# DISEÑO E IMPLEMENTACIÓN DE UNA APLICACIÓN WEB PARA LA ADMINISTRACIÓN Y AUTOMATIZACIÓN DE PROCESOS PARA LA PROTECCIÓN CONTRA INSTALACIÓN Y USO NO LICENCIADO DE APLICACIONES SOFTWARE. TRIONIX 1.1

**PEDRO ERLENDIS GONZÁLEZ PEÑARETE**
**DARWIN WALTER REY CALA**

**UNIUVERSIDAD INDUSTRIAL DE SANTANDER**
**FACULTAD DE CIENCIAS FISICO-MECANICAS**
**ESCUELA DE INGENIERIA DE SISTEMAS E INFORMATICA**
**2006**

# DISEÑO E IMPLEMENTACIÓN DE UNA APLICACIÓN WEB PARA LA ADMINISTRACIÓN Y AUTOMATIZACIÓN DE PROCESOS PARA LA PROTECCIÓN CONTRA INSTALACIÓN Y USO NO LICENCIADO DE APLICACIONES SOFTWARE. TRIONIX 1.1

**Trabajo de grado para optar**
**el título de ingenieros de sistemas**


**DIRECTOR**
**LUIS IGNACIO GONZALEZ RAMIRES**
Magíster en informática, UIS

**CODIRECTOR**
**JUAN GABRIEL QUINTERO PEÑA**
Ingeniero de Sistemas, UIS



**UNIVERSIDAD INDUSTRIAL DE SANTANDER**
**FACULTAD DE CIENCIAS FISICO-MECANICAS**
**ESCUELA DE INGENIERIA DE SISTEMAS E INFORMATICA**
**2006**

## CONTENIDO

## LISTA DE ANEXOS

## LISTA DE TABLAS

## LISTA DE FIGURAS

## GLOSARIO

**Administrador**: Persona encargada de llevar a cabo el proceso de protección y dar un correcto uso de las funciones e información contenida en la aplicación.

**Casos de uso**: Conjunto de acciones que lleva a cabo el sistema con el fin de proporcionar al usuario un resultado importante y representa un requisito funcional.

**Centrado en la arquitectura:** la arquitectura incluye los aspectos mas significativos del sistemas: su estructura, comportamiento, restricciones, plataforma en la que debe funcionar el software, sistemas heredados, reutilización de componentes y requisitos no funcionales.

**Cliente**: persona o entidad que usa las aplicaciones que han sido protegidas con la aplicación Trionix.

**Código ASCII**: Acrєnimo de American Standard Code for Information Interchange (Codigo Normalizado Americano para el Intercambio de Informacієn). En computación, un esquema de codificación que asigna valores numericos a las letras, números, signos de puntuación y algunos otros caracteres.

**Componente:** partes físicas del sistema. Pueden ser reemplazadas.

**Contraseña**: Información confidencial, frecuentemente constitutita por una cadena de caracteres, que puede ser usada en la autenticación de un usuario.

**Desarrollador**: persona con derechos patrimoniales sobre la aplicación que va a ser protegida por Trionix.

**Hexadecimal**: sistema de numeración en base 16. Se emplean las cifras del 0 al 9 y las letras AaF (A=10, B=11, C=12, D=13, E=14, F=15).

**Hash:** función o método para generar claves o llaves que representen de manera unívoca a un documento, registro, archivo.

**Hacker**: experto informático especialista en entrar en sistemas ajenos sin permiso, generalmente para mostrar la baja seguridad de los mismos o simplemente para demostrar que es capaz de hacerlo.

**Interfaz**: sistema de comunicación de un programa con su usuario; la interfaz comprende las pantallas y los elementos que informan al usuario sobre lo que puede hacer, o sobre lo que está ocurriendo.

**Login** : Nombre con el que una persona está registrada en un determinado computador o aplicación.

**Mainframes**: Así se les llama a las grandes computadoras, capaces de atender a miles de usuarios y miles de programas "al mismo tiempo" asignándole un periodo muy pequeño a la atención de cada programa. Su capacidad de trabajo es muy alta, por lo que normalmente se encuentran en empresas de gran tamaño. Sus programas están compuestos por cientos de miles o millones de líneas de código.

**MD5:** Algoritmo de Resumen del Mensaje 5, es un algoritmo de reducción criptográfico de 128 bits ampliamente usado a nivel mundial.

**Modalidad**: forma bajo la cual se desarrolla o implementa un proyecto que va a producir como resultado un producto.

**Protección**: proceso mediante el cual se garantiza que una aplicación funciona bajo unas condiciones establecidas.

**Renovación**: Proceso mediante el cual una aplicación protegida, cuya protección ha espirado puede volver a entrar en correcto funcionamiento.

**Servidor Web**: Máquina conectada 24 horas al día y 365 días al año a Internet y que almacena los documentos que componen uno o varios sitios Web. Cuando un usuario hace clic sobre un enlace (link) a una página Web, se envía una solicitud al servidor Web para localizar los datos nombrados por ese enlace. El servidor Web recibe esta solicitud y suministra los datos que le han sido solicitados (una página HTML, un script interactivo, una página Web generada dinámicamente desde una base de datos) o bien devuelve un mensaje de error.

**Usuario**: Persona a quien inicialmente se le protege una aplicación, esta persona debe contar con los derechos del autor para la manipulación del software en el caso de que el no sea el autor de la aplicación.

**Visitante Web**: persona que oportunamente se ha encontrado con la aplicación navegando por la Web.

**TÍTULO:** **DISEÑO E IMPLEMENTACIÓN DE UNA APLICACIÓN WEB PARA LA ADMINISTRACIÓN Y AUTOMATIZACIÓN DE PROCESOS PARA LA PROTECCIÓN CONTRA INSTALACIÓN Y USO NO LICENCIADO DE APLICACIONES SOFTWARE, TRIONIX 1.1**[*]

**AUTORES:** **PEDRO ERLENDIS GONZÁLEZ PEÑARETE**
DARWIN WALTER REY CALA[**]

**PALABRAS CLAVES:**

Criptografía, DES, RSA, usuario, cliente, protección de aplicaciones.

**DESCRIPCIÓN:**

**TRIONIX 1.1** es un trabajo de grado diseñado con el fin de proteger archivos ejecutables (.exe Win 32 bits) contra la instalación y uso no licenciado aplicando técnicas de criptografía. Busca cubrir una necesidad de la Escuela de Ingeniería de Sistemas: contar con una alternativa que le garantice el control de licencias de aplicaciones software que se desarrollen allí.

Para ello, se implementan dos módulos: un módulo de protección, el cual protege aplicaciones para atender la demanda de usuarios, y el módulo web para la administración de clientes y aplicaciones por parte de sus responsables. El módulo de protección utiliza el algoritmo **DES** (Data Encryption STANDARD) para encriptar los archivos y el algoritmo **RSA** para controlar las licencias de uso. En la web, cada responsable de una aplicación cuenta con un espacio y opciones que le permiten controlar y divulgar sus productos de una manera segura al conocer información de quienes los usan y los permisos otorgados, de acuerdo a sus criterios, para esto se manejó un administrador de perfiles de usuario que permite establecer las opciones y funciones del sistema.

Las aplicaciones protegidas son revisadas y controladas por medio de accesos y tiempos, lo cual permite detectar posibles abusos de las condiciones pactadas, por ejemplo, utilizar la aplicación por más tiempo o realizar una distribución ilegal de la misma. En cualquiera de estos casos, el producto se bloqueará y automáticamente pedirá una nueva clave de uso, que será validada y otorgada por medio de la aplicación web.

---

[*] Trabajo de Investigación

[**] Facultad de Ingenierías Físico Mecánicas, Ingeniería de Sistemas
Director: MSc. Luis Ignacio González Ramírez

**TITLE:** **DESIGN AND IMPLEMENTATION OF A WEB APPLICATION FOR THE ADMINISTRATION AND AUTOMATIZATION OF PROCESES, FOR THE PROTECTION AGAINST INSTALLATION AND NON-LICENSED USE OF SOFTWARE APPLICATIONS, TRIONIX 1.1**[*]

**AUTHORS:** **PEDRO ERLENDIS GONZÁLEZ PEÑARETE**
DARWIN WALTER REY CALA[**]

**KEY WORDS:**

**Cryptography, DES, RSA, user, client, application protection.**

**DESCRIPTION:**

TRIONIX 1.1 is a work to obtain a university degree, it is designed to protect executable files (.exe Win 32 bits), against installation and non-licensed use applying cryptographic techniques. It expects to cover a System Engineering School need: an option to guarantee the control of software application licenses which are developed there.

We might as well to implement two modules: a protection module which protects applications to attend the user demand, and the web module for the client and application administration, by its responsibles. The protection module uses the DES (Data Encryption Standard) algorithm to encrypt, and the RSA algorithm to control use licenses. On the web each responsible for an application has an space and options to control and to spread his products in a safe way because he can get the information about people who use his products and the given permissions, to bearing his criteria in mind, to do this we used a user profile administrator which permits to select the options and the system functions.

The protected applications are checked and controlled by means of accesses and time, it lets to detect some abuses of the established conditions, for example, to use the application more time or to distribute the application ilegally.

In any case, the product will be blocked and it will inmediately ask for a use key, this use key will be validated and will be given by means of the web application.

---

[*] Researching Work

[**] Physical Mechanical Engineering Faculty, System Engineering
Director: MSc. Luis Ignacio González Ramírez

# INTRODUCCIÓN

Un problema que se presenta en la actualidad al desarrollador software, es la distribución de productos en un medio inseguro donde prevalece la cultura de piratería, que margina al desarrollador de los beneficios por su labor desmotivándolo por no desarrollar.

Es por ello que se debe contar con mecanismos de control y divulgación confiables y dinámicos que suplan esta necesidad con un eficiente manejo de la información, en donde la forma de acceder a está, su seguridad, confiabilidad y facilidad juegan un papel importante. Por tal razón surge TRIONIX 1.1, que brinda la posibilidad de distribuir y limitar el uso de un producto, para que solo personal autorizado pueda acceder a él, en medio de un canal masivo de información que permita la disminución de tiempo, como lo es Internet.

TRIONIX 1.1 ha logrado combinar el uso de la Web con una alternativa de protección de aplicaciones, para que el desarrollador de la pequeña y mediana empresa pueda proteger, distribuir autónomamente sus productos y tener un control sobre ellos, para de esta manera obtener un reconocimiento por su trabajo.

El presente documento se clasifica por partes y capítulos. Cada parte esta constituida por capítulos relacionados. La Parte 1 muestra los aspectos teóricos básicos, fundamento de este trabajo como: en el capítulo 1 plantea los objetivos, definición de la problemática planteada, justificación e impacto en la comunidad universitaria. El capítulo 2 presenta la información teórica. En el capítulo 3 se presenta una breve reseña del proceso unificado de desarrollo, el cual es la metodología que se uso.

La parte II describe en detalle cada una de las fases que componen en el proceso unificado de desarrollo: Inicio, elaboración y construcción que analizan los requisitos funcionales y no funcionales del sistema, plasmándolos finalmente en un producto software.

La parte III capitulo 7, se tratan aspectos relevantes de la aplicación desarrollada, describiendo paso a paso los procesos que sigue cada módulo y las opciones para los usuarios, en el capítulo 8 se documentan las pruebas realizadas a la aplicación.

Finalmente se formulan unas recomendaciones con el fin de que este trabajo evolucione y no interrumpa su ciclo de vida; además se realizan conclusiones y anexa documentación que amplia conceptos tratados para la ejecución del proyecto.

**PARTE I**

## FUNDAMENTOS

En esta parte se muestran los aspectos teóricos y técnicos más relevantes; no es el fin mostrar la información en su totalidad, se mostrara a grandes rasgos la información básica y se hará énfasis en los aspectos mas relevantes con el fin de que se entienda a cabalidad el trabajo realizado.

Los objetivos específicos de esta parte son:

- Presentar los objetivos planteados en el proyecto.
- Justificar la elaboración del proyecto.
- Formular el marco teórico.
- Definir el marco metodológico, en donde se muestra la metodología utilizada para el desarrollo del trabajo realizado.

## CAPITULO 1

## AMBITO DEL PROYECTO

### INTRODUCCIÓN

En la EISI[1] se desarrolló a nivel de proyecto de pregrado un producto para proteger aplicaciones .EXE[2] WIN 32[3] llamado: TRIONIX[4], el cual pretendió soportar el desarrollo de software para la pequeña y mediana empresa, y de cierta manera, combatir la cultura de la piratería en nuestro país.

El ***Grupo Software TRIONIX***[5], desarrollador del producto tiene como filosofía la evolución del mismo proporcionando un servicio y ofreciendo más ventajas y garantías para sus usuarios.

A continuación se describe los puntos que se desean mejorar, identificando causas y consecuencias:

- ✓ **ADMINISTRACIÓN:**

La herramienta para la protección TRIONIX[6], maneja la administración de perfiles y los clasifica en: Clientes, usuarios, desarrolladores y administrador, contando cada uno de ellos con opciones disponibles de acuerdo a su perfil.

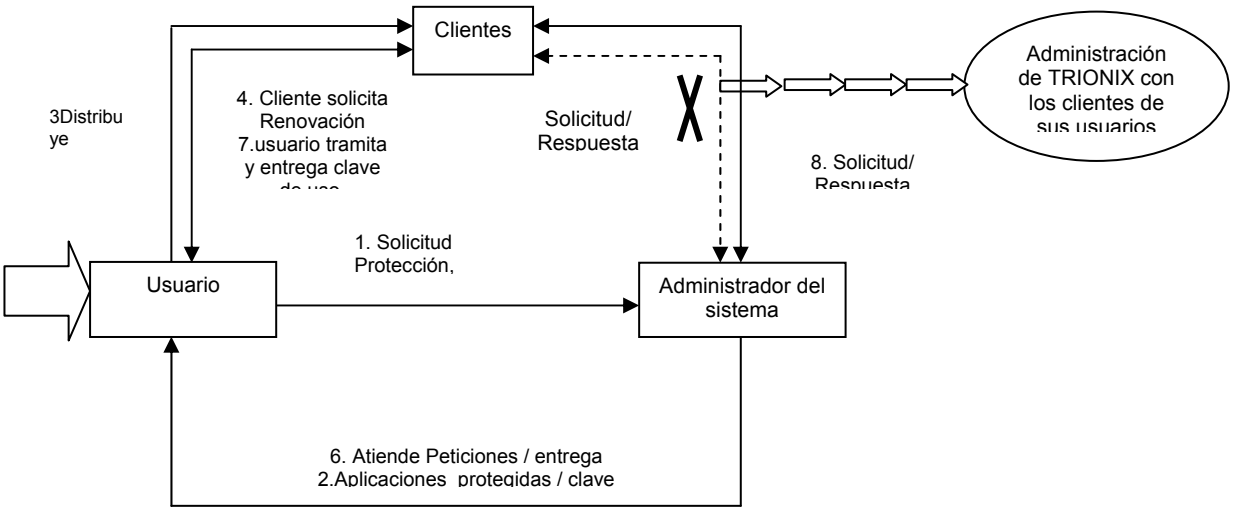



**Figura 1.1** Descripción del problema

[1] (Escuela de Ingeniería de Sistemas e informática) Universidad Industrial de Santander
[2] Extensión de archivos ejecutables; este tipo de archivos puede ejecutarse en un sistema operativo como DOS o Windows.
[3] Es un conjunto de API (Interfaz de Programación de aplicaciones) de 32 bits.
[4] Ver articulo (anexo 8)
[5] Grupo desarrollador de TRIONIX y proponente de la presente propuesta.
[6] Herramienta para la protección contra instalación y uso no licenciado de aplicaciones software…. ver articulo LA PIRATERÍA Y LA PROTECCIÓN DE LA INFORMACIÓN (anexo 1)

Para la clara comprensión de la problemática es necesario definir claramente cada uno de los perfiles anteriormente mencionados:

- Cliente: se llama al sujeto o actor que mediante el uso de aplicaciones protegidas por TRIONIX da respuesta a una necesidad. Es la persona que adquiere la licencia de uso del software con previo consentimiento del dueño de la aplicación (usuario).

- Usuario: es el sujeto que solicita protección de una aplicación de su dominio, o que cuenta con autorización expresa del propietario patrimonial de la misma, para su distribución y uso por parte de los clientes con las restricciones pactadas y controladas por TRIONIX.

- Desarrollador: persona(s) que desarrolla (ron) la(s) aplicación(es).

- Administrador: es quien se encarga de administrar el sistema y disponer de todas las opciones que ofrece la herramienta: protección, renovación y consulta.

El modulo de protección puede ser accedido sólo por el administrador en atención a la demanda de aplicaciones a proteger por parte de los usuarios; dicha protección presenta unos limites de tiempo y accesos, períodos durante los cuales puede ser usada por los usuarios, al finalizar estos períodos se genera una clave pública y solicita un serial, lo que obliga a quien la usa a ponerse en contacto con el responsable de la aplicación y este a su vez con el administrador de la aplicación “TRIONIX” para obtener en definitiva la nueva clave de uso. Este proceso permite que los usuarios administren los clientes.

Es claro, que por ser el cliente quien inicialmente conoce la clave generada por la aplicación, puede ocurrir que éste se remita directamente al administrador, marginando al usuario (dueño del software) de la posibilidad de pactar nuevos términos de licencia de uso. De esta forma, el administrador estaría tomando atribuciones que no le competen, como renovar y otorgar licencias, dado que no existe un mecanismo efectivo de control de autenticación diferente a la buena fe del administrador y del cliente.

Otro problema que se presenta, es la dependencia de un administrador para acceder a las opciones que ofrece el sistema. Para cualquier opción que se desee acceder, se debe seguir un proceso que consiste en: solicitar, esperar respuesta, identificarse e ingresar finalmente al sistema. Este proceso puede tener una duración significativa, y en ocasiones la solicitud puede no tener respuesta debido a la disponibilidad de tiempo del administrador para realizar estas funciones.

Buscando solucionar lo anterior se propone la implementación de una aplicación Web que permita la administración de clientes por parte de los usuarios TRIONIX, además de la automatización de los procesos de renovación y consulta.

- ✓ **BASE DE DATOS**

Dado que en el alcance de la versión original de la base de datos “TRIONIX”, no se contemplaba su acceso a través de la Web, donde la demanda de usuarios es significativa, se hace necesario un nuevo diseño e implementación de la base de datos en busca de cubrir estas necesidades.

La base de datos es un punto relevante en cuanto a la seguridad se refiere, pues en ella es donde se deposita toda la información que la herramienta necesita para su funcionamiento. Una contraseña de administración no sería suficiente pues cualquier persona no autorizada aplicando algún método de descifrado, podría conocer, ingresar y vulnerar la información que se dispone en ella.

Lo descrito anteriormente se soluciona con mecanismos de control como funciones criptográficas HASH[7] que permiten la protección de la información más relevante de la aplicación (contraseñas de usuario y claves de aplicaciones).

- ✓ **TIEMPO DE EJECUCIÓN DE LA PROTECCIÓN**

El algoritmo de protección que se ha venido trabajando hasta el momento es el DES[8] por su seguridad y eficacia en la protección de archivos. Se plantea el estudio de diferentes algoritmos criptográficos buscando seleccionar la mejor alternativa.

[7] funciones criptográficas para proteger información, para más información ver apartado 2.3.1.

[8] (Data Encryption Standard) ver Articulo (anexo 1).

## 1.1 DESCRIPCIÓN DEL PROYECTO

### 1.1.1 OBJETIVO GENERAL

- Diseñar e Implementar una aplicación Web para la administración y automatización de los procesos para la protección contra la instalación y uso no licenciado de aplicaciones software.

### 1.1.2 OBJETIVO ESPECÍFICOS

- Implementar una aplicación Web que soporte la administración de productos software protegidos y regule su acceso y funciones mediante el manejo de perfiles.

- Investigar diferentes tipos de algoritmos criptográficos, comparando fiabilidad y rendimiento frente al algoritmo DES, para de acuerdo a los resultados seleccionar la mejor opción.

- Garantizar la autenticidad e identificación de usuarios al momento del almacenamiento y verificación de contraseñas y claves de aplicaciones en la base de datos, usando funciones criptográficas HASH.

- Diseñar e implementar una base de datos que soporte el almacenamiento y organización de la información referente a la aplicación software protegida y usuarios.

- Implementar el modulo de renovación que permita la actualización de uso de aplicaciones software protegidas por accesos o tiempo determinados al usuario de la aplicación por medio de la Web.

- Diseñar e implementar un módulo Web para la divulgación de aplicaciones protegidas por TRIONIX, que puedan ser descargadas.

## 1.2 JUSTIFICACIÓN

La Escuela de Ingeniería de Sistemas e Informática, como pionera en desarrollo Tecno-Científico debe tener por objeto, el desarrollo masivo y evolutivo de software. Esto se puede lograr con mecanismos de control y divulgación confiables y dinámicos que suplan las necesidades de su entorno.

Estas necesidades se pueden mitigar con un eficiente manejo de la información mediante la utilización de la informática, en donde la forma de acceder a está, su seguridad, confiabilidad y facilidad juegan un papel importante. Por tal razón, y conscientes de las grandes oportunidades que ofrecen las redes y telecomunicaciones en la actualidad, se opta por llevar estas facilidades a los usuarios mediante un canal masivo de información que permita la disminución de tiempo, como lo es Internet.

El desarrollo de la aplicación Web se constituyo en un medio eficiente para el manejo de la información, además de derivar ventajas como:

- Disminución de tiempo en los procesos.
- Mayor velocidad de respuesta al usuario.
- Amigabilidad con el usuario.
- Aprovechamiento de tecnologías informáticas de punta.
- Independencia entre usuarios y administrador.
- Se prescinde de personal experto para acceder la herramienta.
- Mayor divulgación por tratarse de un medio masivo de comunicación.

Aparte de las ventajas técnicas y tecnológicas ofrecidas por la Web, las opciones de renovación y Consulta, se convirtieron en procesos cómodos y menos tediosos, sumados a la evolución del software TRIONIX, acarreando mayor satisfacción a los usuarios.

## 1.3 IMPACTO

La puesta en marcha del proyecto trae consigo grandes beneficios al interior de la EISI (Escuela de Ingeniería de Sistemas e Informática), porque su uso se traduce en herramienta de protección de fácil adquisición. TRIONIX 1.1 además debe permitir:

- La interacción de La EISI con las demás dependencias universitarias internas y con usuarios de Internet.
- Fomentar el desarrollo de aplicaciones software a estudiantes, profesores y egresados, con reconocimiento por su trabajo mediante el uso temporal de la aplicación por parte de usuarios interesados en el tipo de software.
- Enriquecimiento en conocimiento al grupo software.
- La disminución de tiempos y tramitología para la protección de aplicaciones.

## 1.4 VIABILIDAD

En el desarrollo de este trabajo se contó con docentes universitarios con cierta experiencia en el desarrollo Web, además de contar con gran información en libros, Internet, tesis que abordan esta temática, con lo cual se considera que se dieron los medios necesarios para el desarrollo del presente proyecto. Lo anterior sumado con los recursos técnicos como: lenguajes de programación libres (para entorno educativo), manejadores de base de datos, Internet, equipos, servidores, infraestructura en redes, entre otras, que posee la EISI o se podrían adquirir fácilmente sin incurrir en costos.

En cuanto al uso de la aplicación Web de la herramienta se cree que tendrá gran demanda dado que cada día las necesidades de proteger el software crecen, y las soluciones que ofrece el mercado son de alto costo limitando la adquisición a los desarrolladores que inician su vida en el mundo informático.
Además se desea en un futuro no lejano constituir a "TRIONIX" en una plataforma de seguridad que permita proteger aplicaciones de todo tipo; para el logro de esta meta se requiere de una serie de proyectos encaminados a este para lograr un buen producto final: "Plataforma de seguridad para Aplicaciones Software", y el primer paso se dio con el desarrollo de la aplicación Web.

## 1.5 DESCRIPCIÓN DEL PRODUCTO

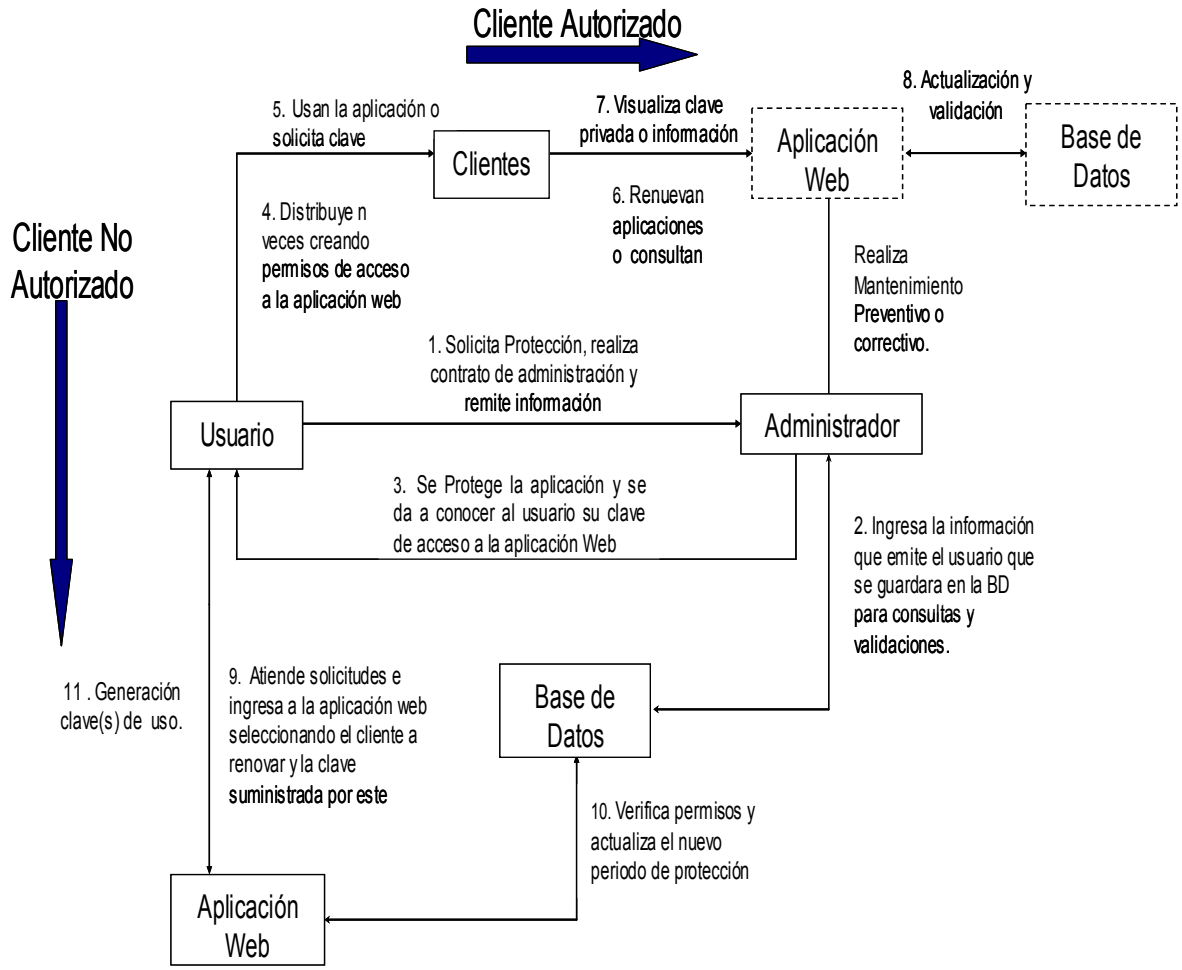



**Figura 1.2** Descripción del Producto

TRIONIX es de gran importancia para los desarrolladores de software, en especial para aquellos que apenas están incursionando, ya que evita el uso no licenciado de las aplicaciones desarrolladas por estos. Es por ello, que TRIONIX se ha desarrollado precisamente con el fin de prestar este servicio, teniendo adicionalmente como uno de sus objetivos el mejoramiento continuo de la herramienta, para lo cual cuenta con personal dispuesto y tecnologías de punta que garantizan la eficiencia y solidez del producto en pro de la satisfacción de los usuarios.

Por esta razón surgió la idea de automatizar los procesos de renovación y consulta con el fin de facilitar el acceso y divulgación de algunas aplicaciones protegidas. A continuación se describen los nuevos procedimientos que se siguen en la aplicación Web:

El usuario que desee proteger una aplicación deberá ponerse en contacto

con el administrador del sistema y celebrar un contrato de administración, donde se estipula las condiciones, como tiempo de prestación del servicio, valor y demás que dispongan las partes. Una vez realizado lo anterior, el administrador podrá proteger las aplicaciones que el usuario solicite. Para tal fin ingresará al modulo de protección de TRIONIX (no disponible en la Web), recopilará información necesaria para validaciones y consultas. Además, le otorgará al usuario una identificación y una contraseña para su ingreso a la aplicación Web para que pueda, de esta manera, acceder a las funciones propias de su perfil en el periodo convenido con el administrador.

El usuario podrá distribuir sus aplicaciones y otorgar licencias de uso a sus clientes. Cuando éstos últimos ejecuten las aplicaciones por primera vez, ellas generan la clave pública particular a la aplicación y solicitaran la privada para permitir su ejecución. De esta manera, el cliente se ve obligado a ponerse en contacto con el usuario, quien tendrá que ingresar a la aplicación Web con la clave que lo identifica y suministrar la información necesaria del cliente, así como la clave pública generada por la aplicación. Una vez finalizado lo anterior, TRIONIX realizará la correspondiente validación de los datos que dará como resultado, en caso favorable, la generación de la clave privada; el usuario si lo desea puede otorgar facultades a sus clientes permitiendo que ingresen y renueven las aplicaciones que usan.

Esta nueva clave privada será la que el usuario le proporcione a su cliente para que habilite la ejecución de la aplicación por un tiempo y/o número de accesos determinados. El cliente también podrá ingresar a la aplicación Web con su contraseña, suministrada por el usuario, para realizar las consultas propias de su perfil.

De igual manera se cuenta con un modulo de divulgación y descarga de aplicaciones protegidas para que cualquier persona que ingrese a la Web las use.

Sumado a lo anterior se implemento una nueva base de datos con las especificaciones analizadas en la descripción del problema, el algoritmo DES que se utiliza en la protección de archivos, sigue siendo utilizado en esta aplicación por el gran nivel de seguridad que ofrece. Ver Anexo 1.

**CAPITULO 2**

# MARCO TEÓRICO

El fin de este capitulo es condensar los aspectos teóricos más relevantes, para comprender en términos generales el desarrollo y enfoque de este trabajo; la metodología a seguir es explicar cada una de las tecnologías usadas; dado lo extenso de cada tema y de la finalidad de este capitulo se anexan bibliografías y direcciones de Internet en donde se encontrará mucho más soporte sobre los temas tratados.

En este capitulo se presentaran los siguientes temas:

- ✓ Aplicaciones Web.
- ✓ Arquitectura del software.
- ✓ Funciones Hash.
- ✓ Bases de Datos.

## 2.1 APLICACIONES WEB

En ingeniería del software una aplicación Web es aquella que los usuarios usan desde un servidor Web a través de Internet o de una Intranet. Las aplicaciones Web son populares debido a la ubicuidad del navegador como un cliente. El contenido final de una página se determina sólo cuando el usuario solicita una página del servidor Web. Dado que el contenido final de la página varía de una petición a otra en función de las acciones del visitante, este tipo de página se denomina página dinámica.

### 2.1.1 FUNCIONAMIENTO DE UNA APLICACIÓN WEB

Una aplicación Web es un conjunto de páginas Web estáticas y dinámicas. Una *página Web estática* es aquélla que no cambia cuando un usuario la solicita: el servidor Web envía la página al navegador Web solicitante sin modificarla. Por el contrario, el servidor modifica las *páginas Web dinámicas* antes de enviarlas al navegador solicitante. La naturaleza cambiante de este tipo de página es la que le da el nombre de dinámica.

### 2.1.2 PROCESAMIENTO DE PÁGINAS WEB ESTÁTICAS

Un sitio Web estático consta de un conjunto de páginas y de archivos HTML relacionados alojados en un equipo que ejecuta un servidor Web.

Un servidor Web es un software que suministra páginas Web en respuesta a las peticiones de los navegadores Web. La petición de una página se genera cuando el usuario hace clic en un vínculo de una página Web, elige un marcador en un navegador o introduce un URL en el cuadro de texto, Dirección del navegador.

El contenido final de una página Web estática lo determina el diseñador de la página y no cambia cuando se solicita la página.

El diseñador escribe todas y cada una de las líneas de código HTML de la página antes de colocarla en el servidor. El código HTML no cambia una vez colocado en el servidor, y por ello este tipo de páginas se denomina página estática.

Cuando el servidor Web recibe una petición de una página estática, el servidor lee la solicitud, localiza la página y la envía al navegador solicitante, como se muestra en la siguiente figura:

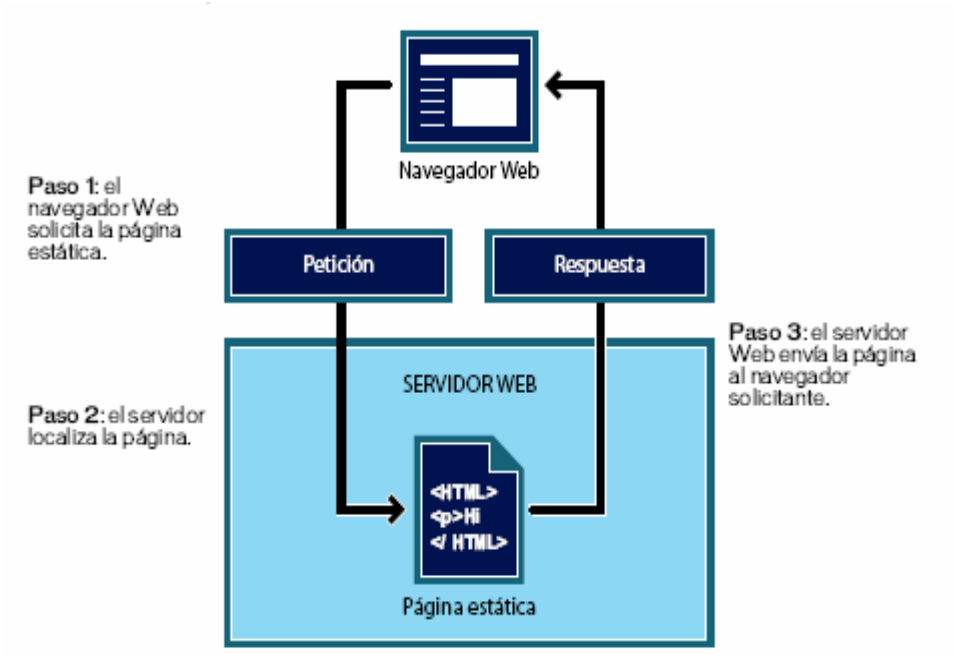



**Figura 2.1** Aplicación Web Navegador-Servidor Web

### 2.1.3 PROCESAMIENTO DE PÁGINAS DINÁMICAS

Cuando un servidor Web recibe una petición para mostrar una página Web estática, el servidor la envía directamente al navegador que la solicita. Cuando el servidor Web recibe una petición para mostrar una página dinámica, sin embargo, reacciona de distinta forma: transfiere la página a un software especial encargado de finalizar la página. Éste software especial se denomina servidor de aplicaciones.

El servidor de aplicaciones lee el código de la página, finaliza la página en función de las instrucciones del código y elimina el código de la página. El resultado es una página estática que el servidor de aplicaciones devuelve al servidor Web, que a su vez la envía al navegador solicitante.

Lo único que el navegador recibe cuando llega la página es código HTML puro. A continuación se incluye una vista de este proceso:

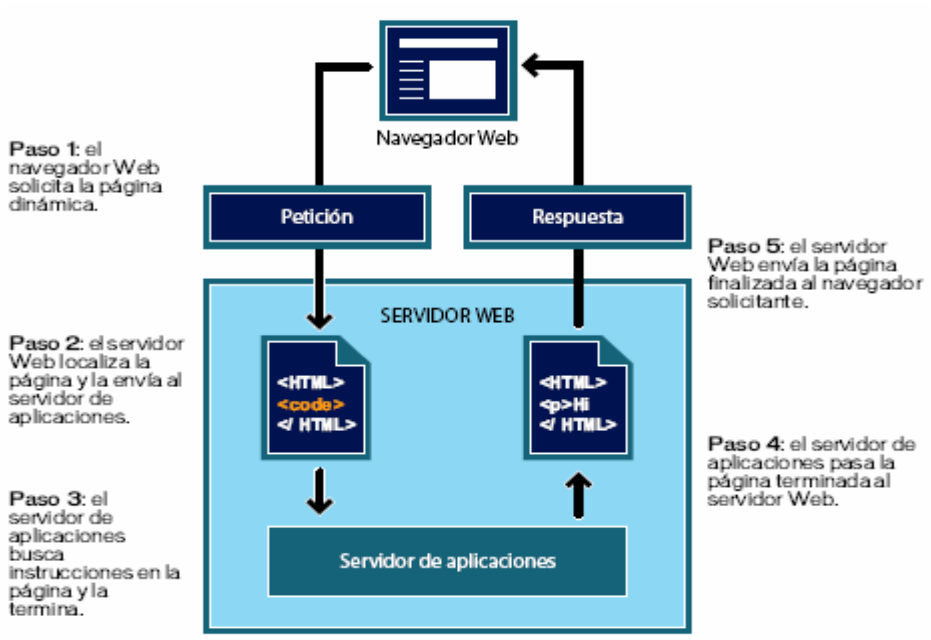



**Figura 2.2** Aplicación Web Navegador - Servidor de aplicaciones

### 2.1.4 ACCESO A UNA BASE DE DATOS

Un servidor de aplicaciones le permite trabajar con recursos del lado del servidor, como una base de datos. Por ejemplo, una página dinámica puede indicar al servidor de aplicaciones que extraiga datos de una base de datos y los inserte en el código HTML de la página. La instrucción para extraer datos de una base de datos recibe el nombre de consulta de base de datos.

Una consulta consta de criterios de búsqueda expresados en un lenguaje de base de datos denominado SQL (Structured Query Language, lenguaje de consulta estructurado). La consulta SQL se escribe en los scripts o etiquetas del lado del servidor de la página.

En el anexo 2 se encuentra una explicación mas amplia sobre base de datos y explícitamente sobre mysql, que es el motor de base de datos usado en este proyecto, por su confiabilidad y rapidez en las transacciones.

### 2.1.5 TECNOLOGÍAS DISPONIBLES PARA EL DESARROLLO

Para desarrollar aplicaciones y dotar a las páginas Web de funcionalidad se puede trabajar tanto en el lado del cliente como en el lado del servidor, las variantes son:

#### 2.1.5.1 PROGRAMACIÓN EN EL CLIENTE

- ✓ El navegador envía un request (petición: digitando la URL de la página o simplemente cualquier solicitud al servidor de aplicaciones).
- ✓ El servidor envía un response que contiene código que el navegador entiende. (envía como respuesta una página HTML con la información solicitada).
- ✓ El navegador interpreta el código enviado por el servidor y realizar una determinada acción. (visualiza la información a quien realizo inicialmente la solicitud).

#### 2.1.5.2 PROGRAMACIÓN EN EL SERVIDOR

- ✓ El navegador envía un request. (recibe la petición del cliente).
- ✓ El servidor ejecuta una aplicación que realiza una determinada acción. (realiza consulta en la Base de Datos o busca la pagina solicitada).
- ✓ El servidor envía el resultado de dicha aplicación al cliente. (página HTML).
- ✓ El navegador muestra el resultado recibido del servidor.

#### 2.1.5.3 ESQUEMA MIXTO: (PROGRAMACIÓN EN EL CLIENTE Y EN EL SERVIDOR)

- ✓ El navegador envía un request.
- ✓ El servidor ejecuta una aplicación que realiza una determinada acción.
- ✓ El servidor envía el resultado de dicha aplicación al cliente conteniendo código a interpretar por el navegador.
- ✓ El navegador interpreta el código enviado por el servidor y realiza una determinada acción.

La programación del lado del cliente tiene como principal ventaja que la ejecución de la aplicación se delega al cliente, con lo cual se evita recargar al servidor de trabajo. El servidor sólo envía el código y es tarea del navegador interpretarlo. La gran desventaja de ésta metodología es que el código que el servidor envía es "sensible" a qué cosas puede o no hacer el navegador. El usuario puede, por ejemplo, decidir deshabilitar una funcionalidad del navegador que es necesaria para que se ejecute un determinado servicio o peor aun, navegadores distintos pueden interpretar el mismo código de distintas formas. Típicamente Firefox y Microsoft, que producen los dos navegadores más usados del mercado, no están de acuerdo sobre como se implementan diversas tecnologías en el cliente.

Programar del lado del servidor tiene como gran ventaja que cualquier cosa puede hacerse sin tener en cuenta el tipo de cliente, ya que la aplicación se ejecuta en el servidor que es un ambiente controlado. Una vez ejecutada la aplicación, el resultado que se envía al cliente puede estar en un formato "normalizado" que cualquier cliente puede mostrar. La desventaja reside en que el servidor se sobrecarga de trabajo ya que además de servir paginas, es responsable de ejecutar aplicaciones. A menudo esto redunda en requisitos de hardware mayores a medida que el servidor ejecuta más y más servicios.

Sin embargo, debido a las incompatibilidades existentes y a la posibilidad de que el usuario controle que cosas se ejecutan y cuales no, la programación del lado del cliente no es muy recomendable y debe limitarse a código altamente estándar que pueda interpretarse de cualquier forma el navegador, lo cual obliga a ejecutar la gran mayoría de las aplicaciones y servicios del lado del servidor.

- **Uso de una API del servidor:**

Otra técnica factible consiste en utilizar una API (application programming interface) provista por el servidor Web para desarrollar aplicaciones, es decir que el servidor provee un lenguaje en el cual se pueden desarrollar aplicaciones. Este esquema, como podemos apreciar, es mucho más eficiente que el anterior ya que el servidor Web es el encargado de ejecutar las aplicaciones en forma directa sin necesidad de crear un proceso. Las desventajas son sin embargo importantes: En primer lugar las aplicaciones creadas en éste marco no son portables ya que solo pueden ejecutarse en un servidor Web determinado, esto es una gran desventaja frente a la aplicaciones CGI que podían una vez desarrolladas ejecutarse en cualquier servidor. La segunda gran desventaja es que frecuentemente un error de programación de una aplicación podría ocasionar que el servidor deje de funcionar, genere un error, pierda memoria u otros problemas. Esto ocasiona que este tipo de aplicación no sea confiable.

❖ **Uso de un Módulo del Servidor Web**

La tecnología más reciente para la ejecución de aplicaciones consiste en anexar a un servidor Web "módulos" que le permitan interpretar un determinado lenguaje. De ésta forma se logra eficiencia ya que el servidor no necesita crear un nuevo proceso por cada aplicación que ejecuta. Las aplicaciones son portables ya que son desarrolladas en un lenguaje estándar que no depende del servidor Web. Las aplicaciones son confiables ya que si bien puede producir un error en el lenguaje en que están diseñadas, si el módulo es sólido, dichos errores no pueden comprometer al servidor.

