la Dirección de Justicia.

**Área:** Es la representación de las áreas por las cuales están compuestos los departamentos.

**Dir. de Justicia:** Es la representación de la Dirección de Justicia de la Administración Provincial de Artemisa.

**Dir. General:** Es la representación de la Dirección Central, Administración Provincial de Artemisa.

### 2.4 Lista de Reserva del Producto (LRP)

La captura de requisitos es uno de los pasos fundamentales a la hora de desarrollar un sistema. Se debe tener en cuenta que al hacer el levantamiento de los mismos sean los adecuados, ya que serán los que faciliten el futuro progreso del producto.

La Lista de Reserva del Producto (LRP) es uno de los primeros artefactos que se genera en fase de Planificación – Diseño de la metodología SXP, específicamente en la etapa de captura de requisitos.

Este artefacto está conformado por una lista priorizada y organizada de requerimientos técnicos y del negocio a tener en cuenta por el equipo de desarrollo en la elaboración de las historias de usuarios y la implementación, definiendo el trabajo que se va a realizar en el proyecto. La LRP es elaborada en conjunto entre el cliente y el analista, puede crecer y modificarse a medida que se obtienen más conocimientos acerca del producto y del cliente, con la restricción de que sólo puede cambiarse entre iteraciones. Su objetivo fundamental es el de asegurar que el producto definido al terminar la lista es el más correcto, útil y competitivo posible.

A continuación un ejemplo de la Tabla de las LRP:

| **Prioridad** | **Ítem*** | **Descripción** | **Estimación** | **Estimado por** |
|---|---|---|---|---|
| **Muy Alta** | | | | |
| | 1 | Insertar información sobre el Registro general de juristas. | 1 día | Analista |

| | | | | |
|---|---|---|---|---|
| | 2 | Buscar información sobre el Registro general de juristas. | 1 día | Analista |
| | 3 | Modificar información sobre el Registro general de juristas. | 1 día | Analista |
| | 4 | Eliminar información sobre el Registro general de juristas. | 1 día | Analista |
| **Alta** | | | | |
| | 5 | Generar reporte Registro General de Juristas. | 1 día | Analista |
| **Media** | | | | |
| | | | | |
| **Baja** | | | | |
| | | | | |
| **Requisitos No Funcionales** | | | | |
| | 6 | Las PC clientes, deberán estar equipadas con: tarjeta de red, con una capacidad mínima de 256 MB de RAM, un disco duro de 10 GB como mínimo. | | |
| | 7 | El servidor deberá estar equipado con: tarjeta de red, el servidor de base de datos debe tener 1GB de RAM y 60 GB de disco duro como mínimo, el servidor web debe tener 1GB de RAM. | | |
| | 8 | Necesita un servidor de bases de datos que soporte una capacidad alta de datos. | | |

| | | | | |
|---|---|---|---|---|
| | 9 | El servidor donde se encuentre instalada la aplicación debe estar situado en un local protegido, donde no esté expuesto a desastres naturales o robo. | | |

**NOTA:** Las LRP son muy extensas, fue necesario dividirlas, la continuación de esta lista se encuentra en los anexos. (Ver **Anexo 2**)

### 2.5 Historias de Usuarios (HU) y Tareas de Ingeniería.

Las Historias de Usuarios reflejan todas las características del sistema, son escritas por los clientes como las tareas que el sistema debe hacer, de esta forma es muy fácil su compresión y su construcción depende principalmente de la habilidad que tenga el cliente para definirlas. Se emplean para hacer estimaciones de tiempo y para especificar los requisitos del *software*, sin necesidad de documentaciones extensas. El lenguaje de éstas debe ser corto y lo más sencillo posible. Servirán de guía si se definen correctamente, para la posterior construcción de las pruebas de aceptación comprobando de esta manera la correcta implementación de las historias de usuario. A cada HU le corresponde una o varias tareas de ingeniería, estas se recogen en una planilla y poseen gran importancia, pues permiten definir cada una de las actividades que estarán asociadas a las historias de usuario y que posibilitarán su implementación, las tareas se van implementando de acuerdo a su prioridad.

A continuación un ejemplo de la Tabla de las HU y sus Tareas de Ingeniería:

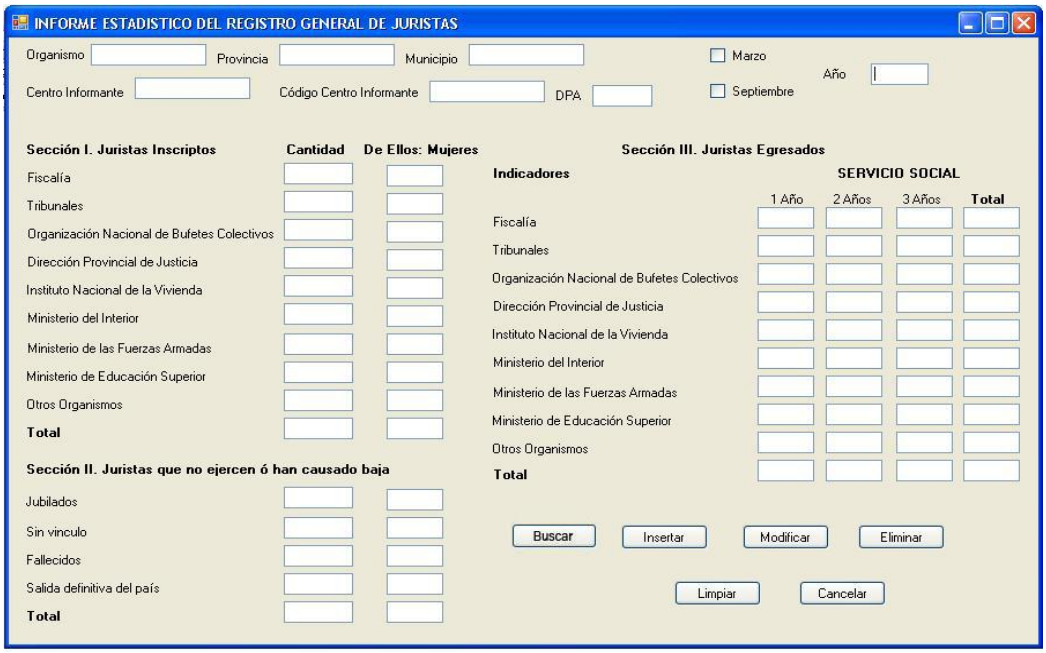
| **Historia de Usuario** | |
|---|---|
| **Número:** HU_1 | **Nombre Historia de Usuario:** Gestionar Información del Registro General de Juristas. |
| **Modificación de Historia de Usuario Número:** Ninguna | |

| | |
|---|---|
| **Usuario:** Lilibeth Medina Pérez<br>Lizbety Laches Macías | **Iteración Asignada: 2** |
| **Prioridad en Negocio:** Muy Alta | **Puntos Estimados: 1/6** |
| **Riesgo en Desarrollo:** Alto | **Puntos Reales: 1/6** |

**Descripción:** La presente historia de usuario tiene como objetivo insertar, modificar, buscar y eliminar toda la información en el Informe Estadístico del Registro General de Juristas en el cual se registra la información referente a los juristas inscriptos, los egresados y los años de servicio social que han cumplido, así como los juristas que no ejercen su profesión o han causado baja, entre otros.