En las siguientes tablas (Tabla 2.1 y Tabla 2.2) se presenta un resumen de los principales lenguajes con una descripción de su uso en programación del lado del servidor así como una breve comparación de los mismos.

| | Descripción |
|---|---|
| Perl | *Practical Extraction and Report Language.* Lenguaje interpretado creado con el objeto principal de simplificar las tareas de administración de un sistema UNIX. Hoy en día se ha convertido en un lenguaje de propósito general. |
| Pitón | Es un lenguaje interpretado que permite escribir programas pequeños, utilizado en desarrollo Web para la creación de CGI. |
| C, C++ | Utilizado para la creación de CGI. |
| PHP | PHP es un lenguaje interpretado diseñado para el desarrollo de sitios dinámicos. La distribución más popular de PHP es como módulo para el servidor Apache, aunque puede funcionar como un interprete para ejecutar aplicaciones CGI. |
| ASP | Active Server Pages. Tecnología creada por Microsoft destinada a la creación de sitios Web. |
| JSP | Java Server Pages. (JSP). Es un lenguaje interpretado insertado en páginas Web y basadas en Java para el desarrollo de sitios dinámicos. |
| Mod_perl | Módulo de Perl para el servidor web. |
| Mod_python | Módulo python para el servidor web. |

**Tabla 2.1** Lenguaje al lado del servidor

El siguiente es un cuadro comparativo entre los lenguajes al lado del servidor,

analizando factores como: tiempo de desarrollo, depuración, confiabilidad y eficiencia:

| | **CGI (Interpretado)** | **CGI (Compilado)** | **API del servidor** | **Módulo del servidor** |
|---|---|---|---|---|
| **Ejemplos** | Perl, Python | C, C++ | Netscape Enterprise | PHP, ASP, JSP,Mod_perl, Mod_python, FastCGI |
| **Tiempo de Desarrollo** | Corto | Largo | Medio | Corto |
| **Depuración** | Sencilla | Compleja | Compleja | Sencilla |
| **Confiabilidad** | Alta | Alta | Baja | Alta |
| **Eficiencia** | Baja | Media | Alta | Alta |

**Tabla 2.2** Comparación de lenguajes al lado del servidor

Para el desarrollo de éste proyecto se ha elegido la tecnología JSP (Java Server Pages), debido a su alta confiabilidad, tiempos cortos en el desarrollo y gran eficiencia en aplicaciones Web, por tanto a continuación se realiza una pequeña introducción a este lenguaje.

### 2.1.6 JSP (JAVA SERVER PAGES)

JSP es una tecnología basada en Java que simplifica el proceso de desarrollo de sitios Web dinámicos. Las *Java Server Pages* son ficheros de texto normalmente con extensión *.jsp*) que sustituyen a las páginas HTML tradicionales. Los ficheros JSP contienen etiquetas HTML y código embebido que permite al diseñador de la página Web acceder a datos desde código Java que se ejecuta en el servidor.

Cuando la página JSP es requerida por un navegador a través de una petición HTTP, las etiquetas HTML permanecen inalterables, mientras que el código que contiene dicha página es ejecutado en el servidor, generando contenido dinámico que se combina con las etiquetas HTML antes de ser enviado al cliente.

Este modo de operar implica una separación entre los aspectos de presentación de la página y la lógica de programación contenida en el código.

#### 2.1.6.1 BENEFICIOS QUE APORTA JSP

Los beneficios ofrecidos por JSP como alternativa a la generación de contenido dinámico para la Web se resumen a continuación.

- **Mejoras en el rendimiento:**
  - Utilización de procesos ligeros (hilos Java) para el manejo de las peticiones.
  - Manejo de múltiples peticiones sobre una página *.jsp* en un instante dado.
  - El contenedor servlet puede ser ejecutado como parte del servidor web.
  - Facilidad para compartir recursos entre peticiones (hilos con el mismo padre: *servlet container*).

- **Soporte de componentes reutilizables:**
  - Creación, utilización y modificación de JavaBeans del servidor.
  - Los JavaBeans utilizados en páginas *.jsp* pueden ser utilizados en servlets, applets o aplicaciones Java.

- **Separación entre código de presentación y código de implementación:**

  Cambios realizados en el código HTML relativos a cómo son mostrados los datos, no interfieren en la lógica de programación y viceversa.

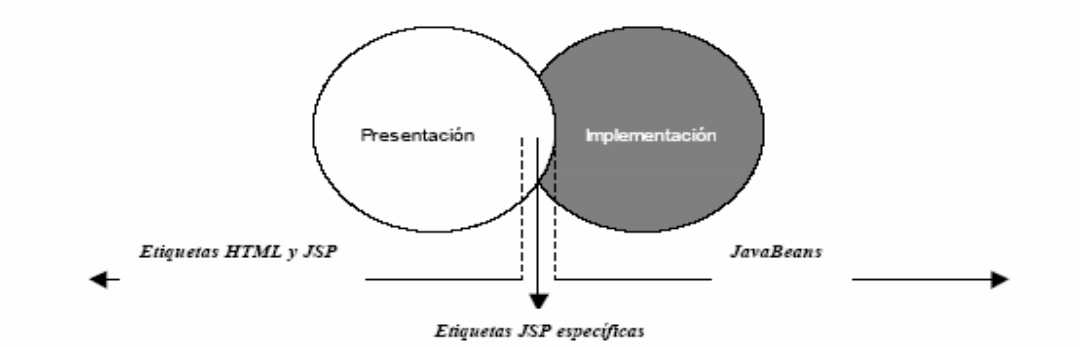



**Figura 2.3** Separación entre código de presentación y código de implementación

- **División del trabajo:**
  - Los diseñadores de páginas pueden centrarse en el código HTML y los Programadores en la lógica del programa.
  - Los desarrollos pueden hacerse independientemente.

- Las frecuentes modificaciones de una página se realizan más eficientemente.

#### 2.1.6.2 ¿COMO TRABAJA JSP?

Para poder utilizar esta tecnología es necesario un servidor Web que de soporte a páginas *.html*, y código que implemente un *contenedor JSP* donde ejecutar las etiquetas JSP. Existen servidores Web que incorporan dicha capacidad dentro de su código (Netscape *Enterprise y Aplication Server 4.0*), así como servidores escritos íntegramente en Java (*Java Web Server* de Sun y *Jigsaw* de W3 Consortium) que dan soporte a esta tecnología directamente.
Sin embargo, la mayoría de servidores Web, están escritos en lenguajes de programación tradicionales, compilados en código nativo por razones de eficiencia. Para estos servidores es necesario añadir código suplementario que implemente el *contenedor JSP*. Para ello se han proporcionado API's del servidor para poder extender su funcionalidad, así Netscape proporciona **NSAPI** (***N****etscape* ***S****erver* ***A****pplication* ***P****rogramming* ***I****nterface*), y Microsoft proporciona **ISAPI** (***I****nternet* ***S****erver* ***A****pplication* ***P****rogramming* ***I****nterface*). Mediante estas API's es posible desarrollar herramientas de soporte para JSP.

Para el servidor Apache, se dispone de módulos. En versiones antiguas del servidor era necesario recompilar el código del servidor para soportar un nuevo módulo, actualmente son cargados dinámicamente basándose en ficheros de configuración.
Una vez que el contenedor JSP ha sido instalado y configurado, los ficheros *.jsp* se tratan igual que los ficheros *.html*, situándolos en cualquier lugar de la jerarquía de directorios. Cualquier clase Java que se utilice en un fichero *.jsp*, debe estar disponible en la variable *classpath* del contenedor JSP.

Aunque la especificación JSP no presupone nada sobre la implementación que da soporte a esta tecnología, la mayoría de las implementaciones disponibles están basadas en *servlets*.

El primer componente de las implementaciones basadas en servlets, es un servlet especial denominado *compilador de páginas*. Este servlet junto con sus clases Java asociadas, se conoce con el nombre de *contenedor JSP*. El contenedor está configurado para llamar al compilador de páginas para todas las peticiones que coincidan con una página *.jsp*. Su misión es la de compilar cada página *.jsp* en un servlet cuya finalidad es la de generar el contenido dinámico especificado por el documento *.jsp* original.
Para compilar una página, el compilador de páginas escanea el documento en busca de etiquetas JSP, generando el código Java correspondiente para cada una de ellas. Las etiquetas estáticas HTML son convertidas a Strings de Java. Las etiquetas que hacen referencia a JavaBeans son traducidas en los correspondientes objetos y llamadas a métodos. Una vez que el código del servlet ha sido construido (se ha codificado su método **service()**), el compilador de páginas llama al compilador de Java para compilar el código fuente y añade el fichero de bytecodes resultante al directorio apropiado del contenedor .

Una vez que el servlet correspondiente a la página *.jsp* ha sido generado, el compilador de páginas invoca al nuevo servlet para generar la respuesta al cliente.

Mientras que el código de la página *.jsp* no se altere, todas las referencias a la página ejecutarán el mismo servlet.

Esto supone una cierta demora en la primera referencia a la página, aunque existen mecanismos en JSP para precompilar la página *.jsp* antes de que se haya producido la primera petición. La figura 2.4 muestra el proceso para crear y ejecutar servlets JSP.

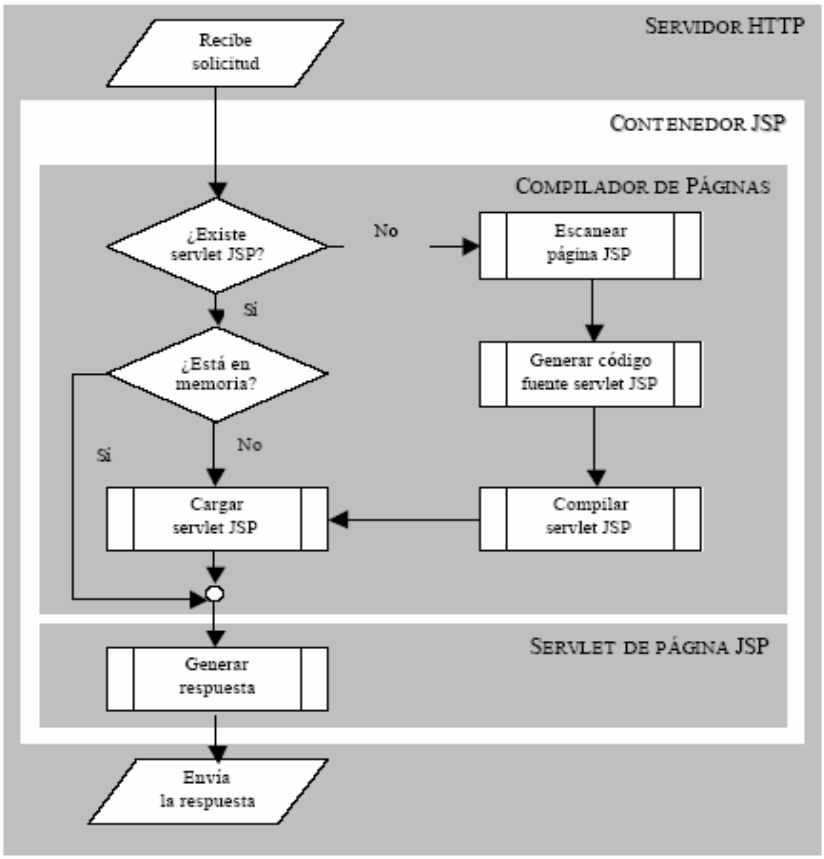



**Figura 2.4** Proceso para crear y ejecutar servlets JSP

## 2.2 ARQUITECTURA DEL SOFTWARE

La arquitectura de un sistema es la vista conceptual de la estructura de éste. Toda aplicación contiene código de presentación, código de procesamiento de datos y código de almacenamiento de datos. Por tanto su arquitectura se define según como éste distribuya el código.

En los últimos años la arquitectura de los sistemas de software desarrollados ha evolucionado junto con el resto de la tecnología informática. Hace no muchos años, cuando los mainframes eran una novedad, se desarrollaban sistemas monolíticos en donde un sistema no era conocido más allá de su entorno de operación.

Cuando la tecnología de redes aparece y se difunde, la industria no tardó en darse cuenta de la ventaja de desarrollar sistemas que contaran con la capacidad de interactuar con otros sistemas residentes en otras máquinas dentro de la red. De aquí surge el modelo Cliente-Servidor donde un "Cliente" solicita servicios de un "Servidor" el cual gestiona las solicitudes de varios clientes a la vez.

Luego llegó el auge de Internet y el desarrollo de nuevas tecnologías para software por componentes. Con éstas se puede construir una aplicación distribuida que reside en uno o más servidores en la red y además se disminuyó la necesidad de software cliente ya que generalmente se utiliza uno ya estandarizado: los navegadores de Internet tales como Netscape, Internet Explorer, Firefox entre otros.

Las organizaciones están aprovechando estas tecnologías que permiten a sus usuarios el tener acceso sencillo y casi universal a sus aplicaciones corporativas sin la necesidad de incurrir en todos los gastos de mantenimiento que conlleva el modelo cliente-servidor como fue planteado en sus inicios.
Para que todo esto funcione, se requiere de incrementar la lógica de programación del lado del servidor ya que la funcionalidad del cliente es mínima. Aquí es donde surge el concepto de sistema multicapa o de "n" capas como una metodología para el diseño de sistemas distribuidos. Esta arquitectura se hizo popular a principios de los años 90 y en la actualidad se ha afianzado como la arquitectura de software de aplicación empresarial.

### 2.2.1 CLIENTE – SERVIDOR

Esta es una versión simplificada de la arquitectura en capas, donde la capa de aplicación se encuentra repartida entre el cliente y el servidor. Los sistemas basados en la arquitectura cliente /servidor están formados por dos partes lógicas o capas: un servidor que proporciona servicios y un cliente que solicita servicio del servidor o servidores. Los dos, juntos, forman un sistema de computación completo con una clara división de responsabilidades.

Un cliente es un proceso que envía un mensaje a un proceso servidor, solicitando que realice una determinada tarea. Los procesos cliente normalmente gestionan la porción de interfaz de usuario de la aplicación, validan los datos introducidos por el usuario, realizan las solicitudes a los servidores y, a veces, ejecutan cierta lógica de negocio.
Un servicio es un proceso que contesta a la solicitud del cliente realizando la tarea propuesta por éste. Los clientes gestionan recursos compartidos como ficheros, impresoras, enlaces de comunicación, base de datos, etc.

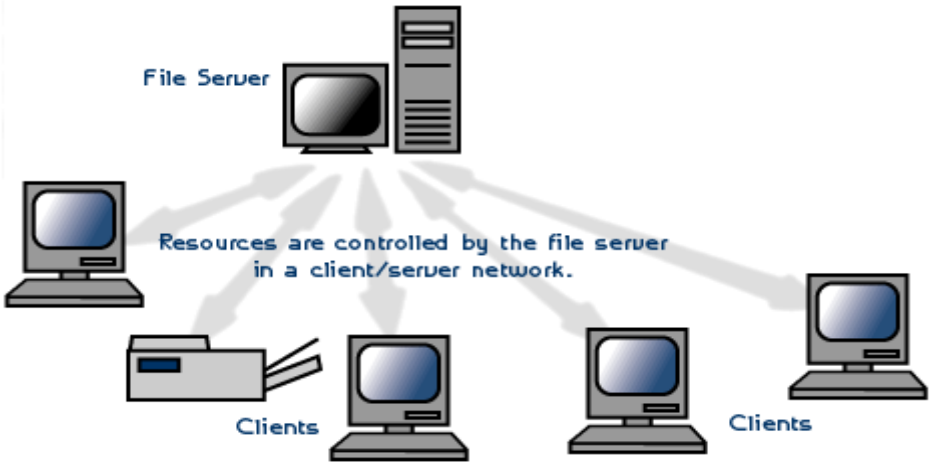



**Figura 2.5** Arquitectura Cliente – Servidor

### 2.2.2 MÚLTIPLES CAPAS

Una arquitectura multicapa consiste en dividir la funcionalidad del sistema total en capas lógicas que pueden ser encapsuladas como componentes que interactúan entre ellos a alto nivel y supone una mayor escalabilidad de la aplicación, un mantenimiento menor y un incremento de la reutilización de componentes.

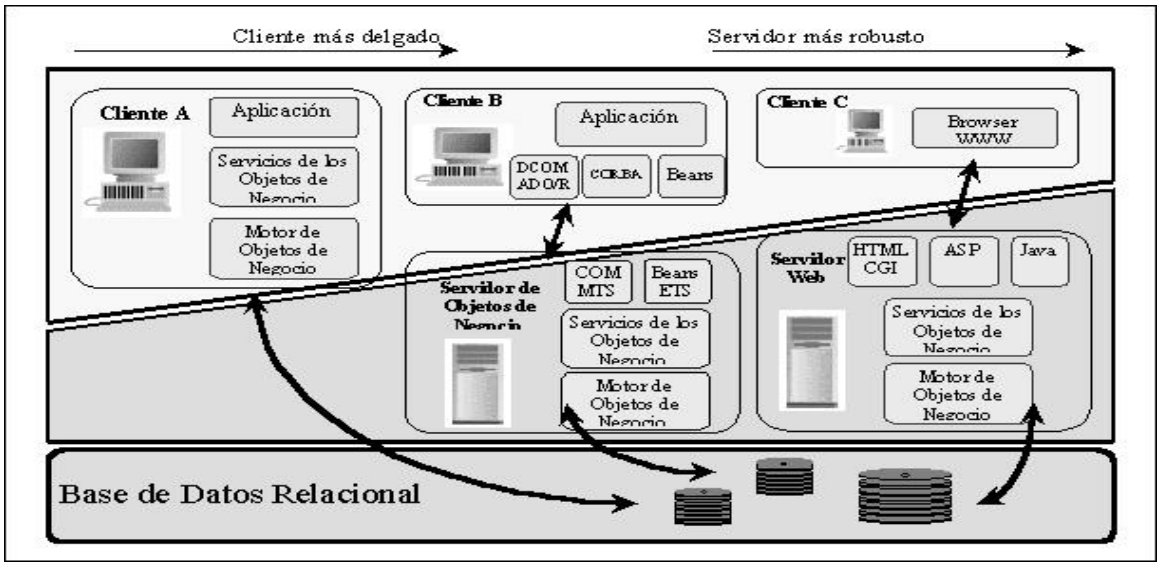



**Figura 2.6** Arquitectura de tres capas. Caso más general de arquitectura multicapas

El número de capas típico que se utiliza en esta arquitectura son tres, capa de datos, de negocio y de presentación. Así en la capa de datos encapsulados todas las funciones de base de datos, las funciones de interfaz de usuario en la capa de "presentación" y finalmente toda la lógica de operación en la capa de "Negocios".

Esta encapsulación tiene como objetivo ofrecer una colección más simple de funciones que permiten desarrollar aplicaciones a alta velocidad y bajo costo. El costo de mantenimiento del sistema disminuye notablemente ya que una modificación en una capa no debe afectar a las demás.

- **Capa de Datos**

El nivel de servicio de datos es responsable de: Almacenar, Recuperar y mantener los datos así como verificar la integridad de los mismos.

Los servicios de datos tienen una variedad de formas y tamaños, incluyendo los sistemas de administración de bases de datos relacionales, servicios de correo electrónico y sistema de archivos.

- **Capa de Aplicación o de Negocio**

La capa de aplicación es el "puente" entre un usuario, representado en la capa de presentación, y los servicios de datos, que se encuentran en la capa de datos. Los servicios de ésta capa responden a peticiones del usuario (u otros servicios de negocio) para ejecutar una tarea de éste tipo. Cumple con esto aplicando procedimientos formales y reglas de negocio a los datos revelantes. Esto aísla al usuario de la interacción directa con la base de datos.

- **Capa de Presentación**

Los servicios de presentación proporcionan la interfaz necesaria para presentar información y reunir datos. También aseguran los servicios de negocios necesarios para ofrecer las capacidades de transacción requeridas e integrar al usuario con la aplicación para ejecutar un proceso de negocios.

Los servicios de presentación generalmente son identificados con la interfaz de usuario, y normalmente resiente en un programa ejecutable localizado en la estación de trabajo del usuario final. El cliente proporciona el contexto de presentación, generalmente un navegador como *Microsoft Internet Explorer, Firefox o Netscape*, que permite ver los datos remotos a través de una capa de presentación HTML, u otro tipo de aplicación.

## 2.3 AUTENTICACION DE USUARIOS

Para la protección de contraseñas de usuarios en la base de datos se utilizo la función HASH MD5 implementada en MYSQL, la cual garantiza que toda contraseña, solo va a ser conocida por quien corresponda, garantizando la autenticidad de los usuarios, a continuación se explicara dicha función en detalle.

### 2.3.1 FUNCIONES DE HASH

Las funciones de hash de una vía (one-way) son una construcción criptográfica empleada en muchas aplicaciones. Son usadas junto con los algoritmos de clave pública para encriptación y firma digital.

#### 2.3.1.1 HISTORIA

En 1990, Ronald L Rivest[9] inventó la función hash MD4. En 1992, mejoró el MD4 y desarrolló otra función hash: MD5. En 1993, la National Security Agency (organismo dependiente del gobierno de EE.UU.) publicó una función hash muy similar al MD5, llamada SHA (Secure Hash Algorithm). Hoy la función más popular de hash es MD5 y ha sido utilizada en gran cantidad de aplicaciones.

Las funciones de hash de una vía tienen dos propiedades:

- Son de una sola vía. Esto significa que es fácil tomar un mensaje y computarlo en el valor del hash, pero es imposible tomar el valor del hash y recrear el mensaje original.

- Están libres de colisiones. Quiere decir que es imposible encontrar dos mensajes que generen un digesto del mismo valor. Quebrar una función hash significa mostrar que cada una o ambas de estas propiedades no son ciertas.

Los algoritmos HASH toman una entrada de longitud arbitraria y calculan un valor de tamaño constante de 32 caracteres, cumpliendo una serie de propiedades criptográficas como:

- Conociendo un HASH, no obtenemos ninguna información sobre el documento original.

- Teniendo un HASH dado, no es factible encontrar un documento cuyo HASH coincida con el primero.

- Un cambio cualquiera en el documento de entrada debe modificar, en media, la mitad de los bits de la salida.

#### 2.3.1.2 EL ALGORITMO MD5

El algoritmo MD5 (acrónimo de Message-Digest Algorithm 5, Algoritmo de Resumen del Mensaje 5) es una función de cifrado tipo hash que acepta una

[9] Profesor De Ingeniería Eléctrica y Electrónica Asociado al MIT(Massachusetts Institute of tecnology).

cadena de texto como entrada y devuelve una cadena de 128 bits. La ventaja de este tipo de algoritmos es la imposibilidad (computacional) de reconstruir la cadena original a partir del resultado.

Ejemplos:

| Mensaje | Fingerprint() o HASH |
|---|---|
| Hola | 4d186321c1a7f0f354b297e8914ab240 |
| sistemas | 102ddaf691e1615d5dacd4c86299bfa4 |
| Gonzalez | E796e897b03dfa33388c5e26154376de |
| | |

#### 2.3.1.3 SEGURIDAD

Puede surgir la pregunta, ¿Se puede desencriptar un hash MD5?, La respuesta es muy sencilla: No, Básicamente porque MD5 no es un algoritmo de encriptación, sino para hallar un message digest.

En dicho message digest no existe información alguna de la cadena original. Es una simple huella digital del mensaje, pero su contenido no se encuentra implícito en los 32 caracteres que componen el hash.
Los hash MD5 son comparables a las huellas dactilares del ser humano. Un ejemplo seria el encontrar la huella dactilar del asesino de turno: Mediante la simple observación de esa huella no obtendrá ninguna información. En la huella en si no hay información sobre si el asesino es hombre, mujer, rubio o moreno, alto o bajo. No seria capaz de decirnos nada aparte de "es una huella".

Solamente mediante la comparación de dicha huella con la base de datos de la policía, y hallando una coincidencia con las allá almacenadas, podremos descubrir la identidad del asesino.

¿Existe un hacker seguro que pueda desencriptar hashes MD5?, No. Ni siquiera Ronald L. Rivest, el programador del algoritmo es capaz de desencriptar un hash MD5, es imposible, porque no se puede sacar información de donde no hay, y si el hash no contiene información de la cadena de entrada, no se puede inventar.

Imagina que hallamos el hash de la Biblia, una cadena enorme, decenas de megabytes. Imagina que el hash resultante es por ejemplo: 4cf78a309d67c6ed32a29fc722ab7d9 (con 32 caracteres); es imposible hallar la cadena original, teniendo en cuenta que si en esa cadena grande de caracteres cambio una c por una s ya el documento no es el mismo.

El algoritmo MD5 estima que, dado un hash cualquiera, la dificultad de encontrarse con el mensaje que lo produjo es de $2^{128}$ intentos

(3.402823669209384634633746074317 7E+38, tarea Computacionalmente muy difícil de llevar a cabo).

#### 2.3.1.4 CODIFICACIÓN

La codificación del MD5 de 128 bits es representada típicamente como un número de 32 dígitos hexadecimal. El siguiente código de 28 bytes ASCII será tratado con MD5 y veremos su correspondiente hash de salida:

MD5("Esto si es una prueba de MD5") = e07186fbff6107d0274af02b8b930b65.
Un simple cambio en el mensaje nos da un cambio total en el la codificación hash, en este caso cambiamos dos letras, el "si" por un "no". MD5("Esto no es una prueba de MD5") = dd21d99a468f3bb52a136ef5beef5034; Otro ejemplo sería la codificación de un campo vacío:

MD5("") = d41d8cd98f00b204e9800998ecf8427e

#### 2.3.1.5 ALGORITMO

- **Terminologías y Notaciones**

Una secuencia de bits puede ser interpretada de manera natural como una secuencia de bytes, donde cada grupo consecutivo de ocho bits se interpreta como un byte con el bit más significativo al principio. Similarmente, una secuencia de bytes puede ser interpretada como una secuencia de 32 bits (palabra), donde cada grupo consecutivo de cuatro bytes se interpreta como una palabra en la que el bit menos significativo está al principio.

El símbolo "+" significa suma de palabras.
X <<< s se interpreta por una rotación a la izquierda 's' posiciones
not(x) se entiende como el complemento de x

- **Descripción del algoritmo MD5**

Empezamos suponiendo que tenemos un mensaje de 'b' bits de entrada, y que nos gustaría encontrar su resumen. Aquí 'b' es un valor arbitrario entero no negativo, pero puede ser cero, no tiene por qué ser múltiplo de ocho, y puede ser muy largo. Imaginemos los bits del mensaje escritos así:
m0 m1 ... m{b-1}
Los siguientes cinco pasos son efectuados para calcular el resumen del mensaje.

**Paso 1. Añadiendo bits**

El mensaje será extendido hasta que su longitud en bits sea congruente con 448, módulo 512.

Esto es, el mensaje se extenderá hasta que se forme el menor número múltiplo de 512 bits. Esta extensión se realiza siempre, incluso si la longitud del mensaje es ya congruente con 448, módulo 512.
La extensión se realiza como sigue: un sólo bit "1" se añade al mensaje, y después bits "0" se añaden hasta que la longitud en bits del mensaje es congruente con 448, módulo 512. En todos los mensajes se añade al menos un bit y como máximo 512.

**Paso 2. Longitud del mensaje**

Una representación de 64 bits de 'b' (la longitud del mensaje antes de añadir los bits) se añade al resultado del paso anterior. En el supuesto no deseado de que 'b' sea mayor que 264, entonces sólo los 64 bits de menor peso de 'b' se usarán.

En este punto el mensaje resultante (después de rellenar con los bits y con 'b') se tiene una longitud que es un múltiplo exacto de 512 bits. A su vez, la longitud del mensaje es múltiplo de 16 palabras (32 bits por palabra). Con M[0 ... N-1] denotaremos las palabras del mensaje resultante, donde N es múltiplo de 16.

**Paso 3. Inicializar el búfer MD**

Un búfer de cuatro palabras (A, B, C, D) se usa para calcular el resumen del mensaje. Aquí cada una de las letras A, B, C, D representa un registro de 32 bits. Estos registros se inicializan con los siguientes valores hexadecimales, los bits de menor peso primero:

palabra A: 01 23 45 67
palabra B: 89 ab cd ef
palabra C: fe dc ba 98
palabra D: 76 54 32 10

**Paso 4. Procesado del mensaje en bloques de 16 palabras**

Primero definimos cuatro funciones auxiliares que toman como entrada tres palabras de 32 bits y su salida es una palabra de 32 bits.

$$F(X,Y,Z) = (X \wedge Y) \vee (\neg X \wedge Z)$$
$$G(X,Y,Z) = (X \wedge Z) \vee (Y \wedge \neg Z)$$
$$H(X,Y,Z) = X \oplus Y \oplus Z$$
$$I(X,Y,Z) = Y \oplus (X \vee \neg Z)$$

Los operadores $\oplus, \wedge, \vee, \neg$ son las funciones XOR, AND, OR y NOT respectivamente.

En cada posición de cada bit F actúa como un condicional: si X,

entonces Y sino Z. La función F podría haber sido definida usando + en lugar de v ya que XY y not(x)Z nunca tendrán unos ('1') en la misma posición de bit.

Es interesante resaltar que si los bits de X, Y y Z son independientes y no sesgados.

Las funciones G, H e I son similares a la función F, ya que actúan "bit a bit en paralelo" para producir sus salidas de los bits de X, Y y Z, en la medida que si cada bit correspondiente de X, Y y Z son independientes y no sesgados, entonces cada bit de G(X,Y,Z), H(X,Y,Z) e I(X,Y,Z) serán independientes y no sesgados.

Este paso usa una tabla de 64 elementos T[1 ... 64] construida con la función seno. Denotaremos por T[i] el elemento i-ésimo de esta tabla, que será igual a la parte entera del valor absoluto del seno de 'i' 4294967296 veces, donde 'i' está en radianes.

#### 2.3.1.6 APLICACIONES DE MD5

- **MD5 para almacenar Contraseñas**

Es muy común que se use MD5 para almacenar las contraseñas de los usuarios en una base de datos. Este método es completamente seguro, ya que quien administre la base de datos solo vera allí los hashes mas no las contraseñas originales.

La única forma de encontrar la contraseña original seria un ataque de fuerza bruta, no desencriptar la contraseña, para esto se tiene una ventaja primordial, (claro si no se ha previsto): sabemos que la longitud de las passwords es limitada y casi nunca supera los 10 caracteres, partiendo de esto se podría hallar el hash de todas las combinaciones de letras y números hasta un máximo de 8 o 10 caracteres e ir comparando cada resultado con el hash guardado en la base de datos. Cuando se allá la coincidencia se habrá encontrado la contraseña, este proceso resultaría inmensamente costoso en tiempo y maquina, pero podría resultar. Se esta hablando de una contraseña de menos de 10 caracteres y con combinaciones comunes como un nombre de persona o un numero telefónico.
Para que la seguridad sea máxima y evitar lo anterior, se recomienda usar una contraseña de mas de 10 caracteres, haciendo una combinación de letras símbolos y números, o de la misma forma una combinación de oraciones. Si se encuentra un hash de una contraseña de estas características resultaría imposible encontrar la contraseña original, puesto que hacer un ataque de fuerza bruta con cadenas de más de 10 caracteres es una locura.

Otros ataques mas sofisticados no prueban todas las combinaciones, sino que

hallan los hashes de palabras de menos de 10 u 8 caracteres a partir

de un diccionario (y probando varias combinaciones numéricas con cada palabra). En este proceso, y dado que las contraseñas de los usuarios suelen ser lamentablemente fáciles de crackear

Bien es cierto que el atacante tendría que tener acceso a los hashes de la base de datos para realizar estos ataques, de cualquier manera una contraseña con las características antes dichas seria bastante difícil vulnerar.

- **¿Cómo se almacenan las Contraseñas de usuario en la Base de Datos?**

Cuando un usuario se registra, se llenan los requisitos exigidos por medio de un formulario, dentro de esos requisitos esta el login de usuario y el password de usuario; luego los datos son ingresados a la base de datos y el usuario queda registrado. Se considera que el dato mas relevante en cuanto a seguridad resulta ser la contraseña de usuario, razón por la cuál al ser introducida por el usuario al registrarse, se le aplica la función hash MD5 y luego se ingresa a la base de datos como una cadena de 32 caracteres, cuando el usuario entra de nuevo a la aplicación las claves encriptadas no se desencriptan para comprobarlas lo que se hace es aplicarle la función hash MD5 a la clave que introduce el usuario al entrar y comprobar que ambas claves generadas por MD5 (la introducida por el usuario y la que está almacenada en la BD) son iguales, si estas claves resultan iguales el usuario podrá acceder a la aplicación, de lo contrario no será posible.

Además de proteger las contraseñas se protegió con las funciones Hash las claves que se guardan en la aplicación y la respuesta secreta; porque no tendría sentido que se protejan las contraseñas si a partir de la respuesta secreta se puede acceder a la aplicación web (caso olvido de contraseña), sin ningún problema.

- **MD5 Para proteger la integridad de archivos**

Otra aplicación de MD5 que se utiliza extensamente en el mundo del software es para proporcionar la seguridad de que un archivo descargado de Internet no se ha alterado. Comparando una suma MD5 publicada con la suma de comprobación del archivo descargado, un usuario puede tener la confianza suficiente de que el archivo es igual que el publicado por los desarrolladores. Esto protege al usuario contra los 'Caballos de Troya' o 'Troyanos' y virus que algún otro usuario malicioso pudiera incluir en el software. La comprobación de un archivo descargado contra su suma MD5 no detecta solamente los archivos alterados de una manera maliciosa, también reconoce una descarga corrupta o incompleta.

## 2.4 INTRODUCCIÓN A MYSQL

MySQL es un sistema de gestión de bases de datos relacional, licenciado bajo la (GPL[10] de la GNU[11]). Este gestor de bases de datos es probablemente el gestor más usado en el mundo del software libre, debido a su gran rapidez y facilidad de uso. Esta gran aceptación se debe en parte, a que existen infinidad de librerías y otras herramientas que permiten su uso a través de gran cantidad de lenguajes de programación, además de su fácil instalación y configuración.

Dentro de los beneficios más relevantes de mysql se encuentra el hecho de ser multiplataforma, por lo que corre bajo UNIX, LINUX y Windows sin problemas. Y, además posee una rapidez y facilidad de uso. Existen infinidad de librerías y otras herramientas que permiten su uso a través de gran cantidad de lenguajes de programación, como son: C, C++, Java, Perl, PHP, Python y TCL, También cabe decir que tiene la opción de protección por contraseña, muy flexible y a la vez segura, la cual ha sido utilizado para la presente aplicación Web como mecanismo de protección de contraseñas de usuario (HASH MD5).

Para más información Ver Anexo2

[10] GPL(General Public License, Licencia Pública General ) mas información http://es.tldp.org/Otros/gples/gples.html

[11] GNU (proyecto de desarrollado de software libre llamado ``GNU'' ) mas información http://es.wikipedia.org/wiki/GNU

**CAPITULO 3**

# MARCO METODOLÓGICO

La función de un modelo de ciclo de vida es establecer un órden en que se especifica, se realizan los prototipos, se implementa, revisa, prueba y se realizan otras actividades dentro de un proyecto. Se deben establecer ciertos criterios para que sean utilizados para determinar el paso de una actividad a otra.

En este capitulo se explica brevemente las diferentes metodologías que pueden ser usadas para el desarrollo de software, como son :

- Cascada
- DRA (desarrollo rápido de aplicaciones)
- Prototipado
- Espiral
- Entrega por etapas
- Proceso unificado

## 3.1 INTRODUCCIÓN

La metodología usada para todo tipo de proyectos es muy importante, porque puede ser definitiva para cumplir con los objetivos propuestos. En el grupo TRIONIX se ha venido utilizando el Proceso Unificado de Desarrollo de Software porque proporciona múltiples herramientas que facilitan la distribución de actividades del equipo de desarrollo, y especifica correctamente los productos dentro del proyecto. Sin embargo se describirán diferentes metodologías haciendo un paralelo entre ellas buscando la mejor alternativa que se adapte a las nuevas necesidades. Para esto se construirá una matriz de decisión que evalué los ítems más relevantes del negocio, y que al final muestre cual de las alternativas satisface los requerimientos, en pro de un mejor producto.

## 3.2 CICLOS DE VIDA DEL DESARROLLO SOFTWARE

### 3.2.1 CASCADA PURA

El predecesor de todos lo modelos de ciclo de vida es el modelo en cascada. En un modelo en cascada, un proyecto progresa a través de una secuencia ordenada de pasos partiendo del concepto inicial del software hasta la prueba del sistema. El proyecto realiza una revisión al final de cada etapa para determinar si está preparado para pasar a la siguiente etapa.

Este modelo en cascada está dirigido por documentos. Los productos principales del trabajo que se pasan entre cada etapa son los documentos.
El modelo en cascada pura es utilizado para ciclos de productos los cuales tienen una definición estable del producto. Además funciona bien con proyectos complejos que se entienden correctamente. El modelo en cascada ayuda a localizar errores en las primeras etapas del proyecto a un bajo costo, además minimiza los gastos de la planificación por que permite realizarla sin problemas.

Sus desventajas se encuentran en la dificultad que tiene para especificar claramente los requerimientos al comienzo del proyecto, antes de que se realice ningún diseño y antes de escribir ningún código.

### 3.2.2 DRA (DESARROLLO RÁPIDO DE APLICACIONES)

El modelo DRA es una adaptación a alta velocidad del modelo en cascada en el que se logra el desarrollo rápido utilizando un enfoque de construcción basado en componentes reutilizables y utilizando técnicas de cuarta generación en lugar de software con lenguajes de programación de tercera generación.

En proyectos donde se comprenden bien los requisitos, se limita correctamente el ámbito del proyecto, y el sistema se puede dividir en módulos, el proceso DRA permite al equipo de desarrollo crear un sistema completamente funcional dentro de periodos cortos de tiempo (60-90 días).

### 3.2.3 PROTOTIPADO

#### 3.2.3.1 PROTOTIPADO EVOLUTIVO

El Prototipado evolutivo es un modelo que toma sus bases del Prototipado desechable, pero posee mayores controles sobre la calidad y desarrolla primero las áreas de mayor riesgo del sistema, de tal forma que el prototipo pueda ser tomado como producto final una vez se llegue a su final. Es decir, en éste modelo se desarrolla el concepto del sistema a medida que avanza el proyecto. El prototipo evolutivo es un enfoque donde se desarrolla primero las partes a seleccionar del sistema y luego el resto a partir de estas partes. A diferencia de otros tipos de prototipos, en el evolutivo no se descarta el código del prototipo; lo transforma en el código entregado finalmente. El desarrollo de prototipos continúa hasta que se decide que el prototipo es lo suficientemente bueno y se puede entregar como producto final.

Este modelo genera signos visibles de progreso, pero se puede correr el riesgo de caer en el esquema de codificar y corregir; sin ninguna planificación, ni gestión.

#### 3.2.3.2 ENTREGA POR ETAPAS O MODELO INCREMENTAL

En éste modelo no se entrega el producto total al final del proyecto, sino que se muestra al cliente en etapas refinadas sucesivas proporcionando una funcionalidad útil antes de entregar el 100% del proyecto. Primero se realiza la definición del concepto del software, el análisis de requerimientos y la creación del diseño global de una arquitectura como en el modelo en cascada.

Este modelo no funciona sin una planificación adecuada tanto para niveles técnicos como para niveles de gestión. En un nivel de gestión, se debe asegurar que las etapas que se planifican son significativas para el cliente y que el trabajo se distribuye entre el personal del proyecto de tal forma que pueden completar su trabajo a tiempo. En un nivel técnico, hay que asegurarse de que se han tenido en cuenta todas las dependencias técnicas entre diferentes componentes de un producto para evitar retrasos por dependencias no previstas.

### 3.2.4 ESPIRAL

"El modelo de espiral es un modelo de ciclo de vida orientado a riesgos que divide un proyecto software en mini proyectos. Cada mini proyecto se centra en uno o más riesgos importantes hasta que todos estos estén controlados". El concepto "riesgo" puede referirse a requerimientos o arquitecturas poco comprensibles, problemas de ejecución importantes o con la tecnología subyacente.

El método parte de una escala pequeña en medio de la espiral, se localiza los riesgos, se genera un plan para manejarlos, y a continuación se establece una aproximación a la siguiente iteración. Después de controlar todos los riesgos más importantes, el modelo en espiral finaliza con un ciclo de vida en cascada, o con Prototipado o con otro modelo. También se puede incorporar otros modelos de ciclo de vida en sus iteraciones.

### 3.2.5 PROCESO UNIFICADO

Es un proceso de desarrollo de software que proporciona normas para el desarrollo eficiente de software de calidad dentro de plazos y presupuestos planeados. Entendiendo proceso de desarrollo de software como el conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema software.

El proceso unificado está dirigido por casos de uso, centrado en la arquitectura y es iterativo e incremental.[12]

[12] Steve McConnell. *Desarrollo y Gestión de Proyectos Informáticos*. McGraw Hill, 1996.

**Dirigido por casos de uso**: el desarrollo de software se centra en la importancia del desarrollo para el usuario y no en términos de funciones que debe cumplir el sistema. Los casos de uso dirigen el proceso durante todos los flujos de trabajo de las distintas fases.

**Centrado en la arquitectura:** Al describir la arquitectura se debe obtener una mayor comprensión del sistema, se organiza el desarrollo y se fomenta la reutilización. Esta arquitectura abarca la organización del software, los elementos estructurales que compondrán el sistema y sus interfaces, así como su comportamiento y colaboraciones entre elementos.

**Es iterativo e incremental:** Un proceso iterativo permite una comprensión creciente de los requerimientos, a la vez que se va haciendo crecer el sistema abordando las tareas más riesgosas primero. El trabajo de desarrollo se divide de manera planeada en partes más pequeñas llamadas iteraciones lo cual genera progresivamente un incremento en el proyecto total. Si una iteración cumple con sus objetivos el desarrollo continuo con la siguiente iteración, en caso contrario se revisa las decisiones previas y se prueba un nuevo enfoque.

Como se dijo anteriormente, el proceso unificado divide el proceso de desarrollo en ciclos el cual se divide en cuatro fases: inicio, elaboración, construcción y transición. Cada una de estas fases concluye con un hito bien definido donde deben tomarse decisiones respecto al proyecto como la reestructuración del cronograma de trabajo. Cada una de estas fases se divide a su vez en iteraciones. Cada iteración sigue la estructura de un pequeño ciclo de vida en cascada, pasando a través de los cinco flujos de trabajo fundamentales: requisitos, análisis, diseño, implementación y prueba. En la iteración también incluye la planificación que precede a los flujos de trabajo y la evaluación que va detrás de ellos.

## 3.3 SELECCIÓN DEL CICLO DE VIDA

Distintos proyectos tienen necesidades diferentes, incluso todos necesitan ser desarrollados lo más rápido posible.

No existe un modelo de ciclo de vida de desarrollo rápido debido a que el modelo más efectivo depende del contexto en que se utilice. Un modelo de ciclo de vida que a menudo trabaja bien, puede suceder que no funcione bien si no se utiliza correctamente.

Para seleccionar el modelo de ciclo de vida más conveniente para el proyecto, se debe examinar éste y responder a las siguientes preguntas:

- ¿Me compenetro con el cliente para la especificación de los requerimientos al comienzo del proyecto?
- ¿Comprendo bien la arquitectura del sistema? Es probable que necesite llevar a cabo modificaciones importantes en la arquitectura a mitad del proyecto?

- ¿Cuánta fiabilidad necesito?
- ¿Cuánto tiempo extra necesito para planificar y diseñar durante el proyecto, para las versiones futuras?
- ¿Cuántos riesgos conlleva el proyecto?
- ¿Si necesito puedo realizar modificaciones a medio camino?
- ¿Cuánta sofisticación necesito para utilizar el modelo de ciclo de vida con éxito?