**Observaciones:** Ninguna

**Prototipo de interface:**




**Tarea de Ingeniería**

| | |
|---|---|
| **Número Tarea: 1.1** | **Número Historia de Usuario:** HU_1 |
| **Nombre Tarea:** Investigar sobre cómo implementar insertar, buscar, modificar y eliminar modelos. | |
| **Tipo de Tarea:** Investigación | **Puntos Estimados:** 2/6 |

| | |
|---|---|
| **Fecha Inicio:** 30/01/12 | **Fecha Fin:** 31/01/12 |
| **Programador Responsable:** Lilibeth Medina Pérez y Lizbety Laches Macías | |
| **Descripción:** La presente tarea de ingeniería tiene como objetivo investigar cómo se desarrolla la implementación de cómo insertar, buscar, modificar y eliminar los modelos estudiados. | |
| **Tarea de Ingeniería** | |
| **Número Tarea: 1.2** | **Número Historia de Usuario:** HU_1 |
| **Nombre Tarea:** Insertar la información en el Informe Estadístico del Registro General de Juristas | |
| **Tipo de Tarea:** Desarrollo | **Puntos Estimados: 1/6** |
| **Fecha Inicio:** 01/02/12 | **Fecha Fin:** 01/02/12 |
| **Programador Responsable:** Lilibeth Medina Pérez y Lizbety Laches Macías | |
| **Descripción:** Inserta datos referente a la información del Informe Estadístico del Registro General de Juristas. | |
| **Tarea de Ingeniería** | |
| **Número Tarea: 1.3** | **Número Historia de Usuario:** HU_1 |
| **Nombre Tarea:** Buscar la información en el Informe Estadístico del Registro General de Juristas | |
| **Tipo de Tarea:** Desarrollo | **Puntos Estimados: 1/6** |
| **Fecha Inicio:** 01/02/12 | **Fecha Fin:** 01/02/12 |
| **Programador Responsable:** Lilibeth Medina Pérez y Lizbety Laches Macías | |
| **Descripción:** Buscar datos referente a la información del Informe Estadístico del Registro General de Juristas. | |
| **Tarea de Ingeniería** | |
| **Número Tarea: 1.4** | **Número Historia de Usuario:** HU_1 |
| **Nombre Tarea:** Modificar la información en el Informe Estadístico del Registro | |

| | |
|---|---|
| General de Juristas | |
| **Tipo de Tarea:** Desarrollo | **Puntos Estimados: 1/6** |
| **Fecha Inicio:** 02/02/12 | **Fecha Fin:** 02/02/12 |
| **Programador Responsable:** Lilibeth Medina Pérez y Lizbety Laches Macías | |
| **Descripción:** Modificar datos referente a la información del Informe Estadístico del Registro General de Juristas. | |
| **Tarea de Ingeniería** | |
| **Número Tarea: 1.5** | **Número Historia de Usuario:** HU_1 |
| **Nombre Tarea:** Eliminar la información en el Informe Estadístico del Registro General de Juristas | |
| **Tipo de Tarea:** Desarrollo | **Puntos Estimados: 1/6** |
| **Fecha Inicio:** 02/02/12 | **Fecha Fin:** 02/02/12 |
| **Programador Responsable:** Lilibeth Medina Pérez y Lizbety Laches Macías | |
| **Descripción:** Eliminar datos referente a la información en el Informe Estadístico del Registro General de Juristas. | |
| **Historia de Usuario** | |
| **Número:** HU_28 | **Nombre Historia de Usuario:** Generar reporte Registro General de Juristas. |
| **Modificación de Historia de Usuario Número:** Ninguna | |
| **Usuario:** Lilibeth Medina Pérez<br>Lizbety Laches Macías | **Iteración Asignada: 3** |
| **Prioridad en Negocio:** Alta | **Puntos Estimados: 1/6** |
| **Riesgo en Desarrollo:** Alto | **Puntos Reales: 1/6** |
| **Descripción:** La presente historia de usuario tiene como objetivo generar un reporte con toda la información referente al Registro general de juristas. | |
| **Observaciones:** Para que esto sea posible se debe realizar anteriormente la | |

| | |
|---|---|
| inserción de los datos de al menos un Registro general de juristas. | |
| **Prototipo de interface:** ***Ident al HU_1*** | |
| | **Tarea de Ingeniería** |
| **Número Tarea: 28.1** | **Número Historia de Usuario: HU_28** |
| **Nombre Tarea:** Investigar sobre cómo se puede implementar un generar reportes de modelos. | |
| **Tipo de Tarea:** Investigación | **Puntos Estimados: 1/6** |
| **Fecha Inicio:** 06/04/12 | **Fecha Fin:** 07/04/12 |
| **Programador Responsable:** Lilibeth Medina Pérez y Lizbety Laches Macías | |
| **Descripción:** La presente tarea de ingeniería tiene como objetivo investigar cómo se desarrolla la implementación de cómo generar reportes a partir de información almacenada en modelos. | |

| | |
|---|---|
| | **Tarea de Ingeniería** |
| **Número Tarea: 28.2** | **Número Historia de Usuario: HU_28** |
| **Nombre Tarea:** Generar reporte del modelo Registro General de Juristas. | |
| **Tipo de Tarea:** Desarrollo | **Puntos Estimados: 1/6** |
| **Fecha Inicio:** 09/04/12 | **Fecha Fin:** 09/04/12 |
| **Programador Responsable:** Lilibeth Medina Pérez y Lizbety Laches Macías | |
| **Descripción:** Generar reportes a partir de la información almacenada en modelos del Registro General de Juristas. | |

**NOTA:** Debido a la cantidad de HU, solo se puso un breve ejemplo, las correspondientes a los otros modelos, con sus respectivas tareas de ingeniería se encuentran en los anexos. (Ver **Anexo 3 y 4**)