Debido a la gran variedad de aspectos involucrados para el desarrollo del software, es necesario definir ciertos aspectos relevantes para seleccionar la mejor alternativa e implementarla para el desarrollo del software. Estos aspectos son:

A. Grado de identificación de los requerimientos.
B. Comprensión de la arquitectura a utilizar.
C. Grado de fiabilidad del sistema.
D. Grado de desarrollo en la generación del sistema.
E. Nivel de manejo de riesgos.
F. Estado de la planificación del proyecto.
G. Tiempo necesario en la gestión.
H. Existencias de modificaciones durante el transcurso del proyecto.
I. Nivel de sofisticación para directivos y desarrolladores.

Para evaluar dichos aspectos se utilizó la siguiente matriz de decisión[13]:

| CAPACIDAD DEL MODELO[14] | CASCADA PURA | DRA | PROTOTIPO EVOLUTIVO | ESPIRAL | INCREMENTAL | PROCESO UNIFICADO |
|---|---|---|---|---|---|---|
| A: alto | 10 | 10 | 0 | 0 | 10 | 10 |
| B: alto | 10 | 10 | 8 | 0 | 8 | 10 |
| C: alto | 10 | 5 | 5 | 10 | 10 | 10 |
| D: alto | 10 | 6 | 10 | 10 | 10 | 10 |
| E: medio | 2 | 10 | 6 | 10 | 7 | 10 |
| F: definido | 5 | 10 | 0 | 5 | 6 | 10 |
| G: medio | 2 | 5 | 6 | 6 | 7 | 8 |
| H: medio | 2 | 5 | 10 | 6 | 8 | 6 |
| I: alto | 2 | 10 | 8 | 10 | 9 | 10 |
| **TOTALES** | 53 | 71 | 53 | 57 | 75 | **84** |

**Tabla 3.1** Comparación ponderada de los modelos de ciclo de vida

Según los resultados se ve claramente que el puntaje mayor es la metodología del Proceso Unificado de Desarrollo, por tal razón se opta por seguir usando

[13] Calificación de 0 a 10 puntos, siendo 10 el puntaje máximo.
[14] En esta columna se especifica el numeral correspondiente a la característica evaluada y frente a él se coloca el valor correspondiente para este proyecto.

dicha metodología, por suplir las necesidades del negocio, y que al final conlleva a un producto software adecuado.

### 3.3.1 DESCRIPCIÓN METODOLOGÍA SELECCIONADA

El proceso unificado de desarrollo software utiliza el lenguaje unificado de modelado (UML), para preparar todos los esquemas de un sistema software. Además permite llevar a cabo el proyecto de una manera eficiente en términos de costo calidad y tiempo, cumpliendo con el cronograma y presupuesto pactados.

Como ya se dijo, el proceso unificado esta basado en componentes, dirigido por casos de uso, centrado en la arquitectura y es iterativo e incremental.

- ***Basado en componentes*:** el software a desarrollar esta formado por componentes software que son partes físicas del sistema y que pueden ser reemplazadas.

- ***Dirigido por casos de uso*:** un caso de uso es un conjunto de acciones que lleva a cabo el sistema con el fin de proporcionar al usuario un resultado importante y representa un requisito funcional. Los casos de uso guían el proceso a través de todos los flujos de trabajo y permiten desarrollar el sistema en función de las necesidades del usuario; además guían la arquitectura del sistema quien a su vez influye en la selección de los casos de uso.

- ***Centrado en la arquitectura*:** la arquitectura incluye los aspectos más significativos del sistema: su estructura, comportamiento, restricciones, plataforma en la que debe funcionar el software, sistemas heredados, reutilización de componentes y requisitos no funcionales; debe permitir el desarrollo de todos los casos de uso requeridos y estos deben encajar en la arquitectura cuando se llevan a cabo. Los Casos de Uso presentan relación con la Arquitectura, puesto que cada producto tiene tanto una función (Caso de Uso) como una forma (Arquitectura), ambas deben equilibrarse si se desea lograr un producto de éxito; con base en esto, los casos de uso y la arquitectura deben evolucionar en paralelo.

El desarrollo software complejo implica un aumento en el esfuerzo, por esta razón es práctico dividir el desarrollo en pequeñas partes. En el Proceso Unificado de Desarrollo cada entrega corresponde a una iteración, la cual finaliza en un incremento (crecimiento del producto).Cada iteración se realiza con un ciclo en cascada, es decir, se manejan cuatro fases: análisis, diseño, implementación y prueba, al finalizar se realiza una evaluación de los objetivos, si estos se cumplen se continúa con la siguiente iteración, de lo contrario es conveniente replantear el enfoque con el que se está trabajando.

Es de vital importancia seleccionar primero los casos de uso más relevantes con el fin de construir una arquitectura robusta en las primeras iteraciones que soporte el sistema de una manera eficiente. Se deben seleccionar y programar solo las iteraciones necesarias para lograr los objetivos del proyecto y desarrollar un proceso iterativo controlado.

Cada vez que se hace un incremento en el sistema, se debe validar la funcionalidad del incremento y de los demás componentes desarrollados en ese momento. El proceso unificado reduce el riesgo de retrasos en el calendario previsto, acelera el ritmo de desarrollo e involucra al usuario en todas las fases refinando los objetivos que dirigen el trabajo en cada iteración.

El enfoque iterativo e incremental, guiado por los casos de uso y centrado en la arquitectura, permite una comprensión creciente de los requerimientos, al mismo tiempo que el sistema crece, por esta razón, presenta sustanciales beneficios, puesto que reduce el costo de los riesgos, al costo de un solo incremento, a su vez, logra tener un sistema ejecutable mucho más pronto.

El proceso unificado de desarrollo de software se repite durante la vida del sistema a manera de ciclos que constituyen cada uno una versión del producto. Cada ciclo se subdivide en cuatro fases (inicio, elaboración, construcción, transición) que a su vez se subdividen en iteraciones que se desarrollan a lo argo de 5 flujos de trabajo fundamentales: **requisitos, análisis, diseño, implementación.** Como se observa en la siguiente figura.

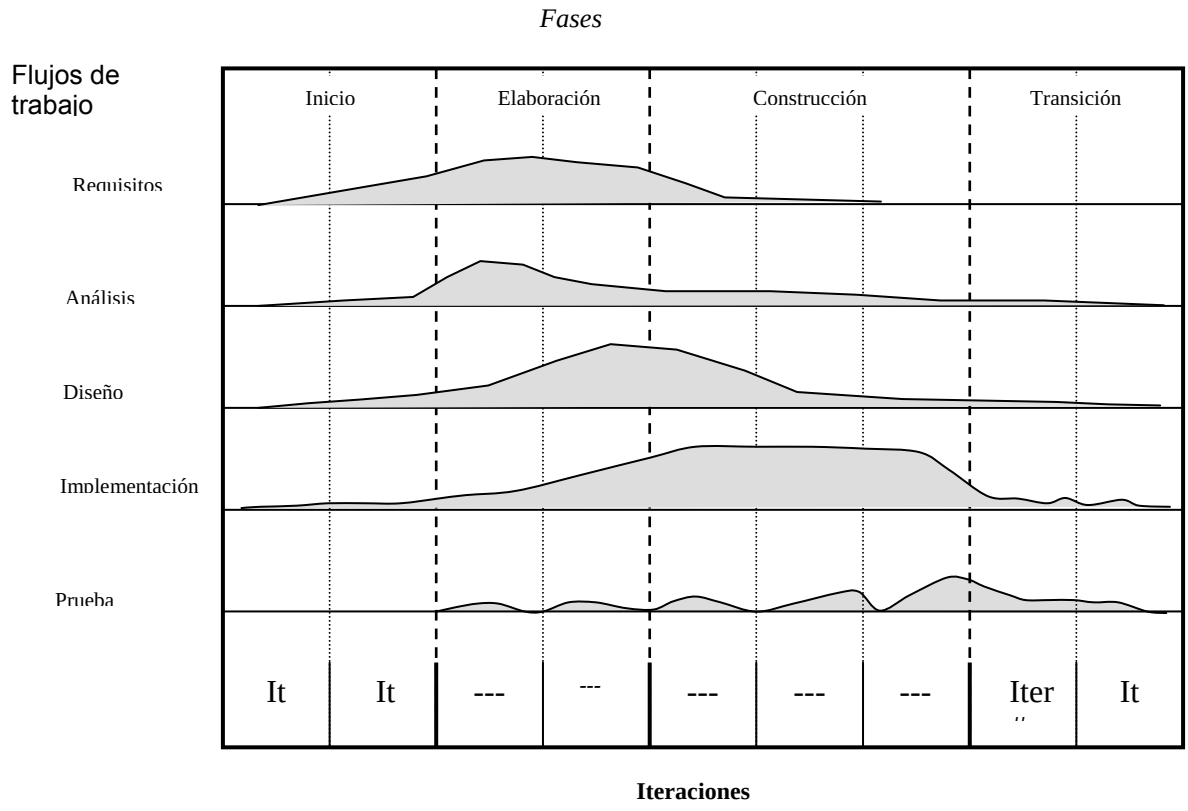



**Figura 3.1** Flujos de trabajo

otro aspecto relevante de esta metodología es la correcta organización del tiempo que permite cumplir con los plazos planeados, es decir la división del proyecto en iteraciones o mini-proyectos los cuales traen consigo muchos beneficios dentro de los que están, una reducción de costos por desarrollos no óptimos, pues efectuar entregas en las fechas estipuladas evita que la mayoría de los problemas no se presenten en la fase final como suele ocurrir cuando se siguen otras metodologías, sino que los riesgos se logran identificar en etapas muy tempranas.
Otra característica es la realización de tareas simultáneas, es decir se puede ir recolectando información documentada del tema de investigación, mientras se captura el muestreo de datos.

## 3.4 PLAN DE TRABAJO

Para la elaboración del proyecto se seguirá la metodología planteada por el Proceso Unificado de Desarrollo Software. Comprende las siguientes fases: **Inicio, Elaboración, Construcción y Transición**. Cada una de estas fases estará subdivida en iteraciones, las cuales constituyen pequeños ciclos de vida dentro de los cuales se recorrerán los flujos de trabajo: **Requisitos, Análisis, Diseño, Implementación y Prueba**.

A continuación se detalla de manera general cada una de las fases de desarrollo con sus respectivas actividades, mencionando los aspectos más relevantes para la construcción de la aplicación Web y las labores que requieren más atención en cada flujo de trabajo. Las actividades mencionadas pueden llegar a ampliarse o reducirse según las dificultades encontradas a lo largo del proyecto.

### 3.4.1 FASE DE INICIO

La fase de inicio tiene como objetivo desarrollar una descripción del producto final a partir del objetivo principal del proyecto y hacer un análisis de la problemática que permita determinar cuales son las principales funciones que debe realizar el sistema.
Las actividades que se llevaran a cabo en esta fase son las siguientes:

- Realizar una búsqueda de información en el área de seguridad informática para poder conocer las herramientas existentes en el mercado que buscan solucionar la problemática presentada. De esta manera se espera poder ubicarnos de una manera mas precisa en el contexto en el cual se desarrollan las soluciones.
- Estudiar el lenguaje UML, utilizado como soporte al Proceso Unificado de Desarrollo de Software.
- Establecer los requisitos fundamentales, empezando por los más representativos.

- Establecer el modelo de casos de usos iniciales.
- Definir el ámbito y los límites del sistema.
- Desarrollar un prototipo que muestre el funcionamiento del sistema.
- Descripción de la arquitectura candidata.

### 3.4.2 FASE DE ELABORACIÓN

En esta fase la labor principal es seleccionar una arquitectura estable con el fin de construir el sistema dentro de un marco de trabajo económico y planificar adecuadamente las labores de construcción.
Las actividades a realizar en esta fase son:

- Construir el modelo de análisis a partir del modelo de Casos de Uso.
- Crear una base para la arquitectura que cubra la funcionalidad del sistema.
- Indagar sobre los lenguajes de programación existentes para poder definir cual se adapta más a la idea a desarrollar.
- Recopilar la mayor parte de los requisitos pendientes de la fase de Inicio.
- Definir el diseño de la base de datos, que contendrá la información acerca de la aplicación protegida y sus implicaciones.

### 3.4.3 FASE DE CONSTRUCCIÓN

El propósito de esta fase consiste en desarrollar el producto software que implemente la funcionalidad del sistema que fue establecido en las fases anteriores, y plasmado en la arquitectura. El proceso de desarrollo software se realizar de manera iterativa e incremental, de tal forma que a través del tiempo se generan prototipos que garantizan el cumplimiento e implementación de las alternativas que satisfacen los requisitos establecidos.
Ésta fase cubre las siguientes actividades:

- Probar diferentes algoritmos criptográficos buscando rapidez en función de seguridad e implementar la mejor alternativa.
- Implementar la base de datos que contendrá la información relacionada con las aplicaciones protegidas.
- Construir una interfaz que le permita al usuario interactuar con la aplicación Web.
- Realizar pruebas que garanticen el correcto funcionamiento de la aplicación Web.
- Integrar el sistema completo.

### 3.4.4 FASE DE TRANSICIÓN

En esta fase se realizan las pruebas de la versión obtenida en la fase de construcción; inicialmente se realiza una prueba con usuarios expertos en pruebas de software, quienes dan conocer sus deficiencias y defectos.

Del resultado de todo este proceso podemos descubrir si la herramienta cubre las expectativas del usuario, si hay riesgos inesperados, además podemos tener en cuenta los problemas no resueltos, eliminar ambigüedades y centrarnos en las áreas donde el usuario muestra deficiencias en cuanto al uso. Las actividades que se realizaran en esta fase son las siguientes:

- Determinar si el sistema hace lo que demanda el usuario.
- Descubrir problemas no resueltos y encontrar fallas no detectadas.

Ver Anexo 3 Pruebas usuarios finales.

**PARTE II**

## PROCESO UNIFICADO DE DESARROLLO

El proceso unificado de desarrollo software recopila y detalla el conjunto de actividades necesarias para transformar los requisitos de usuario en un sistema software; dicho sistema debe realizar con éxito lo que sus futuros usuarios necesitan. Por lo tanto se debe llevar un proceso planificado, ordenado y evolutivo que permita finalmente lograr los objetivos.

Este proceso se centra en la arquitectura donde se incluyen los aspectos estáticos y dinámicos significativos del sistema que surgen de las necesidades del negocio, los cuales se reflejan en los casos de uso. Sin embargo, estos aspectos están acompañados de otros factores como son: plataforma software, arquitectura hardware, sistema operativo, sistema de gestión de base de datos y protocolos de comunicación en red, los cuales con una serie de ciclos constituyen la vida de un sistema.

Cada ciclo concluye en una versión del producto para los usuarios. Cada uno de estos, se desarrolla a lo largo del tiempo y se dividen en fases, con una secuencia de modelos que visualizan lo que esta sucediendo en las fases (Inicio, Elaboración y Construcción), cada una con objetivos a cumplir.

Los objetivos específicos de esta parte son:

**Fase Inicio.**

- Desarrollar una descripción del producto final a partir de la identificación y análisis de requisitos.
- Desarrollar el análisis del negocio es busca de justificar la ejecución del proyecto.

**Fase Elaboración**

- Especificar el mayor porcentaje de casos de uso del producto.

**Fase Construcción**

- Completar la línea base hasta transformarla en una versión operativa del sistema.

**CAPITULO 4**

**FASE DE INICIO**

## 4.1 PLANIFICACIÓN

El objetivo principal de esta fase es definir el alcance del sistema propuesto e identificar las características y funciones que tendrá la aplicación Web en desarrollo, las cuales deberán satisfacer los requerimientos de los usuarios.

Luego de desarrollar los flujos de trabajo correspondientes a esta fase se espera tener una parte de los requisitos del proyecto bien definidos, así como los riesgos del negocio con su plan de contingencia que permita constituir un esquema estructurado, que sirva como arquitectura candidata y pueda soportar el ámbito del sistema.

## 4.2 FLUJOS DE TRABAJO

Los flujos de trabajo de esta fase giran entorno a dos grupos de actividades como son: la determinación del ámbito del sistema y la comprensión de la arquitectura candidata. Los participantes del proyecto deben adquirir varios roles en esta fase.

### 4.2.1 FLUJO DE TRABAJO DE LOS REQUISITOS

- **Actividad: enumerar los requisitos candidatos**

A continuación se presenta una primera aproximación al listado de requisitos de la aplicación Web:

**Lista de Requisitos**

- ✓ El sistema permitirá la administración de los clientes por parte de los usuarios.
- ✓ El sistema debe realizar todas las operaciones sin tener acceso al código fuente de la aplicación a proteger.

- ✓ El sistema debe permitir renovar la protección por un tiempo definido por el usuario o dueño de una aplicación.

- ✓ El sistema debe permitir renovar la protección por una cantidad de usos definidos por el usuario.

- ✓ El sistema debe permitir la restricción de accesos a la totalidad de opciones.

- ✓ El sistema debe proporcionar mecanismos de seguridad para la protección de contraseñas y permitir su variación.

- ✓ El sistema debe contar con un canal de comunicación entre todos los actores operativos (correo).

- ✓ El sistema permitirá la publicación y descarga de aplicaciones protegidas.

- ✓ Permitir al administrador del sistema llevar información relacionada con las aplicaciones protegidas mediante una base de datos, además de contar con estadísticas que en un momento dado se constituyan en base para la toma de decisiones en pro de un mejoramiento.

- ✓ El sistema debe proporcionar a la aplicación protegida los mecanismos necesarios para desactivarse en caso de vencimiento o que se haya tratado de vulnerar (retroceder la fecha del sistema).

- **Actividad: Identificación de riesgos críticos**

La siguiente tabla muestra cada uno de los riesgos a los que puede estar sometido el producto que se esta desarrollando asociando sus impacto, responsables y contingencias

| *Descripción* | *Nivel de riesgo* | *Impacto* | *Monitor* | *Responsable* | *Contingencia* |
|---|---|---|---|---|---|
| **Incumplir con los tiempos y objetivos planificados** | Critico | Global | Autores | Autores y director del proyecto | Adaptar los mecanismos de control de la gestión |
| ***Problemas con la utilización de DLLS por la variación de plataforma*** | Significativo | Global | Autores | Autores | Re-implementación de los procesos de renovación y protección |
| ***Permanencia operativa del servidor Apache Tomcat*** | Crítico | Global | Autores | Autores | La EISI deberá contar con un servidor de respaldo con similares especificaciones técnicas y al titu actualizado que permita el continuo funcionamiento la aplicación. |
| ***Capacidad de usuarios concurrentes del servidor Web y de acceso a la base de datos.*** | Crítico | Global | Autores | Administrador de la Aplicación | Si en un momento dado el sistema llegase al tope de usuarios, se notaria con una notable demora en los procesos y operaciones de la aplicación, que iría mejorando a medida que los usuarios abandonan el sistema. |

**Tabla 4.1** Riesgos Críticos y significativos

- **Actividad: comprender el contexto del sistema**

✓ **Descripción del contexto**

Debido a la dependencia que tienen los usuarios con el administrador para realizar cualquier operación (consulta, modificación…) y la imposibilidad para administrar directamente su software, surge la idea de automatizar los procesos de renovación y consulta, permitiendo así la administración de aplicaciones protegidas por parte de los usuarios (ver tabla 4.2) mediante la implementación de la aplicación Web.

- **Actividad: comprender el contexto del sistema**

✓ **Identificación y Descripción de Actores**

El sistema maneja la administración de perfiles de usuario y los clasifica según su relevancia en la aplicación en: Administrador, usuario, cliente y visitantes, cada uno de los cuales con privilegios de acuerdo a su perfil.

La siguiente tabla describe a cada uno de los actores participantes en la aplicación Web en desarrollo.

| ***Actor*** | ***Descripción*** | ***Responsabilidades*** | ***Necesidades(funcionamiento)*** |
|---|---|---|---|
| **Administrador** | Representa la persona encargada de la administración del sistema, quien protege aplicaciones y celebra contratos de administración de aplicaciones protegidas con los usuarios. | **1.** Mantenimiento de Usuarios.<br>**2.** Proteger Aplicaciones.<br>**3.** Mantenimiento módulo de renovación.<br>**4.** Mantenimiento módulo de consulta.<br>**5.** Mantenimiento módulo aplicaciones protegidas.<br>**6.** Analizar estadísticas y a partir de estas realizar correcciones o complementos con respecto al uso de la aplicación Web. | El administrador utiliza la aplicación para:<br><br>**1.** La creación, modificación y eliminación (inactivar) de usuarios y aplicaciones.<br>**2.** Proteger proyectos por días o número de ejecuciones.<br>**3.** Celebrar contratos con los usuarios para que ellos puedan divulgar su aplicación a terceros y obtengan beneficios por su trabajo. |

| | | | |
|---|---|---|---|
| **Usuario** | Persona o entidad propietaria patrimonial de aplicaciones software protegidas. | **1.** Mantenimiento de sus Clientes.<br>**2**. Mantenimiento de sus proyectos.<br>**3.** Renovar aplicaciones solicitadas por sus clientes en un periodo menor al pactado en el contrato de administración celebrado con el administrador del sistema.<br>**4.** Otorgar permisos a sus clientes para renovar proyectos. | **1.** El sistema de acuerdo a su perfil le brinda las opciones relacionadas al mantenimiento de clientes.<br>**2.** Recepcionar solicitudes de sus clientes o otros interesados en alguna de sus aplicaciones protegidas.<br>**3.** Conocer el comportamiento de sus clientes por medio de las estadísticas que arroja el sistema. |
| **Cliente** | Persona u organización que usa aplicaciones protegidas por la herramienta TRIONIX | **1.** Renovar o Solicitar renovación de una aplicación al usuario, quien según el caso lo clasificara en: Cliente Automático o Cliente No Automático.<br>**2.** Consultar información referente a las aplicaciones que usa o a las que no y le interese usar. | **1.** Renovar con previa autorización del usuario.<br>**2.** Consultar y ver información general o específica de aplicaciones protegidas.<br>**3.**Descargar proyectos y usarlos. |
| **Visitante** | Representa a cualquier persona que ingrese a la pagina principal de la aplicación interesada o no en usar aplicaciones software protegidas. | **1.** Ver información general de la aplicación Web.<br>**2.** Consultar información general de desarrolladores y proyectos.<br>**3.** Descargar aplicaciones protegidas a usar durante un periodo determinado de tiempo o accesos.<br>**4.** Renovar para el caso de aplicaciones libres o emitir solicitud al usuario responsable de la aplicación protegida. | **1.** Descargar y Usar aplicaciones protegidas.<br>**2.** Renovar o solicitar renovación.<br>**3.** Consultar. |

**Tabla 4.2** Actores del Sistema

- **Casos de uso y actores de los módulos de servicio de la aplicación Web**

En la siguiente tabla se presentan los casos de uso identificados en esta fase, y los actores que intervienen en cada uno de ellos.

| | Casos de Uso | Actores |
|---|---|---|
| 1 | **Protección** | |
| 1.1 | Proteger aplicaciones | Administrador |
| 1.2 | Crear Usuario | Administrador |
| 1.3 | Actualizar Usuario | Administrador |
| 1.4 | Contraseñas usuario | Administrador |
| 2 | **Renovación** | |
| 2.1 | Solicitud-Respuesta Renovación | Cliente-Usuario |
| 2.2 | Renovación | Cliente-Usuario |
| 2.3 | Crear Cliente | Usuario |
| 2.4 | Actualizar Cliente | Usuario |
| 2.5 | Contraseñas Cliente | Usuario |
| 3 | **Consulta** | |
| 3.1.a | Consultar información General | Todos |
| 3.1.b | Consultar información Especifica | Todos excepto visitantes. |
| 4 | **Aplicaciones Protegidas** | |
| 4.1 | Descargar aplicaciones | Todos |
| 4.2 | Renovar | Cliente (cuando el usuario le ha otorgado el permiso para hacerlo), usuario (aplicaciones con restricciones de uso). |
| 4.3 | Solicitud de renovación | Cliente o usuario a usuario si es el caso de aplicaciones con restricciones de uso. |

**Tabla 4.3** Casos de uso

- **Descripción de Módulos de Servicio**

### 1. Descripción del Módulo de servicio Protección

Este módulo es exclusivo del administrador del sistema, en donde podrá proteger aplicaciones software atendiendo la demanda de los usuarios; la función del administrador además de proteger es el mantenimiento de usuarios, lo cual involucra: crear, modificar y borrar, así como el mantenimiento de la aplicación Web.
El proceso de protección no está disponible en la Web, sino que se realiza de manera personal, donde tanto administrador como usuario celebran un contrato de uso con condiciones de tiempo, que autorizan la administración de clientes por parte de los usuarios, los cuales tomarán dominio exclusivo de sus aplicaciones y clientes durante el periodo convenido.

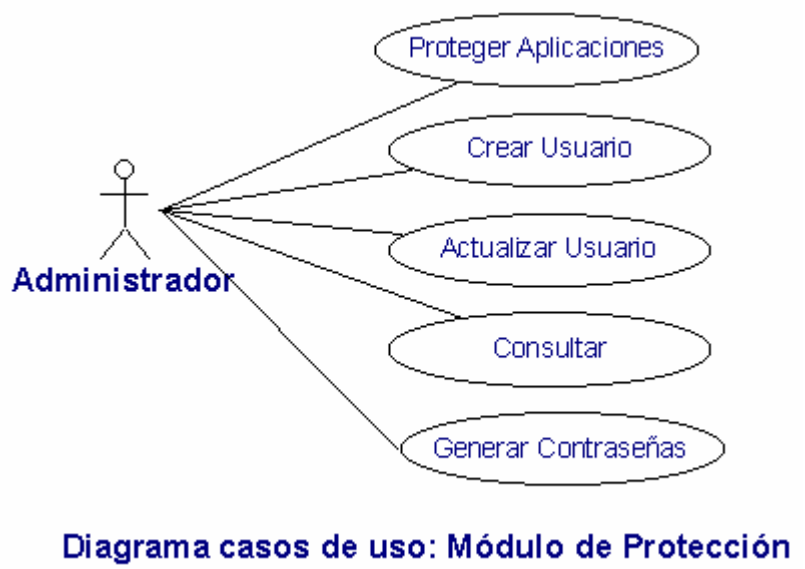



**Figura 4.1** Caso de uso Protección de Aplicaciones

### 1.1 Detalle del caso de uso Proteger Aplicaciones

**Precondición:** Celebración contrato Administrador-Usuario.

**Camino Básico:**
Una vez celebrado el contrato entre las partes (administrador-usuario) el módulo de protección se dispone para la protección de aplicaciones software solicitando inicialmente información de interés del usuario, la cual se validará para evitar errores e inconsistencia en los datos, una vez culminado este proceso el administrador le otorgará al usuario un login y contraseña para que acceda a la aplicación Web, luego se remitirá a proteger la aplicación. Los procedimientos de este módulo son exclusivos del administrador del sistema.

FIN

### 1.2 Detalle del caso de uso Crear Usuarios

**Precondición:** El administrador debe haber ingresado al sistema con su contraseña, la cual será validada verificando con la información recopilada en la Base de datos, si son correctos se presentará la interfaz de la aplicación que no se dispone en la Web, solicitando el ingreso de un usuario siempre y cuando este tenga una aplicación a proteger y haya celebrado un contrato de administración.

**Camino Básico:**
**Captura de Datos:** el administrador llena los campos con información del usuario y su aplicación como: nombre, organización, dirección, teléfono, e-mail, entre otros.

**Validación de los Datos:** los datos suministrados son validados por el sistema con el fin de evaluar que todos los campos considerados como de obligatorio llenado hayan sido suministrados, esta opción se realiza antes de guardar la información.

**Registro de Usuario:** en este estado la solicitud de creación de usuario es registrada en cada uno de los correspondientes campos de la base de datos, generando como respuesta la activación de la opción de proteger.
FIN
Detalle del caso de uso Actualizar Usuarios

**Precondición:** el administrador debe haber ingresado a la aplicación Web con su login y contraseña, y estos deben ser correctos.

**Camino Básico:**

**Captura de Datos:** el administrador selecciona la opción Mantenimiento de usuarios, en donde elige al usuario que desea actualizar e inmediatamente se visualizará toda su información disponible para ser modificada, además se dispone de la opción de inactivar un usuario, siempre y cuando haya caducado su período de administración.

**Validación de Datos:** haya o no modificaciones en la información el sistema realizara procesos de validación para evitar inconsistencias en el tipo de datos, o campos nulos.

**Registro de Datos:** luego de la validación de los datos y visto bueno, el sistema registra los datos y genera como respuesta afirmativa un mensaje de confirmación de la actualización de la información del usuario.

### 1.4 Detalle del caso de uso Contraseñas Usuarios

**Precondición:** el administrador debe haber ingresado al sistema y mediante la verificación del login de usuario y contraseña validará el acceso a la aplicación; una vez allí debe ingresar la información de un usuario y asignar a su criterio un login y password, para que el usuario administre la aplicación que protege. Esta contraseña posteriormente puede ser cambiada por el usuario.

**Camino Básico:**
**Captura de Solicitud:** en este estado el usuario llena el formulario de cambio de contraseña de acceso a la aplicación Web. Por seguridad la contraseña debe tener una longitud mayor o igual a 6 caracteres.
**Validación de los datos:** la información descrita y llenada en el formulario de contraseñas es validado por el sistema, para de esta manera evaluar que el tipo de dato es correcto y que los campos requeridos fueron suministrados.

**Registro:** una vez validado los datos correctamente, el sistema generara una respuesta afirmativa de cambio de contraseña de usuario.

FIN

**Diagramas de Estados:**

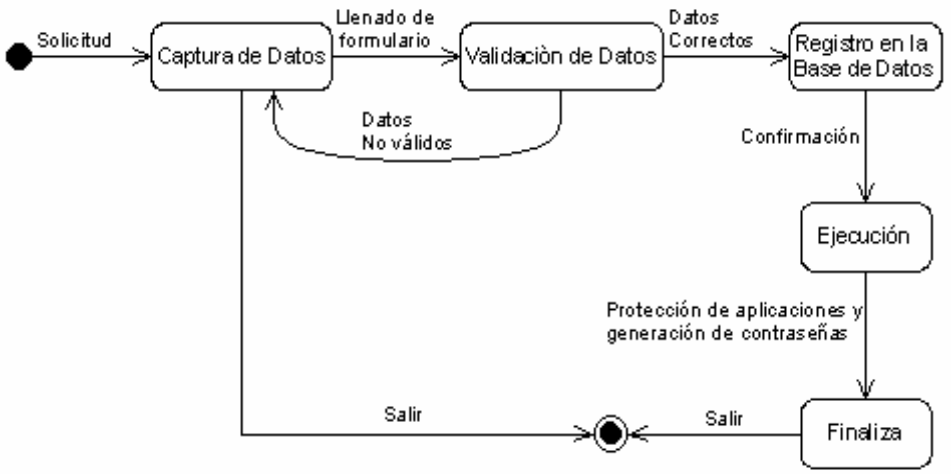



**Diagrama de Estados: caso de Uso módulo de Protección**

**Figura 4.2** Diagrama de Estado. Caso de uso Contraseñas de Usuario

**2. Descripción del Modulo de servicio Renovación**

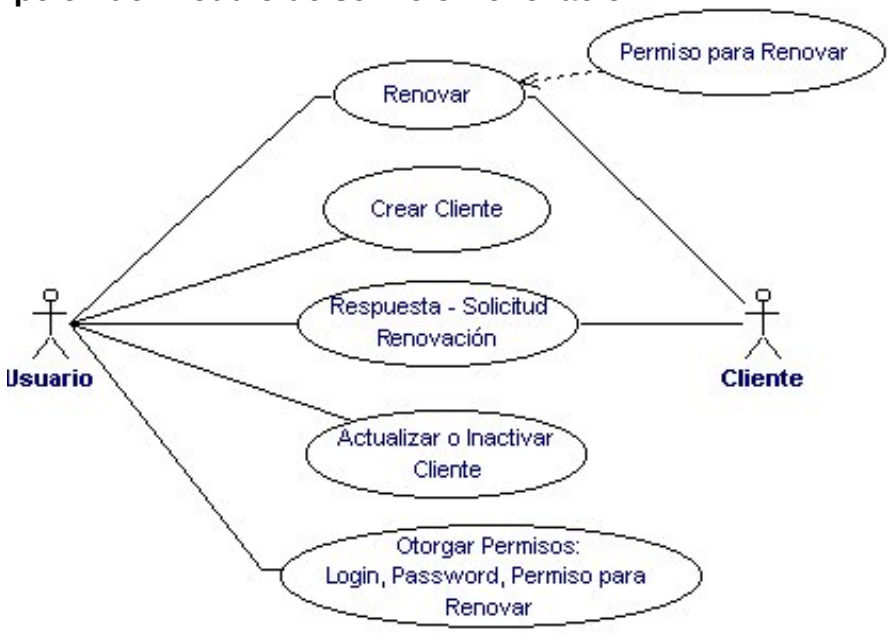



**Diagrama de Casos de uso: Módulo de Renovación**

**Figura 4.3** Diagrama Caso de uso Renovación

**Precondición:** el proceso de renovación inicia una vez caduque el periodo o el numero de accesos de una aplicación protegida, el sistema automáticamente genera una clave única (clave pública), la cual debe suministrar el cliente al usuario vía e-mail o diferente (caso Cliente No Automático y aplicación con restricción de uso), para que este ingrese al sistema con su contraseña de usuario y realice la renovación, seleccionando el cliente que realiza la solicitud y el proyecto a renovar.

**Camino Básico**:

**Envío de la Solicitud:** el cliente envía por correo u otro medio la petición de renovación de uso de una determinada aplicación al usuario, suministrándole la clave pública que genera el sistema una vez caduque la protección o la primera vez que se ejecute esta; en la solicitud además de la clave debe suministrar información acerca del número de días o accesos que el cliente desea usar la aplicación.

**Respuesta o captura de Solicitud:** Dependiendo del contrato de uso que hayan realizado el cliente y el usuario, este último dará como respuesta afirmativa la generación de la nueva clave de uso de la aplicación. Para ello ingresa al modulo de renovación y suministra la clave remitida por el cliente, la cual se validará por el sistema, si es correcta se activan las opciones de renovación y a partir de estas se generará la nueva clave de uso, que remitirá al cliente.

Si el usuario lo desea, activa al cliente para que una vez caduque la nueva protección, ingrese directamente a la aplicación Web y renueve el proyecto.

FIN

### 2.1 Detalle Caso de Uso Solicitud – Respuesta Renovación

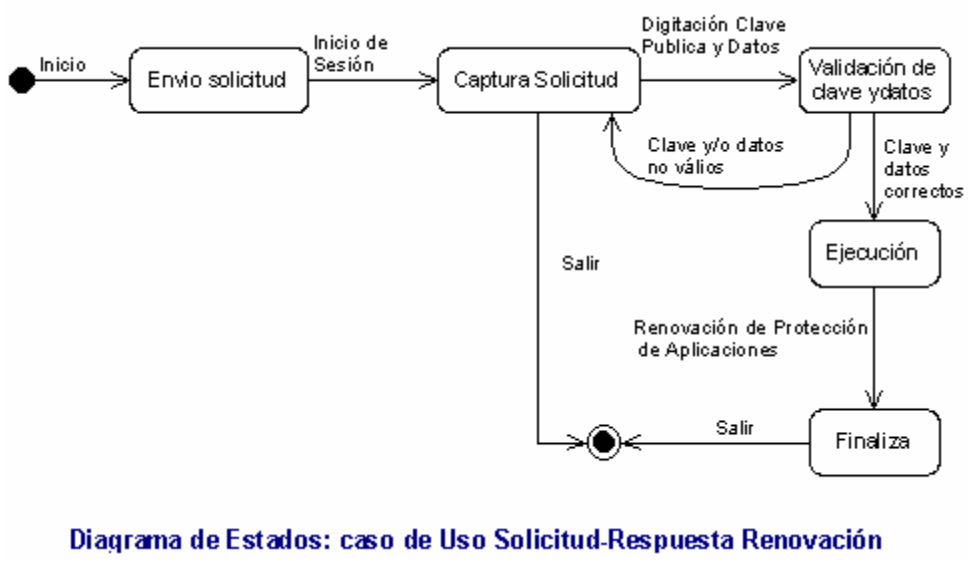



**Figura 4.4** Diagrama de Estados. Caso de uso Solicitud-Respuesta Renovación

### 2.1 Detalle Caso de Uso Renovación

**Precondición:** ingreso al modulo de renovación de la aplicación Web y suministro de la clave pública de la aplicación.

**Camino Básico:**

**Validación Clave:** Este estado se inicia cuando caduca el período de protección de una aplicación y TRIONIX genere una clave pública y solicite un serial para continuar su uso; el cliente quien inicialmente conoce esta clave, ingresará a la aplicación Web a la opción Renovar y digitará dicha clave; el sistema validará la autenticidad de esta y revisará si el cliente tiene permiso para renovar, si lo tiene generará el serial, de lo contrario deberá enviar una solicitud al usuario y el proceso a seguir es el del caso de uso: Solicitud – Respuesta Renovación.

**Registro Solicitud:** una vez finalizada la validación de la clave, se guardará en la Base de Datos para realizar validaciones; además se guardara el tipo de protección derivado del número de días o accesos seleccionados. La respuesta del proceso de renovación es la generación de la nueva clave de uso particular de la aplicación.

FIN

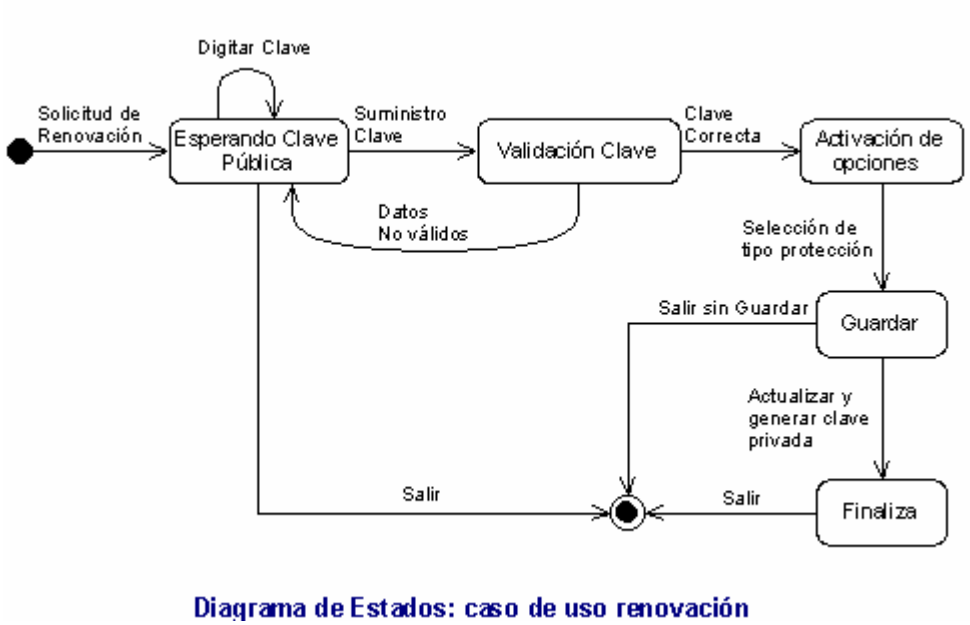



**Figura 4.5** Diagrama de Estados. Caso de uso Renovación

## 2.2 Detalle Caso de Uso Crear Cliente

**Precondición:** El usuario debe haber iniciado sesión y seleccionado la opción Mantenimiento de clientes “Nuevo Cliente”.

**Camino Básico:**

**Captura de Datos:** El usuario llena el formulario correspondiente a la creación de clientes suministrando cierta información de interés como: nombre, organización, teléfono, dirección, e-mail, login, contraseña y que proyecto(s) de su dominio va usar el cliente con sus permisos (días y accesos).

El login y contraseña otorgada al cliente en creación, le servirá a este para ingresar a la aplicación Web a consultar, enviar solicitudes y/o renovar.

**Validación de Datos:** proceso que realiza la aplicación cuando se inserte algún dato como mecanismo de control que permita evaluar la coherencia y tipo de dato.

**Registro de Cliente:** En este estado la solicitud de creación de cliente es registrada en cada uno de los campos de la Base de datos correspondientes, la respuesta que genera el sistema es un mensaje de confirmación de la inserción.

FIN

### 2.3 Detalle Caso de Uso Actualizar Cliente

**Precondición:** El usuario debe haber iniciado sesión, seleccionado la opción Mantenimiento de clientes y escogido del listado presentado el cliente a actualizar.

**Camino Básico:**

**Captura de datos:** una vez seleccionado el cliente a actualizar se desplegará toda su información que puede ser modificada o borrada (inactivación).

**Validación de los Datos:** Los datos suministrados son validados por el sistema para evitar inconsistencias en el tipo de dato o nulidades en los campos.

**Registro Solicitud:** La información validada por el sistema se registrará en la Base de datos, generando como respuesta afirmativa la activación de otras opciones o un mensaje de que el proceso ha culminado satisfactoriamente.

FIN

### 2.5 Detalle del caso de uso Contraseñas Clientes

**Precondición:**

1. El Usuario debe haber ingresado al sistema y seleccionado la opción Mantenimiento clientes, una vez allí escoge del listado el cliente a variar el login y contraseña, o bien crea uno nuevo. La contraseña del cliente sólo es conocida inicialmente por el usuario, pero una vez el cliente la cambie solo éste la conocerá.

2. El cliente debe haber iniciado sesión con el login y contraseña que le ha suministrado el usuario.

**Camino Básico:**

**Captura de Solicitud:** En este estado el cliente llena el formulario de cambio de contraseña de acceso a la aplicación Web.

**Validación de los datos:** la información descrita y llenada es validada por el sistema, para de esta manera evaluar que el tipo de dato es correcto y que los campos requeridos fueron suministrados.

**Registro de Datos:** una vez validado los datos correctamente, el sistema generara una respuesta afirmativa de cambio de contraseña.

FIN

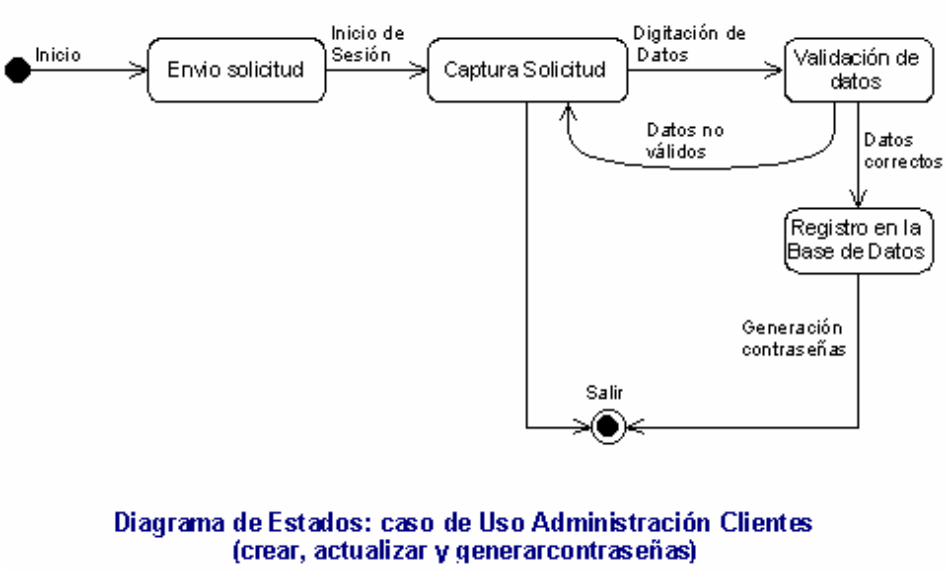



**Figura 4.6** Diagrama de Estados. Caso de uso Contraseñas Clientes

### 3. Descripción del Modulo de servicio Consulta

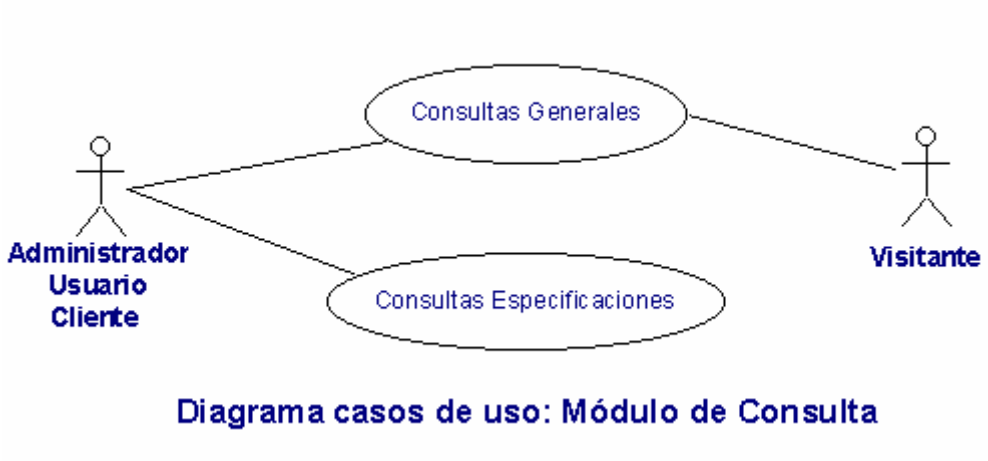



**Figura 4.7** Diagrama Caso de uso consultar

### 3.1 Detalle Caso de Uso Consultar

**Precondición:** haber iniciado sesión (caso: Administrador, usuario y cliente) o ingresado a la página principal de la aplicación (caso: Visitante).