### 2.6 Plan de *release*

El plan de *release*, divide el proceso de desarrollo de *software* en iteraciones, planificando el trabajo a realizar en cada una de ellas, define además cuales son

las historias de usuario más significativas, y su ubicación en las iteraciones según su prioridad. Seguidamente se muestra la Tabla del Plan de *release*:

| Release | Descripción de la iteración | Orden de la HU a implementar | Duración total |
|---|---|---|---|
| 2 | En esta iteración se implementarán las historias de usuario que tienen prioridad muy alta. | HU_1, HU_2, HU_3, HU_4, HU_5, HU_6, HU_7, HU_8, HU_9, HU_10, HU_11, HU_12, HU_13, HU_14, HU_15, HU_16, HU_17, HU_18, HU_19, HU_20, HU_21, HU_22, HU_23, HU_24, HU_25, HU_26, HU_27, HU_28. | 9 Semanas |
| 3 | En esta iteración se desarrollarán las historias de usuario de prioridad alta y se integrarán con las historias de usuario ya implementadas. | HU_29, HU_30, HU_31, HU_32, HU_33, HU_34, HU_35, HU_36, HU_37, HU_38, HU_39, HU_40, HU_41, HU_42, HU_43, HU_44, HU_45, HU_46, HU_47, HU_48, HU_49, HU_50, HU_51, HU_52, HU_53, HU_54, HU_55, HU_56. | 2 Semanas |
| 4 | En esta iteración se continuará desarrollando las historias de usuario de alta prioridad y se integrarán con todas las anteriores para conformar la aplicación para la Dirección de Justicia. | HU_57, HU_58, HU_59, HU_60, HU_61, HU_62, HU_63, HU_64, HU_65, HU_66, HU_67, HU_68, HU_69, HU_70, HU_71, HU_72, HU_73, HU_74, HU_75, HU_76, HU_77, HU_78, HU_79, HU_80, HU_81, HU_82, HU_83, HU_84. | 2 Semanas |

## 2.7 Arquitectura de *Software*

La Arquitectura en n-Capas se refiere a la distribución de una aplicación desde el punto de vista lógico, es decir, como está estructurado el código. Específicamente consiste en separar la capa de presentación de la lógica de negocio y la de datos**.** Lo que se conoce como arquitectura en capas es en realidad un estilo de programación. La ventaja principal de esta arquitectura es que el desarrollo se puede llevar a cabo en varios niveles y en caso de que ocurra alguna modificación, sólo se atenderá al nivel requerido sin tener que afectar directamente los otros niveles. En la **Figura 4** se muestra como está conformada dicha arquitectura:

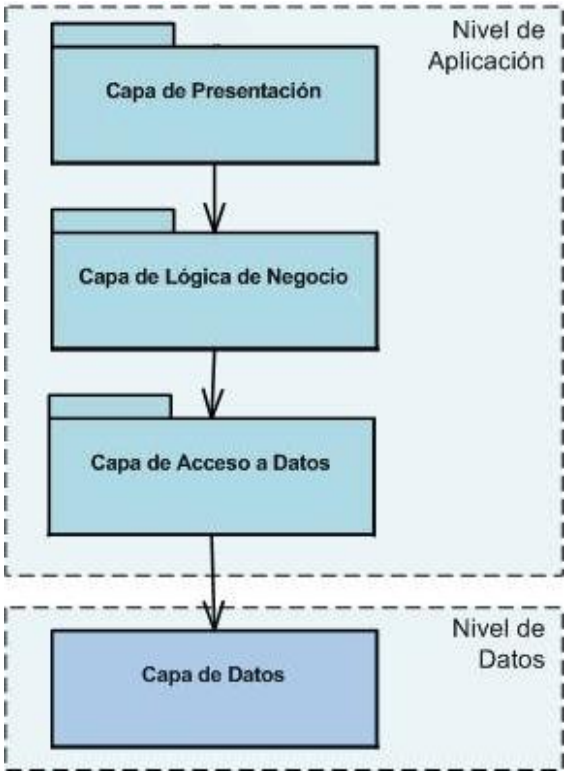



**Figura 4:** Arquitectura en n-Capas

**Capa de presentación:** es la que ve el usuario, también se la denomina "capa de usuario". Presenta el sistema al usuario, le comunica la información y captura la información del usuario en un mínimo de proceso, realiza un filtrado previo para

comprobar que no hay errores de formato. También es conocida como interfaz gráfica y debe tener la característica de ser "amigable" (entendible y fácil de usar) para el usuario. Esta capa se comunica únicamente con la capa de negocio.

**Capa de negocio o Lógica del negocio:** es donde residen los programas que se ejecutan, se reciben las peticiones del usuario y se envían las respuestas tras el proceso. Se denomina capa de negocio e incluso de lógica del negocio porque es aquí donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, y con la capa de datos, para solicitar al gestor de base de datos almacenar o recuperar datos de él. También se consideran aquí los programas de aplicación.

**Capa de acceso a datos:** sirve como puente entre la capa lógica de negocio y el proveedor de datos. Este capa pretende encapsular las especificidades del proveedor de datos tales como (SQL, Oracle, Sybase, archivos XML, texto, hojas electrónicas), a la siguiente capa. Para que si cambia el proveedor de datos solo necesitemos cambiar en una sola capa el proveedor de datos

**Capa de datos:** es donde residen los datos y es la encargada de acceder a los mismos. Está formada por uno o más gestores de bases de datos que realizan todo el almacenamiento de datos, reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio.

En el diseño de sistemas informáticos actual se suelen usar las arquitecturas multinivel o programación por capas. En dichas arquitecturas a cada nivel se le confía una misión simple, lo que permite el diseño de arquitecturas escalables (que pueden ampliarse con facilidad en caso de que las necesidades aumenten).

### 2.8 Diseño con Metáforas

El Diseño con Metáforas es un esbozo inicial del diseño del sistema, que no necesita de especificaciones, ni detalles; se define solamente como una solución simple que pueda funcionar y ser implementada en un momento determinado del proyecto, es generado por el artefacto del Modelo de Diseño y está compuesto por un diagrama de paquetes. El diseño con metáforas es la base para la definición de una futura arquitectura. (Ver **Figura 5**)