**a) Caso de Uso Consultas Generales:**

Las consultas generales no necesitan el llenado de ninguna clase de información, sólo se cuenta con estas como mecanismo de búsqueda rápido y de dar a conocer que aplicaciones han sido protegidas y como poder acceder a ellas.

**b) Caso de Uso Consultas Específicas:**

Una vez validada la contraseña de ingreso por parte de administrador, usuarios o clientes se activaran todas las opciones de consulta disponibles, en este caso la información mostrada es de sólo lectura.

FIN

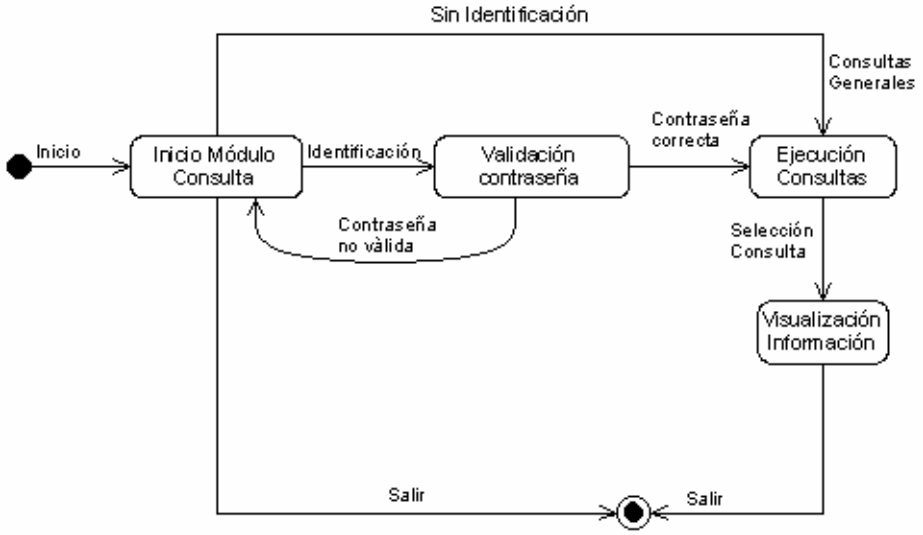



Diagrama de Estados: caso de Uso Módulo de Consulta

**Figura 4.8** Diagrama de Estados. Caso de uso consultar

**4. Descripción del Módulo Aplicaciones Protegidas**

Este módulo consta de las aplicaciones libres y No libres, las primeras presentan la opción de descarga para que sean usadas gratuitamente por un período definido por su dueño, y las segundas solo se visualizaran como una manera de publicación y contacto entre el usuario y los interesados en usarla.

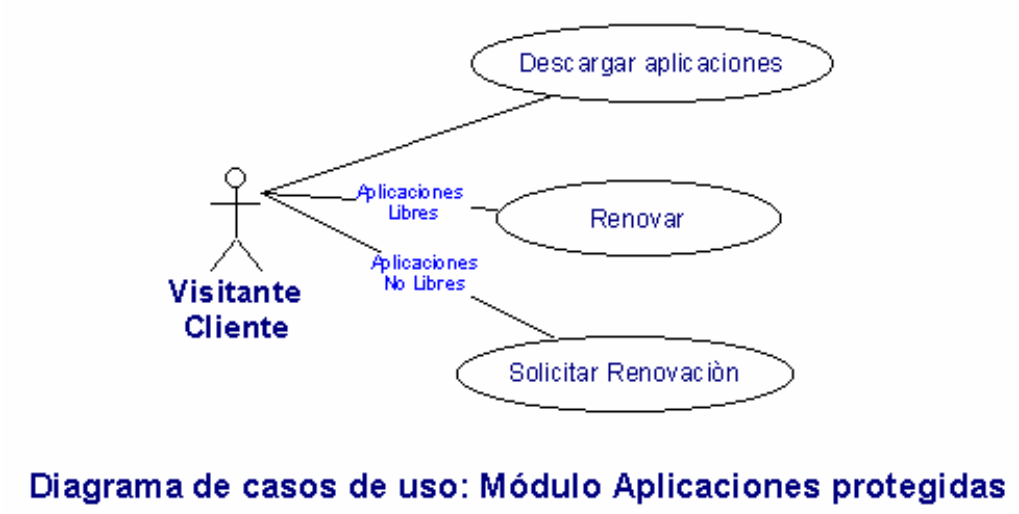



**Figura 4.9** Diagrama Caso de uso Módulo de Aplicaciones protegidas

### 4.1 Detalle Caso de Uso Descargar Aplicaciones

**Precondición:**

1. ***Administrador-Usuario-Cliente:*** Haber iniciado sesión y seleccionado la opción “descargas”.

2. ***Visitante:*** El visitante Web debe haber ingresado al módulo de software protegido (pagina principal), en donde encontrara aplicaciones clasificadas en dos categorías: Libre y No Libre, la primera de ellas necesita del suministro de cierta información para que este se convierta en cliente, las no libres presentan la opción de envió de correo al usuario responsable de la aplicación.

**Camino Básico:**

**Captura de Solicitud:** El visitante Web llena el formulario de inscripción, el cual solicita cierta información de interés para que se active el proceso de descarga. Para usuarios de “TRIONIX” seleccionan cual(es) aplicación(es) desean descargar.

**Validación de los datos:** la información descrita y llenada en el formulario de inscripción es validada por el sistema, como mecanismo de autenticación en futuros procesos (Renovación y consulta) y como medio de control.

**Registro de Datos:** una vez validado los datos correctamente, el sistema generara como respuesta afirmativa la opción de descarga.

FIN

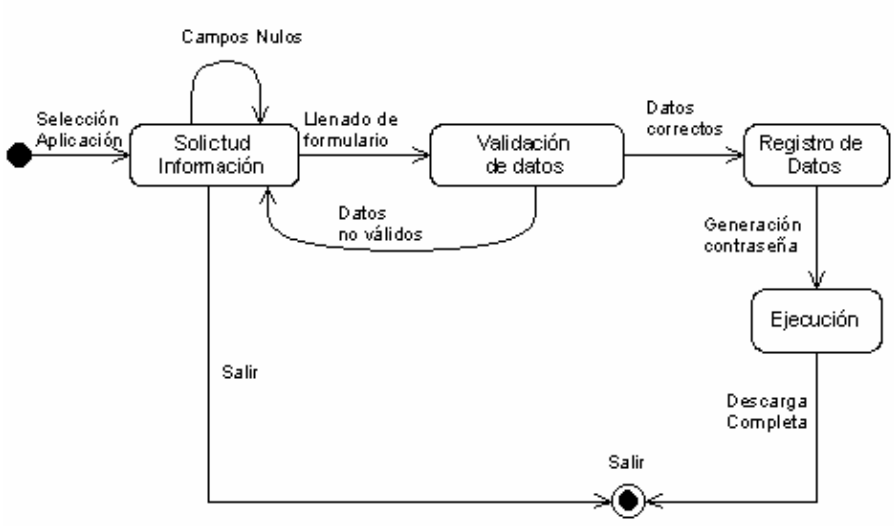



Diagrama de Estados: caso de Uso Aplicaciones Protegidas

**Figura 4.10** Diagrama de Estados. Caso de uso Descargar Aplicaciones

### 4.2 Detalle Caso de Uso Renovar Aplicaciones: Cliente Web

Como el visitante Web a la hora de registrarse se constituye en un cliente normal, para el aplica todas las opciones del cliente; por lo que para renovar lo puede realizar siempre y cuando el usuario le otorgue el permiso para hacerlo, de lo contrario le enviara una solicitud vía e-mail expresando su interés de seguir usando la aplicación.

**Precondición:** El cliente debe ingresar al modulo de renovación y haber digitado la clave pública que le suministró la aplicación vencida.

**Camino Básico:**

**Captura de Solicitud:** Si el cliente tiene el permiso de renovar, selecciona el número de días y accesos que desea usar la aplicación, sin exceder los establecidos por el usuario.

**Validación de los datos**: Se valida la clave pública suministrada y que se hayan llenado los parámetros de protección (días y accesos).

**Registro de Datos:** una vez validado los datos correctamente, el sistema generara como respuesta afirmativa la generación de la clave particular de uso de la aplicación (clave privada).

Cuando un cliente descarga una aplicación por primera vez no aplica ninguna restricción de renovación, por lo que puede hacerlo teniendo en cuenta los permisos otorgados por el dueño de la aplicación.

FIN

### 4.3 Detalle Caso de Uso solicitud de Renovación

Es similar al caso de uso solicitud-respuesta Renovación (ver numeral 2.1) con la variante que puede presentarse en que el usuario coloque condiciones económicas para continuar su uso.

**Precondición:** Envió de correo expresando el interés por renovar una aplicación

**Camino Básico**:
**Envío de la Solicitud:** en este estado el cliente Web envía al usuario por correo u otro medio la petición de renovación de uso de una determinada aplicación; el usuario responderá de manera afirmativa o negativa.

**Respuesta Solicitud:** si la respuesta es afirmativa, el usuario solicitara al cliente la clave pública del proyecto y pactaran los permisos de uso; posteriormente reenviara la nueva clave privada, si es negativa le dará a conocer las condiciones de uso del proyecto.
FIN

## 4.3 EVALUACIÓN DE LA FASE DE INICIO

El desarrollo de esta fase fue producto de un análisis preliminar acerca del Por qué, el para qué y el cómo del desarrollo de esta aplicación; lo cual permitió la identificación de factores relevantes del sistema, descritos en los casos de uso y actores involucrados en cada uno de estos que se constituyen en base factible de la arquitectura del sistema.

Una vez identificados los casos de uso se entró a analizarlos en detalle, describiéndolos en forma clara e individualizada con la ayuda de diagramas UML de casos de uso y estado; estos últimos permitieron la unificación de los casos de uso respectivos a los módulos de servicio, con el objetivo de evaluarlos; además en la fase que culmina se trataron los riesgos de manera particular imposibilitando su presencia.

## 4.4 PLANEACIÓN DE LA FASE DE ELABORACIÓN

El resultado de la fase de inicio es un modelo de casos de uso parcialmente completo y una descripción de la arquitectura candidata; ahora se recopilará la mayor parte de los requisitos, los cuales serán clasificados según su prioridad y analizados de acuerdo a su importancia en la arquitectura que se busca.

Para el logro del objetivo fundamental se detallaran un porcentaje mayoritario de casos de uso, de modo tal que satisfagan la línea base y que esta sea lo suficientemente robusta para resistir la fase que le sigue (construcción); sin embargo se deberá abordar y mitigar los riesgos que pueden interferir en la consecución de este objetivo.

# CAPITULO 5

## FASE DE ELABORACION

El principal objetivo de esta fase es el establecimiento de una arquitectura estable que encamine el sistema a lo largo de su vida de desarrollo, para esto se recopilan la mayor parte de los requisitos, expresando los requisitos funcionales como casos de uso y asociando a estos sus actividades, procesos y riesgos que de una manera dificulten la consecución de los objetivos.

Además de lo anterior se identificará y detallará aquellos puntos relevantes del sistema, que permitan la constitución de una sólida línea base de la arquitectura viable y evolutiva.

## 5.1 IDENTIFICACIÓN DE RIESGOS

La planificación del desarrollo de un producto software esta influenciada por factores de riesgo, por lo que antes de iniciar cualquier fase de desarrollo se debe identificar, detallar y proponer soluciones que impidan la ocurrencia de eventos adversos; todo esto se consigue con una buena PRE-planificación que involucre factores que puedan agudizarse y constituirse en “Riesgos Críticos” que afecten el normal proceso de desarrollo; una vez terminada esta planificación preliminar se continuara con un exitoso desarrollo que conlleve al logro de los objetivos.

Los factores de riesgos más relevantes que pueden presentarse, son los riesgos técnicos involucrados en los casos de uso, para que sean mitigados y su presencia no afecte el proceso de desarrollo; entre estos:

| *Descripción* | *Nivel de riesgo* | *Impacto* | *Control* |
|---|---|---|---|
| ***Establecimiento de una arquitectura no flexible*** | Critico | Global | Se identifica los casos de uso significativos, que corresponden a aquellos que cubren las tareas funcionales que el sistema ha de realizar; además de la identificación de los casos de uso de requisitos no funcionales, casos de uso secundarios, auxiliares y opcionales, que aunque no son importantes arquitectónicamente hablando permiten la mayor cobertura de requisitos y encontrar una base de la arquitectura. |
| ***recopilación incorrecta de requerimientos*** | Critico | Global | Una vez recopilada la información y definidos los requisitos, se sigue con la identificación de casos de uso necesarios para asegurar que se esta desarrollando el sistema correcto, que asegura la evolución del sistema en las siguientes fases. Todo esto se consigue con la realización de un flujo de trabajo de los requisitos de usuario. |

**Tabla 5.1**. Listado de Riesgos

## 5.2 FLUJOS DE TRABAJO

En los flujos de trabajo de esta fase se recopilan, analizan, diseñan, implementan y prueban solo los requisitos relevantes desde el punto de vista de la arquitectura. Lo que se implemente en esta fase servirá para probar lo que se diseña, pero puede que no sirva para construcción.
Los riesgos en esta fase no deben eliminarse sino reducirlos a un nivel aceptable para la fase de construcción

### 5.2.1 FLUJO DE TRABAJO DE LOS REQUISITOS

Se identificará y describirá en detalle el 80 por ciento de casos de uso, estableciendo prioridades y examinando de acuerdo a su importancia en la definición de la arquitectura.

- ✓ ***Actividad: encontrar actores y casos de uso***

Se detallarán aquellos casos de uso que hayan sido modificados (complementos) con respecto a la fase anterior, y dado el caso, se identificaran casos de uso adicionales derivados de la evolución del desarrollo del software que conlleven a la satisfacción de los objetivos.

Los actores se han identificado y detallado plenamente en la fase de inicio, por lo que no se realizará en esta fase (ver Tabla 4.2).

**1. Caso de uso Proteger Aplicaciones**

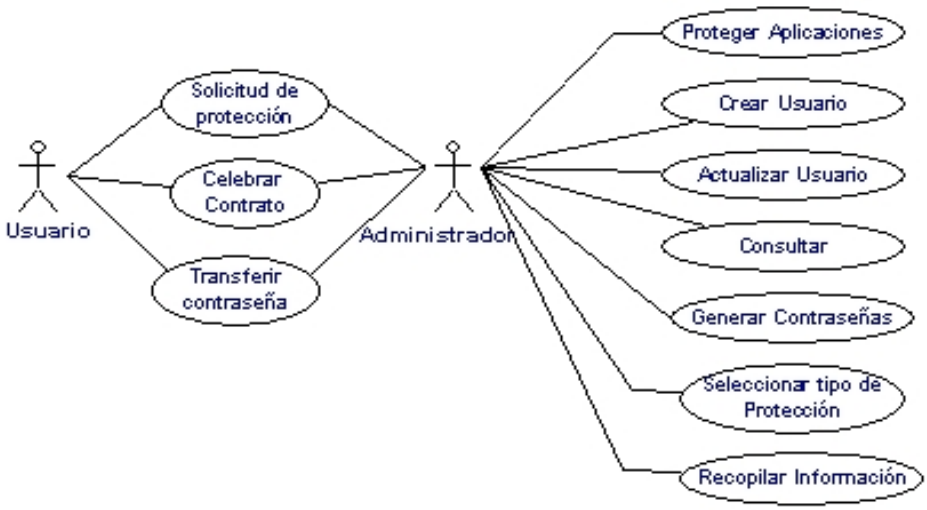



**Figura 5.1** Diagrama de Caso de uso proteger Aplicaciones

***Descripción***

El uso de la aplicación Web se adquiere una vez se celebre un contrato de administración entre administrador-usuario y se proteja una aplicación de su dominio; dicho contrato estará compuesto por condiciones de tiempo en días y demás que se dispongan; durante este período el usuario podrá administrar sus clientes y aplicaciones.

Luego de la elaboración y aprobación del contrato, el administrador creará un nuevo usuario solicitando al mismo cierta información que será validada por el sistema; posterior a esto el administrador se remitirá a proteger la aplicación y a suministrarle al usuario, el login y contraseña que le ha creado, para que el usuario ingrese a la aplicación Web e inicie su periodo de administración.

**2. Caso de Uso Renovación**

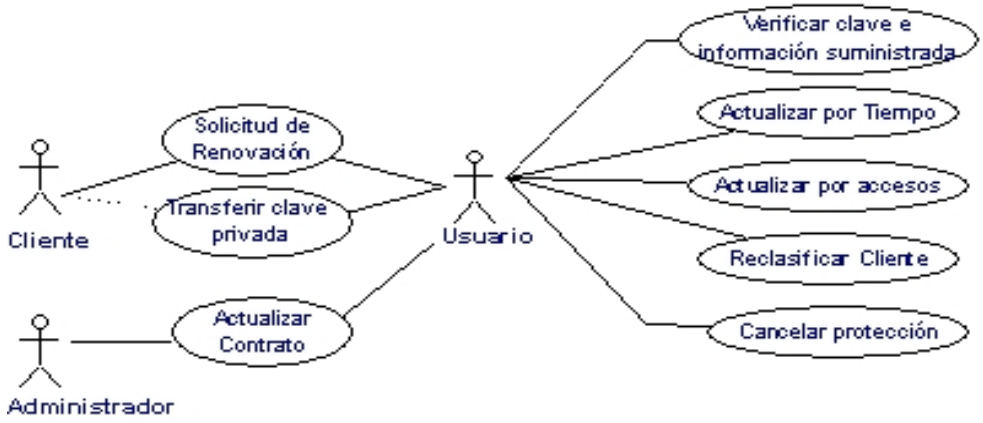



**Figura 5.2** Diagrama Caso de uso Renovación

***Descripción***

Una vez finalizada la protección de una aplicación el sistema automáticamente generará una clave pública particular, dicha clave será componente al igual que los parámetros de protección (días y/o accesos) de la solicitud que emita el cliente al usuario (caso cliente No Automático), este último verificara y autorizara la actualización de la protección, de dicho proceso puede reclasificarse al cliente otorgándole la facultad de auto renovación por un periodo de tiempo (cliente Automático), es decir que con su clave podrá renovar de acuerdo a los criterios establecidos por el usuario.

La respuesta final a la solicitud del cliente es la actualización de la protección y el suministro de la clave privada de uso; el caso de cancelación de protección se da cuando tanto cliente como usuario están en desacuerdo con las condiciones de uso de una determinada aplicación.

El proceso de renovación se extiende también a la actualización del contrato entre administrador y usuario, donde este último solicitará previamente a la caducidad del contrato inicial, su interés de renovación y uso de la aplicación Web.

El proceso descrito anteriormente aplica cuando un aplicación se ejecuta por primera vez.

El proceso descrito anteriormente aplica cuando un aplicación se ejecuta por primera vez.

Lo que se ha hecho hasta ahora es agrupar los requerimientos de acuerdo a su importancia detallando aquellos casos de uso que involucre, tratándolos de manera específica de tal manera que describan la totalidad de requerimientos y que guíen la construcción de la línea base de la arquitectura.

### 5.2.2 FLUJO DE TRABAJO DEL ANÁLISIS

Durante la fase de inicio, se empezó a realizar un esquema del modelo de análisis, ahora este se refinará basado en la utilización de los casos de uso arquitectónicamente significativos para de esta manera comprender los detalles del sistema.
Las actividades que encierra el análisis de la arquitectura, están en la identificación y análisis de los casos de uso significativos arquitectónicamente.

- ✓ ***Actividad: análisis de la arquitectura***

En la fase anterior, se desarrollo el análisis de la arquitectura solo hasta determinar que había una estructura factible; en esta etapa se extiende dicho análisis a un punto tal que pueda servir de base a una línea de la arquitectura ejecutable.

Para lograr este propósito se realiza una partición en paquetes de análisis representados en los casos de uso, identificando los paquetes específicos de la aplicación; luego se identifica con facilidad los paquetes de servicio y clases de análisis significativos arquitectónicamente y obvios.

- **PAQUETE DE ANÁLISIS GESTIÓN DE PROTECCIÓN**

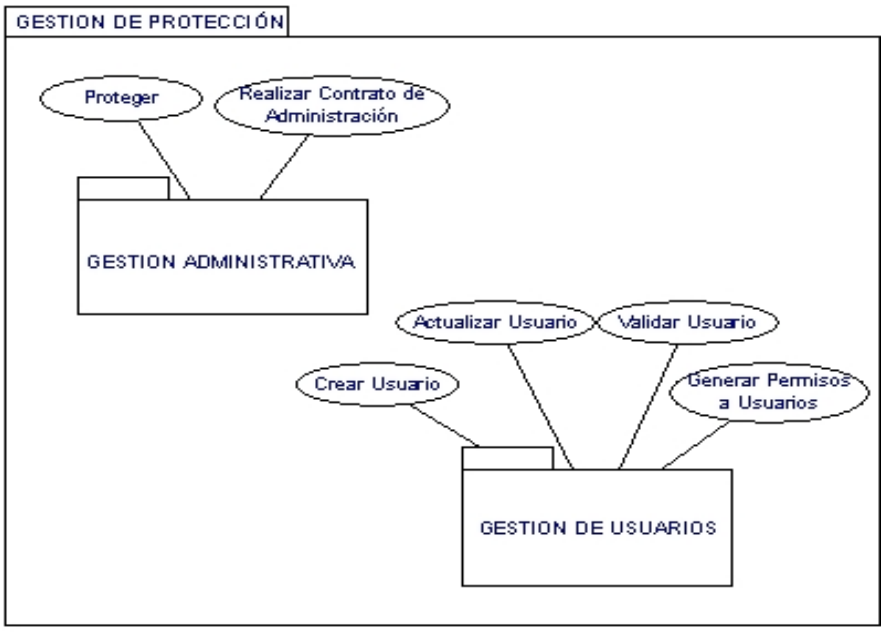



**Figura 5.3** Paquetes de Análisis Gestión de Protección

Esta compuesto por el paquete de gestión administrativa encargado de la funcionalidad operativa del sistema y del Gestión de Usuario, encargado de la relación y funciones con los usuarios.

**PAQUETE DE ANÁLISIS GESTIÓN RENOVACIÓN**

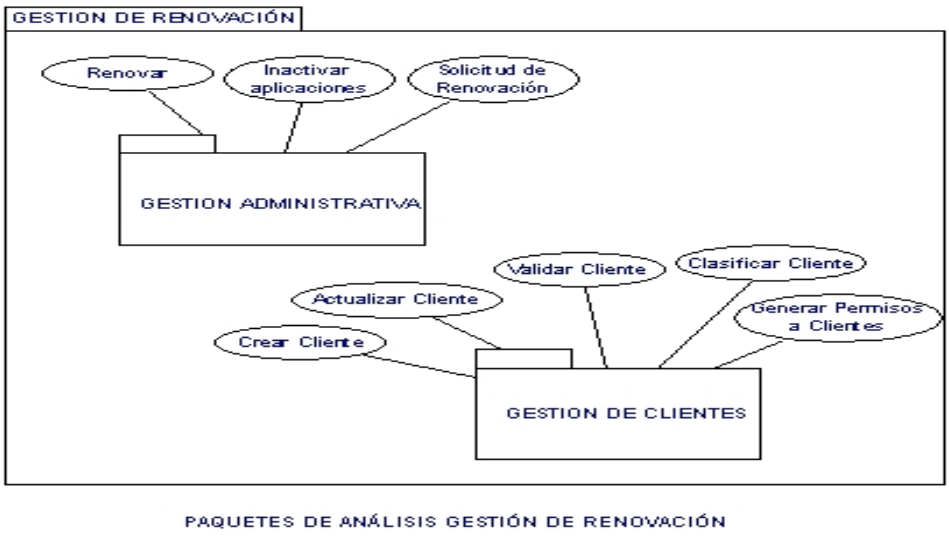



**Figura 5.4** Paquetes de Análisis Gestión de Renovación

Se compone por los paquetes de gestión administrativa y de Gestión de cliente, el primero de ellos contiene todas las funciones que realiza el usuario a petición de sus clientes (Renovar, atender solicitudes....) y el segundo encargado del mantenimiento y administración de clientes.

- **PAQUETE DE ANÁLISIS GESTIÓN DE CONSULTA**


PAQUETES DE ANÁLISIS GESTIÓN DECONSULTA

**Figura 5.5** Paquetes de Análisis Gestión de Consulta

- **PAQUETE DE ANÁLISIS GESTIÓN DE APLICACIONES PROTEGIDAS**


PAQUETES DE ANÁLISIS GESTIÓN DE APLICACIONES PROTEGIDAS

**Figura 5.6** Paquetes de Análisis Gestión de Aplicaciones Protegidas

- ✓ **ACTIVIDAD: Análisis de casos de uso**

La descripción de algunos casos de uso no es tan compresible de acuerdo a los modelos de análisis elaborados, por lo que deben refinarse ya sea por su complejidad o por su significado en la arquitectura en función de las clases de análisis que existen en el ámbito de los requisitos.

Los casos de uso que deben ser refinados son aquellos que tienen un gran significado y relevancia en la definición de la arquitectura o para la comprensión de requisitos; los restantes deben ser comprendidos en su totalidad en función de su impacto en el sistema para ser implementados durante la siguiente fase de desarrollo.

De acuerdo al análisis de casos de uso se hace necesario identificar las clases de análisis (entidades, controles e interfaces) y como se relacionan entre si para llevar un flujo de sucesos de un caso de uso; para esto se ha de utilizar la ayuda de diagramas de colaboración que permitan una clara esquematización del proceso derivando además una detallada identificación de requisitos y responsables sobre las clases.

### ❖ PAQUETE DE GESTIÓN DE PROTECCIÓN

Se decidió detallar el análisis al caso de uso proteger aplicaciones, debido a su importancia en el modulo de protección, aunque este para su funcionamiento depende también de los restantes; sin embargo la aplicación tiene por objeto la protección de software y este es quien realiza este proceso.
El caso de uso proteger aplicaciones cuenta con las clases de interfaz IU (Interfaz Única) de identificación y IU de la aplicación, que se encargan de mostrar al administrador el formulario de validación de acceso a la aplicación, y el menú de opciones para la protección de software respectivamente. Las clases de control gestor de identificación, gestor de proyectos y gestor de protecciones son las que permiten la ejecución de los procesos derivados de la protección: validación de acceso, validación de datos y protección respectivamente, estas clases interactúan con las clases de identidad de la base de datos porque son el soporte de sus procesos y permiten la continuación de la protección.

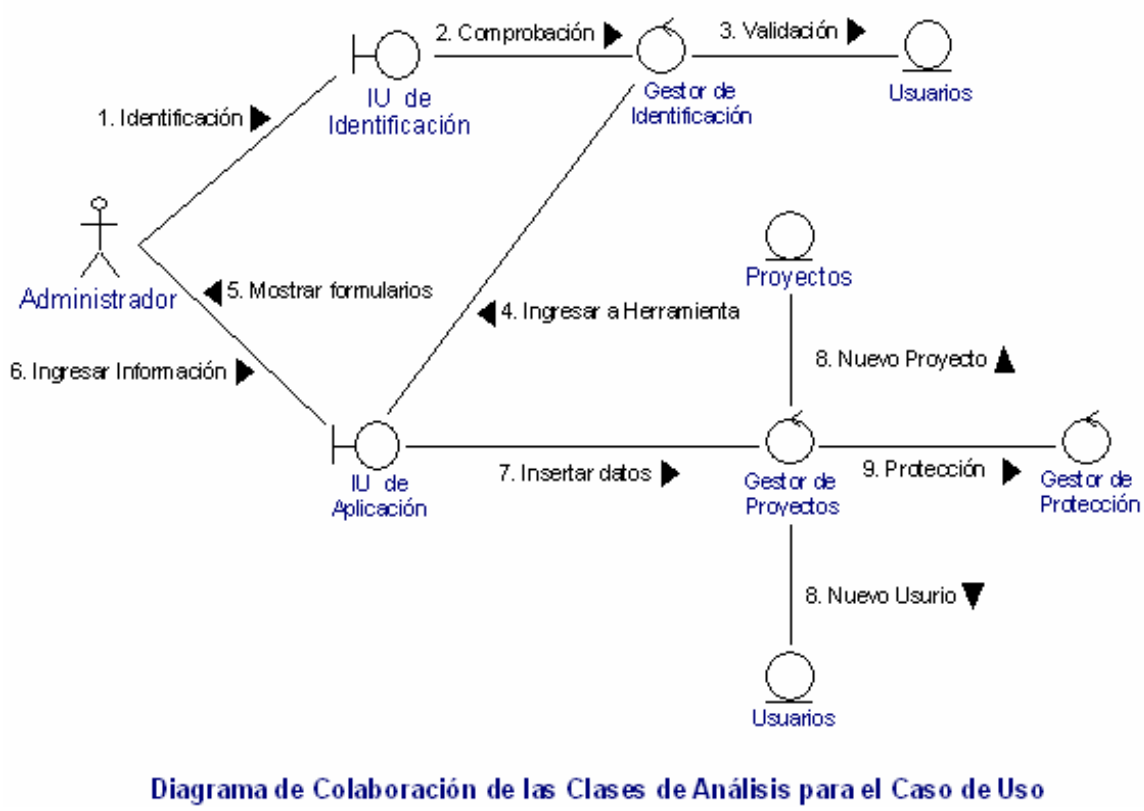



**Figura 5.7** Diagrama de Colaboración de las Clases de Análisis para el caso de uso proteger Aplicaciones

El administrador ingresa al sistema con un nombre de usuario y contraseña los cuales son comprobados por el gestor de identificación quien valida que se hayan digitado, que la longitud de caracteres cumpla con los parámetros establecidos y la autenticidad de los mismos, realizando un proceso de comprobación entre los suministrados por el usuario y los que se encuentran en la base de datos; si la respuesta a este proceso es afirmativa se activa la interfaz de la aplicación para que el administrador inserte información referente a un usuario y aplicación a proteger; para ello se despliega un formulario solicitando cierta información de obligatorio llenado, la información será validada posteriormente por el gestor de proyectos y depositada en la base de datos para futuras validaciones y consultas, finalmente se protege la aplicación.

### ❖ PAQUETE DE GESTIÓN DE RENOVACIÓN

Se analizará el caso de uso renovar por tratarse del eje fundamental del modulo de renovación, es este quién solicita, valida, actualiza y genera la nueva clave de uso de la aplicación.

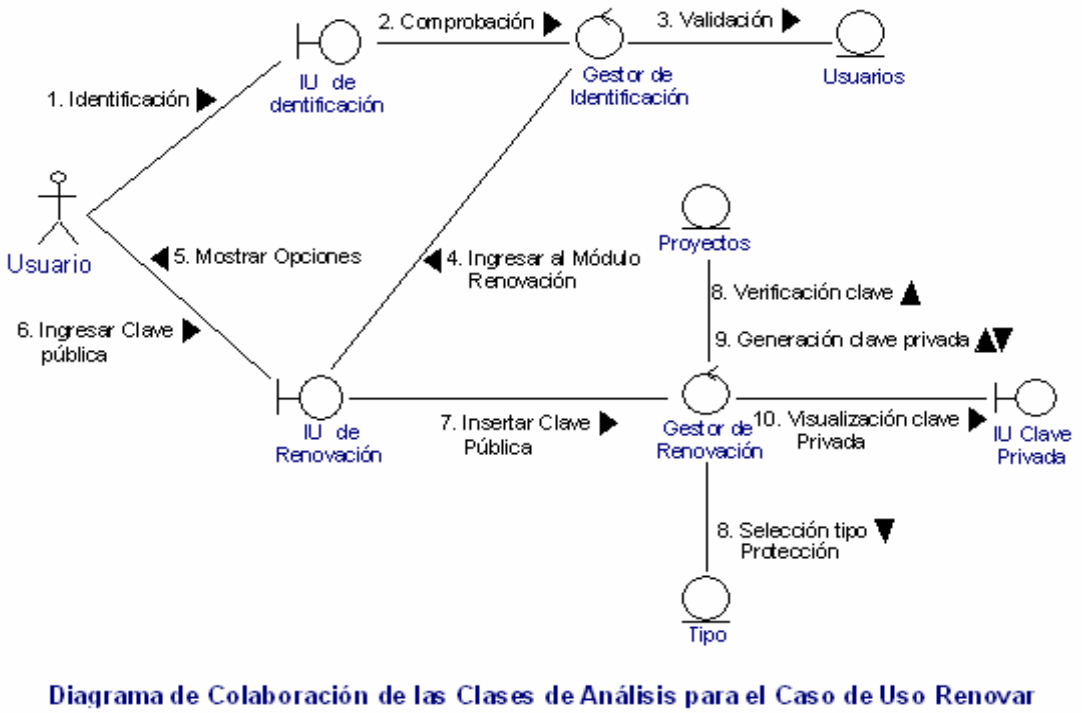



**Figura 5.8** Diagrama de Colaboración de las Clases de Análisis para el caso de uso Renovar

Cuando un cliente desee renovar y tiene permiso para hacerlo, debe ingresar a la aplicación Web, modulo de renovación y suministrar la clave pública del proyecto a proteger, esta clave se valida y actualizara la información correspondiente generando la nueva clave de uso; si no tiene permiso para renovar debe realizar la solicitud al usuario por medio de un correo electrónico u otro medio. La solicitud debe contener la clave pública de la aplicación, generada una vez caduque la protección y los parámetros de actualización: tiempo y accesos.
El usuario para atender las solicitudes de sus clientes debe ingresar a la aplicación Web, ver los mensajes de sus clientes y según su criterio optar o no por renovar. Si va a renovar ingresa al modulo renovación, selecciona el cliente y el proyecto, digita la clave suministrada por el cliente la cual será validada, selecciona los parámetros de protección y el sistema le generará la nueva clave privada, la cual remitirá al cliente.

### ❖ PAQUETE DE GESTIÓN DE CONSULTA

En este paquete se han integrado los casos de uso consultar información particular e información general por su similitud y proceso que siguen diferentes a la identificación.

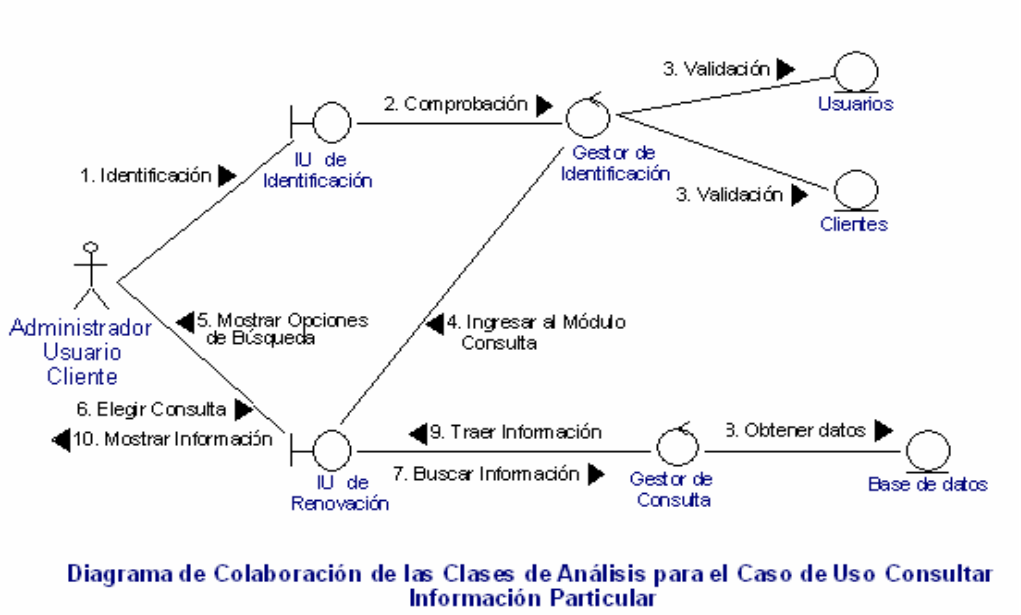



**Figura 5.9** Diagrama de Colaboración de las Clases de Análisis para el caso de uso Consultar Información Particular

Para el caso de uso consultas particulares, solo esta disponible para los clientes o usuarios de la aplicación Web, por esto se solicitará identificación con login y contraseña, la cual se validará verificando su existencia en la de la base de datos, si son correctos se visualizará las opciones de búsqueda de información que se disponen, el usuario seleccionará el tipo de consulta y de acuerdo a esta buscara en las tablas y registros que correspondan a lo seleccionado; como resultado el sistema mostrará la información solicitada.

Para consultas generales el sistema no solicitará identificación y se ofrecerá información limitada acerca de desarrolladores y aplicaciones protegidas a cualquier navegante.

### ❖ PAQUETE DE GESTIÓN DE APLICACIONES PROTEGIDAS

La opción de descarga está disponible para toda persona que desee utilizar un software durante un periodo de tiempo establecido por los desarrolladores o responsables de la aplicación software; quienes además optarán por divulgarlo a manera de publicación (caso software con restricciones) o uso (software libre). Si se trata del primer caso, el sistema presentará cierta información referente al usuario para que el interesado se ponga en contacto con él, para el segundo caso, el sistema solicitará cierta información de interés de obligatorio llenado además de un login y contraseña, para que el visitante se constituya en un cliente y posteriormente ingrese a la aplicación Web y consulte o renueve de acuerdo a los permisos que le haya otorgado el usuario.

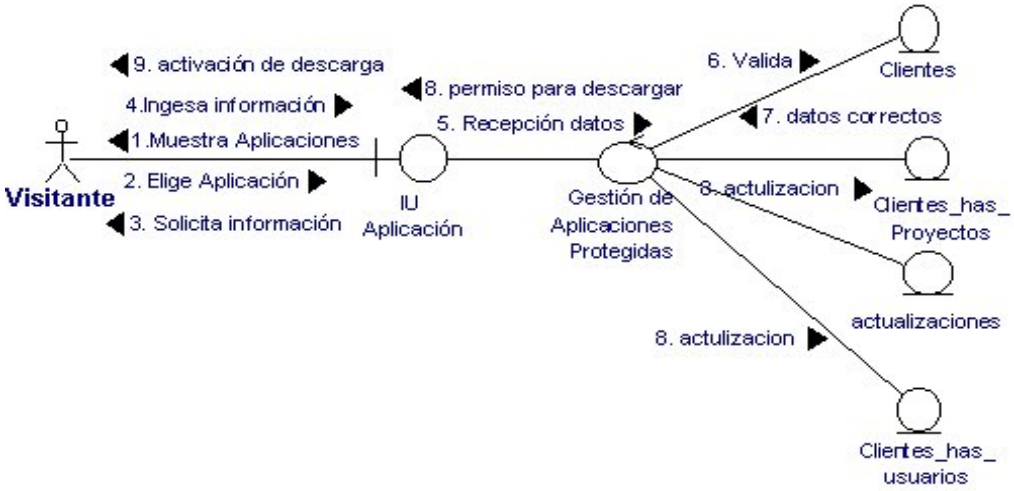



**Figura 5.10** Diagrama de Colaboración de las Clases de Análisis para el caso de uso Descargar Aplicaciones

### 5.2.3 FLUJO DE TRABAJO DEL DISEÑO

En el diseño se busca modelar el sistema y lograr que este soporte los requisitos funcionales, no funcionales y otras restricciones. Un punto de entrada en el diseño es el resultado del análisis, modelo de análisis, el cual proporciona una comprensión en detalle de los requisitos e impone una estructura del sistema a consolidar y tratar hasta conseguir en definitiva una base sólida de la arquitectura.

El diseño tiene como objeto la comprensión en detalle de aspectos relacionados con los requisitos no funcionales y restricciones en cuanto a lenguajes de programación, componentes reutilizables, sistemas operativos, tecnologías de interfaz de usuario, entre otros.

En esta fase se diseñan una pequeña fracción de casos de uso identificados, en particular aquellos arquitectónicamente significativos, los cuales conforman el modelo de diseño que incluye subsistemas, clases, interfaces de gran importancia para la arquitectura que se busca.

- ✓ **Actividad: DISEÑO DE LA ARQUITECTURA**

El objetivo del diseño de la arquitectura es identificar y plasmar los modelos de diseño, despliegue y su arquitectura mediante la identificación de los nodos, configuración de red, subsistemas, interfaces, software del sistema y capa intermedia

El siguiente diagrama muestra a grandes rasgos la arquitectura del sistema especificando los subsistemas particulares de la aplicación (los que realmente utilizan los usuarios) y aquellos de fondo que utiliza la aplicación para su funcionamiento.