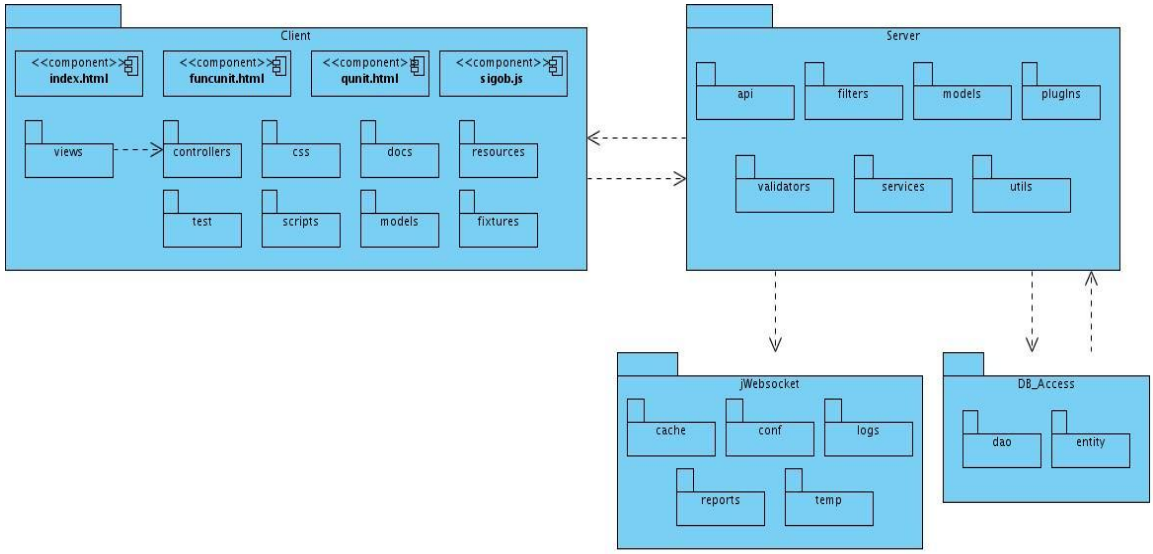



**Figura 5:** Diagrama de Paquetes

## Descripción

El Diagrama de Paquetes anterior se usa para reflejar la organización de los paquetes y sus elementos de la Dirección de Justicia.

En la capa de *Client_Side* contiene paquetes y componentes con los que debe interactuar el cliente. Para tener una mejor organización se han separado los componentes en paquetes diferentes, para facilitar el trabajo a la hora de la implementación. El componente Index.html, es la clase principal que se utiliza en el desarrollo de la interfaz visual del sistema, de la cual dependen las demás vistas.

En la capa *Controller_Server* se representan los componentes que son ubicados en diferentes paquetes, cada uno de ellos tiene funcionalidades específicas en el desarrollo del sistema, en esta capa se encuentran ubicados los *plugins* y dentro de este los eventos para poder dar cumplimiento a cada uno de los requisitos funcionales del sistema. Esta capa depende de la capa *DB_Access* y jwebsocket para poder obtener los datos.

En la capa *DB_Access* se encuentran los componentes entity y dao. En el paquete entity se guardan las clases que son persistentes en el sistema, que tendrán durabilidad y por lo tanto sus datos permanecerán guardados en la base de datos del sistema y en paquete dao se encuentran las archivos que permiten la interacción con la capa *Controller_Server*.

El paquete jwebsocket representa las librerías del *Framework* jwebsocket.

### 2.9 Diagrama de Componentes

Los diagramas de componentes describen los elementos físicos del sistema y sus relaciones, los mismos pueden ser archivos, librerías, módulos, ejecutables, o paquetes y pueden ser usados para modelar y documentar cualquier arquitectura de sistema. Muestran los elementos de diseño del *software* y por otra parte permiten visualizar con más facilidad la estructura general del mismo y el comportamiento del servicio que estos componentes proporcionan y utilizan a través de las interfaces. Estos además tienen la organización y las dependencias lógicas entre un conjunto de componentes *software*. En la **Figura 6** se muestra como quedó conformado el Diagrama de Componentes:

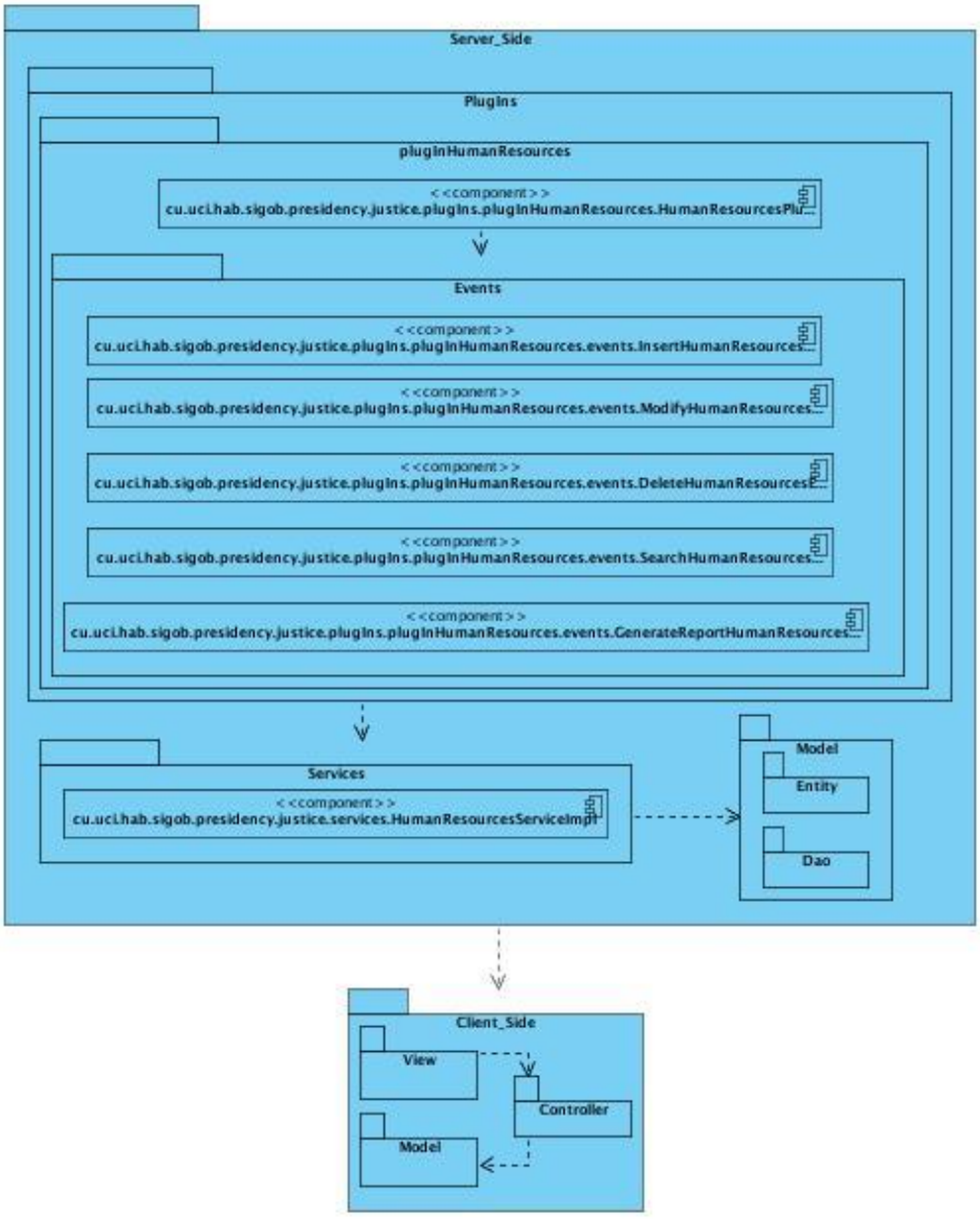



**Figura 6:** Diagrama de Componentes

## Conclusiones parciales

En este capítulo se hizo una propuesta del sistema. Se realizó la planificación del proyecto por roles, para asignar las tareas y garantizar la realización de un trabajo organizado. Se identificaron los requisitos funcionales del sistema cubriendo las

necesidades del cliente, se realizó además un levantamiento de las historias de usuarios a partir de los requisitos identificados y las principales tareas asociadas las misma para su planificación, se precisó la prioridad de cada una de estas historias, puntualizando así el orden en el que serán implementadas las mismas. Se han generado los principales diagramas necesarios parar llevar a cabo un buen desarrollo de *software*.