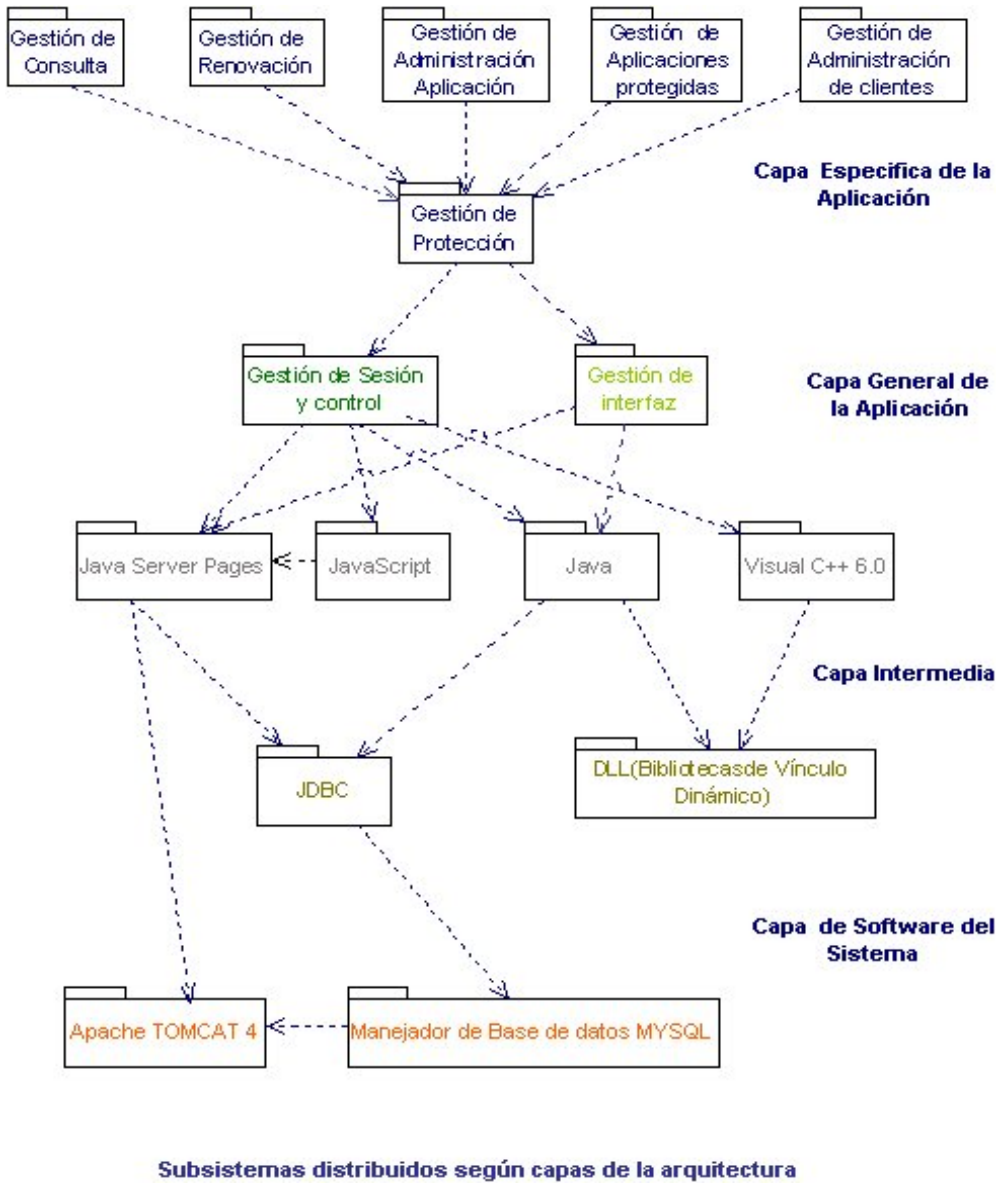



**Figura 5.11** Subsistemas distribuidos según capas de la arquitectura

## ❖ NODOS Y CONFIGURACIÓN DE RED

La aplicación Web cuenta con dos clases de nodos: Un nodo servidor compuesto por una capa de datos y capa lógica, la una permite la transferencia de información (solicitudes) y la otra interpretar los datos y dar respuesta; el nodo cliente encargado de la capa de presentación en un modelo de arquitectura de tres capas, el cual se utilizara para la distribución del sistema.

En cuanto a protocolos de comunicación entre nodos se utilizaran los protocolos HTTP y TCP/IP. Los requerimientos de hardware y software se encuentran descritos en el Manual de Usuario.

## ❖ SUBSISTEMAS

Los subsistemas permiten organizar el modelo de diseño en piezas manejables, de modo tal que ayudan a dividir el trabajo de diseño y posteriormente el de implementación. Un subsistema debe estar fuertemente asociado entre sus componentes y su dependencia con los demás subsistemas debe ser mínima.

Los subsistemas se derivan a partir de los paquetes de análisis tratados en detalle anteriormente por lo cual es posible utilizarlos como base para los subsistemas de diseño, entre los subsistemas tenemos: Subsistema de Gestión de protección, Subsistema de Gestión de Renovación, Subsistema de Gestión de Consulta, Subsistema de Gestión de administración de clientes y Subsistema de Gestión de Aplicaciones Protegidas, encargados de la parte operativa de la aplicación.

Además de los subsistemas mencionados y aparte de los paquetes de análisis han surgido los Subsistema de Gestión de Sesión - Control y Subsistema de Gestión Interfaz, el primero de ellos es el encargado de controlar a través de validaciones, el acceso de los usuarios al sistema, además de validar la autenticidad de claves en el proceso de renovación y la información que se depositará en la base de datos, esta funcionalidad es característica de todos los subsistemas, lo cual hace posible la integración de estos procedimientos en un nuevo subsistema.
El Subsistema de Gestión de Interfaz es el encargado de controlar la consistencia, coherencia, diseño de interfaces, colores, simbología, en pro de satisfacción y uso por parte de los usuarios de la aplicación, además este subsistema es el canal de comunicación entre el sistema y cualquier actor del mismo.

Los Subsistemas de Gestión de Sesión - Control y Gestión de Interfaz son subsistemas generales o de segundo orden, debido a que no son accedidos directamente por los usuarios, sino que son soporte de los demás subsistemas por su dependencia en sus procesos, por esto en el modelo de diseño se encuentran en la parte inferior.

## ❖ SUBSISTEMA DE GESTIÓN DE SESIÓN - CONTROL

La seguridad del sistema es uno de los aspectos vitales y relevantes en importancia, debido a que no todos los usuarios pueden realizar las mismas actividades y disponer de las opciones del sistema, por ello se hace necesario establecer un mecanismo de control que garantice que solo personas autorizadas e identificadas tengan acceso a las zonas y opciones que el sistema dispone de acuerdo a su perfil.

El subsistema de gestión de Sesión-control además de validar el acceso al sistema tiene por objeto la validación de la clave pública en el proceso de renovación y la verificación de tipo de datos y no nulidades en los campos cuando el sistema solicite información. Para explicar en detalle este subsistema se utilizará un diagrama de secuencias que describa un caso particular y como interviene en él, el subsistema de Gestión de sesión-control, para este caso el diagrama no contendrá detalle de restricciones y caminos alternativos, pues se desea explicar los procedimientos normales del proceso del caso de uso a tratar.

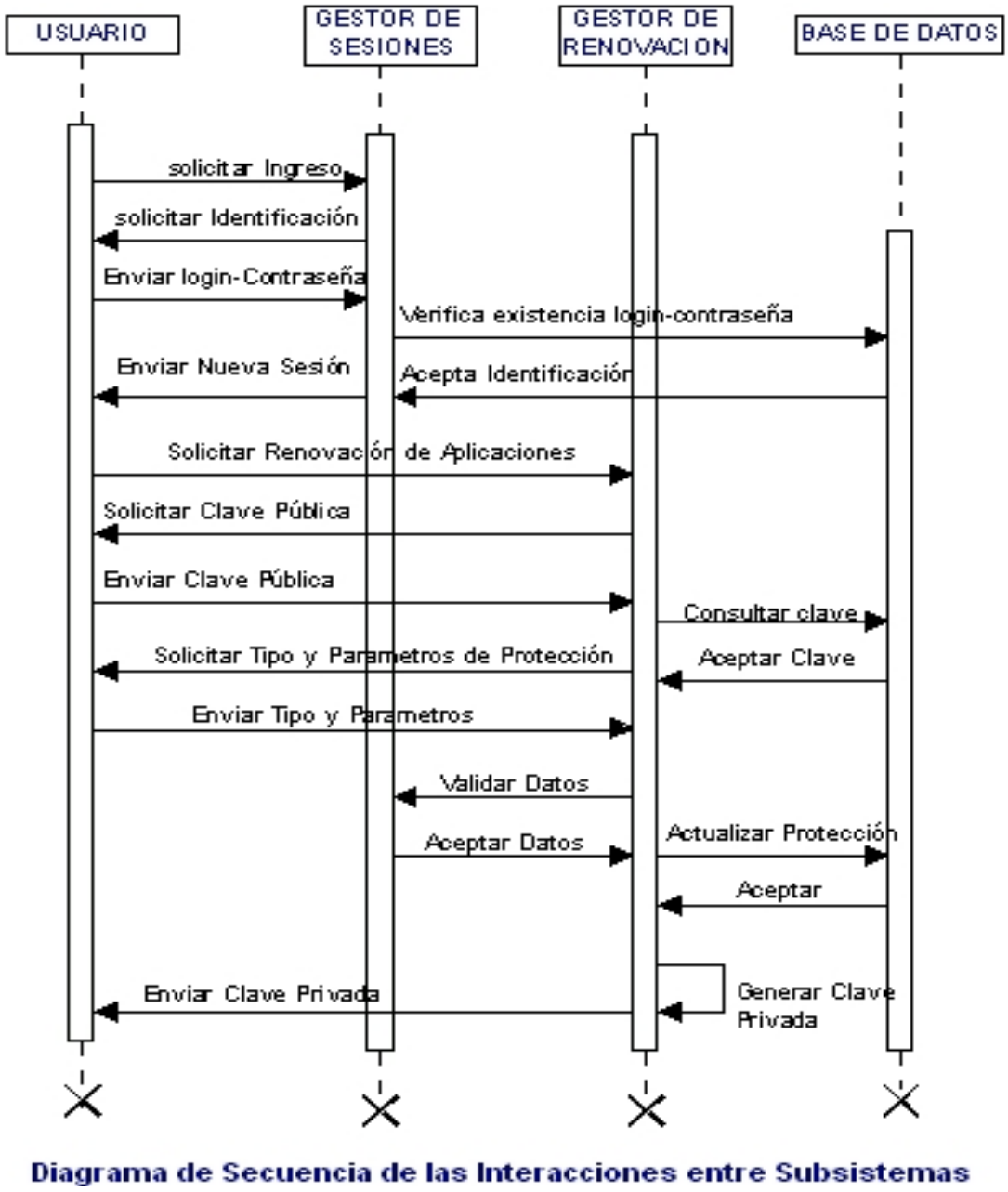



**Figura 5.12** Diagrama de Secuencias caso de uso Renovación

El diagrama anterior describe el proceso seguido por el caso de uso renovación, como puede notarse cuando el usuario o cliente (clientes Automático: permiso para renovar) desea ingresar a la aplicación Web primero debe solicitar su acceso a la misma, el subsistema de gestión de sesión y control se encarga de solicitar identificación al usuario el cual deberá suministrar su nombre de usuario y contraseña, una vez se verifique que sean correctos, el gestor de sesión-control permitirá el ingreso al módulo de renovación, en este punto el usuario solicita renovación de una aplicación, el gestor de renovación inicia su proceso solicitando la clave pública de la misma y parámetros de protección, una vez ingresados se validaran por el sistema verificando su consistencia de tipo y no nulidades, si lo anterior es correcto el gestor de renovación genera la clave privada particular de uso de la aplicación para que finalmente termine la renovación de dicha aplicación satisfactoriamente.

El subsistema de gestión de sesión-control es invocado para el acceso a los módulos y la validación de la información que se desee remitir a la base de datos, por esto se compone de controles adecuados que garantizan la seguridad del sistema.

### ❖ SUBSISTEMA DE GESTIÓN DE INTERFAZ

La interfaz es el medio de comunicación entre el usuario y el sistema; de está depende el uso masivo de la aplicación, pues será lo que realmente verán los usuarios.

Este subsistema es el encargado de todo lo relacionado con la interfaz de usuario y de que cumpla los requisitos básicos tales como: estándares de diseño, consistencia, coherencia, colorido, en pro de la satisfacción a los usuarios.

Cuando se habla de *consistencia* se refiere a una adecuada distribución de imágenes, texto y controles gráficos en cada una de las interfaces y opciones que componen el sistema.

*Coherencia*: el paso de una pantalla a otra debe ser acorde al trabajo que en ese momento se intente realizar, por ejemplo, si se escoge una opción se debe activar la pagina que acompaña esta y no otra que no tenga relación alguna, o bien que la interfaz de la opción elegida no se preceda de interfaces que el usuario deba desactivar para llegar finalmente a la deseada.

*No Sobrepoblación de Pantallas*: la óptima división de controles gráficos permite una buena comprensión en las pantallas, es decir en una sola interfaz no se acumula toda la información que se desea brindar sino que se divide en interfaces sucesivas y organizadas para la ilustración de la información de acuerdo a la opción elegida.

*Control de Colores*: si la gama de colores que componen la interfaz principal y las interfaces secundarias son muy intensas, será incomodo la utilización de la aplicación.

### ❖ SUBSISTEMA DE GESTIÓN DE CONSULTA

Subsistema generador de información general o particular referente a desarrolladores o aplicaciones protegidas de acuerdo al rol que posee el actor.

La información general esta disponible para cualquier persona que ingrese a la aplicación Web, sin embargo si se desea ver información particular debe identificarse con nombre de usuario y contraseña, por lo que obligatoriamente debe ser Administrador, Usuario o Cliente de la aplicación.

### ❖ SUBSISTEMA DE GESTIÓN DE RENOVACIÓN

El proceso de renovación es una de las facultades más importantes que poseen los usuarios y algunos clientes, es el módulo de actualización de protección de aplicaciones. Inicia su proceso una vez caduque una protección por número de accesos o limite de tiempo, esta estrictamente ligado con el subsistema de Sesión-control por invocarlo para la validación de la clave pública y datos de actualización; el resultado de este proceso es la generación de la clave privada compuesta por el código de tipo de protección, tiempo de protección, clave como tal, accesos y código del proyecto protegido(aplicación).

### ❖ SUBSISTEMA DE GESTIÓN DE ADMINISTRACION DE CLIENTES

Este subsistema cuenta con las opciones de administración de clientes, donde los usuarios pueden crear, modificar, inactivar, otorgar permisos y clasificar un cliente de su dominio, y garantizar así que toda aplicación tenga licencia de uso.

### ❖ SUBSISTEMA DE GESTIÓN DE APLICACIONES PROTEGIDAS

Visualiza cada una de las aplicaciones protegidas por TRIONIX, las unas a manera de publicación por su restricción de uso y las otras disponibles para ser descargadas y utilizadas por cualquier persona, si es este último caso, el sistema solicitará cierta información, que será validada y remitida a la base de datos; finalmente se permitirá el descargue e ingreso a la aplicación Web en calidad de cliente.

### ❖ SUBSISTEMA DE GESTIÓN DE PROTECCIÓN

Realiza la principal función de la herramienta, proteger aplicaciones software Win32, para esto inicialmente solicita información del usuario como: nombre, dirección, organización, tiempo de administración, entre otros, además de solicitar un login y contraseña para quien protege acceda posteriormente a la aplicación Web y administre sus aplicaciones.

Este subsistema no esta disponible en la Web, sino que interactúa con la aplicación accediendo y remitiendo información remotamente a la base de datos, como soporte de los demás módulos componentes.

- **SUBSISTEMA ADMINISTRACIÓN**

Subsistema exclusivo del administrador, en donde tiene todo el control operativo sobre la aplicación Web y las opciones de mantenimiento de información que se manejan.

- **ACTIVIDAD: DISEÑO DE SUBSISTEMAS DE RENOVACIÓN Y CONSULTA**

Cada subsistema cuenta con su interfaz y opciones particulares, los unos interactúan con los otros para los procesos de validación y consulta, por ejemplo los subsistemas anteriores se relación entre si porque al renovar se necesita validar y esta ultima realiza un proceso interno de consulta para verificar la validez y aceptar la información.

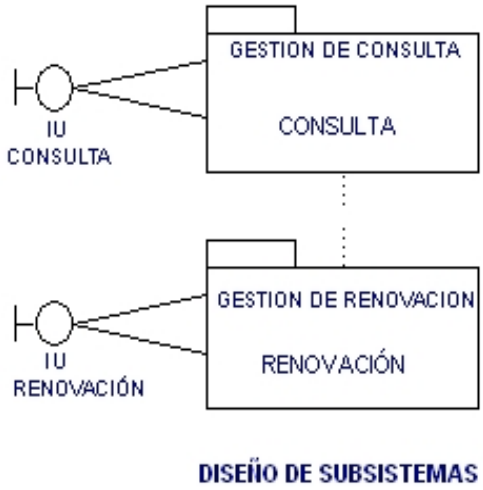



**Figura 5.13** Subsistemas de Renovación y consulta.

### 5.2.4 FLUJO DE TRABAJO DE LA IMPLEMENTACIÓN

Este flujo de trabajo implementa y prueba los componentes arquitectónicamente significativos a partir de los modelos y elementos de diseño construidos. El producto de esto es la base de la arquitectura que se busca.

El objetivo fundamental de la implementación es desarrollar la arquitectura y el sistema como un todo, planificando y distribuyendo el sistema asignando componentes ejecutables a nodos en el diagrama de despliegue, implementando clases y subsistemas encontrados durante el diseño.

## 5.3 EVALUACIÓN FASE DE ELABORACIÓN

La evaluación deberá determinar si la arquitectura escogida es sólida y estable que guié el trabajo a lo largo de la fase de construcción y durante toda la vida útil del sistema. Esta arquitectura debe ser flexible, es decir permitir el crecimiento y modificaciones al sistema.

Para obtener esta arquitectura, se debe recopilar la mayoría de casos de uso y analizar aquellos de gran impacto e importancia sobre la arquitectura, además debe continuarse el control sobre los riesgos surgidos anteriormente, en esta fase y los que eventualmente puedan surgir durante el desarrollo.

### 5.3.1 VISTA DE LA ARQUITECTURA

Esta fase se centro en la determinación de una línea base de la arquitectura, y para ello se recopiló y analizó en detalle aquellos casos de uso relevantes desde el punto de vista arquitectónico, para así obtener un esquema base del sistema en desarrollo.

Durante la fase de elaboración se analizaron los casos de uso que no fueron detallados lo suficientemente en la fase de inicio y que componen los casos de uso más significativos para la arquitectura, a partir de estos se obtuvieron las clases de entidad, control e interfaz, base del modelo entidad relación, interfaces y funciones o scripts necesarios, debido a que permiten una plena identificación de entradas, salidas y procesos del sistema.

Además en el flujo de diseño se identificaron los subsistemas de la aplicación, su interacción y dependencia con los demás para realizar su función y organizar el sistema de manera tal que describa el procedimiento seguido por los subsistemas al invocar algún proceso, otro aspecto importante de la arquitectura es su flexibilidad de expansión gracias al diseño por subsistemas, los cuales hacen fácil la adición de un nuevo subsistema sin necesidad de reestructurar toda la aplicación, esta característica también es particular del diseño de capas, la modificación en alguna no afecta la arquitectura.

### 5.3.2 CUMPLIMIENTO DE CRITERIOS

La fase de elaboración tiene por objetivo principal establecer la base de una arquitectura sólida que guié el desarrollo del sistema, por tal razón las tareas de los flujos de trabajo se han enfocado y tratado específicamente para de esta manera obtener la vista de una arquitectura estable, que satisfaga los objetivos de la fase y del desarrollo.

Esta fase inicialmente se centro en un PRE-análisis, identificación y análisis de los requisitos del sistema traducidos en casos de uso y actores que intervienen en los mismos; el PRE-análisis se realizó a partir de los requisitos tratados en la versión anterior del producto TRIONIX,

de aquí se determino qué requisitos deberían ser modificados, eliminados y generados a partir de estos, el análisis posterior consistió en detallar los casos de uso más significativos, en donde se pudo establecer que la mayoría de casos de uso están bien definidos y claramente descritos como para ser implementados en la siguiente fase.

Sin embargo, el análisis se extendió para aquellos casos de usos significativos y relevantes de cada paquete de análisis, debido a que reúnen los aspectos más importantes de los restantes casos de uso y funciones características de su paquete, y que guían el diseño del modelo entidad-relación y las interfaces de usuario.

Los perfiles de usuario surgieron a partir de los actores y las facultades asociadas a cada uno de ellos, limitando el acceso y opciones del sistema contribuyendo a verificar que la línea base de la arquitectura satisfaga los requerimientos de los usuarios.

## 5.4 PLANEACIÓN DE LA FASE DE CONSTRUCCIÓN

Es importante comenzar a planificar de forma detallada la fase de construcción y determinar el número de iteraciones necesarias, si es el caso. Para el desarrollo de la aplicación software a implementar, en cada fase realizada se han analizado en detalle la mayoría de los aspectos que define el proceso unificado; por ello se considera que si se continúa con esta línea de desarrollo bastara con una sola iteración para obtener la versión operativa del sistema.

Al final de la fase de construcción se debe haber identificado el 100% de los casos de uso del sistema debidamente detallados, realizando pruebas de unidad con resultados satisfactorios, mitigados los riesgos identificados en esta fase y en las anteriores; y finalmente la versión terminada del sistema.

**CAPITULO 6**

# FASE DE CONSTRUCCIÓN

El objetivo más importante de esta fase es enfatizar en la implementación y las pruebas del software. Se espera obtener una versión de la herramienta que cumpla con los requerimientos establecidos al comienzo del proyecto. Existen algunos documentos que deben ir de la mano con la construcción tal como el manual de usuario y la documentación de la implementación, los riesgos deben estar casi mitigados en su totalidad en esta fase.

## 6.1 FLUJOS DE TRABAJO

Los flujos de trabajo se centran en la construcción del sistema, es decir en completar la realización de los casos de uso, implementar los subsistemas y clases como componentes que será el producto de un desarrollo iterativo y guiado por los casos de uso. Las pruebas toman importancia ya que en las fases anteriores solo se implementaban para probar los casos de uso y en esta fase deben estar encaminadas hacia el correcto funcionamiento de la versión ejecutable.

### 6.1.1 EL FLUJO DE TRABAJO DE LOS REQUISITOS

- ✓ **Actividad: Desarrollo de prototipos de interfaz**

Los prototipos de interfaz de usuario nos ayudan a comprender y especificar las interacciones entre actores humanos y el sistema durante la captura de requisitos.

- **Interfaz de prueba del proceso de protección.**

Esta interfaz permite la recopilación de la información del usuario y la aplicación a proteger.

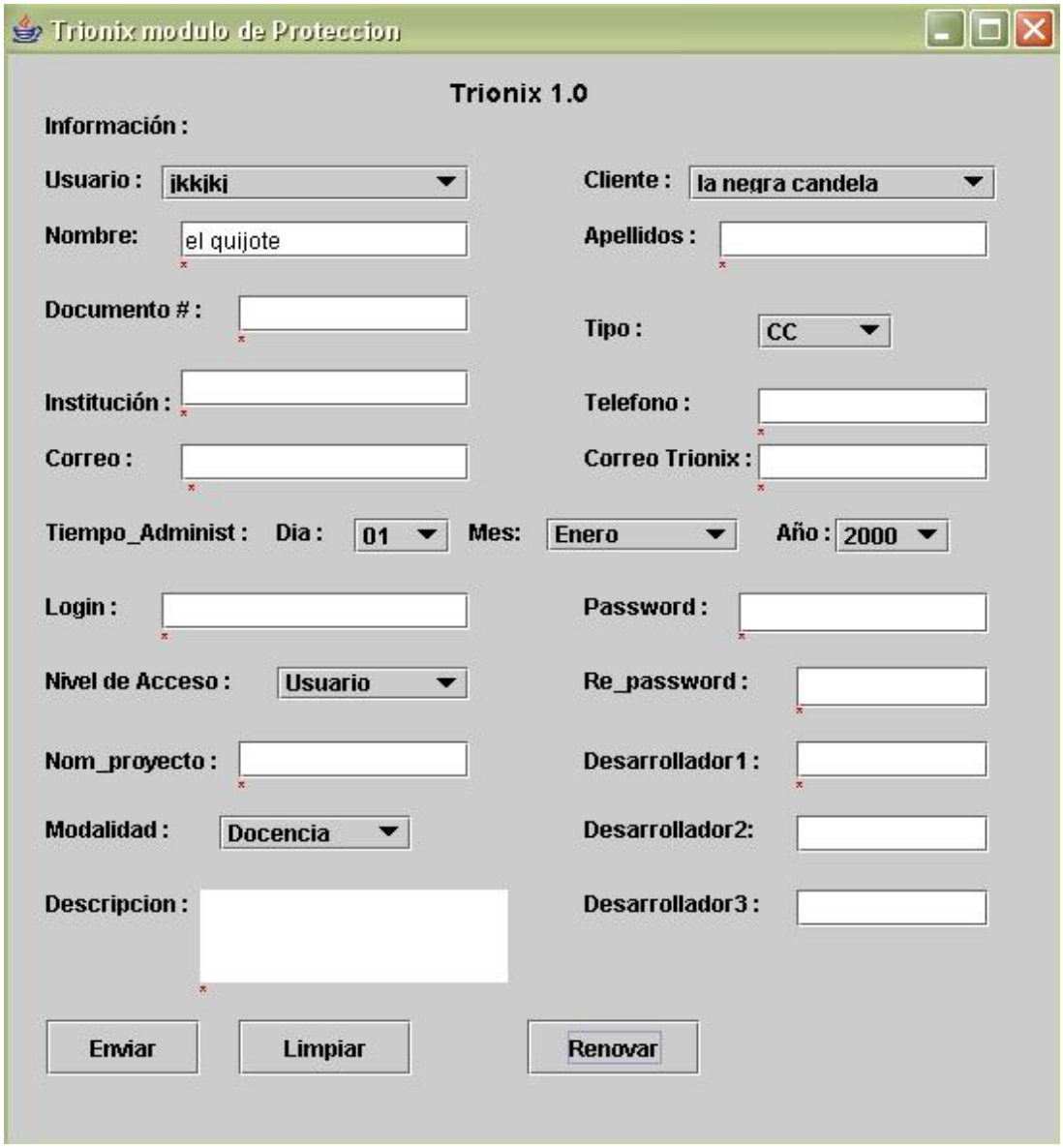



**Figura 6.1** Interfaz1. Primer prototipo de la Interfaz de proteger Software.

La siguiente interfaz recopila y calcula los parámetros necesarios para la protección: ruta origen (ruta donde se encuentra el proyecto a proteger), ruta destino (ruta donde se guardara el nuevo ejecutable protegido), selección de opción de descarga y parámetros de uso.

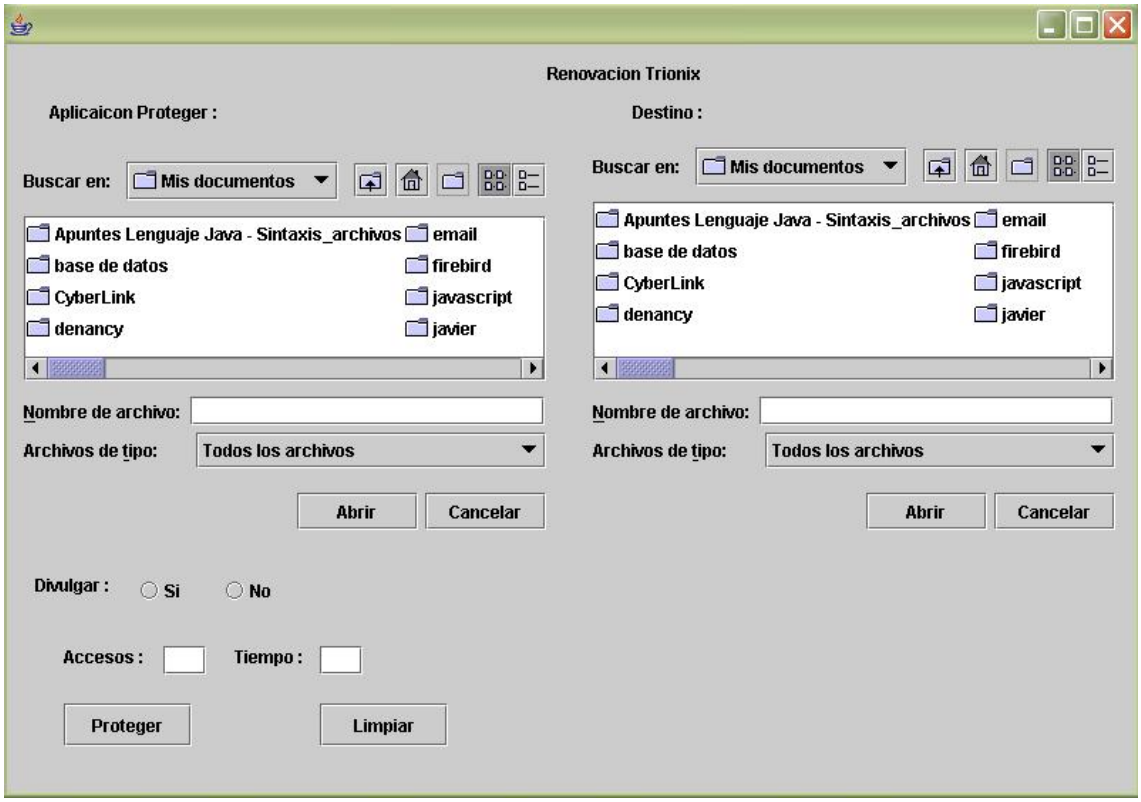



**Figura 6.**2 Interfaz2. Primer prototipo de la Interfaz de proteger Software.

- **Interfaz final de la aplicación Web (Administración)**

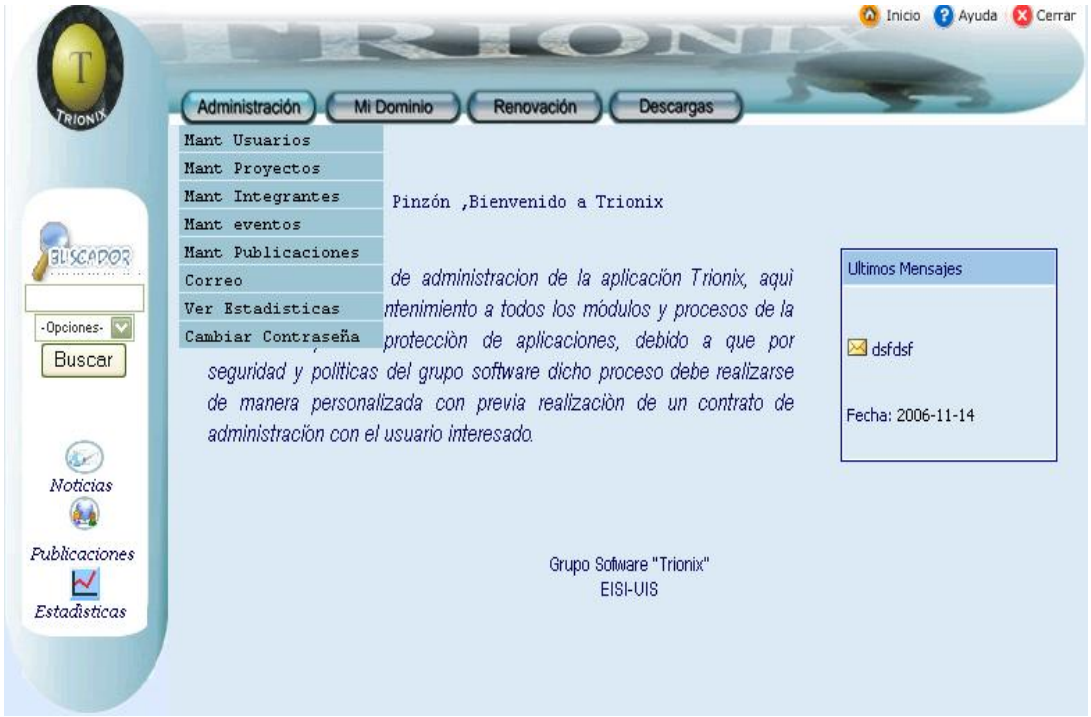



**Figura 6.3** Interfaz final aplicación Web: Modulo de Administración

- **Interfaz final del Modulo de Protección**

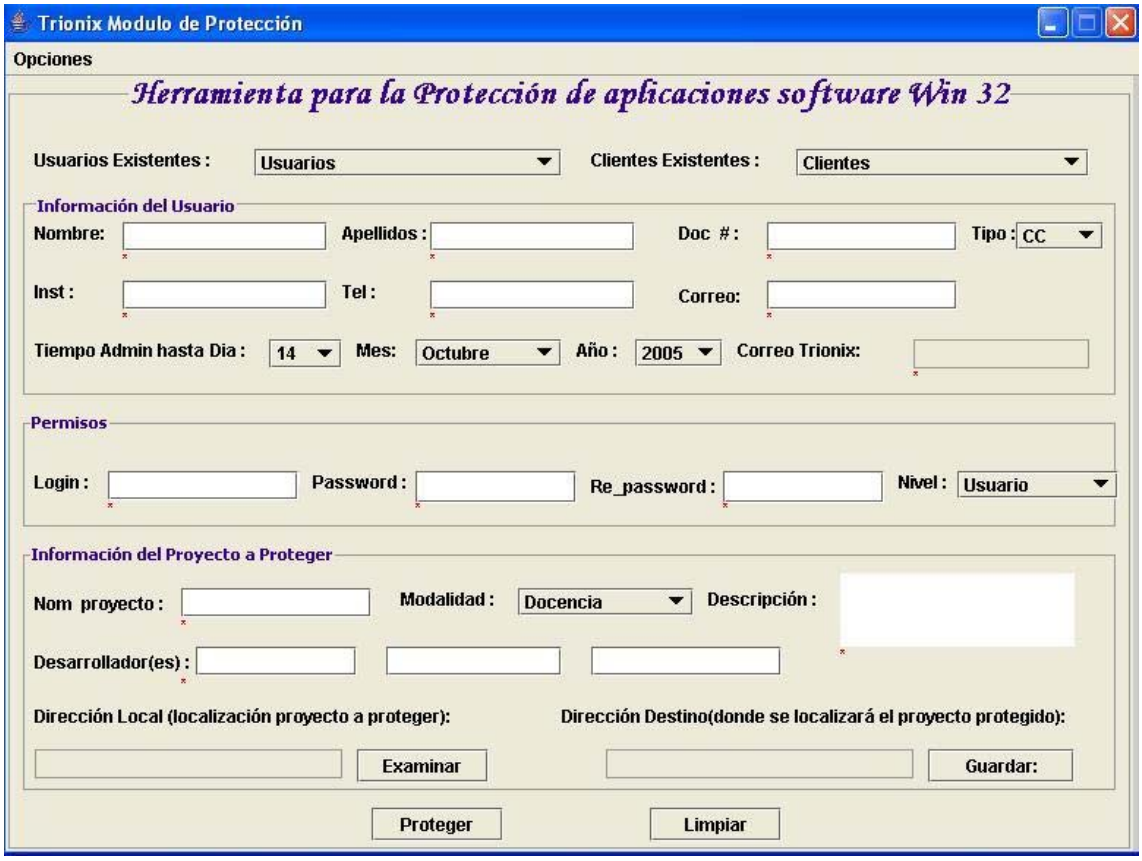



**Figura 6.4** Interfaz final: Modulo de Protección de aplicaciones

✓ **Actividad: Determinar la prioridad de los casos de uso**

| ***Orden de prioridad*** | ***Caso de Uso*** | ***Riesgo*** | ***Impacto*** |
|---|---|---|---|
| 1 | Protección | Identificación de clases y subsistemas | No permitirá determinar una arquitectura correcta |
| 2 | Renovación | Un usuario renueve proyectos ajenos a su dominio | Violación a la información de otro usuario. |
| 3 | Consulta | Confiabilidad de la base de datos | Imposibilidad de renovación y consulta |
| 4 | Administración | Integridad del sistema de acuerdo al perfil de usuario | Deterioro de la confiabilidad de la aplicación Web. |

**Tabla 6.1** Prioridad de los casos de uso

### 6.1.2 EL FLUJO DE TRABAJO DEL ANÁLISIS

El objetivo de esta fase es obtener un producto software en su versión operativa inicial. Este debe contar con ciertas características en cuanto a calidad y aseguramiento de requisitos.

- ✓ **Actividad: Analizar un caso de uso**

Se identifica y detalla el cien por ciento (100%) de requisitos en casos de uso, de manera que se busca satisfacer los requerimientos funcionales trazados al inicio del proceso de desarrollo y durante su evolución.

- **Caso de uso protección**

***Descripción**:* El usuario o representante de una aplicación software solicita su protección al administrador que a su vez puede realizar las siguientes actividades:

- ✓ Celebrar contratos de uso con los usuarios.
- ✓ Recopilar información del usuario.
- ✓ Crear perfil de usuario.
- ✓ Recopilar información del proyecto.
- ✓ Proteger la aplicación.

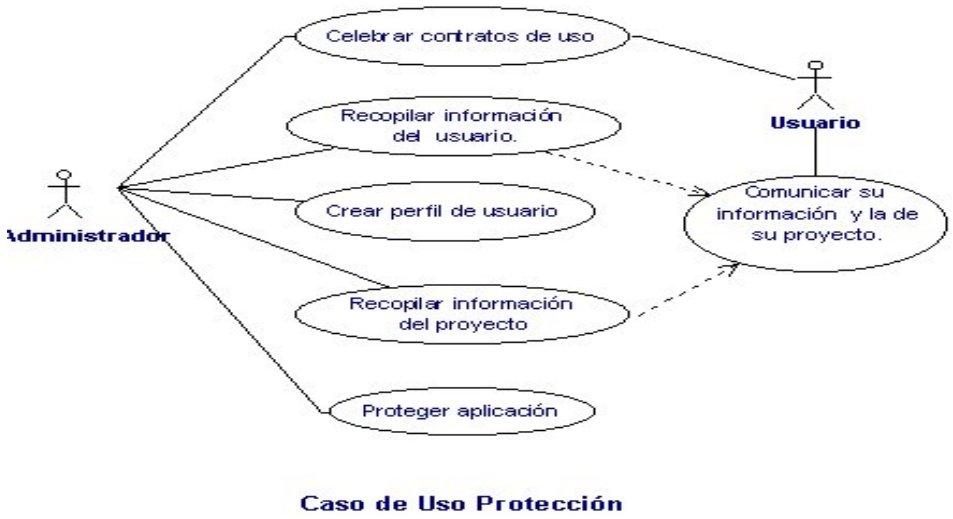



**Figura 6.5** Diagrama de caso de uso protección

- **Caso de uso Renovación**

El cliente solicita renovación o renueva dependiendo del permiso que le ha otorgado el usuario. Para renovar el sistema solicita una clave (pública) que genera TRIONIX cuando caduca la protección, esta clave se valida y si es correcta, solicita unos nuevos parámetros de protección con los cuales se genera la nueva clave de uso (privada).

El usuario podrá negar la actualización de una protección de su dominio en cualquier caso.

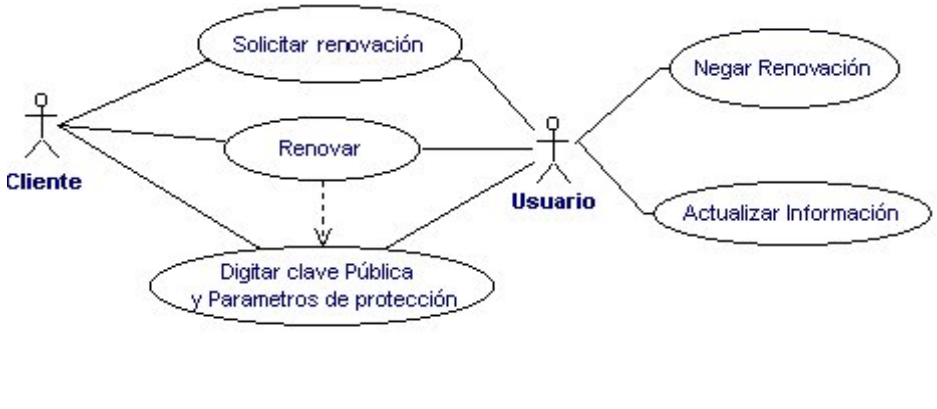



**Figura 6.6** Diagrama de caso de uso Renovación

- **Caso de uso consulta**


**Figura 6.7** Diagrama de caso de uso Consulta

*Descripción:* Tiene acceso a las consultas: el administrador, los clientes, usuarios y cualquier navegante Web; sin embargo para este último las opciones de consultas son generales referentes a que proyectos están protegidos y a que persona pertenece; Las estadísticas se disponen solo para el administrador y cada usuario, donde obtendrán información de su dominio.

- **Caso de uso Administración**

  El administrador es la persona encargada de la regulación y control operativo de la aplicación; puede realizar el mantenimiento a toda la

información que se maneja. Sin embargo es ajeno al mantenimiento de la información de los clientes de los usuarios.

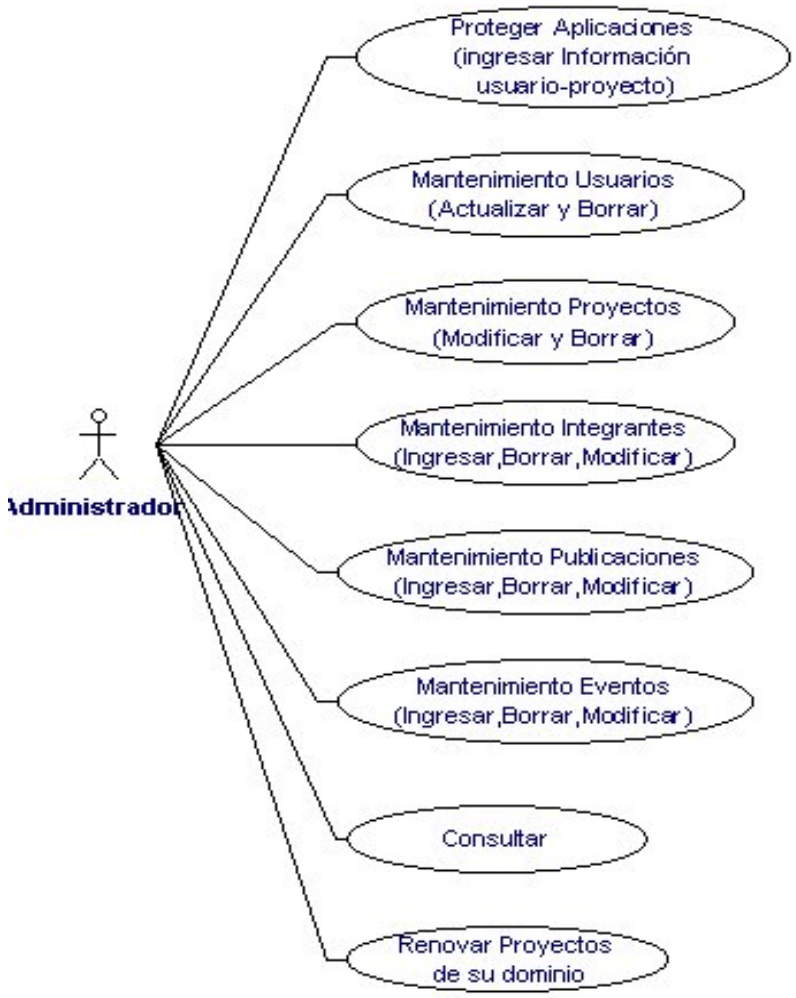



**Figura 6.8** Diagrama de caso de uso Administración

### 6.1.3 EL FLUJO DE TRABAJO DEL DISEÑO

En este flujo de trabajo nos centraremos en los casos de uso que no fueron utilizados para la obtención de la línea base de la arquitectura, teniendo en cuenta que en la fase anterior se diseñaron los casos de uso arquitectónicamente significativos como son: protección, consulta, renovación y administración. Se deja al margen los restantes casos de uso ya que la prioridad que presenta es baja y se considera que han sido detallados suficientemente.

- ✓ **Actividad: representar un caso de uso en clases participantes**


**Figura 6.9** Representación de clases que intervienen en el caso de uso protección

El administrador interactúa con la IU identificación utilizando su login y contraseña, la cual validará el gestor de control verificando con la guardada en la Base de datos, si son correctas permitirá el acceso a la IU de protección, donde el administrador podrá proteger un proyecto.

### 6.1.4 EL FLUJO DE TRABAJO DE LA IMPLEMENTACIÓN

Este flujo representa la versión operativa inicial de la aplicación, la cual recopila el 100% de los casos donde se realiza la interacción de todos los componentes analizados en las fases anteriores.

En el siguiente diagrama se refleja la situación que se presenta cuando un usuario solicita la protección de una aplicación. El administrador ingresa a la IU de protección, la cual le solicitará el llenado de cierta información referente al usuario y la aplicación a proteger, si todo es correcto el gestor de protección cifrará y generará un nuevo ejecutable de la aplicación, el cual se entrega al usuario.

La aplicación protegida inicialmente está vencida y genera una clave (pública) solicitando una nueva (clave privada); entonces el usuario ingresa a renovar digitando dicha clave (pública), el gestor de renovación la valida y corrobora información contenida en esta, con la de la base de datos, si es correcta solicitará los nuevos parámetros de protección y generará finalmente la nueva clave de uso.

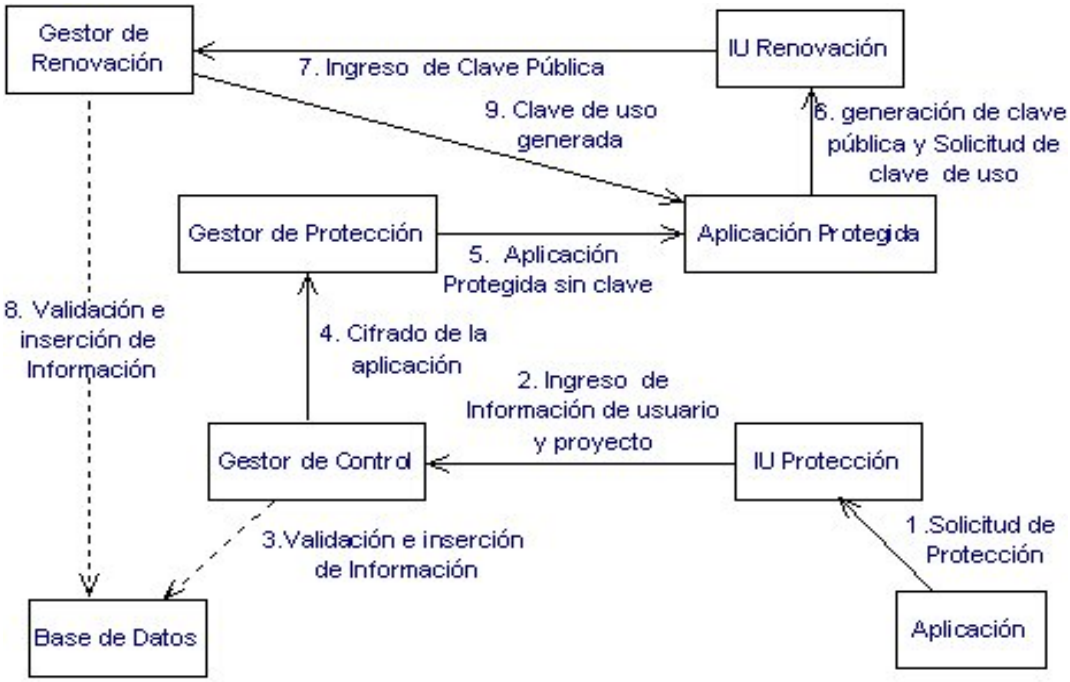



**Diagrama de colaboración de clases del sistema**

**Figura 6.10** Diagrama de Colaboración de Clases del Sistema

- ✓ **Actividad: Implementar una clase e implementar un subsistema**

Las clases se agrupan en sus correspondientes subsistemas. A continuación se implementará y detallará cada una de las clases de los subsistemas de gran importancia en la aplicación.

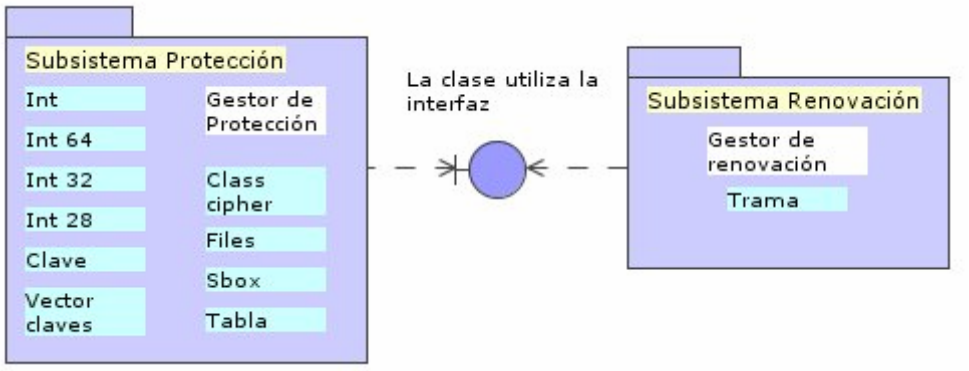



**Figura 6.11** Implementación de clases y subsistemas

La línea de trazo discontinuo de una clase a una interfaz significa que la clase usa la interfaz; esta interfaz permite la asociación de los subsistemas.

#### 6.1.4.1 CLASES DEL ALGORITMO DES

Las clases en donde se encuentra el algoritmo DES implementado están contenidas en la librería de vinculo dinámico llamada desfin.dll. Estas clases están agrupadas en el subsistema de protección.

##### 6.1.4.1.1 CLASES PARA EL MANEJO DE LOS DATOS

El algoritmo DES contiene operaciones de sustitución y de transposición, cada una ellas fue desarrollada en diferentes tipos de clases con el fin de diferenciar al momento de manipular el código fuente.

- **Datos numéricos**

En el siguiente código se encuentran implementada la definición de los diferentes tipos de objetos necesarios para la implementación del algoritmo DES.

```
class Int {
  public:
  virtual void writeBit(int nbit, bool valor) = 0;
  virtual bool readBit(int nbit) = 0;
  virtual Int* clone() = 0;
  void permutation(Tabla tabla);
};
```

*Creada como una clase modelo que servirá para definir os tipos que se utilizan en el cifrado y descifrado del algoritmo DES.*

```
class Int64: public Int {
  protected:
  int64 entero;

  public:
  void setInt(int64 ent){entero = ent;}
  int64 getInt(){return entero;}
  void writeBit(int nbit, bool valor);
  bool readBit(int nbit);
  Int* clone();
  Int32 get32Left();
  Int32 get32Right();
  static Int64 concatenate(Int32 high, Int32 low);
  static Int64 xor(Int64 a, Int64 b);
  void xor(Int64 a);
};
```

*Clase creada para la clave y el texto a procesar.*

*Esta clase hereda del Int.*

*Se encapsulan las funciones básicas como lo es escribir, leer, asignar y operaciones lógicas*

*Clase para realizar las operaciones internas del DES.*

```
class Int32: public Int{
  protected:
          int32 entero;

  public:
```

*Hereda de la clase Int.*

*Con esta clase se soluciona el problema que se presenta en el DES cuando se trabaja con la mitad de la trama.*

```
  void setInt(int32 ent){entero = ent;}
  int32 getInt(){return entero;}
  void writeBit(int nbit, bool valor);
  bool readBit(int nbit);
  Int* clone();
  static Int32 xor(Int32 a, Int32 b);
  void xor(Int32 a);
};
```

*Clase para realizar las operaciones internas del DES.*

*Hereda de la clase Int32.*

```
class Int28: public Int32 {

  public:

  void setInt(int32 ent){entero = ent; clean();}

  void clean(){entero &= 0x0FFFFFFF;}

  void lShift(int desp);
};
```

*Cuenta con funciones como clean que limpia los bits sobrantes (más allá de los 28 de interés).*

```
class Clave: public Int64 {

  public:

  Int28 get28Low();

  Int28 get28High();

  static Clave concatenate(Int28 high, Int28 low);

};
```

*Clase para el manejo de la Clave del algoritmo DES.*

*Hereda de la clase Int64.*

*Cuenta con funciones que permiten obtener la parte alta y baja de las tramas analizadas. También permite concatenar cadenas (unir parte alta con la parte baja)*

- **Tablas de datos.**

Para aplicar el algoritmo DES, se deben realizar transformaciones y permutaciones con base a información consignadas en tablas predefinidas por el estándar DES.

```
class VectorClaves {
  Clave vector[NCLAVES];
   public:
  void setItem(int pos,Clave clave){vector[pos]=clave;}
  Clave getItem(int pos){return vector[pos];}
};
```

*Clase para alojar el vector de claves requerido en el proceso de cifrado y descifrado.*

*Se cuenta con funciones para el manejo de los datos almacenados en el interior del vector.*

```
class Tabla {
        int *tabla;
        int tamano;
```

*Clase implementada para la creación, asignación y manejo de las tablas utilizadas en el DES.*

```
  public:
  Tabla(){tabla = NULL; tamano = 0;}
  Tabla(int *vector, int tam){set(vector, tam);};

  void set(int *vector, int tam);
  int getItem(int i);
  int getLength() {return tamano;}
};
class Sbox {
  static int32 tabla[8][4][16];

   public:
    int32 getItem(int i, int j, int k){ return (tabla[i][j][k]);}
    Int32 sustitute(Int64 data);
};
```

*las*

### ❖ Clase de Agrupación

Para obtener el resultado final del algoritmo DES, se implementa la siguiente clase que utiliza todas las clases descritas anteriormente.

*Clase que encapsula todas las funciones necesarias para el proceso de cifrado y descifrado.*
*Lo que se necesita ya esta implementado como clases, por lo cual se deben crear objetos de dichas clases e invocar las funciones.*

```
class Cipher {
  static int vectorPc1[];
  static int vectorPc2[];
  static int vectorLs[];
  static int vectorIp[];
  static int vectorIpInv[];
  static int vectorBitSel[];
  static int vectorP[];
  Tabla pc1;
  Tabla pc2;
  Tabla ls;
  Tabla ip;
  Tabla ipInv;
  Tabla bitSel;
  Tabla p;
  VectorClaves keys;
  Sbox sbox;
  void generateKeyVec(Clave key, VectorClaves *keyVec);
  Int32 function(Int32 a, Clave key);
  Int64 cipher(Int64 input, VectorClaves *keyVec);
  Int64 decipher(Int64 input, VectorClaves *keyVec);
  public:
  Cipher() {
    pc1.set(vectorPc1, 56);
    pc2.set(vectorPc2, 48);
    ls.set(vectorLs, 16);
    ip.set(vectorIp, 64);
              ipInv.set(vectorIpInv, 64);
              bitSel.set(vectorBitSel, 48);
```

```
    p.set(vectorP, 32);
  }
  void generateKeys(Clave key);
  Int64 cipher(Int64 input);
  Int64 decipher(Int64 input);
};
```

### 6.1.4.2 CLASE PARA EL MANEJO DE TRAMAS

#### 6.1.4.2.1 Clave Pública

```
class Trama {
        ulong parte1;
        int64 parte2;
        ulong parte3;
        ulong parte4;
        char *aux_trama;
        char *trama_1;
        char *trama_2;
        char *trama_21;
        char *trama_22;
        char *trama_3;
        char *trama_4;
        char *trama;
        char *cero;
        char* Armar_trama(ulong tipo_p, ulong tiempo, int64 clave, ulong
        acceso, ulong codigo);
        void Desarmar_trama(char *trama,ulong &tipo_p, ulong &tiempo, int64
        &clave, ulong &acceso, ulong &codigo);
};
```

*Clase utilizada para armar las tramas que serán mostradas en el momento de vencimiento y de renovación.*

*También se utiliza para desarmar las tramas y poder obtener la información que se almacena internamente.*

*Los atributos char* son utilizados como variables auxiliares para el proceso de armado y desarmado.*

#### 6.1.4.2.2 Clave Privada

```
class RSA
{
//Métodos de la clase
public static long TestCoprimo(long a, long b)
public static long Encontrar(long a, long modulo)
public static long SetPrimo(int a)
public static boolean TestPrimo (long n)
public static long GetClavePrivate(long clave_publica_1){
public static long GetClavePublic()
public  String Armar_trama(long tipo_p, long tiempo, long clave, long acceso,
long codigo)
void    Obtener_ClavePrivate(long clave_public_1,long clave_public_2, long
nuevotipo1, long nuevotiempo1,long nuevoacceso1,long codigo_proy1)
void  Obtener_ClavePublic(int clave_public_1, int clave_public_2)
        public String Desarmar_trama(String tramaentrada,long nuevotipo_p,
        long nuevotiempo, long nuevoacceso)
 }
```

```
//validación de claves
class Claves
{
user.Desarmar_trama(tramaentrada,nuevotipo_p,nuevotiempo,nuevoacceso);
}
```

#### 6.1.4.3 CLASES PARA EL MANEJO DE ARCHIVOS

Para poder realizar un manejo eficiente de la fecha, se crea la siguiente estructura de datos:

```
typedef struct {
        int anno;
        int mes;
        int dia;
        int hora;
        int minuto;
        int segundo;
}Fecha_Date;
```

*Clase implementada para la creación, protección y manejo de archivos.*

```
class File {
  public:
  int handle;
  int64 marca;
  int64 marca1;
  int64 marca2;
  int64 marca3;
  int64 marca4;
  int64 marca5;

  void CifrarFile(int handle_1, int handle_2);
  void DescifrarFile(int handle_1, int handle_2, int64 longitud);
  void ArmarFile(int handle_1,int handle_2,int handle_3,int acceso, int tiempo,int
tipo);
  void DesarmarFile(int handle_1, int handle_2);
  void Ubicar_punto(int handle, int64 patron);
  void Ubicar_principio(int handle_1);
  void Ubicar_fin(int handle_1);
  bool Localizar(int64 info, int64 patron);
  int64 Establecer_marca(int64 key);
  void Colocar_fecha(int handle,Fecha_Date fecha,int64 patron);
  Fecha_Date Consultar_fecha(int handle,int64 patron);
  void Colocar_acceso(int handle, int64 acceso);
  int64 Consultar_acceso(int handle);
  int64 Consultar_tamano(int handle);
  void ModificarTipoProteccion(int handle, int tipo);
          int ConsultarTipoProteccion(int handle);
         };
```

*La clase File cuenta con funciones que le permiten localizar información que se encuentra dentro de los archivos protegidos.*

## 6.2 PRUEBAS DE IMPLEMENTACIÓN

Se realizaran pruebas a los subsistemas para comprobar el funcionamiento del sistema. Para esto previamente la base de datos se ha poblado con datos de prueba.

- ✓ **Subsistema de Gestión Sesión y Control**

| **Estado del Sistema** | | |
|---|---|---|
| *El Administrador no ha accedido al sistema, por lo cual no se ha inicializado la variable de sesión.* | | |
| Parámetros | | |
| Nombre | **Valor** | **Descripción** |
| ---------- | ---------- | ---------- |

**Tabla 6.2.** Prueba I al subsistema de Gestión Control

- ✓ **Prueba de validación de datos**

- **Caso 1 Ingreso al sistema.**

| **Datos de Entrada** | | |
|---|---|---|
| *Datos necesarios para validarse en el sistema* | | |
| **Nombre** | **Valor** | **Descripción** |
| *Login* | Administrador | Identificación valida |
| *Contraseña* | Xxxxxxxx | Clave de acceso valida |

**Tabla 6.3**. Caso 1. Datos validos

- **Caso 1.1 Datos validos para Administrador.**

| Datos de Entrada | | |
|---|---|---|
| *Accediendo al modulo de protección* | | |
| **Nombre** | **Valor** | **Descripción** |
| *Opción del sistema* | Proteger | Acceso exclusivo para administrador. |

**Tabla 6.4.** Caso 1.1 Opción para el Administrador

- **Caso 2 Datos inválidos.**

| **Datos de Entrada** | | |
|---|---|---|
| *Datos para realizar la validación de acceso* | | |
| **Nombre** | **Valor** | **Descripción** |
| *Login* | Administrador | Identificación valida |
| *Contraseña* | Xxx | Clave invalida. La longitud debe ser mayor de 6 caracteres |

**Tabla 6.5** Caso 2. Datos inválidos

| Estado del Sistema | | |
|---|---|---|
| *Inicio de sesión. Se ha accedido al sistema.* | | |
| Parámetros | | |
| **Nombre** | **Valor** | **Descripción** |
| *Identificador de perfil* | 1 | Identificador de perfil correspondiente al administrador, presentación de página inicial del administrador. |
| *Identificación de perfil* | 2 | identificador de perfil para el actual usuario |

**Tabla 6.6** Prueba II al subsistema de Gestión Control

- **Caso 2.1 Datos Inválidos para el Administrador**

  En ningún caso presenta restricción de acceso al total de opciones que presenta el sistema, solo que no puede disponer de los clientes de sus usuarios.

- **Caso 2.2 Datos Inválidos para el Usuario**

| Datos de Entrada | | |
|---|---|---|
| *Accediendo al modulo Renovación* | | |
| **Nombre** | **Valor** | **Descripción** |
| *Opción del sistema* | Protección | Proceso no permitido para el usuario. Exclusivo del administrador. |

**Tabla 6.7.** Caso 2.2 Datos inválidos para Usuarios

- ✓ **Subsistema de Gestión de Interfaz**

| Estado del Sistema | | |
|---|---|---|
| *El usuario ya ha accedido al sistema* | | |
| Parámetros | | |
| **Nombre** | **Valor** | **Descripción** |
| *identificador de perfil Id _ usuario* | 2 | Identificador de perfil de usuario correspondiente a un usuario del sistema, presentación de la pagina inicial de usuario |

**Tabla 6.8.** Prueba I al subsistema de Gestión de Interfaz

- **Interfaz para una zona determinada**

| **Estado del Sistema** | | |
|---|---|---|
| *El usuario accede al sistema* | | |
| **Nombre** | **Valor** | **Descripción** |
| *Opción Cliente* | -------- | Opciones con respecto a clientes, Nuevo, Ver, Actualizar y Modificar. |
| Resultado: Datos de Salida | | |
| **Información** | **Valor esperado** | **Valor obtenido** |
| *Zonas a las que tiene acceso* | Clientes<br>Proyectos<br>Renovar<br>Descargas<br>Estadísticas<br>Consultar | tiene acceso<br>tiene acceso<br>tiene acceso<br>tiene acceso<br>tiene acceso<br>tiene acceso |

**Tabla 6.9.** Prueba I de la interfaz

✓ **Subsistema renovación**

| **Estado del Sistema** | | |
|---|---|---|
| *El usuario ha accedido al sistema.* | | |
| Parámetros | | |
| **Nombre** | **Valor** | **Descripción** |
| *ID_Usuario* | 2 | identificador de perfil correspondiente a un usuario |

**Tabla 6.10.** Acceso al subsistema de Renovación

- **Caso 1. Renovar una protección**

| ***Datos de Entrada*** | | | |
|---|---|---|---|
| *Clave pública de un proyecto.* | | | |
| Nombre | **Valor Predeterminados** | **Valores Obtenidos** | **Observación** |
| ***Trama***<br><br>*1* \| *2* \| *3* \| *4* \| *5* \| *6*<br><br>***1.Tipo de protección***<br>***2. Acceso***<br>***3. Clave pública***<br>***4. Tiempo***<br>***5.Código proyecto***<br>***6.Tipo cliente*** | 32868T-24512-3A000-52754-1B000-50A00-20040-00001 | ------ | Trama generada por el sistema una vez expire la protección de una aplicación. |

| | | | |
|---|---|---|---|
| ***Tipo de protección*** | 3 | 49302T-51245-7A000-95684-5B000-80A00-20040-00001 | Selección de tipo de protección. (Acceso y tiempo), actualización de datos y generación de clave privada de aplicación |

**Tabla 6.11.** Renovar una protección

☑ **Caso 2. Validación de datos**

| **Datos de Entrada** | | |
|---|---|---|
| Datos necesarios para realizar el proceso de renovación | | |
| Nombre | **Valor** | **Observación** |
| ***Trama*** | 49252T-12452-1A000-51246-2B00C-50A00-20040-00 | Clave invalida, no correspondiente a la longitud predeterminada: 48 caracteres. Además las letras que componen la claves, son marcas de clase que han sido predeterminadas; por tanto la trama solo estará compuesta por 4 letras a lo máximo |
| **Resultado de salida** | | Ninguno. Debido a que no se pudo realizar el proceso de renovación. |

**Tabla 6.12.** Caso 2 Validación de datos

✓ **Subsistema de Consulta**

| **Estado del Sistema** | | |
|---|---|---|
| Un cliente accedió al sistema. Se ha iniciado sesión. | | |
| Parámetros | | |
| Nombre | **Valor** | **Descripción** |
| ***ID_CLIENTE*** | 3 | identificador de perfil correspondiente a un cliente. |

**Tabla 6.13.** Acceso al sistema como cliente

☑ **Caso 1. Consultar.**

| **Datos de entrada** | | |
|---|---|---|
| Estar en Sesión | | |
| Parámetros | | |
| Nombre | **Valor** | **Descripción** |
| ***Consultar*** | Proyectos que uso | El cliente ve los proyectos que usa y su protección (días y accesos). |

**Tabla 6.14.** Caso1 Consultar

✓ **Subsistema de Protección**

- **Crear Protección**

| **Datos de Entrada** | | |
|---|---|---|
| Nombre | **Valor** | **Descripción** |
| ***Nombre_usuario*** | Usuario1 | Recopilación de información referente al usuario. |
| ***Apellidos_usuario*** | Apellidos1 | |
| ***Documento #*** | 95236541 | |
| ***Tipo_documento*** | C.C | |
| ***Institución*** | UIS | |
| ***Telefono*** | 6412589 | |
| ***correo*** | correo@hotmail.com | |
| ***correoTrionix*** | prueba@trionix.co | |
| ***Tiempo _ administración*** | 2005-10-25 | |
| ***Login*** | Prueba | |
| ***Password*** | ****** | |
| ***Re-Password*** | ****** | |
| Nombre | **Valor** | **Descripción** |
| ***ID_Proyecto*** | 1 | Identificador de proyecto |
| ***Nombre_Proyecto*** | Proy1 | Nombre de la aplicación a proteger |
| ***Fecha_creación*** | 2005/05/01 | Fecha de creación del proyecto |
| ***Desarrollador*** | 3 | Nombre de los desarrolladores de la aplicación |
| ***Descripción*** | Es un proyecto prueba | Descripción de la aplicación |
| ***Dirección Origen*** | C:\Mis Documentos\prueba.exe | Dirección origen donde se encuentra la aplicación a proteger. |
| ***Dirección Destino*** | C:\Mis Documentos\destino.exe | Dirección destino donde se depositará el nuevo ejecutable de la aplicación. |
| ***Resultado de Salida. Protección de la aplicación.*** | | |

**Tabla 6.15.** Creación de un proyecto

☑ **Validación de datos**

| ***Precondición***: El usuario ha creado un cliente y le ha otorgado un permiso para ingresar a la aplicación Web y para renovar. | | | | |
|---|---|---|---|---|
| ***Estado del Sistema*** | | | | |
| ***Inicio de sesión de un Cliente y Validación de clave pública y permiso para renovar.*** | | | | |
| Nombre | **Valor** | **Descripción** | **Observación** | |
| ***Tipo_proteccion*** | 3 | Protección por acceso y tiempo | Valido | Se debe seleccionar el número de días y accesos a usar la aplicación. |
| ***Accesos*** | 50 | Protección por 50 accesos | Valido | Valor debe ser estrictamente mayor que cero (0) y no puede exceder, los días de diferencia entre la fecha actual y la fecha de administración del usuario dueño de la aplicación. |
| ***Tiempo*** | 20 | Protección por 20 días | Valido | Valor en días mayor que cero (0) y no exceder las 200 ejecuciones. |

**Tabla 6.16.** Validación de datos

Estas pruebas se complementan con las documentadas en el Capitulo 8 y las realizadas a los usuarios finales (anexo 3).

**PARTE III**

## PROBANDO EL SISTEMA

En esta parte se muestra de manera detallada, los procesos más importantes que encierran la aplicación: Protección y Renovación, además se describe cada una de las opciones con que cuenta la aplicación para facilidad de los usuarios. También se realizan pruebas al sistema tomando determinados procesos claves, buscando reducir al mínimo los errores

**CAPITULO 7**

# ASPECTOS RELEVANTES

## 7.1 ADMINISTRACIÓN

El modulo de administración de la aplicación Web cuenta con todas las opciones del sistema, las cuales se encuentran entrelazadas para un óptimo funcionamiento. A continuación se realizará una breve explicación de cada una de estas opciones.

### OPCIONES NO DISPONIBLES EN LA WEB

### 7.1.1 PROTEGER UNA APLICACIÓN

Esta opción no se encuentra disponible en la Web, por lo cual es necesario establecer un encuentro personal entre el administrador y usuario, quienes celebran un contrato de administración; una vez realizado, el administrador se dispondrá a introducir la información del usuario y del proyecto a proteger; remotamente accederá a la base de datos y guardará esta información. A continuación se describe en detalle el proceso de protección.

**Condición inicial**

Inicialmente se encuentran las aplicaciones en su estado original, es decir como archivos ejecutables en el disco duro del administrador para poder ser sometidas al proceso de protección. Las tres aplicaciones son:

- ✓ **Aplicación a proteger**
  Archivo ejecutable de la aplicación que se desea proteger.

- ✓ **Protector**
  Archivo ejecutable de la aplicación encargado de verificar los permisos otorgados (aplicación de TRIONIX).

- ✓ **Pprotector**
  Archivo ejecutable de la aplicación encargado de monitorear el correcto flujo de eventos durante la ejecución de la aplicación protegida (aplicación de TRIONIX).

Deben existir estas dos últimas aplicaciones (protector y pprotector) para que al momento de la ejecución de la aplicación protegida por parte del cliente, protector verifique los permisos y pprotector asegure que protector nunca será finalizado estando en ejecución la aplicación a proteger.

De ocurrir esto, pprotector debe finalizar y borrar la aplicación a proteger que se ha descifrado en el disco.

#### 7.1.1.1 PROCESO DE PROTECCIÓN DE APLICACIONES

Para poder llevar a cabo el proceso de protección se deben seguir una serie de pasos que permitan obtener los recursos necesarios para proteger la aplicación. Estos pasos son:

- ✓ **Paso 1**. Celebración de contrato entre Administrador-Usuario (Período en el cual el usuario administrará sus proyectos y sus clientes[15]).
- ✓ **Paso 2**. Capturar información relacionada con:
  - **Usuario y otorgamiento de permiso de acceso a la aplicación Web** (Login y Password).
  - **Aplicación a Proteger**.
    Dirección donde se localiza la aplicación a proteger (dirección local en el equipo del administrador).
  - **Ruta de destino**. Dirección en la cual se desea grabar el nuevo ejecutable correspondiente a la aplicación protegida (dirección local).
  - **Tipo de protección**. Protección asignada automáticamente: Ejecuciones (valor predeterminado = 0), Tiempo (valor = 0), tipo de protección (valor predeterminado = 3).

- ✓ **Paso 3.** Creación de un nuevo archivo ejecutable. Luego de cumplir con el paso 1, 2 y las condiciones iniciales, se procede a armar el nuevo archivo ejecutable de la siguiente manera:

  ☑ **Abrir el protector para copiarlo bit a bit desde el principio**, en el nuevo archivo ejecutable.
  - **Armar tramas** (ver numeral 5.2.4.1) con la información obtenida en el paso 2. Luego, se procede a copiar en el nuevo archivo ejecutable cada una de las tramas, colocando una bandera que las identifique. Estas banderas son números de 64 bits que se establecen de manera arbitraria, pero por seguridad se cifra con el algoritmo DES[16], con la misma clave utilizada para cifrar la aplicación a proteger.

  ☑ **Abrir la aplicación a proteger para cifrarla** con el algoritmo DES(seleccionado por su eficiencia y seguridad, ver Anexo 1) en paquetes de 64 bits en intervalos variables de paquetes. La razón por la cual no se cifran todos los paquetes es por motivo de tiempo de ejecución y recursos consumidos, tanto al momento de la protección como en la ejecución de la aplicación protegida por parte del cliente, siendo esta ultima uno de los puntos críticos del proceso, antes de copiar esta aplicación se debe colocar una bandera para identificarla.

[15] Persona(s) que usa(n) aplicaciones protegidas
[16] Data Encryption Standard, algoritmo criptográfico implementado para la protección de aplicaciones por su seguridad y eficiencia. Ver anexo 1.

☑ **Abrir la aplicación protector para copiarla** a continuación de la aplicaron a proteger byte a byte. Para poder identificarla dentro del nuevo archivo ejecutable se debe colocar una bandera.

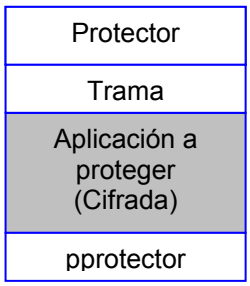



**Figura 7.1** Estructura de la aplicación protegida

Este proceso de protección tiene un tiempo de duración proporcional al tamaño del archivo a proteger.

TRIONIX tiene incluido todos los algoritmos necesarios para el cifrado y elaboración de nuevos ejecutables, haciendo de la protección una labor transparente.

#### 7.1.1.2 EJECUCIÓN DE UNA APLICACIÓN PROTEGIDA

Un archivo ejecutable consta del protector, tramas, aplicación a proteger y pprotector.

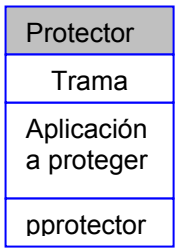



**Figura 7.2** Estructura de la aplicación protegida

Este archivo al momento de ser ejecutado, empieza por la primera aplicación contenida en su interior (protector), la cual se encarga de realizar la validación de los permisos otorgados, comparando con la información contenida en tramas almacenadas dentro del archivo de la aplicación protegida con la información del sistema como fecha y hora actual. Si estos permisos son validos, se descifra la aplicación a proteger que se encuentra en el archivo de la aplicación protegida y la graba como un archivo ejecutable en el directorio Windows/System32, al igual que la aplicación pprotector.
Esto se realiza para poder ejecutar ambas aplicaciones (aplicación a proteger,

pprotector) como subprocesos.

La ejecución de las aplicaciones como subprocesos requiere que el proceso padre (protector) cree un hilo por cada subproceso a crear. Esto se realiza para poder tener algún control sobre los subprocesos hijos. En caso que el proceso padre sea eliminado, los subprocesos hijos se independizan y pasan a ser procesos sin padre. En este tipo de protección planteada esto traería graves consecuencias, ya que en caso de eliminar el protector, la aplicación a proteger que ha sido descifrada y gravada en el disco como un archivo ejecutable no podría ser controlada para impedir que se ejecute por sí misma. Por esta razón se utiliza la aplicación pprotector, que tiene como función principal "proteger" a la aplicación protectora; esta protección se basa en que no puede permitir que el protector sea cerrado antes que la aplicación a proteger. También tiene como tarea actualizar la información contenida en tramas dentro de la aplicación protegida como lo es: número de acceso, número de días de uso de la aplicación y última fecha de acceso. Esta información es almacenada para que en la próxima ejecución de la aplicación protegida, el protector la pueda extraer y realizar la validación de los permisos otorgados.

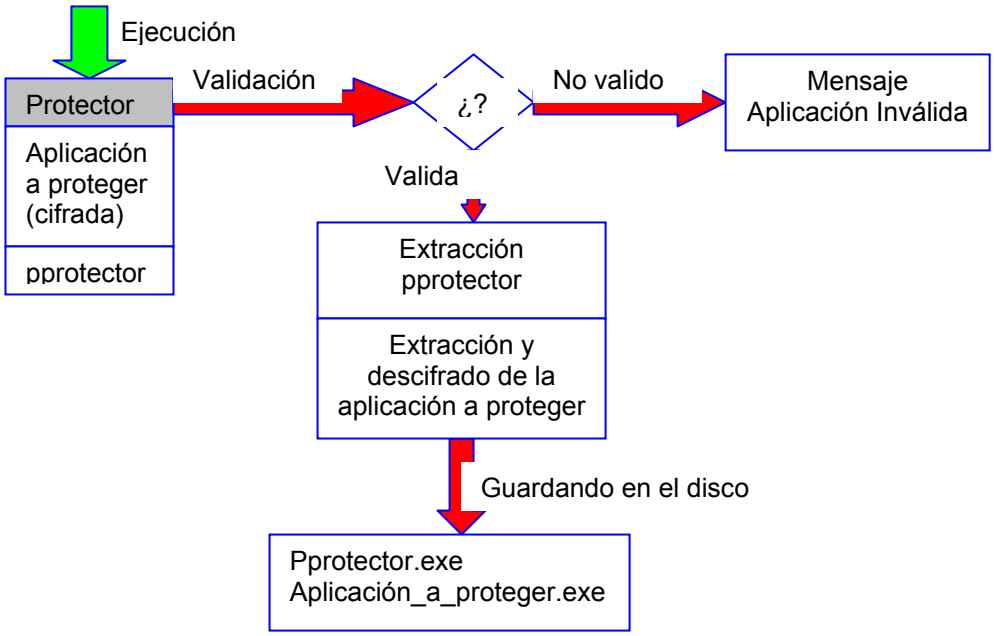



**Figura 7.3** Validación de la aplicación protegida y extracción de aplicaciones

La aplicación pprotector permanece oculta en ejecución y solo reaccionara ante los eventos:

- ✓ **Finalización del proceso protector.**
  Pprotector debe finalizar el proceso de la aplicación a proteger y eliminarla del disco duro (máquina del administrador) para que esta no pueda ser ejecutada sin protección.

- ✓ **Finalización de la ejecución de la aplicación a proteger.**

Finaliza el proceso de la aplicación protector y elimina del disco duro (máquina del administrador) el archivo correspondiente a la aplicación a proteger.

De esta manera se espera que la aplicación a proteger no sea copiada cuando se encuentra descifrada en el disco; solo seria posible copiarla cuando se termina de grabar (fin del proceso descifrar), pero inmediatamente comienza su ejecución, lo cual por seguridad el sistema operativo impide copiar un archivo en ejecución. Cuando la aplicación a proteger deja de estar en ejecución es borrada por pprotector de manera que en ningún momento puede ser copiada. Además de esto se toman medidas que mejoran la seguridad de la aplicación a proteger cuando se encuentra en disco descifrada, como son:

- ☑ El nombre de este archivo en el disco es generado de manera aleatoria. Su longitud es de mínimo cuatro (4) caracteres y máximo diez caracteres (10) que comprender mayúsculas y minúsculas sin incluir signos.

- ☑ Al terminar de descifrar el archivo, se coloca invisible al usuario.

Se espera dificultar la labor de cualquier persona que intente violar el esquema de seguridad aplicado, aunque se supone que el equipo del administrador es de uso exclusivo y que este debe tomar medidas propias de seguridad (clave de acceso).

#### 7.1.1.3 Primera ejecución

La primera vez que se ejecuta la aplicación protegida aparece en un estado inactiva, mostrando un serial y solicitando otro para su comparación. Para ello se debe ingresar a la aplicación Web identificándose con login y contraseña, y en el modulo de renovación ingresar la clave mostrada y obtener la nueva clave de uso, a partir de los parámetros de protección establecidos (tiempo y accesos).

#### 7.1.1.4 Vencimiento

Cuando una aplicación se vence (por tiempo o acceso) la aplicación protector invoca funciones que le permiten generar la clave pública y la trama que le será suministrada al usuario al momento de iniciar la ejecución de la aplicación protegida. La generación de la clave (serial) se realiza mediante el algoritmo RSA[17] que esta implementado en una librería dll. Esta librería es invocada por la aplicación protector cuando alguno de los permisos otorgados se vence.

### OPCIONES DISPONIBLES EN LA WEB

La siguiente figura ilustra la interfaz inicial del administrador del sistema con

[17] El algoritmo RSA se utiliza para la validación y generación de claves. ver Anexo 1

cada una de las opciones con que este dispone:

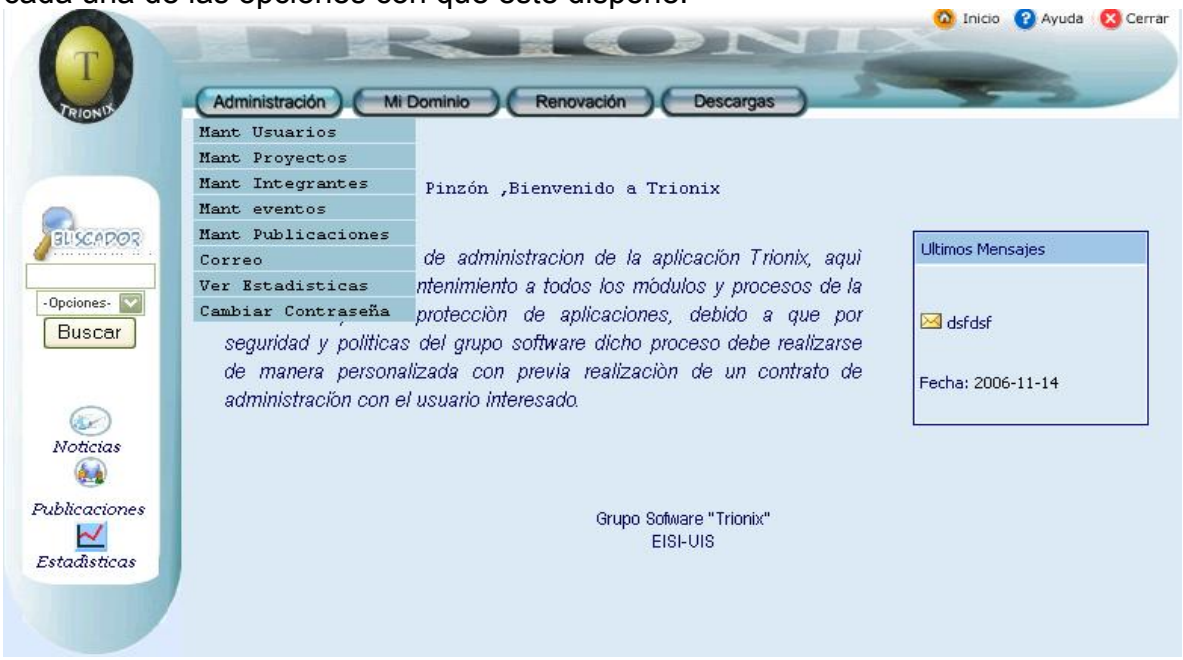



**Figura 7.4** Interfaz del Administrador

### 7.1.2 MANTENIMIENTO DE USUARIOS

Se entiende como usuario al dueño patrimonial de una aplicación software protegida, quién puede administrar sus aplicaciones otorgando permisos de tiempo y accesos por medio de la aplicación Web a sus clientes, durante el período convenido con el administrador a la hora de proteger inicialmente una aplicación (necesariamente un usuario debe tener una aplicación software protegida).

La opción Mantenimiento de usuarios permite al administrador del sistema la actualización de la información del usuario, sus permisos, período de administración, así como la de sus aplicaciones; el borrado (inactivación) se puede realizar posteriormente al vencimiento del periodo actual de administración del usuario y que este ultimo no exprese su interés de renovar o incumpla con alguna de las condiciones pactadas inicialmente.

### 7.1.3 MANTENIMIENTO DE PROYECTOS

Se presenta el listado de las aplicaciones protegidas con las opciones de actualizar y borrar, el ingreso de datos de proyectos se realiza a la hora de su protección.

- ***Actualizar.***
  La información referente a una aplicación protegida puede ser actualizada por el administrador o el usuario responsable. Una aplicación presenta la opción de permitir su descarga, con la cual cualquier navegante Web podrá hacerlo, registrándose y usándola por el período de tiempo y/o accesos

establecidos por el usuario; el no permitir la descarga le dará la posibilidad de publicación a la aplicación, es decir aparecerá en el paquete de software no libre(pagina Principal) como una forma de darle divulgación a la aplicación y un contacto entre el navegante o cliente y el usuario por medio de la opción de envió de correo expresando su interés o alguna inquietud.

- ***Borrar.***

El administrador del sistema podrá borrar una aplicación software de algún usuario siempre y cuando esta no sea utilizada por algún cliente, de lo contrario implicaría la in activación de quienes la usan. Si es el único proyecto del usuario, de inmediato le vencería su periodo de administración; esta opción se deja disponible en especial para el caso de borrar un proyecto que no se deseaba proteger o que erróneamente se creo.

### 7.1.4 MANTENIMIENTO DE INTEGRANTES

Opción para el ingreso, actualización y borrado de los integrantes del grupo software TRIONIX, para dar un reconocimiento a cada uno de ellos por su labor.

### 7.1.5 MANTENIMIENTO DE PUBLICACIONES

Este es el espacio para divulgar el trabajo intelectual del grupo: artículos, ponencias, o trabajos de otros autores referentes a la línea investigativa del grupo, estas publicaciones presentan la opción de descarga para que cualquier persona lo haga, lea y conozca acerca del tema de criptografía y seguridad informática.

### 7.1.6 MANTENIMIENTO DE EVENTOS

Espacio para las noticias y eventos del grupo software, las cuales se visualizaran las cuatro (4) más recientes en la pagina principal de la aplicación.

### 7.1.7 ESTADISTICAS

Soporte para el administrador para la toma de decisiones, allí podrá conocer el comportamiento mes a mes de los ingresos al sistema, el número de usuarios activos e inactivos, el número de proyectos, el más usado y el menos usado. Con dicha información se proponen hacer variantes en los procesos o proponer soluciones para optimizar el funcionamiento de la aplicación.

### 7.1.8 CORREO

Permite la intercomunicación entre todos los actores del sistema, envió y recepción de solicitudes referente a una aplicación o simplemente como canal de comunicación entre los usuarios y clientes de TRIONIX. Todo correo tiene una vigencia de 30 días, posteriormente se borrará automáticamente.

### 7.1.9 CAMBIAR CONTRASEÑA

Opción para que el administrador varié su contraseña. Por seguridad se recomienda cambiarla una vez por mes y guardarla físicamente en un lugar seguro.

Además de las opciones anteriores se cuenta con la opción de hacer backup a la base de datos obteniendo el archivo SQL para restaurarla en un momento dado o simplemente como copia de seguridad.

## 7.2 MI DOMINIO

El administrador del sistema es un usuario con el privilegio de disponer de todas las opciones que presenta el sistema. A diferencia de un usuario normal no es necesario que tenga al menos un proyecto protegido por lo que esta restricción no aplica para él. En mi dominio él ingresa si tiene proyectos protegidos y realiza las operaciones como si fuese un usuario normal y administra con sus criterios los proyectos y clientes.