# Capítulo 3: Adquisición y validación de los resultados del sistema

## Introducción

En este capítulo se exponen los resultados obtenidos al aplicar las pruebas unitarias, con el objetivo de medir la calidad del diseño realizado y de la implementación. A continuación se describen las pruebas que se le realizaron a la aplicación.

## 3.1 Pruebas Unitarias

Dentro de las distintas fases del desarrollo de *software*, además de los requerimientos, diseño, desarrollo, una de las más importantes es la fase de pruebas, en esta se comprueba si el código desarrollado realmente está haciendo lo que se acordó, por otra parte es donde se verifica que el sistema cumpla con los requerimientos que fueron especificados por el cliente en la fase de análisis. Existen varios tipos de pruebas, sin embargo, algunas de las más básicas que existen son las pruebas unitarias o también llamadas pruebas de caja blanca, igualmente conocidas como pruebas modulares que prueban los distintos módulos que conforman el proyecto y permiten determinar si un módulo del programa está listo y correctamente terminado. Además, de que son el inicio para la correcta realización de los demás tipos de pruebas y en este caso son las pruebas realizadas sobre cada clase desarrollada analizando cada uno de los métodos de la misma.

Las pruebas unitarias se plantean a pequeña escala, y consiste en ir probando uno a uno los diferentes módulos que constituyen una aplicación. Una prueba unitaria es una forma de probar el correcto funcionamiento del código. Esto sirve para asegurar que cada uno de los módulos funcione correctamente por separado.

**Características de las Pruebas Unitarias**

Para que una prueba unitaria sea buena se deben cumplir los siguientes requisitos:

- **Automatizable**: no debería requerirse una intervención manual. Esto es especialmente útil para integración continua.
- **Completas**: deben cubrir la mayor cantidad de código.
- **Repetibles o Reutilizables**: no se deben crear pruebas que sólo puedan ser ejecutadas una sola vez. También es útil para integración continua.
- **Independientes**: la ejecución de una prueba no debe afectar a la ejecución de otra.
- **Profesionales**: las pruebas deben ser consideradas igual que el código, con la misma profesionalidad, documentación, etc.

Aunque estos requisitos no tienen que ser cumplidos al pie de la letra, se recomienda seguirlos o de lo contrario las pruebas pierden parte de su función.

Estas pruebas se centran en los detalles procedimentales del *software*, por lo que su diseño está fuertemente ligado al código fuente. Se escogen distintos valores de entrada para examinar cada uno de los posibles flujos de ejecución del programa y cerciorarse de que se devuelven los valores de salida adecuados.

El objetivo de estas pruebas es aislar cada parte del programa y mostrar que las partes individuales son correctas. Proporcionan un contrato escrito que el trozo de código debe satisfacer. Estas pruebas aisladas proporcionan cinco ventajas básicas:

- **Fomentan el cambio:** Las pruebas unitarias facilitan que el programador cambie el código para mejorar su estructura (lo que se ha dado en llamar refactorización), puesto que permiten hacer pruebas sobre los cambios y así

asegurarse de que los nuevos cambios no han introducido errores.

- **Simplifica la integración:** Puesto que permiten llegar a la fase de integración con un grado alto de seguridad de que el código está funcionando correctamente. De esta manera se facilitan las pruebas de integración.

- **Documenta el código:** Las propias pruebas son documentación del código puesto que ahí se puede ver cómo utilizarlo.

- **Separación de la interfaz y la implementación:** Dado que la única interacción entre los casos de prueba y las unidades bajo prueba son las interfaces de estas últimas, se puede cambiar cualquiera de los dos sin afectar al otro, a veces usando objetos mock *(mock object)* para simular el comportamiento de objetos complejos.

- **Los errores están más acotados y son más fáciles de localizar:** dado que se tienen pruebas unitarias que pueden desenmascararlos.

Para las distintas pruebas que se pueden realizar, existen distintas herramientas, una de ellas es JUnit.

JUnit es un conjunto de clases que permite realizar la ejecución de clases Java de manera controlada, para poder evaluar si el funcionamiento de cada uno de los métodos de la clase se comporta como se espera. Es decir, en función de algún valor de entrada se evalúa el valor de retorno esperado; si la clase cumple con la especificación, entonces JUnit devolverá que el método de la clase pasó exitosamente la prueba; en caso de que el valor esperado sea diferente al que regresó el método durante la ejecución, JUnit devolverá un fallo en el método correspondiente.

El IDE de desarrollo Netbeans cuenta con *plugins* que permiten que la generación de las plantillas necesarias para la creación de las pruebas de una clase Java se

realice de manera automática, facilitando al programador enfocarse en la prueba y el resultado esperado, y dejando a la herramienta la creación de las clases que permiten coordinar las pruebas.

Por lo anteriormente descrito y apoyándose en las ventajas que proporcionan las pruebas unitarias, se decidió aplicar estas pruebas al módulo de la dirección de justicia. A continuación se muestra como se realizó este proceso de pruebas.

Se tomó como ejemplo la Clase *CriminalHistoryServiceImpl* (Servicio del Modelo Antecedentes Penales) y se probaron todas sus funcionalidades, obteniéndose en cada una de ellas resultados satisfactorios. Este mismo proceso se le realizó a los restantes servicios logrando los mismos resultados.

```
    /**
     * Test of insertCriminalHistory method, of class CriminalHistoryServiceImpl.
     */
    @Test
    public void testInsertCriminalHistory() throws Exception {

        InsertCriminalHistoryEvent aEvent = new InsertCriminalHistoryEvent();
        aEvent.setDate("2011-02-04");
        aEvent.setExterior(5);
        aEvent.setNational(6);
        aEvent.setIdMonths(4);
        aEvent.setYear(2001);
        aEvent.setIdProvince(1);
        aEvent.setRecords("artemisa");
        CriminalHistoryServiceImpl instance = new CriminalHistoryServiceImpl();
        boolean expResult = true;
        boolean result = instance.insertCriminalHistory(aEvent);
        assertEquals(expResult, result);
        // TODO review the generated test code and remove the default call to fail.
         System.out.println("insertCriminalHistory");
          System.out.println(result);
    }
```