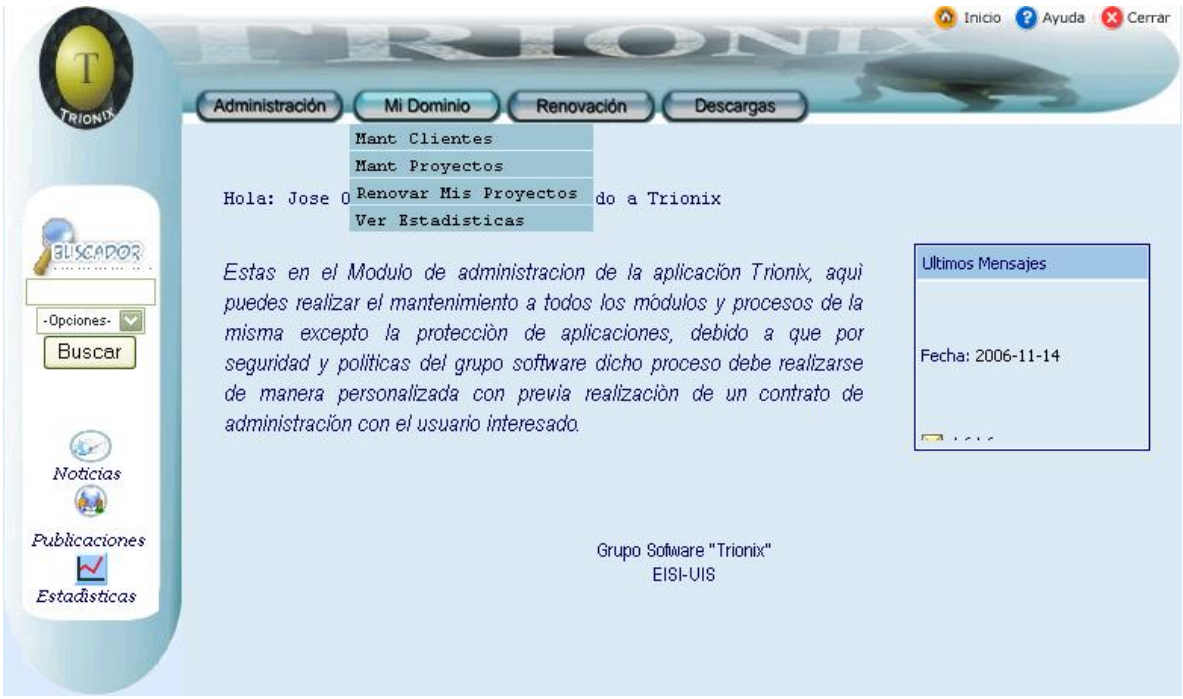



**Figura 7.5** Interfaz del Administrador como Usuario

### 7.2.1 MANTENIMIENTO DE CLIENTES

Un usuario cuando protege una aplicación lo hace para poder divulgarla y tener un control sobre ella. La aplicación Web le brinda esta alternativa, donde el usuario podrá crear sus clientes, otorgar permisos de acceso y tiempos de uso, en busca de garantizar el licenciamiento de uso para toda aplicación.

### 7.2.2 MANTENIMIENTO DE PROYECTOS

El usuario podrá actualizar o modificar la información de una aplicación de su dominio. Goza con iguales opciones que el administrador del sistema en el mantenimiento de proyectos, con la diferencia que podrá hacerlo sólo con sus aplicaciones protegidas y no con la de otros usuarios. *Ver numeral 7.1.3 ADMINISTRACIÓN MANTENIMIENTO DE PROYECTOS.*

### 7.2.3 RENOVACIÓN

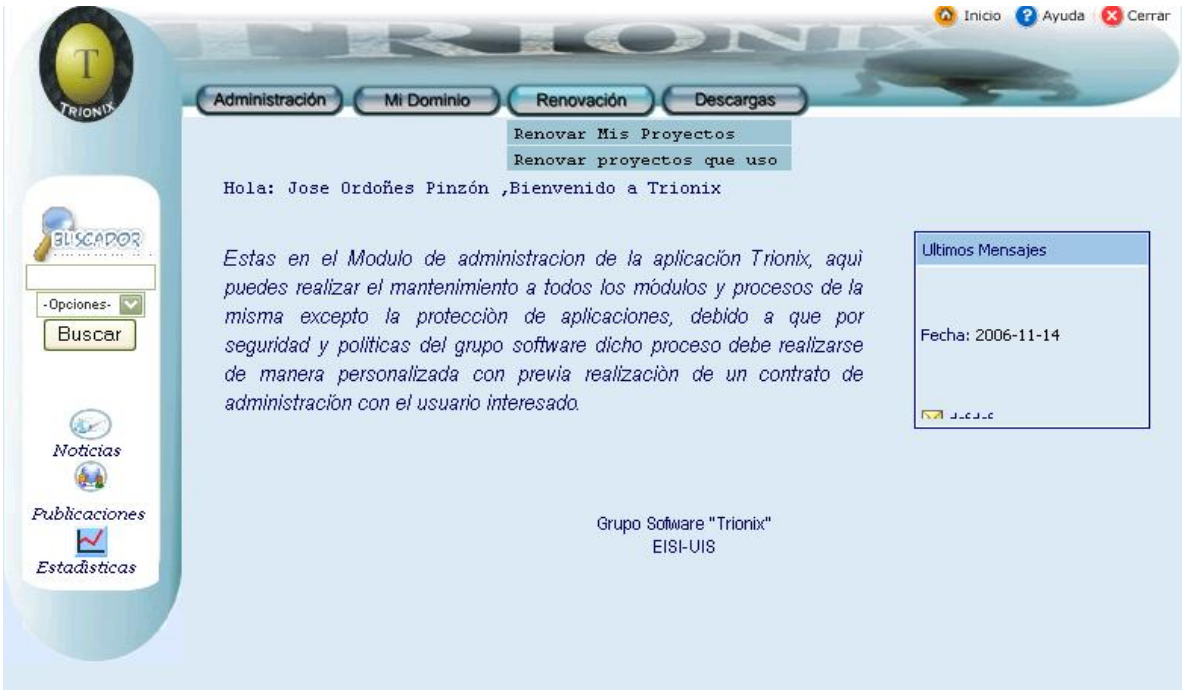



**Figura 7.6** Opciones de Renovación

Una aplicación cuando se protege inicialmente está vencida. Cuando se ejecuta la aplicación por primera vez o cuando caduque el período de protección, TRIONIX genera una clave (pública) y solicita una clave privada (serial), lo que obliga a quien lo usa a renovar el período de protección y obtener la clave de uso.

Para este proceso la aplicación Web presenta el módulo de renovación el cual solicita la clave pública, realiza una validación verificando la longitud de la clave, si cumple satisfactoriamente con todas las condiciones se constituirá en una clave verdadera y verificará el tipo de Cliente: Automático, No Automático;
si es el primero permitirá la escogencia de los parámetros para el nuevo período de protección (tiempo y accesos); posterior a esto invocará el algoritmo RSA quien revalidara la clave para determinar la validez y autenticidad de la misma. Si lo es, generará con el mismo algoritmo y con los nuevos parámetros de protección la nueva clave de uso particular de la aplicación (clave privada); en cambio si el cliente es no Automático le mostrará un mensaje con la opción de ir y modificar los permisos que el había otorgado inicialmente.
A continuación se explicara en detalle el manejo de claves (trama) como se

obtiene y cual es su estructura:

#### 7.2.3.1 TRAMA

La trama es una estructura de datos utilizada para el transporte de información. Su estructura está definida según las necesidades presentadas en el proceso de protección y renovación.

La trama cuenta con información correspondiente a:

**Tipo de protección.**

Existen tres (3) tipos de protección:

1. **Acceso:** se controla el número de acceso, por defecto 0 y máximo 200.
2. **Tiempo:** controla el número de días a utilizar establecido por el usuario o dueño de la aplicación.
3. **Accesos y tiempo:** En caso de realizar este tipo de protección, se controla según lo descrito anteriormente en el numeral 1 y 2 (Acceso y Tiempo).

- **Acceso**

Al momento de vencimiento, la aplicación protector[18] incluirá en la trama el número de accesos realizados durante la protección.

En caso de no renovar por acceso, la trama incluye un cero (0) en esta parte de la trama. El valor máximo de accesos es 200; sí la protección se desea por más se debe proteger por días.

- **Tiempo**

Al momento de renovación, TRIONIX incluye en la trama el tiempo máximo de duración dado en días. Al momento de vencimiento, la aplicación protector incluirá en la trama el tiempo de uso dado en días; este tiempo es el número total de días en los cuales se uso la aplicación protegida. El número máximo de días a renovar está en función del tiempo de administración, es decir, sólo se puede renovar máximo por el número de días que le resta al usuario para que finalice su período.

En caso de no realizar una protección por tiempo se incluye en la trama un valor igual a cero (0).

- **Clave**

Cuando alguno de los permisos otorgados a la aplicación protegida(tiempo y/o accesos) se vence, mediante un llamado a la librería mdtrio.dll[19] que contiene el algoritmo RSA para la generación de claves, la aplicación protector obtiene la clave publica y la incluye en la trama para que pueda ser recibida por la aplicación Web mediante el proceso de renovación.

[18] Archivo ejecutable de la aplicación estándar encargada de verificar los permisos otorgados

[19] Biblioteca de vínculo Dinámico que se invoca para generar y validar claves, ver Anexo 4.

Si la trama es correcta, la aplicación Web TRIONIX extrae la clave pública e invoca la clases Claves y RSA que contienen el RSA, para que validen la clave actual y genere la clave privada. Para obtener la clave privada reúne toda la información necesaria para armar la trama con los nuevos parámetros y visualiza la nueva clave.

- **Código**

Al momento de proteger la aplicación se incluye en la trama el código del proyecto y se diferencia un proyecto de otro por el código de quien lo usa, Ejemplo:

| *Código Proyecto* | *Código Cliente que lo usa* | *Nombre Cliente* |
|---|---|---|
| 1 | 2 | Darwin Rey |
| 1 | 5 | Pedro González |

Como se visualiza en el Ejemplo, aunque Darwin Rey y Pedro González usan el proyecto 1, éste se diferencia por sus códigos, por lo cual se garantiza que cualquier actualización o renovación de un proyecto no implica el cambio en el otro por lo que son completamente independientes.

Al momento de renovar, La aplicación Web TRIONIX se encarga de extraer de la trama el código del proyecto y realiza una validación de validez, consultando en la Base de Datos que ese proyecto es usado por el cliente o usuario en sesión, si lo es, entonces la clave es verdadera, e invoca al RSA que hará la validación final.

### 7.2.3.2 ESTRUCTURA DE LA TRAMA

La trama utilizada está compuesta por números y letras; se agrupan los datos en ocho (8) paquetes separados por el signo menos (-); el primer bloque está compuesto por 6 caracteres y los siguientes paquetes están formados por grupos de 5 caracteres. Los (-) son banderas que se tienen para la validación preliminar.

La trama se construye por medio de una función que se encuentra en la librería dll para el manejo de tramas (mdtrio.dll) y por la clase RSA. Se envía los parámetros a la función la cual retorna la trama como una cadena de caracteres.

```
char* Armar_trama ( ulong tipo_p, ulong tiempo, int64 clave, ulong acceso, ulong codigo, ulong tipo_cli)
```

Encabezado de la función utilizada para la generación de la Clave Pública.

```
public  String Armar_trama(long tipo_p, long tiempo, long clave, long acceso, long codigo)
```

Encabezado de la función utilizada para la generación de la Clave Privada.
Las funciones que se encuentran en la librería para el manejo de tramas

trabajan con desplazamiento de bits para poder armar las tramas que serán devueltas. Si enumeramos bit a bit la trama de izquierda a derecha se puede detallar como está compuesta:

- ✓ **Tipo de protección.** Tiene como longitud 2 bit. Ocupa el bit 1 y 2.
- ✓ **Tiempo.** Tiene de longitud 16 bits, empieza en el bit 3 y termina en el bit 18.
- ✓ **Clave.** Tiene longitud de 64 bit. Va del bit 19 al 83 bit.
- ✓ **Accesos.** Longitud 16 bit. Ocupa del bit 84 al bit 100;
- ✓ **Código Proyecto.** Longitud 32 bit. Ocupa del bit 101 al bit 132

Un ejemplo de la trama es el siguiente:

***49154T-0A000-00000-11559-B0000-3A000-20040-00001***

### 7.2.4 UTILIDAD DE LA TRAMA

La trama es de gran utilidad en los procesos de renovación y protección, ya que facilita el transporte de información. Cuando una aplicación se quiere renovar, se debe ingresar la clave devuelta por la aplicación Web TRIONIX después de realizado el proceso de renovación. Si esta clave es valida para la aplicación protegida, el protector extrae el número de acceso máximo, tiempo de uso máximo y tipo de protección, para sobrescribir la información que se encuentran en las tramas internas de la aplicación protegida y reiniciar el monitoreo.

*Nota. Cuando la aplicación se vence se debe renovar sin cerrar la aplicación vencida, ya que cada vez que se ejecuta, se genera una clave diferente. Esto se hace para evitar que la clave pública a suministrar sea alterada por medio de la copia o reemplazo*.

### 7.2.5 RENOVAR MIS PROYECTOS


**Figura 7.7** Renovando Mis proyectos

El usuario puede renovar los proyectos de su dominio, ya sea por solicitud de un cliente o si ve que es necesario hacerlo. Para esto selecciona el cliente a renovar y cual de los proyectos que éste usa. Luego de esta selección, digita la clave que el cliente le suministre, selecciona los parámetros de protección (si el cliente es automático) y el sistema le generara la nueva clave que el usuario debe suministrarle a su cliente, de lo contrario le modificara los permisos(cliente no Automático) o negará en definitiva la renovación.
*Nota: En caso de que un cliente atrase la fecha de su equipo buscando* ganar un mayor tiempo de uso, TRIONIX de inmediato vencerá la aplicación, cancelara los permisos de renovación y emite un correo al usuario responsable para que tome las medidas del caso sobre el cliente.

### 7.2.6 RENOVAR PROYECTOS QUE USO

TRIONIX maneja 3 perfiles de usuario: administrador, usuario y cliente. Es flexible al cambio de perfil, es decir un administrador puede ser usuario, si tiene aplicaciones protegidas y cliente, si usa aplicaciones de otros usuarios; de similar manera ocurre con un usuario, que puede ser cliente de otro usuario; y un cliente puede constituirse en usuario si protege una aplicación de su dominio; para estos casos la aplicación Web le da privilegio al mayor de los perfiles; es decir si un usuario es cliente le deja el perfil de usuario, y si un cliente se constituye en usuario le cambia su perfil a usuario, aunque para el caso de renovación se conservaran las condiciones de uso y restricciones como cliente que hayan sido establecidas por los dueños de las aplicaciones.


**Figura 7.8** Renovar proyectos que uso

En el caso de la opción de Renovar proyectos que uso en el módulo de administración, quien se identifica es el administrador del sistema, si esta usando proyectos de otro usuario, la renovación se restringe a lo establecido por el dueño de la aplicación a renovar, por lo que debe digitar la clave publica verificar que clase de cliente es, y seleccionar los parámetros de protección (Cliente Automático: días y accesos) o simplemente remitir un correo al usuario expresando su interés por renovar.

### 7.2.7 ESTADISTICAS

Sirve como soporte al usuario para conocer el comportamiento de accesos de sus clientes, sus proyectos y de esta forma conservar o modificar sus criterios de administración.

### 7.2.8 CORREO

Canal de comunicación con sus clientes y otros usuarios, para el envió y recepción de solicitudes referentes a las aplicaciones protegidas.

### 7.2.9 DESCARGAS

Esta opción permite la descarga de aplicaciones protegidas que puede usar el administrador durante un período de tiempo y/o accesos que ha determinado el dueño del proyecto, esta característica se le otorga a la aplicación a la hora de actualizar su información.

## 7.3 USUARIO

El modulo de usuario presenta ciertas opciones para garantizar la administración de clientes y proyectos. No se entrar a detallar las opciones pues en el numeral 7.2 MI DOMINIO, esta descrito para el administrador quien en esta opción se comporta simplemente como un usuario más.

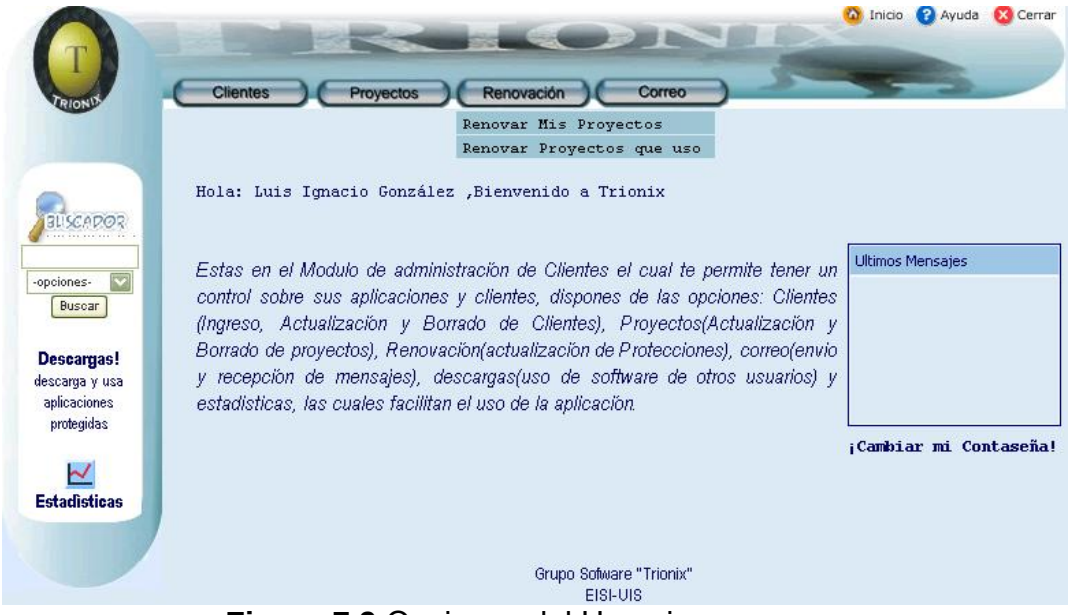



**Figura 7.9** Opciones del Usuario

## 7.4 CLIENTE

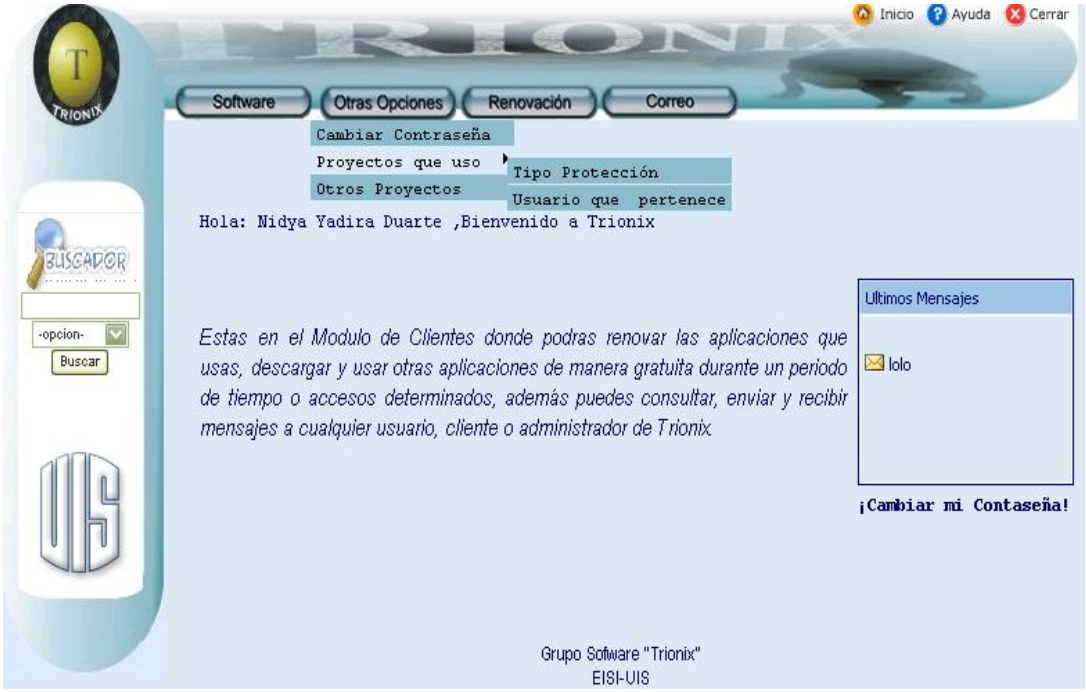



**Figura 7.10** Opciones del Cliente

El modulo cliente tiene como principal objetivo el permitir renovar aplicaciones protegidas por parte de los clientes previamente autorizados por el usuario, además de dar a conocer la información que relaciona las aplicaciones que usa el cliente y los permisos que le han sido otorgado. A continuación se detallan la totalidad de opciones que presenta este módulo.

### 7.4.1 SOFTWARE

Espacio de proyectos disponibles para descargar y usar durante un período de tiempo y/o accesos que ha determinado su dueño. Esta característica se le otorga a la aplicación a la hora de actualizar su información por parte del usuario.

### 7.4.2 OTRAS OPCIONES

Estas son opciones adicionales que se le dan al cliente, con el fin de facilitar consultas e información de los proyectos que usa y los que podría usar.

### 7.4.3 CAMBIAR CONTRASEÑA

Opción que se le brinda al cliente para que varié la contraseña que le otorgue el usuario inicialmente y tenga un espacio exclusivo en la aplicación Web. Por seguridad se recomienda cambiarla una vez por mes.

#### 7.4.3.1 PROYECTOS QUE USO

Información relacionada con los proyectos que usa el cliente, su tipo y parámetros de protección, usuario al cual pertenece cada aplicación usada.

#### 7.4.3.2 OTROS PROYECTOS

Pequeña descripción de proyectos que no usa el cliente y que en un momento dado pueden interesarle.

### 7.4.4 RENOVACION

Permite la renovación de una aplicación solicitando la clave pública, verificando la autenticidad de la clave y permisos del cliente; si son correctos se remite a generar la nueva clave de uso particular de la aplicación.

### 7.4.5 CORREO

Medio de comunicación entre el cliente y cualquier usuario expresando alguna inquietud o interés en el uso de una determinada aplicación.

**CAPITULO 8**

# PRUEBAS

Las pruebas consisten en la elaboración de un test, el cual permita establecer si el sistema funciona correctamente, de lo contrario tomar las medidas correspondientes para llevar a cabo las modificaciones necesarias.

Para tal fin se tomaron los procesos más importantes, los cuales deben satisfacer cada una de las condiciones descritas en cada cuestionario; además la base de datos se ha poblado con datos consistentes para poder llevar a cabo un buen proceso de prueba.

## 8.1 OBJETIVOS

- ✓ Analizar los riesgos a los que puede estar expuesta la aplicación y desarrollar dado el caso un plan de acción que permita contrarrestarlos y que no afecte su normal funcionamiento.

- ✓ Realizar cuestionarios particulares en cada proceso, donde los resultados deben satisfacer el 100% de condiciones.

- ✓ Verificar que las validaciones y controles de seguridad funcionen correctamente.

## 8.2 MÓDULO DE ADMINISTRACIÓN

Se divide en dos: opciones no disponibles en la Web (protección) y disponibles en la Web (las restantes).

### 8.2.1 PROTEGER UNA APLICACIÓN

En este módulo, el administrador podrá proteger aplicaciones software y realizar mantenimiento a toda la información que maneja la aplicación excepto a la de los clientes de sus usuarios. Además, podrá consultar la información insertada para poder verificar su consistencia. Cada formulario de inserción o modificación de datos, debe validar que se han suministrados los datos considerados como obligatorios y que los tipos de datos correspondan, es decir datos numéricos para el número de cedula y teléfono y alfanumérico para nombres.

Las pruebas a realizar deben demostrar el correcto funcionamiento del módulo, con el fin de evitar errores.

| PASOS | DESCRIPCIÓN |
|---|---|
| **1** | Entrar a la aplicación con su nombre de usuario y contraseña. |
| **2** | Ingresar la información del usuario y la aplicación a proteger |
| **3** | Después de llenados los datos, guardar la información ejecutando el botón *proteger*, el cual invoca las funciones para proteger software y guardar la información en la base de datos. |
| **4** | Ir a la lista (parte superior derecha), seleccionar y consultar la información del usuario insertado; Para cualquier modificación debe ingresar a la aplicación Web y en la opción *Mantenimiento usuarios* hacer las correcciones o actualizaciones a los datos. |

**Tabla 8.1** Descripción del proceso proteger una aplicación

#### 8.2.1.1 DATOS DE PRUEBA(Prototipo 1)

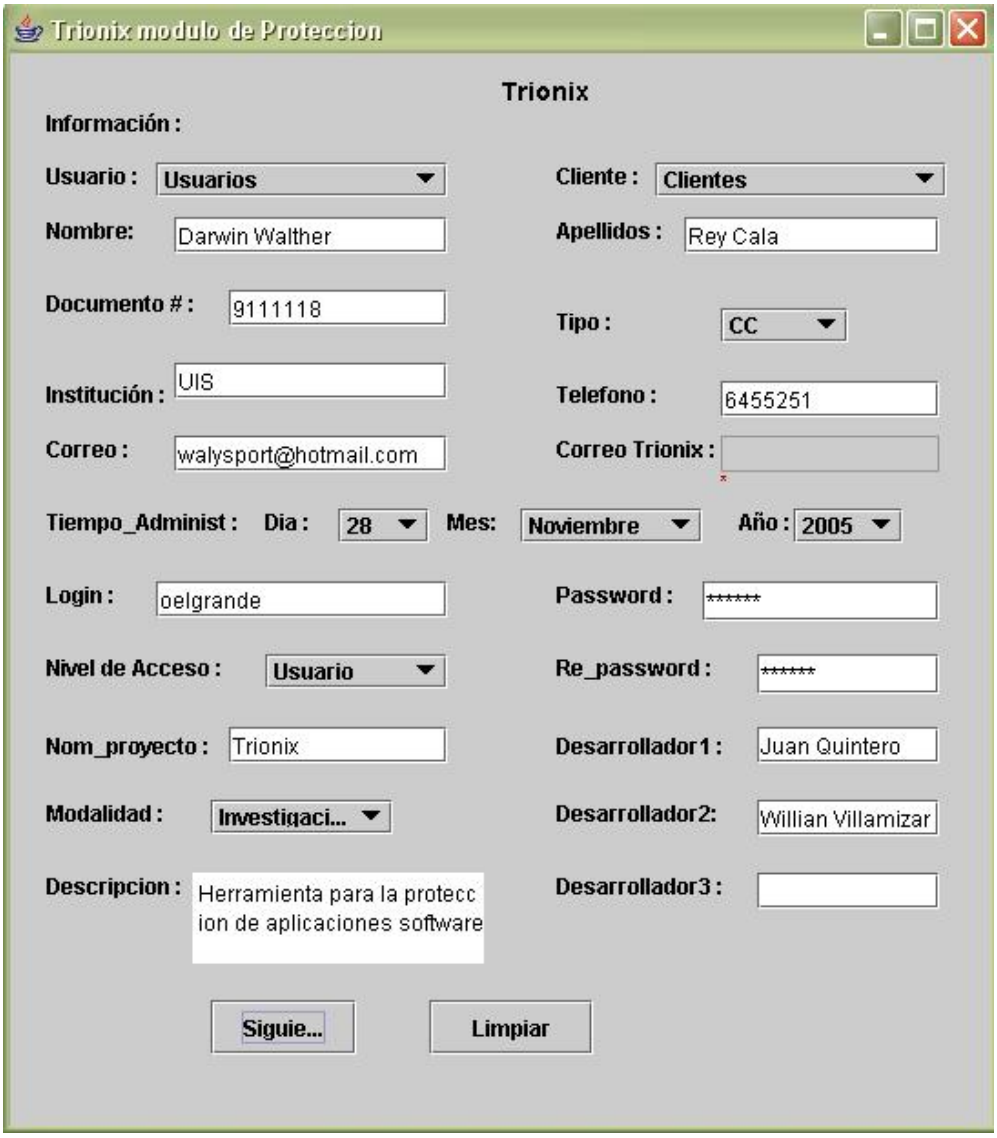



**Figura 8.1.** Inserción de datos del usuario y proyecto a Proteger

Los datos descritos en la anterior interfaz se insertaron a manera de prueba, para que permitan contestar el cuestionario posterior y se puede determinar el correcto funcionamiento de la aplicación o los errores que se presentan.

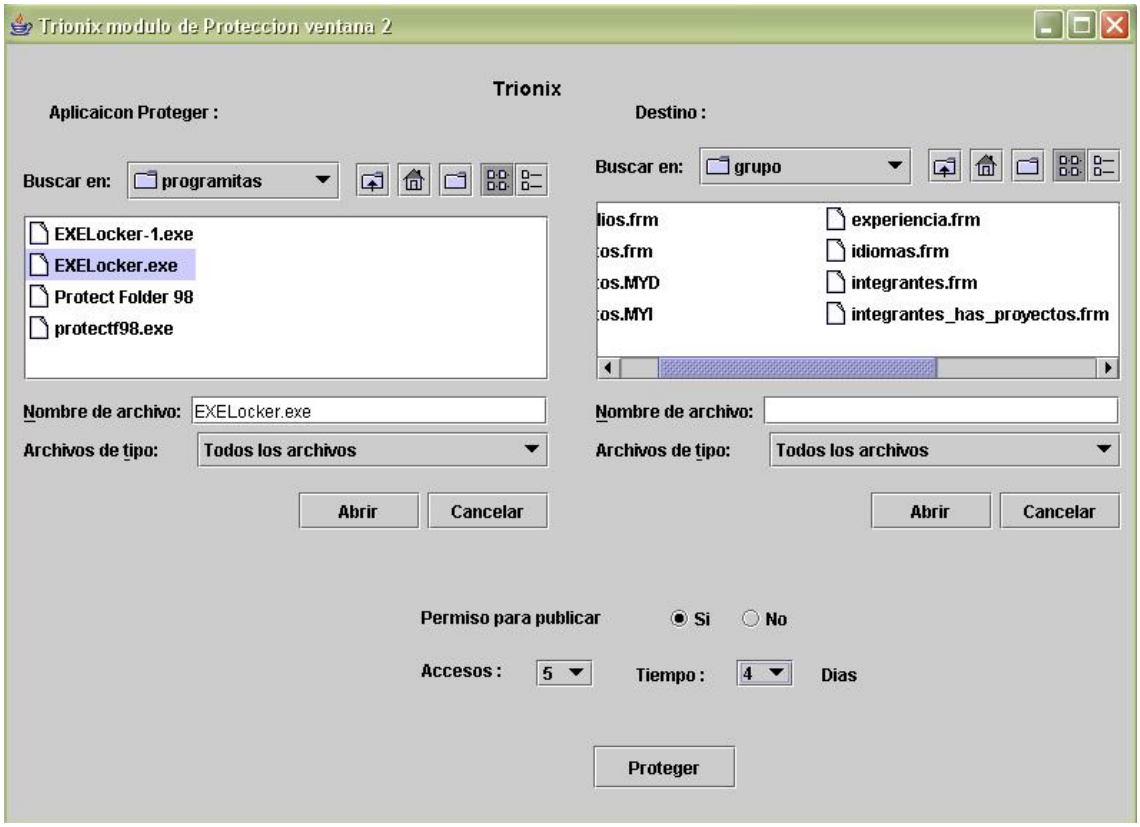



**Figura 8.2**. Inserción de datos del proyecto a Proteger

### 8.2.1.2 OBSERVACIONES

- ✓ La inserción de usuarios y proyectos no esta disponible en la Web.
- ✓ Para cualquier modificación o actualización de algún dato se debe realizar en la aplicación Web.

### 8.2.1.3 TABLA DE CHEQUEO

El siguiente cuestionario permite determinar el cumplimiento o no, de determinados aspectos en el módulo de administración:

| No | Pregunta | Respuesta | |
|---|---|---|---|
| | | **SI** | **NO** |
| **1** | El formulario permitió introducir en todos los campos la información con un ancho de palabra o frase correcto. | **X** | |
| **2** | El formulario validó los campos obligatorios y su tipo de dato (numérico o alfanumérico). | **X** | |
| **3** | Es clara la descripción de cada campo. | **X** | |
| **4** | Después de seleccionar el botón “proteger” no ocurrió ningún error imprevisto. | **X** | |
| **5** | Al proteger la aplicación se genero el nuevo archivo ejecutable. | **X** | |
| **6** | Al Consultar el usuario creado anteriormente la información coincide con la que emitió el usuario. | **X** | |

**Tabla 8.2** cuestionario del proceso de proteger una aplicación

El proceso de protección de software se realizó sin ningún problema, además cumplió satisfactoriamente con cada uno de los ítems analizados y descritos en el anterior cuestionario; por ello se considera que su objetivo: La protección de software se ha cumplido.

### 8.2.2 MODIFICACIÓN Y BORRADO DE PROYECTOS

Dentro del módulo de administración el usuario-administrador podrá consultar la información de aplicaciones protegidas y hacer las debidas modificaciones; además puede borrar proyectos teniendo en cuenta que esto implica la in activación de quienes lo usan y si es el único proyecto del usuario de inmediato le vence el período de administración (Esta opción se dejo disponible para que el administrador modifique algún dato o que en un momento dado inactive a un usuario por incumplimiento en lo que pactado).

**Precondición:** Para elaborar esta prueba es necesario que se hayan insertados datos previamente.

La prueba a continuación deberá demostrar la funcionalidad del mantenimiento de proyectos respecto a la modificación y borrado de un registro de cualquier proyecto.

| PASOS | DESCRIPCIÓN |
|---|---|
| **1** | Entrar a la aplicación Web con el nombre de usuario y contraseña asignada (Administrador,123456) |
| **2** | Seleccionar la opción Mantenimiento de Proyectos y escoger del Listado presentado el nombre del proyecto o escribir este en el buscador, opción actualizar, una vez seleccionado se visualizará la información del proyecto que puede ser modificada. |
| **3** | En el formulario se debe modificar los datos que se desean del proyecto. |
| **4** | Escoger el botón Actualizar (parte inferior del formulario), para guardar los cambios. Aparecerá nuevamente el listado de proyectos para seleccionar otro proyecto o salir. |
| **5** | Para verificar los datos modificados se debe seleccionar nuevamente del listado presentado el proyecto anteriormente actualizado. |
| **6** | Repetir paso 2. |
| **7** | Escoger el botón Borrar Proyecto(s) (parte inferior del formulario). |
| **8** | Buscar por nombre y verificar que el proyecto fue borrado. |

**Tabla 8.3** Descripción del proceso Mantenimiento de Proyectos.

**Ilustración paso 2**

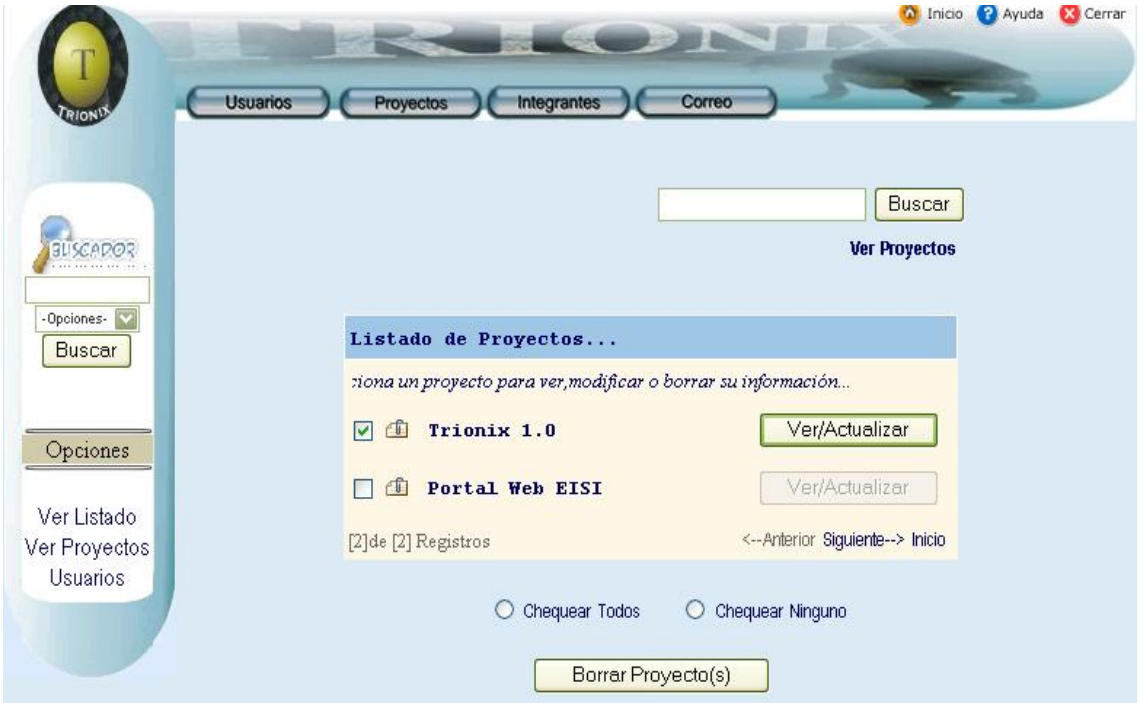



**Figura 8.3.** Listado de proyectos protegidos con al opción de Actualizar y Borrar

Para actualizar se debe seleccionar un único proyecto, en cambio para borrar se pueden seleccionar varios proyectos.

#### 8.2.2.1 DATOS DE PRUEBA

La siguiente interfaz contiene la información del proyecto a modificar:

**Información Inicial**


**Figura 8.4.** Información a Actualizar del proyecto Trionix 1.0

**Nota:** Se tratará de ingresar algún dato erróneo para verificar la validación que se realiza al insertar y modificar cualquier información.

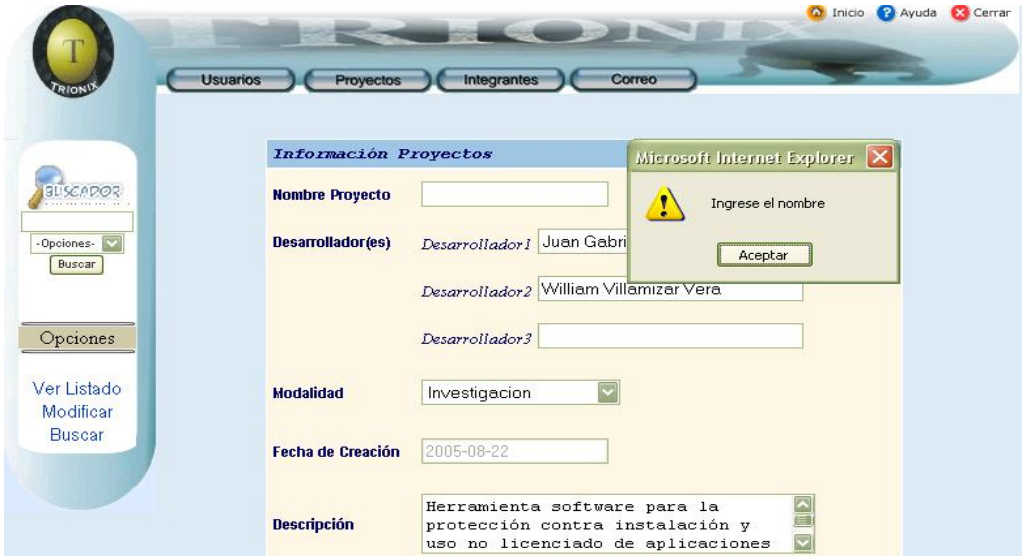



**Figura 8.5.** Validando la información de un proyecto

#### 8.2.2.2 TABLA DE CHEQUEO

Los siguientes cuestionamientos determinarán el cumplimiento o no de determinados aspectos para la modificación y borrado de datos de un proyecto.

| No | Pregunta | Respuesta | |
|---|---|---|---|
| | | **SI** | **NO** |
| **1** | Es fácil encontrar un proyecto que se desea modificar. | **X** | |
| **2** | La validación al modificar los datos fue correcta. | **X** | |
| **3** | La verificación de los datos modificados fue correcta y no modifico datos de otro proyecto | **X** | |
| **4** | El borrado de los proyectos seleccionados fue correcto y no borro otro(s) proyecto(s). | **X** | |
| **5** | Se visualizo un mensaje de confirmación para el borrado de un proyecto y lo que ello implica (caso selección errónea de la opción de Borrar un proyecto). | **X** | |

**Tabla 8.4** cuestionario de los procesos de actualización y borrado de un proyecto.

Las respuestas son claras y describen por si solas, que los procesos de actualización y borrado de un proyecto funcionan correctamente.

### 8.3 MODULO DE USUARIOS

#### 8.3.1 MANTENIMIENTO DE CLIENTES

El módulo de usuarios le permite al usuario la inserción, modificación y borrado de sus clientes y la interacción de estos con sus proyectos (permisos).
Para realizar una prueba a este proceso, el usuario debe tener al menos un proyecto protegido y su período de administración debe estar vigente.

| PASOS | DESCRIPCIÒN |
|---|---|
| **1** | Entrar a la aplicación con el nombre de usuario y contraseña asignado. |
| **2** | Seleccionar la opción *Clientes* (Botón superior izquierdo) |
| **3** | Para actualizar o borrar, seleccionar el cliente del listado presentado. |
| **4** | Para Ingresar un Nuevo Cliente seleccionar de la barra lateral izquierda la opción *Nuevo Cliente.* |
| **5** | Si paso 3 entonces: Se visualiza la información del cliente seleccionado, la cual puede ser modificada. |

| | |
|---|---|
| **6** | En el formulario se debe modificar los datos del cliente. |
| **7** | Escoger el botón Actualizar (parte inferior del formulario), para guardar los cambios. Aparecerá nuevamente el listado de clientes para seleccionar otro cliente o salir. |
| **8** | Para verificar los datos modificados se debe seleccionar nuevamente del listado presentado el cliente anteriormente actualizado. |
| **9** | Si paso 4 entonces: Se visualiza el formulario de inserción de clientes solicitando cierta información y visualizando los proyectos de dominio del usuario. |
| **10** | En el formulario se deben insertar los datos del cliente a crear, los proyectos que este va usar y los permisos sobre estos. |
| **11** | Escoger el botón Ingresar (parte inferior del formulario), para guardar la información. Aparecerá un formulario para insertar un nuevo cliente o salir. |
| **12** | Para verificar que el cliente fue creado, se busca en el listado de clientes donde debe estar disponible para actualizar o borrar. |

**Tabla 8.5** Descripción del proceso Mantenimiento de Clientes.

#### 8.3.1.1 DATOS DE PRUEBA

La siguiente interfaz contiene información prueba para verificar el correcto funcionamiento de los proceso en cuestión:

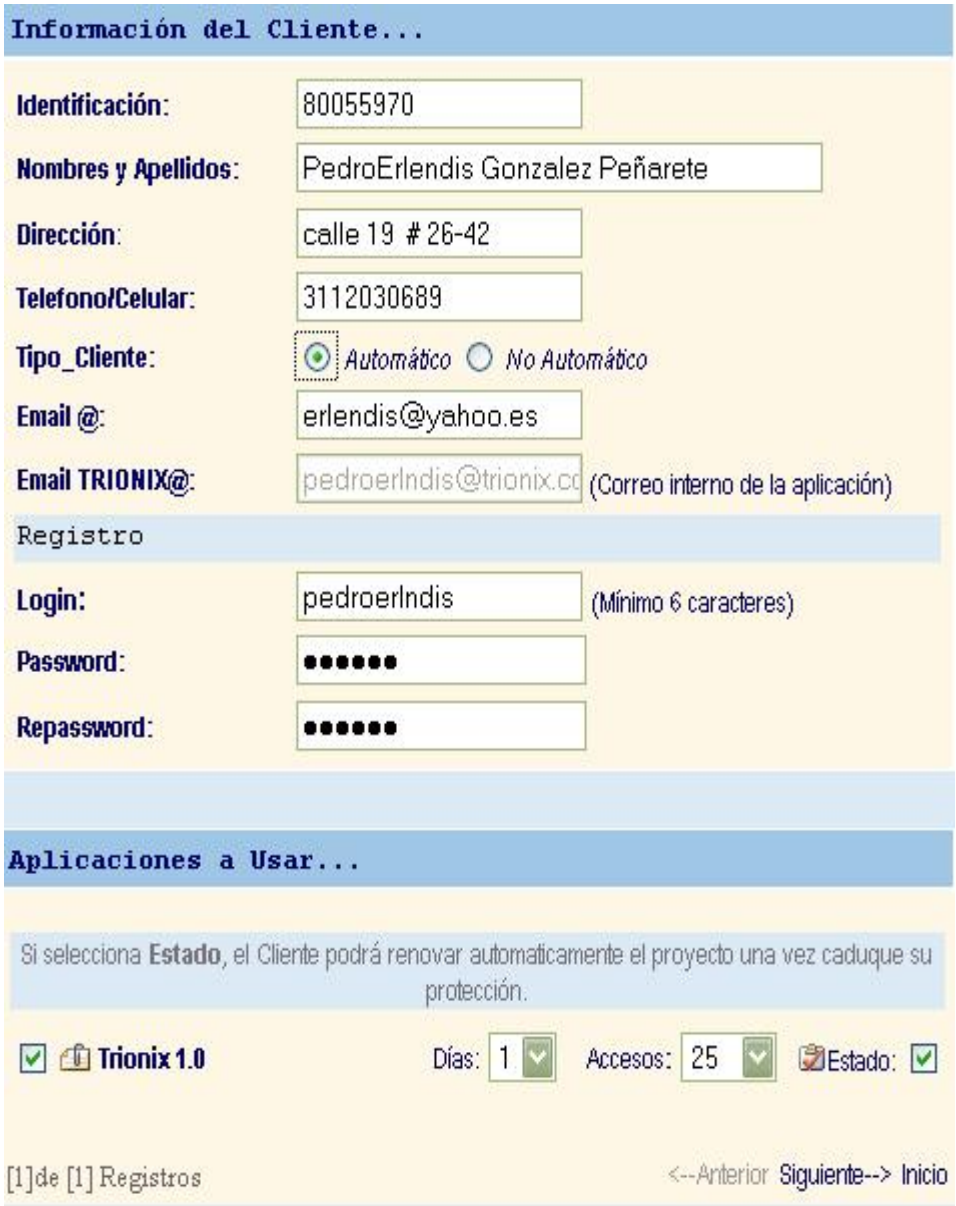



**Figura 8.6.** Inserción de un Cliente

#### 8.3.1.2 TABLA DE CHEQUEO

El siguiente cuestionario permite determinar el cumplimiento o no de determinados aspectos en los procesos de inserción, actualización y borrado de clientes.

| No | Pregunta | Respuesta | |
|---|---|---|---|
| | | **SI** | **NO** |
| **1** | La validación al insertar o modificar datos fue correcta. | **X** | |
| **2** | En los proyectos que usa o puede usar el cliente solo aparecen los del dominio del usuario en sesión. | **X** | |
| **3** | El máximo número días de uso de cualquier proyecto del usuario, corresponde a la diferencia en días entre la fecha de administración del usuario y la fecha del sistema. | **X** | |
| **4** | El máximo número de accesos de un proyecto supera los 200. | | **X** |
| **5** | La información que se recopila es necesaria y suficiente | **X** | |
| **6** | Al realizar cualquier proceso (inserción, actualización o borrado) se presentan mensajes de confirmación o error según el caso. | **X** | |
| **7** | Si borro un cliente desaparece la relación de este con los proyectos que uso. | **X** | |
| **8** | Se tiene acceso a la información de otros clientes fuera del dominio del usuario en sesión | | **X** |

**Tabla 8.6** cuestionario de los procesos de inserción, actualización y borrado de clientes

Los procesos relacionados con el mantenimiento de clientes funcionan correctamente y garantizan que un usuario administre sus aplicaciones y clientes.