**Figura 7:** Prueba Unitaria realizada al Insertar del Modelo Antecedentes Penales

```
 @Test
public void testDeleteCriminalHistory() throws Exception {

    DeleteCriminalHistoryEvent aEvent = new DeleteCriminalHistoryEvent();

    aEvent.setIdMonths(4);
    aEvent.setYear(2001);

    CriminalHistoryServiceImpl instance = new CriminalHistoryServiceImpl();
    boolean expResult = true;
    boolean result = instance.deleteCriminalHistory(aEvent);
    assertEquals(expResult, result);
    // TODO review the generated test code and remove the default call to fail.
     System.out.println("deleteCriminalHistory");
      System.out.println(result);
}
```

**Figura 8:** Prueba Unitaria realizada al Eliminar del Modelo Antecedentes Penales

```
/**
 * Test of modifyCriminalHistory method, of class CriminalHistoryServiceImpl.
 */
@Test
public void testModifyCriminalHistory() throws Exception {

    ModifyCriminalHistoryEvent aEvent = new ModifyCriminalHistoryEvent();
    aEvent.setIdCriminalHistory(1);
     aEvent.setDate("2011-02-04");
    aEvent.setExterior(5);
    aEvent.setNational(6);
    aEvent.setIdMonths(4);
    aEvent.setYear(2001);
    aEvent.setIdProvince(1);
    aEvent.setRecords("artemisa");
    CriminalHistoryServiceImpl instance = new CriminalHistoryServiceImpl();
    boolean expResult = true;
    boolean result = instance.modifyCriminalHistory(aEvent);
    assertEquals(expResult, result);
    // TODO review the generated test code and remove the default call to fail.
    System.out.println("modifyCriminalHistory");
    System.out.println(result);
}
```

**Figura 9:** Prueba Unitaria realizada al Modificar del Modelo Antecedentes Penales

```
@Test
public void testSearchCriminalHistory() throws Exception {

    SearchCriminalHistoryEvent aEvent = new SearchCriminalHistoryEvent();

    aEvent.setIdMonths(4);

    CriminalHistoryServiceImpl instance = new CriminalHistoryServiceImpl();
    boolean expResult = true;
    boolean result = instance.searchCriminalHistory(aEvent);
    assertEquals(expResult, result);
    // TODO review the generated test code and remove the default call to fail.
     System.out.println("searchCriminalHistory");
      System.out.println(result);
}
```

**Figura 10:** Prueba Unitaria realizada al Buscar del Modelo Antecedentes Penales

```
@Test
public void testGenerateReportCriminalHistory() throws Exception {

    GenerateReportCriminalHistoryEvent aEvent = new GenerateReportCriminalHistoryEvent();

    aEvent.setIdMonths(4);
    aEvent.setYear(2001);

    CriminalHistoryServiceImpl instance = new CriminalHistoryServiceImpl();
    boolean expResult = true;
    boolean result = instance.generateCriminalHistory(aEvent);
    assertEquals(expResult, result);
    // TODO review the generated test code and remove the default call to fail.
     System.out.println("generateCriminalHistory");
      System.out.println(result);
}
```

**Figura 11:** Prueba Unitaria realizada al Generar Reporte del Modelo Antecedentes Penales

Finalmente el resultado que se lanzó luego de probar las cinco funcionalidades de este servicio, fue exitoso, como se muestra en la **Figura 12**.

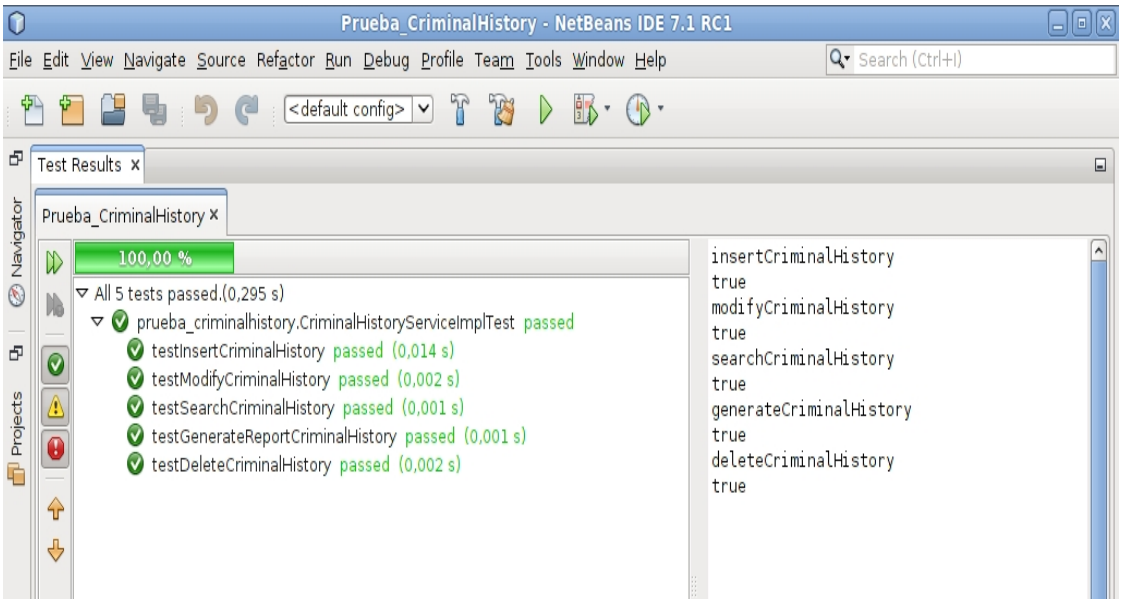



**Figura 12:** Resultados de la prueba unitaria realizada al Servicio Antecedentes Penales (*CriminalHistoryServiceImpl*)

La utilización de las pruebas unitarias permite arreglar los errores detectados sin cambiar la funcionalidad del componente probado, contribuyen al proceso de integración de componentes del sistema al permitir asegurar el funcionamiento correcto de los componentes de manera individual y proporcionan grandes ventajas, permitiendo a los programadores principalmente medir la calidad de su trabajo y garantizar la entrega de un producto con calidad y en correspondencia con las necesidades del cliente.

### 3.2 Resultados Prácticos

Luego de la realización de las pruebas unitarias al Módulo de la Dirección de Justicia de la Administración Provincial de Artemisa y obtenidos resultados satisfactorios, queda disponible el módulo para la gestión de la información de la dirección correspondiente en su versión 1.0 que cumple con las especificaciones pedidas por el cliente.

## Conclusiones Parciales

Una vez terminado el sistema y aplicadas las pruebas necesarias para evaluar la calidad del diseño y de la implementación, se llega a la conclusión de que las pruebas unitarias realizadas permitieron comprobar el correcto funcionamiento de las estructuras lógicas del sistema, con el fin de obtener un producto con la mayor calidad posible.

# Conclusiones

Durante el desarrollo del trabajo de diploma y el tiempo dedicado a la investigación, diseño, implementación y prueba de la solución propuesta, se adquirieron conocimientos necesarios para el desarrollo del sistema y fundamentales para la formación profesional de la autora. Luego de la realización de las actividades anteriores:

- Se establecieron los fundamentos teórico-metodológicos para el desarrollo de los procesos de gestión de información.
- Se caracterizó el proceso de gestión de la información en la dirección de Justicia.
- Se desarrolló el módulo para la dirección de Justicia.
- Se validaron los requisitos propuestos mediante el uso de técnicas para este fin.

# Recomendaciones

Luego de haber concluido el trabajo se recomienda:

- Implementar nuevas funcionalidades para mejorar la calidad y robustez del producto.
- Realizar encuestas periódicas sobre la efectividad del sistema y el grado de satisfacción de los usuarios.

Todo esto previendo un futuro perfeccionamiento del sistema.

# Referencias bibliográficas


1. **Acebey, W. S. Y V..**. Métodos Ágiles Última actualización: Septiembre 22, 2008. (Métodos Ágiles). Disponible en: http://métodos ágiles.html.
2. **Alvero, F. (1976).** Cervantes. Diccionario Manual de la Lengua Española. La Habana, Cuba. Ed. Pueblo y Educación.
3. **Augier Escalona, Alejandro...** Gestión de la Información - Ecured Última actualización: Febrero 13, 2012. (Ecured). Disponible en: http://www.ecured.cu/index.php/gesti%c3%b3n_de_la_informaci%c3%b3n.
4. **Barros, Oscar (1994)** - "Reingeniería de Procesos de Negocio", Editorial Dolmen, Chile, 1994.
5. **Castro Morell, Msc. Daniel E.** 2008. Sistema para la Tramitación de. Villa Clara: S.N., 2008.
6. **Contreras, Luis Carlos.** Metodología RUP Última actualización: Marzo 2006. (Scribd). Disponible en: http://es.scribd.com/doc/64933584/62/metodologia-rup.
7. **Dangel, A. D..**. Sistemas de Información Última actualización: 2009. (econlink.com.ar). Disponible en: http://www.econlink.com.ar/sistemas-informacion/definicion.
8. **De Heredia, R. (1995).** Dirección Integrada de Proyectos. 2da Edición. Madrid, España. Ed. Servicio de Publicaciones de la E.T.S. de Ingenieros Industriales de la UPM.
9. **Enjoiras, M.** Beneficios del uso de Java en las aplicaciones modernas de bibliotecas Última actualización: 1998. (FESABID 98). Disponible en: http://www.ciepi.org/fesabid98/comunicaciones/m_enjolras.htm.
10. **Gutierres, Javier...** ¿Qué es un Framework Web? (lsi.us.) Disponible en: http://www.lsi.us.es/~javierj/investigacion_ficheros/framework.pdf.
11. **Lastre Reynaldo, Reyvis.E.** Análisis y Diseño del Módulo de

Reportes para el portal de la UJC en la UCI. Ing. Susana Reina Consuegra. Universidad de las Ciencias en Informática de la Ciudad Habana, 2008.

12. **Manios, A...** Metodología para la Construcción de Software Última actualización: 2009 (slideshare). Disponible en: http://www.slideshare.net/andresmanios/metodologa-para-la-construccin-de-software.
13. **Menéndez, R.**.. Apuntes Informática Aplicada a la Gestión Pública. Capítulo 2, Ingeniería del Software, Metodologías de Desarrollo. 2011/12. Universidad de Murcia (España). Rafael Barzanallana Última actualización: Octubre 13, 2011. (Universidad de Murcia). Disponible en: http://www.um.es/docencia/barzana/iagp/iagp2-metodologias-de-desarrollo.html.
14. **Navarro, J...** Netbeans frente a Eclipse - Más que código, Última actualización: Diciembre 9, 2003. (NetBeans frente a Eclipse). Disponible en: http://www.juanjonavarro.com/masquecodigo/2003/12/09/Netbeans-frente-a-Eclipse.
15. **Pressman, Roger S.** Ingeniería de Software. Un Enfoque Práctico 6ta Edición. 2000. Entorno Virtual de Aprendizaje. Disponible en: http://eva.uci.cu.
16. **Rodríguez, J. M. De B..**. Atlante. Sistema de Gestión Procesal de Justicia del Gobierno de Canarias. Última actualización: 2001. (Artículos Técnicos). Disponible en: http://historico.pnj.cgpj.es/numero7/articulos9.html.
17. **Rumbaugh, James, Jacobson, Ivar Y Booch, Grady.** El Lenguaje Unificado de Modelado. Entorno Virtual de Aprendizaje. Disponible en: http://eva.uci.cu.

# Bibliografía


1. **Acebey, W. S. Y V...** Métodos Ágiles Última actualización: Septiembre 22, 2008. (Métodos Ágiles). Disponible en: http://métodos ágiles.html.
2. **Berberena, I. R. P...** Los Sistemas Integrados de Gestión de la Información Gestiopolis Última actualización: Julio 2, 2010. (GestioPolis.com). Disponible en: http://www.gestiopolis.com/administracion-estrategia-2/sistemas-integrados-gestion-informacion.htm.
3. **Heri Felix Castro.** Sistemas de Información Última actualización: 2012. (Scribd). Disponible en: http://es.scribd.com/doc/34332883/sistemas-de-informacion.
4. **Letelier Torres, Patricio Y Penadés, Ma Carmen.** Metodologías Ágiles en el Desarrollo De Software, Alicante – España: Noviembre 12, 2003. **E.V.A. UCI, I. D. S.** Conferencia #1. Introducción a la Ingeniería de Software, ISW 1.
5. **Peñalver Romero, G.M.**, "Trabajo de Diploma: Metodología Ágil para Proyectos de Software Libre". Ciudad de la Habana, Universidad de las Ciencias Informáticas. 2008.
6. **Acerca de los websockets« jWebSocket: »A dream is coming true…**" Última actualización: Septiembre 13, 2011. (jWebSocket: «A dream is coming true…»). Disponible en: http://softwarelibre.hab.uci.cu/jwsblog/?p=207.
7. **Concepto de gestión** - Definición, Significado y Qué es, Última actualización: 2012. (Definición. De). Disponible en: http://definicion.de/gestion/.
8. **Concepto de información** - Definición, Significado y Qué es, Última actualización: 2012. (Definición. De). Disponible en: http://definicion.de/informacion/.
9. **Definición de sistema** - Qué es, Significado y Concepto Última actualización: 2012. (Definición. De). Disponible en: http://definicion.de/sistema/

10. **Eclipse, entorno de desarrollo integrado** - Ecured Última actualización: 2012. (Ecured). Disponible en: http://www.ecured.cu/index.php/eclipse,_entorno_de_desarrollo_integrado.
11. **El Sistema de Gestión Procesal Atlante II supera el Test De Compatibilidad del Consejo General del Poder Judicial** Última actualización: 2012. (europapress.es). Disponible en: http://www.europapress.es/islas-canarias/noticia-sistema-gestion-procesal-atlante-ii-supera-test-compatibilidad-consejo-general-poder-judicial-20110608141714.html.
12. **Free download manager** Visual Paradigm For UML (ME) - (Paradigma Visual para UML (ME)) (Visual Paradigm For UML (ME)) por Visual Paradigm International Ltd. - Reporte y Descarga Última actualización: Marzo 5, 2007. (Free Download Manager). Disponible en: http://www.freedownloadmanager.org/es/downloads/paradigma_visual_para_uml_(m%c3%8d)_14720_p/.
13. **Jwebsocket.** Google Traductor Última actualización: Marzo 2010. (jWebSocket, El Websocket de alta velocidad de comunicación de soluciones). Disponible en: http://translate.google.com.cu/translate?
14. **Metodologías de desarrollo de software** - Ecured (Ecured). Disponible en:http://www.ecured.cu/index.php/metodolog%c3%adas_de_desarrollo_de_software.
15. **Metodologia Agil de Desarrollo SXP** - Ecured Última actualización: 2012. (Ecured). Disponible en: http://www.ecured.cu/index.php/metodologia_agil_de_desarrollo_sxp.
16. **Netbeans IDE - features** Última actualización: 2012. (NetBeans). Disponible en: http://netbeans.org/features/index.html.
17. **¿Qué es Jwebsocket?** Última actualización: Septiembre 9, 2011. (jwebsocket: «A dream is coming true…»" next generation of rt Applications…). Disponible en: http://softwarelibre.hab.uci.cu/jwsblog/?paged=8.

18. **Springhispano.org** | Spring Framework para la Comunidad Hispanoamericana. Última actualización: Septiembre 30, 2010 Disponible en: http://springhispano.org/?q=node/35.
19. **Sistema de gestión procesal Minerva** Última actualización: 2010. Disponible en: http://oficinajudicial.justicia.es/portaloj/sistema_minerva.
20. **Sistema Informático para la Gestión del Proceso de Diligencias Previas en los Tribunales Provinciales Cubanos.** 2010. Universidad de las Ciencias Informáticas. Disponible en: http://bibliteca.uci.cu
21. **Tutorial Maven**. Enero 15, 2009 Disponible en: http://viviendoconjavaynomoririntentandolo.blogspot.com/2009/01/tutorial-maven.html

# Glosario de Términos

**A**

**AP:** Administración Provincial de Artemisa.

**C**

**Código abierto:** en inglés *Open Source*, es el término con el que se conoce al *software* distribuido y desarrollado libremente. El código abierto tiene un punto de vista más orientado a los beneficios prácticos de compartir el código.

**CASE:** herramienta de Ingeniería del *Software* asistida por computadoras, una herramienta CASE es un conjunto de aplicaciones informáticas que tienen el objetivo de aumentar la productividad en el desarrollo de un *software* mitigando los costes en términos de tiempo.

**CVS:** Sistema de control de versiones.

**E**

**Entorno de Desarrollo Integrado** llamado también **IDE** (sigla en inglés de *Integrated Development Environment*), es un programa informático compuesto por un conjunto de herramientas de programación. Puede dedicarse en exclusiva a un solo lenguaje de programación o bien poder utilizarse para varios.

**F**

***Framework*:** en español marco de trabajo, se refiriere a una estructura *software* compuesta de componentes personalizables e intercambiables para el desarrollo de una aplicación.

**P**

***PlugIn*:** es un módulo de *hardware* o *software* que añade una característica o un servicio específico a un sistema más grande.

**POM:** responde a las siglas de Project Object Model, es un fichero XML, que es la unidad principal de un proyecto Maven. Contiene información acerca del proyecto, fuentes, test, dependencias, plugins, versión.

**R**

**RUP**: en inglés significa *Rational Unified Process*). El Proceso Unificado de Rational es un producto del proceso de ingeniería de *software* que proporciona un enfoque disciplinado para asignar tareas y responsabilidades dentro de una organización del desarrollo.

**S**

**SXP:** es una metodología ágil, compuesta por las metodologías Scrum y XP ofrece una estrategia tecnológica, a partir de la introducción de procedimientos ágiles que permitan actualizar los procesos de *software* para el mejoramiento de la actividad productiva fomentando el desarrollo de la creatividad, aumentando el nivel de preocupación y responsabilidad de los miembros del equipo, ayudando al líder del proyecto a tener un mejor control del mismo.

**Scrum:** es un proceso en el que se aplican de manera regular un conjunto de mejores prácticas para trabajar en equipo y obtener el mejor resultado posible de un proyecto.

**T**

**Tokens:** en programación, un elemento individual en un lenguaje de programación. Es un bloque de texto categorizado.

**TIC:** Tecnologías de la Información y las Comunicaciones.

**U**

**UML** es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de *software*. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocios y funciones del sistema, y aspectos concretos como expresiones de lenguajes de

programación, esquemas de bases de datos y componentes de *software* reutilizables.

**W**

**Websockets:** es una tecnología que proporciona un canal de comunicación bidireccional y full-dúplex sobre un único socket TCP. Está diseñada para ser implementada en navegadores y servidores web, pero puede utilizarse por cualquier aplicación cliente/servidor.

**X**

**XP:** del inglés *Extreme Programing*, Programación Extrema es una metodología ágil para el desarrollo de *software* y consiste básicamente en ajustarse estrictamente a una serie de reglas que se centran en las necesidades del cliente para lograr un producto de buena calidad en poco tiempo.

**XML:** son las siglas de *Extensible Markup Language,* una especificación/lenguaje de programación desarrollada por el W3C. XML es una versión de SGML, diseñado especialmente para los documentos de la web.

# Anexos

**Anexo 1: Encuesta realizada en la Dirección de Justicia de la Administración Provincial de Artemisa.**

- ¿Cuáles son las funciones y misiones de la Dirección?
- ¿De qué forma se obtienen habitualmente la información?
- ¿Con qué frecuencia se obtiene la información?
- ¿Cómo son procesados los datos?
- ¿Qué acciones se realizan con las informaciones una vez obtenidas y registradas?
- ¿Cuáles son las informaciones específicas que se manejan en la Dirección?
- ¿Cuáles son las capacidades de la infraestructura tecnológica existente en la Dirección?
- ¿Cómo le gustaría que fueran presentados los datos en la aplicación, en tablas, gráficos u otros?