El usuario puede renovar los proyectos que usa y los que usan sus clientes; para esto debe digitar en cualquier caso, la clave pública que genera el sistema una vez culmine un período de protección y escribir los nuevos parámetros de uso (días y accesos), para de esta manera conocer la nueva clave privada.

La prueba a continuación deberá demostrar la funcionalidad del proceso de renovación de un proyecto, los pasos que debe seguir el proceso son:

| PASOS | DESCRIPCIÒN |
|---|---|
| **1** | Seleccionar la opción *Renovar Mis proyectos*. |
| **2** | Seleccionar el cliente y proyecto a renovar. |
| **3** | Ingresar la clave pública del proyecto (la remitida por el cliente). |
| **4** | Seleccionar Parámetros de protección (número de días y accesos). |
| **5** | Esperar que el sistema valide que la clave es correcta. Si lo es sigue el paso 6 si no, el paso 3. |
| **6** | Digitar la nueva clave en la ventana del proyecto vencido. |

**Tabla 8.7** Descripción del proceso de Renovación de Proyectos.

### 8.3.2.1 DATOS DE PRUEBA

La siguiente interfaz contiene información prueba para verificar el correcto funcionamiento del proceso en cuestión.


**Figura 8.7.** Renovando un Proyecto

### 8.3.2.2 TABLA DE CHEQUEO

El siguiente cuestionario permite determinar el cumplimiento o no de determinados aspectos en el proceso de renovación de proyectos.

| No | Pregunta | Respuesta | |
|---|---|---|---|
| | | **SI** | **NO** |
| **1** | Cada vez que ejecuto la aplicación vencida se genera una clave diferente | **X** | |
| **2** | Se puede renovar proyectos que usan los clientes, del usuario en sesión, pero que no son de su dominio | | **X** |
| **3** | La validación de la clave pública es correcta y no se aceptan claves validas de anteriores procesos. | **X** | |
| **4** | El máximo número de días de renovación de cualquier proyecto del usuario, corresponde a la diferencia en días entre la fecha de administración del usuario y la fecha del sistema. | **X** | |
| **5** | El máximo número de accesos de un proyecto supera los 200. | | **X** |
| **6** | La clave privada que genera el sistema es aceptada por la aplicación vencida. | **X** | |

**Tabla 8.8** cuestionario para probar el proceso de renovación.

El proceso de renovación realiza las validaciones correspondientes y genera la nueva clave de uso a partir de los parámetros que establezca el usuario.

## 8.4 CONCLUSIONES DE LAS PRUEBAS

- Cada proceso que fue analizado obtuvo respuestas favorables, cumpliendo a cabalidad con ciertos aspectos descritos particularmente en cada cuestionario; por ello se considera que la aplicación puede iniciar su uso y satisfacer la demanda de usuarios que deseen proteger su software.

- Las pruebas permiten realizar correctivos y determinar errores y defectos del producto a entregar y que medidas se deben tomar.

- Las pruebas realizadas son preliminares, pues se considera que el sistema en realidad será probado cuando a él incurra un número significativo de usuarios y la base de datos este parcialmente poblada, en donde además de funcionamiento se podrá evaluar rendimiento, eficacia y otros aspectos.

  Estas pruebas se complementan con las efectuadas a los usuarios finales por medio de la encuesta realizada. Ver anexo 3.

## RECOMENDACIONES

- Garantizar un continuo desarrollo de este trabajo para que no quede relegado frente a las nuevas tecnologías de manera que se aprovechen las facilidades y demás ventajas que estas ofrecen.

- Se debe garantizar el buen funcionamiento de la aplicación, considerando que en la actualidad las especificaciones de hardware del servidor en donde se aloja la aplicación presenta limitaciones; teniendo en cuenta que la funcionalidad a la luz de los usuarios también depende considerablemente de la agilidad y el tiempo de respuesta cuando se interactúa con la aplicación Web.

- Aunque la escuela (Escuela de Ingeniería de sistemas) cuenta con un servidor aceptable para el buen funcionamiento de la aplicación, se recomienda adaptar un servidor de respaldo con similares o mayores especificaciones técnicas, que garanticen en un momento dado la continuidad del funcionamiento de la aplicación en caso de que el servidor titular no responda por alguna causa.

- Por seguridad, los usuarios de la aplicación deben cambiar la contraseña de acceso periódicamente, ya que su seguridad podría disminuir, aumentando la posibilidad de vulnerabilidad.

- Para garantizar la operatividad de la aplicación se debe contar con un administrador, quien preferiblemente sea participante del grupo “TRIONIX”; quien atienda la protección de aplicaciones y realice el mantenimiento de la aplicación, además de continuar trabajando en la implementación de nuevos controles supliendo las nacientes necesidades.

- Se recomienda en una próxima versión, implementar controles y hacer más portable el paquete para la protección de archivos, para que de esta manera se permita proteger aplicaciones a los usuarios sin necesidad de remitirse al administrador.

- La seguridad informática es un área que está en auge a nivel mundial, debido al aumento que ha tenido en el mundo de las telecomunicaciones. La escuela de Ingeniería de Sistemas en Informática de la Universidad Industrial de Santander debería incentivar su estudio y desarrollo, en búsqueda de obtener un campo que nos garantice un aporte a la sociedad.

## CONCLUSIONES

- Las tecnologías usadas en el desarrollo de este trabajo fueron de gran importancia debido a que se ha contado con herramientas modernas para el desarrollo de aplicaciones Web (como jsp,tomcat y dreamweaver), al igual que con un motor de base de datos ágil y confiable como mysql.

- Aplicar una metodología adecuada permite que el proceso de desarrollo de software se lleve a cabo dentro de los márgenes establecidos y con la calidad esperada. En este proyecto se trabajo el proceso unificado de desarrollo (el cual es guiado por casos de uso, iterativo e incremental y centrado en la arquitectura) debido a su facilidad, acople a lo que se desarrolla y por la distribución de actividades en el desarrollo del proyecto.

- El manejo de perfiles de usuario permite que sólo personal autorizado acceda a opciones particulares. “TRIONIX” permite que sólo el administrador de la aplicación pueda llevar a cabo el proceso de protección, diseñado de esta manera debido a que es critico, y de no ser controlado puede llegarse a una perdida de control que pondría en duda la protección de aplicaciones; de otra parte, el proceso de renovación se ha colocado como una opción en la Web para de esta manera facilitar a los usuarios su uso, teniendo en cuenta que se han implementado mecanismos necesarios para que el proceso de renovación se haga de una manera controlada evitando posibles vulnerabilidades al sistema.

- En el desarrollo de este trabajo, se ha aprovechado el gran potencial de java para interactuar con diferentes sistemas operativos y aprovechar recursos de diferentes lenguajes de programación, integrándose con relativa facilidad. En el trabajo predecesor a este proyecto se trabajo con un lenguaje como lo es visual c++, creando dll (librerías de vínculos dinámicos) para el cifrado de archivos; la necesidad de interactuar con la base de datos mysql sobre linux y dll propias de windows; se optó por usar jni (Java Native Interface ) que hacen posible esta tarea.

- Los algoritmos criptográficos utilizados en este proyecto previamente probados son el DES y el RSA, los cuales presentan ventajas como:

  **DES**

  Permite cifrar la aplicación protegida, garantizando que su descifrado sin conocer la clave (fuerza bruta) sea tan complejo que se requiere costos (tiempo, hardware y económicos) elevados, terminando por declinar en la intención del descifrado.

Su tiempo de ejecución es mayor que los algoritmos de cifrado que incluyen transposición o sustitución; pero esto se compensa con la seguridad brindada, además se optó por seguir con este pero combinando con técnicas de manejo de archivos, manejo de memoria y funcionalidades del sistema operativo, lo que hacen del proceso de protección un paquete seguro.

Luego de cifrar un archivo con este algoritmo se conserva el tamaño original, debido a que este un algoritmo simétrico de clave secreta.

**RSA**
Al ser un algoritmo de clave pública, puede ser utilizado como generador de claves (publica y privada) de manera que se garantice que para cada clave publica existe una clave privada única.

La persona que desee conocer la clave privada de un proyecto sin ninguna autorización de renovación, se enfrentará al Problema de la Factorización de Números Grandes puesto que la solución para conocer esa clave privada pasa por deducir el valor del Indicador de Euler f(n) = (p-1)(q-1) para así poder encontrar el inverso de la clave pública d = inv [e,f(n)], además lo que muestra la trama al usuario, es el resultado de operaciones binarias AND OR y de desplazamiento de bits lo que dificulta mucho más el determinar el significado de cada parte de la trama y los valores originales de la clave, garantizando así la autenticidad de las mismas.

Cifrar un archivo con el RSA implica un aumento en el tamaño original, debido a la naturaleza del algoritmo de clave publica. Por esta razón no se utilizo al momento de cifrar la aplicación a proteger.

Se concluye que para el cifrado de archivos se recomienda los algoritmos simétricos de clave secreta y para el manejo de claves y transacciones seguras se recomienda el algoritmo RSA.

- Las herramientas de software libre para entorno educativo que en la actualidad se encuentran en el medio, fueron suficientes para el desarrollo del proyecto. La velocidad de consulta en las base de datos, tiempos de acceso a la aplicación, capacidad de almacenamiento, cifrado de contraseñas (función HASH: MD5) entre otros aspectos, son puntos a favor que poseen este tipo de tecnologías. Lo cual suplió y complementó las necesidades de la protección y administración de software, que llevaron al éxito del producto entregado.

## BIBLIOGRAFÍA


**[1]** PINO, Gil. Seguridad Informática: Técnicas Criptográficas. México: Editorial Alfaomega, 1997. 137 p.

**[2]** JACOBSON, Ivar : BOOCH, Grady y RUMBAUGH James. El Proceso Unificado de Desarrollo de Software. Madrid : Editorial Addison Wesley,1999. 438 p.

**[3]** JACOBSON, Ivar : BOOCH Grady y RUMBAUGH James. El Lenguaje Unificado de Modelado. Madrid : Ed. Addison Wesley,2000.

**[4]** Pressman, Roger. Ingeniería del software : Un enfoque práctico. Tercera Edición. México: Editorial McGraw – Hill,1993.

**[5]** CEBALLOS, Francisco Javier. Java 2 : Curso de Programación. México: Editorial Alfaomega Ra_ma, 2000.

**[6]** Visual C++ 6: Aplicaciones para Win32. Segunda edición. México: Editorial Alfaomega Ra_ma. 2000.

**[7]** Visual C++ 6: Aplicaciones para Win32. Segunda edición. México: Editorial Alfaomega Ra_ma. 2000.

**[8]** Stallings, William. **Comunicaciones y redes de computadores.** Quinta edición. Prentice hall. 1997.

**[9]** Hanna, Phil. **JSP Manual de referencia.**

**[10]** Andrés Fernando García Agudelo, Federico Vladimir García Moya **Diseño e Implantación del sitio WEB UNIRED, y desarrollo de los sistemas de consulta del material bibliográfico de sus bibliotecas y librerías a través de Internet.** Tesis de grado, Bucaramanga. Junio 2003.

**[11]** Carolina Inés Vanegas González, Sergio Andrés Moreno Acevedo **Desarrollo de un modulo de información y servicios internet para la escuela de medicina, integrado al sistema de información web de la Universidad Industrial de Santander.** Tesis de Grado, Bucaramanga junio 2003

**[12]** Quintero Peña Juan Gabriel, Villamizar Vera Willian **Herramienta para la protección contra instalación y uso no licenciado de aplicaciones software** Tesis de Grado, Bucaramanga junio 2004

**Direcciones electrónicas:**

- http://www.programacion.com/java/tutorial/jni/ Utilizar el Java Native Interface (JNI, Pasar parametros desde java a una dll en Visual C++).
- http://www.fit.qut.edu.au/. Computer Science & Electrical Engineering
- http://pisuerga.inf.ubu.es/lsi/Invest/Java/Tuto/A_III.htm Códigos Nativos JNI.
- http://www.cryptored.edu.es Red Iberoamericana de Criptografía.
- http://www.programacion.net/java/foros/ foros de Ayuda de Java
- http://pisuerga.inf.ubu.es/lsi/Invest/Java/Tuto/A_III.htm
- http://www.javahispano.org/articles.article.action?id=36
- http://dev.mysql.com/doc/

## ANEXO 1 ALGORITMOS CRIPTOGRAFICOS[20]

En la actualidad, el principal problema que se presenta a la hora de desarrollar un producto software, es la incertidumbre que genera su distribución en un medio inseguro donde personas no autorizadas podrían llegar a redistribuirlo, marginado al verdadero desarrollador de los beneficios de su labor.

Todo esto hace posible cada vez más tener técnicas y algoritmos de encriptación de datos seguros para transmitir información a través de Internet u otro medio de comunicación o para protegerlo de personal no autorizado.

En este anexo se verán algunas características básicas de algoritmos criptográficos como el Cifrador Afín y el cifrador de Vigeneré frente al Algoritmo DES (Data Encryption Standard) buscando velocidad en tiempo de ejecución en función de seguridad, para ello se describen las ventajas y desventajas de cada uno de estos algoritmos, se realizan pruebas cuantificando el tiempo que demoran en cifrar archivos de un mismo tamaño.

De este análisis se seleccionará la mejor alternativa, analizando además del tiempo factores como seguridad, rendimiento y eficacia; el algoritmo seleccionado se implementará para el cifrado en la protección de archivos del presente proyecto.

Además se describe el algoritmo RSA como soporte para quien desee conocer un poco más del proceso que sigue en el tratamiento de claves.

### DESCRIPCIÓN DE ALGUNOS ALGORITMOS CRIPTOGRAFICOS

### 1. 1 CIFRADO AFÍN

Es el caso general del algoritmo de César[21]. Su transformación sería:

E**(a,b)**(M) = (aM + b) mód N

Siendo a y b dos números enteros menores que el cardinal N del alfabeto, y cumpliendo que mcd(a, N) = 1. La clave de cifrado k viene entonces dada por el par (a, b). El algoritmo de César sería pues una transformación afín con k = (1; 3).

Ejemplo:

[20] Tomado de http://www1.tiendalinux.com/documentacion/manuales/unixsec/unixsec-1.2/node22.html

[21] El cifrado Cesar es uno de los más antiguos que se conocen. Matemáticamente, para trabajar con este, tomamos el alfabeto A = Zm (enteros de módulo m). Cuando $a$ y $b$ son primos entre sí, la aplicación, f(x) = ax + b a $\neq$ 0, recibe el nombre de *codificación módulo m con parámetros a, b*; el par (a,b) es la clave de este criptosistema. Más información en: http://www1.tiendalinux.com/documentacion/manuales/unixsec/unixsec-1.2/node22.html

| | A | B | C | D | E | F | G | H | I | J | K | L | M | N | Ñ | O | P | Q | R | S | T | U | V | W | X | Y | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 |

Texto : pedro
Texto cifrado: WOMBT
b = 1
a = 3

Para la p se hace el siguiente procedimiento:

Para este caso el número de letras del alfabeto seria 28, incluyendo la posición 0 que representa el espacio en blanco.

Según este alfabeto la letra p ocupa la posición 17 por lo tanto M = 17, de otro lado a = 3 , b =1, y la operación será E**(a,b)**(M) = (3*17 + 1) mód N, donde N tendrá un valor de 28 que es el numero de letras de este alfabeto, de esta manera se tendrá:
P = (p*3+1) modulo 28
P= (17*3 + 1 ) modulo 28
P= 52 modulo 28
P = posición 24 = w

Y sucesivamente con las siguientes letras se sigue similar proceso.

| | | | | | | |
|---|---|---|---|---|---|---|
| p | * | 3 | + | 1 | = | W |
| e | * | 3 | + | 1 | = | O |
| d | * | 3 | + | 1 | = | M |
| r | * | 3 | + | 1 | = | B |
| o | * | 3 | + | 1 | = | T |

**VENTAJAS**

Si se tiene un alfabeto, ejemplo el español, se tiene un número grande de combinaciones para generar un alfabeto nuevo, lo que dificultaría notablemente la vulnerabilidad del cifrador.
Si al alfabeto se le agrega las combinaciones de letras y símbolos, se crea un nuevo alfabeto mas extenso y por ende la dificultad de violación del cifrador aumentaría razonablemente.

**DESVENTAJAS**

Este cifrador puede ser vulnerado mediante las propiedades estadísticas de los lenguajes naturales, es decir, por la frecuencia con que una letra se presenta en un alfabeto dado.

Si se tiene una pequeña fracción del texto claro, las probabilidades de vulnerar el algoritmo serian mayores, esto se da por el echo de que el desplazamiento del texto cifrado se da en forma circular y la frecuencia de repetición facilitaría el descifrado.
Si se tiene una aplicación computacional que recree este método de cifrado, las probabilidades de descifrado serian elevadas.

## 1.2 CIFRADO DE VIGENÉRE

El sistema de cifrado de Vigenére (en honor al criptógrafo francés del mismo nombre) es un sistema polialfabético o de sustitución múltiple. Este tipo de criptosistemas aparecieron para sustituir a los monoalfabéticos o de sustitución simple, basados en el Cesar, que presentaban ciertas debilidades frente al ataque de los criptoanalistas relativas a la frecuencia de aparición de elementos del alfabeto. El principal elemento de este sistema es la llamada Tabla de Vigenére, una matriz de caracteres cuadrada, con dimensión 26x26.

Se pueden usar diversas cifras de César para distintas letras. Supongamos que las letras en las posiciones impares las ciframos mediante una sustitución, y las pares mediante otra. Digamos que cada letra impar la sustituimos por la que hay dos lugares más adelante (la A pasa a ser la C), lo que llamaremos cifras de César de clave 2; las letras pares las cifraremos con una cifra cesariana de clave 5 (la A pasa a ser la que tiene cinco lugares más adelante, es decir la F). Ahora se escribirán los alfabetos correspondientes:

```
Texto plano:              abcdefghijklmnopqrstuvwxyz
Cifrado (César clave 2) :CDEFGHIJKLMNOPQRSTUVWXYZAB
Cifrado (César clave 5): FGHIJKLMNOPQRSTUVWXYZABCDE
```

Se cifrara la frase "bienvenidos al boletin enigma" (sin signos de puntuación ni mayúsculas). La primera letra (b) la cifraremos con la cifra César de clave 2; en la segunda fila, vemos que la b se transforma en la D. Para la segunda letra (i), tomamos la fila inferior, y vemos que la i se transforma en la N. Para la tercera letra, volvemos a usar la cifra César de clave 2: la e se transforma en la G. Así, sucesivamente, el cifrado será de esta forma:

```
Cifra (César clave...):   25252525252525252525252525252
Texto plano:              bienvenidos al boletin enigma
Texto cifrado:            DNGSXJPNFTU FN GQQGYKS GSKLOF
```

Bien, pues eso es una cifra de Vigenére. Claro que no se usan solamente dos alfabetos, sino que se puede tomar mas. Para poder visualizarlo mejor, se construida la siguiente tabla:

| | a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
| B | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A |
| C | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B |
| D | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C |
| E | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D |
| F | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | E | F |
| G | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F |
| H | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G |
| I | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H |
| J | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I |
| K | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J |
| L | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K |
| M | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L |
| N | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M |
| O | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N |
| P | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O |
| Q | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P |
| R | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q |
| S | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R |
| T | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S |
| U | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T |
| V | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U |
| W | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V |
| X | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W |
| Y | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X |
| Z | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y |

El funcionamiento se verá mejor con un ejemplo. Supongamos se quiere cifrar "bienvenidos al boletín enigma" mediante la cifra Vigenére con clave DAVID. Para la primera letra cifrada, tomaremos la intersección entre la cuarta fila (correspondiente al alfabeto D) y la segunda columna (correspondiente a la letra b). El resultado es la letra cifrada E. A continuación, tomamos la fila correspondiente a la siguiente letra de la clave (A, primera fila) y la columna correspondiente a la siguiente letra del texto (i, novena columna). La intersección de fila primera y columna novena nos da la letra cifrada I. Así sucesivamente.

Siguiendo este proceso, obtendremos lo siguiente:

| | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Clave (fila): | D | A | V | I | D | D | A | V | I | D | D | A | V | I | D | D | A | V | I | D | D | A | V | I | D | D |
| Texto (columna): | b | i | e | n | v | e | n | i | d | o | s | a | l | b | o | l | e | t | i | n | e | n | i | g | m | a |
| Texto cifrado: | E | I | Z | V | Y | H | N | D | L | R | V | A | G | J | R | O | E | O | Q | Q | H | N | D | O | P | D |

El ejemplo anterior, que usaba las cifras de César 2 y 5, correspondería a un Vigenére con clave CF.

La que hoy se conoce como cifra Vigenére tiene una interesante leyenda. Para empezar, resulta que Vigenére desarrolló otros sistemas de cifra más avanzados, que al parecer no fueron muy populares en su época (se creía que eran más complejos de uso, así que durante tres siglos se usó el típico libro de código). La cifra que ahora conocemos bajo su nombre fue

sacada del olvido en el siglo XIX, cuando fue redescubierto. La primera solución criptoanalítica data de 1860.

La cifra Vigenére parece haber adquirido un aura de invulnerabilidad con el tiempo. David Kahn narra cómo, en 1917, la revista Scientific American (hoy traducida bajo el nombre de Investigación y Ciencia) afirmaba que la cifra Vigenére era "imposible de traducir [atacar]". Incluso en fecha tan tardía como la década de los ochenta, Bruce Schneier afirmó que los primeros modelos de teléfonos móviles estadounidenses usaban un sistema de cifrado basado en la cifra Vigenére. !Sorprendente récord de longevidad!.

Este tipo de cifrado es razonablemente seguro, si se usa correctamente, sin embargo han sido vulnerados mensajes codificados con este sistema, mediante el estudio de la repetición de bloques de letras: la distancia entre un bloque y su repetición suele ser múltiplo de la palabra tomada como clave; por ello ha tenido modificaciones para impedir su vulnerabilidad.

En conclusión se puede decir que el cifrador vigenére, resulta seguro si se tiene una clave con longitud considerable;  de la misma manera se debe contar con un canal seguro de envió.

El cifrador vigenere también presenta la misma debilidad del cifrador Afín en donde por  análisis de frecuencias se podría vulnerar el cifrador.

## 1.3 EL ALGORITMO RSA

Este algoritmo se basa en que es muy fácil encontrar números primos de gran tamaño, pero muy difícil factorizar su producto. Básicamente el algoritmo funciona de la siguiente forma:

Obtención de la pareja de llaves(clave pública y clave privada):
Elegimos dos números primos grandes, p y q.
Calculamos su producto n = p * q
Calculamos otro número $\Phi(n) = (p) * (q)$ : que se constituye en la clave de descifrado.
Escogemos un número e, donde $\Phi(n)$  y d sean primos entre si (no tienen factores comunes).
Buscamos un valor W, donde $e* d = 1 \bmod \Phi(n)$
La clave pública obtenida estaría constituida por la pareja de números (e, n) y la clave secreta por (d, n).

**Ejemplo:**

Para poder comprender mejor el funcionamiento del RSA, se ilustra el siguiente ejemplo:

p=3, q=5. Se eligen dos números primos.
n=p*q $\rightarrow$ n=3*5=15.
$\Phi$(n)=(3-1)*(5-1)=8.
Tomando e=3, se obtiene d=3 por medio de la formula
e* d = 1 mod $\Phi$(n) = (3*3)mod8 = 9mod8 = 1.
Clave pública es(n,e)=(15,3)
Tomando el mensaje m=2 su cifrado es:
c = m^e mod n
c = 2^3 mod 15 = 8.

El mensaje que se enviaría sería el número 8. Para poder interpretar de manera correcta, el mensaje se debe someter a un proceso de descifrado, como lo es:

m = c^d mod n = 8^3 mod 15 $\rightarrow$ m = 512mod15 = 2.

Con lo cual se obtiene el mensaje correcto. Se debe anotar que si la clave empleada no es la correcta, el mensaje que se recibe no será el adecuado.

**Ventajas**

La ventaja de RSA es que no se tiene que enviar el password con el mensaje cifrado, en lugar de ello se envia una LLAVE PUBLICA con el mensaje (e y n), por lo cual, la persona usa entonces esta clave para cifrar el archivo.

La única forma de decifrar el mensaje es usando la llave PRIVADA (d y n);
las llave se guarda en secreto, d y n es como su password particualr, e y n es como el password que tu le das a otros para que puedan cifrar mensajes que tu leas.

### 1.4 **DES** ( *Data Encryption Algorithm* )

El DEA ( *Data Encryption Algorithm* ) o DES ( *Data Encryption Standard* ) es desde 1977 de uso obligatorio en el cifrado de informaciones gubernamentales no clasificadas (anunciado por el *National Bureau of Standards* , USA). Este criptosistema fué desarrollado por IBM como una variación de un criptosistema anterior, Lucifer, y posteriormente, tras algunas comprobaciones llevadas a cabo por la NSA estadounidense, pasó a transformarse en el que hoy conocemos como DES. Este criptosistema puede ser implementado tanto en software como en chips con tecnología VLSI ( *Very Large Scale Integration* ), alcanzando en hardware una velocidad de hasta 50 Mbs. Un ejemplo de implantación hard puede ser PC-Encryptor, de Eracom, y un ejemplo de implantación en software es DES-LOCK, de la empresa Oceanics.

El DES es un sistema de clave privada tanto de cifrado como de descifrado. Posee una clave de entrada con una longitud de 64 bits, produciendo una salida también de 64 bits, con una clave de 56 bits (el octavo bit de cada byte es de paridad), llamada clave externa, en la que reside toda la seguridad del criptosistema, ya que el algoritmo es de dominio público.

Cada trozo de 64 bits de los datos se desordena según un esquema fijo a partir de una permutación inicial conocida como IP. A continuación, se divide cada uno de los trozos en dos mitades de 32 bits, que se someten a un algoritmo durante 16 iteraciones. Este algoritmo básico que se repite 16 veces (llamadas vueltas), utiliza en cada una de ellas 48 de los 56 bits de la clave (estos 48 bits se denominan clave interna, diferente en cada vuelta). Estas claves internas se utilizan en un orden para cifrar texto (llamemoslas $K^1$, $K^2$,...,$K^{16}$) y en el orden inverso ($K^{16}$,..., $K^1$) para descifrarlo. En cada una de las vueltas se realizan permutaciones, sustituciones no lineales (que constituyen en sí el núcleo del algoritmo DES) y operaciones lógicas básicas, como la XOR. La mitad derecha se transfiere a la mitad izquierda sin ningún cambio; también se expande de 32 hasta 48 bits, utilizando para ello una simple duplicación. El resultado final de una iteración es un XOR con la clave interna de la vuelta correspondiente. Esta salida se divide en bloques de 6 bits, cada uno de los cuales se somete a una sustitución en un bloque de 4 bits (bloque-S, con un rango 0...63) dando una salida también de 4 bits (rango decimal 0...15) que a su vez se recombina con una permutación en un registro con longitud 32 bits. Con el contenido de este registro se efectua una operación XOR sobre la mitad izquierda de los datos originales, convirtiéndose el nuevo resultado en una salida (parte derecha) de 32 bits. Transcurridas las dieciseis vueltas, las dos mitades finales de 32 bits cada una se recombinan con una permutación contraria a la realizada al principio (IP), y el resultado es un criptograma de 64 bits.

Aunque no ha sido posible demostrar rigurosamente la debilidad del criptosistema DES, y actualmente es el más utilizado en el mundo entero, parece claro que con las actuales computadoras y su elevada potencia de cálculo, una clave de 56 bits (en la que recordemos, reside toda la seguridad del DES) es fácilmente vulnerable frente a un ataque exhaustivo en el que se prueben combinaciones de esos 56 bits. Hay que resaltar que el tamaño inicial de la clave, en el diseño de IBM, era de 128 bits; la razón de la disminución no se ha hecho pública hasta el momento. Por si esto fuera poco, otro factor que ha aumentado las controversias y discusiones acerca de la seguridad del DES son dos propiedades del algoritmo: la propiedad de complementación, que reduce el tiempo necesario para un ataque exhaustivo, y la propiedad de las claves débiles, dada cuando el proceso de cifrado es idéntico al de descifrado ($K^1=K^{16}$, $K^2=K^{15}$,..., $K^8=K^9$), que sucede con cuatro claves del criptosistema. Otro *secreto* de IBM (a instancias de la NSA) es la elección y diseño de las *cajas* que DES utiliza para el cifrado.

En varias ocasiones se ha dicho que el DES ha sido vulnerado, sin embargo nunca se ha hecho oficial a pesar de las grandes cantidades de dinero que se ofrecen a quien lo descifre.

Sin embargo el conocer por fuerza bruta la clave del DES acarrea costos (tiempo, hardware y económicos) elevados, terminando por declinar en la intención de descifrado.

**VENTAJAS**

Es el sistema más extendido del mundo, el que más máquinas usan, el más barato y el más probado.

Es rápido y fácil de implementar. Desde su aparición nunca ha sido roto con un sistema práctico.
La fortaleza matemática que presenta. Si se hiciese un seguiemiento al algoritmo pareciese que se perdiera informaciòn pero no, recupera hasta el ultimo bit.

## 2. PRUEBAS

El siguiente cuadro contiene información de tiempos obtenidos en la prueba de cifrado de algunos archivos de texto con los algoritmos estudiados; se considera que esta prueba es suficiente para probar cual de los algoritmos es más rápido sin importar el tamaño del archivo que se cifre.

| Cifrador / Tamaño | Cifrado Afín | Cifrado De Vigenére | DES |
|---|---|---|---|
| 20Kb | 3515ms | 3360 ms | 4015ms |
| 24 kb | 9094ms | 8688ms | 9832ms |
| 28 kb | 14594ms | 14437ms | 17420ms |
| 32Kb | 24454ms | 24140ms | 28215ms |

Como puede notarse en el cuadro anterior, el DES en todos los casos es el de mayor tiempo de duración y el cifrador de Vigeneré el más rápido, todo esto indica que la mejor alternativa es este último, sin embargo analizando más en detalle no tendría sentido cambiar un algoritmo seguro y un poco más lento a uno que es más fácilmente vulnerado y con un tiempo menor de ejecución y dejar a lado otras ventajas importantes como eficacia, robustez y eficiencia; lo anterior sumado a que el objeto principal de este proyecto es la protección de software como mecanismo de regulación en el uso licenciado del mismo, por lo cual se opta por seguir usando el DES por la seguridad que brinda y seguir en el estudio de criptografía tratando cada vez en buscar controles que impidan su vulnerabilidad.

**CONCLUSIONES**

- Los métodos clásicos han demostrado su ineficacia ante la potencia de cálculo de los modernos ordenadores, por lo que no se usan en aplicaciones que necesiten una gran seguridad; sin embargo el DES ofrece una seguridad aceptable que garantiza que las aplicaciones protegidas, objeto de nuestro análisis, difícilmente serán vulneradas por lo que se optó por seguir usando este algoritmo para el cifrado.

- A pesar que el algoritmo DES es el relativamente más lento en su tiempo de ejecución en comparación a los restantes algoritmos analizados, se compensa con la seguridad que brinda, ya que para obtener la clave de descifrado solo se puede determinar con un ataque de fuerza bruta implicando un gran tiempo, costo de adquisición en Hardware y mucho más, lo que dificulta su vulnerabilidad.

- La seguridad ya no se basa en el algoritmo mismo sino en encontrar funciones matemáticas irreversibles, que no tengan solución conocida como es el caso de la factorización de grandes números, las permutaciones y transformadas. Los nuevos métodos dependen más de la investigación y el avance científico que del conocimiento de alguna clase de "secreto" del algoritmo.

## ANEXO 2 INTRODUCCIÓN A MYSQL[22]

[22] Tomado de: http://www.aditel.org/~dpecos/docs/mysql_postgres/x57.html

http://es.tldp.org/Otros/gples/gples.html

http://www.netpecos.org/docs/mysql_postgres/x57.html

MySQL es un sistema de gestión de bases de datos relacional, licenciado bajo la (GPL[23] de la GNU[24]). Este gestor de bases de datos es probablemente el gestor más usado en el mundo del software libre, debido a su gran rapidez y facilidad de uso. Esta gran aceptación se debe en parte, a que existen infinidad de librerías y otras herramientas que permiten su uso a través de gran cantidad de lenguajes de programación, de la misma forma se encuentra el hecho de ser multiplataforma, por lo que corre bajo UNIX, LINUX y Windows sin problemas y además de su fácil instalación y configuración.

MySQL empezó como un proyecto de programación hace mas de 10 años, cuando un programador sueco decidió crear su propio gestor de datos para la aplicación que estaba desarrollando. Hasta ese momento usaba msql, pero vio que se le quedaba corto en algunos aspectos que el podría mejorar directamente. Pronto MySQL fue el número 1 en rendimiento en el ámbito de las bases de datos de código abierto. Posteriormente se creo una empresa sueca, llamada Tux que se dedicó a crear nuevas versiones de MySQL y comercializarlo en ciertos casos.

Una característica importante que le hacen falta a mysql, son las subconsultas aunque gran parte de las veces que se necesitan, es posible rescribirlas de manera que no sean necesarias; pero esta deficiencia entre comillas, se puede superar con JOIN, LEFT JOIN y RIGTH JOIN, que son capaces de suplir las subconsultas en gran parte de los casos, obteniendo, por otra parte, una mayor eficiencia).

En términos generales, mysql resulta ser un gestor de base de datos con importantes características como flexibilidad, rapidez y robustez, que fueron aprovechados en el desarrollo de la aplicación Web, A continuación se dará una breve explicación de la sintaxis que maneja MYSQL y su forma de conexión.

## 2.1 CONEXIÓN A LA BASE DE DATOS MEDIANTE JAVA

### 2.1.1 JDBC SQL RESULTSET

El modelo de datos de java se basa en una serie de objetos especializados que facilitan el procesamiento de una base de datos.

Se debe comunicar un programa o aplicación con una base de datos y más que comunicar se pretende que el programa o aplicación realice una serie de

---

[23] GPL(General Public License, Licencia Pública General ) mas información http://es.tldp.org/Otros/gples/gples.html

[24] GNU (proyecto de desarrollado de software libre llamado ``GNU'' ) mas información http://es.wikipedia.org/wiki/GNU

procesos u operaciones con la base de datos o mejor aun con el conjunto de tablas que contiene una base de datos.

El modo de comunicarse entre un programa o aplicación y la base de datos implica que ambos manejen un lenguaje de programación común, es decir no se puede mandar una instrucción en un lenguaje determinado y esperar que se entienda, una razón muy sencilla es que la base de datos tendría que conocer o comprender todos los lenguajes de programación, para resolver este problema de comunicación es que se usa un lenguaje común de bases de datos que tanto los lenguajes de programación existentes como las bases de datos entienden, este lenguaje común de bases de datos es el **SQL (structured query languaje)** o lenguaje estructurado de consultas.

Para mandar las instrucciones a la base de datos, se usan los siguientes objetos:

- MYSQL CONNECTORJ: Es un objeto (clase) especializado que se utiliza para enlazar e intercambiar información entre MYSQL y JAVA.

- OBJETO CONNECTION: Objeto que se utiliza para establecer la conexión o enlace entre el programa **jsp** y la base de datos en **mysql**.
  Ejemplo: mysql-connector-java-3.0.10-stable-bin.jar

- OBJETO RESULTSET: Es la representación en memoria de las tablas de la base de datos en disco, se puede entender como una tabla virtual, generalmente todos los procesos que se realicen con la tabla (insertar registros, eliminar registros, etc) se realizaran realmente contra un resulset y no provocaran ningún cambio en la tabla física en disco, resulset tiene un conjunto de métodos muy útiles y muy usados para el proceso de los renglones de la tabla virtual.

- OBJETO STATEMENT: Este objeto y sus dos métodos executequery (solo para select de sql) y executeupdate ( solo para insert, update y delete de sql) son los métodos que se utilizaran para comunicarse con la tabla física en disco.

### 2.1.2 PROCEDIMIENTO PARA MANIPULAR UNA BASE DE DATOS CON JSP

El procedimiento que se intenta seguir cuando se construya un programa **jsp** que tenga que manipular una tabla en disco deberá seguir los siguientes pasos:

- Crear una conexión o enlace a la base de datos.
- Abrir la conexión a la base de datos.
- Crear el enlace y cargarlo con la instrucción sql
  - Crear el RESULTSET y cargarlo.
  - Cargar un objeto table de html con el RESULTSET

- Procesar el table de html
- Cerrar resultset, statement, driver o conexión

Un ejemplo es el Siguiente:

```
<%
//creo la  conexion
String DRIVER= "com.mysql.jdbc.Driver";
Class.forName(DRIVER);
String URL = "jdbc:mysql://localhost:3306/final";
//habro la conexion
Connection con = DriverManager.getConnection(URL);
Statement stmt = con.createStatement();
// Crear el enlace y cargarlo con la instrucción sql
ResultSet rs = stmt.executeQuery("select *   from usuarios ");
// Para visitar o procesar todos los renglones de la tabla se usa un ciclo while, y
el metodo RESULTSET.NEXT(),
while (rs.next()) {
%>
<!—hacer una tabla donde se publicaran los resultados -->
<table width="200" border="1">
<tr>
<td width="108">Nombre Usuario </td>
<td width="76"><%=rs.getString("NOM_USUARIO")%></td>
</tr>
</table>
<% }%>
```

#### 2.1.2.1 INSERCIÓN O ADICIÓN DE REGISTROS

Insertar o agregar registros o renglones nuevos a una tabla en la base de datos, es un proceso que usa la siguiente instrucción sql:

INSERT INTO TABLA(CAMPO1,CAMPO2..) VALUES(VALOR1,VALOR2.);

Acá solo se esta usando lo mínimo de cada instrucción sql.

#### 2.1.2.2 BÜSQUEDA DE UN REGISTRO

Es de gran importancia la búsqueda de un registro o renglón determinado, en este proceso el usuario del programa quiere que se despliegue uno y solo un registro de información, proporcionando un dato de búsqueda generalmente la clave o el id, (numero de identificación) del registro.

La solución es sencilla, solo usar otra vez la instrucción select, con el siguiente formato:

```
//Selecciona la tabla y busca en ella los registros correspondientes
ResultSet rs = stmt.executeQuery("select *   from usuarios where ID_USUARIO = 1 ");
```

#### 2.1.2.3 BAJA O ELIMINACIÓN  DE UN REGISTRO

La eliminación es otro proceso  común con las bases de datos La instrucción sql a usar es:

DELETE FROM TABLA WHERE (condición) Y RESULTSET. EXECUTEUPDATE()

### 2.1.2.4 EDICIÓN DE REGISTROS

Editar registros significa cambiar el contenido de algunos de los campos o columnas por nueva información o para corregir algún error de captura original. La instrucción sql a usar es:

UPDATE NOM_TABLA SET(campos) valor Y RESULTSET.EXECUTEUPDATE()

## ANEXO 3 PRUEBAS A USUARIOS FINALES

Este anexo contiene los resultados de pruebas realizadas a la aplicación Web con el fin de conocer la opinión y el nivel de aceptación sobre ciertos aspectos de diseño y funcionamiento; para de esta manera tratar de reducir al mínimo los errores y tomar medidas con fines correctivos.

Aunque el examen trato diversos aspectos relevantes y la base de datos se encontraba poblada con un número considerable de registros, no se considera que la prueba es totalmente satisfactoria; porque el verdadero funcionamiento se puede cuantificar en una situación en que la base de datos se encuentre parcialmente llena y se realicen diferentes transacciones a un mismo tiempo; sin embargo como prueba preliminar se cumplieron todos los objetivos y los resultados obtenidos fueron los esperados.

Para la prueba realizada se tomo como población objetivo a estudiantes de la Escuela de Ingeniería de Sistemas UIS de diferentes niveles, de la misma forma se le asignaron diferentes tipos de usuario; a quienes se les explico ciertos aspectos básicos y se les indago por medio del siguiente cuestionario:

**ENCUESTA GENERAL A USUARIOS FINALES**

Nombre usuario: _____ Fecha: _____
Tipo usuario: _____

**FORMATO DE PREGUNTAS:**

| **Pregunta 1:** |
|---|
| ¿Como le pareció la interfaz de la aplicación? |
| **Explicación:** |
| La interfaz es fácil de manejar y le proporciona las opciones suficientes para la navegación en la aplicación, además los colores son agradables a la vista |
| **Respuesta:** |
| Excelente_____Bueno____Regular_____ Malo_____ |
| **Comentarios:** |
| |

| **Pregunta 2:** |
|---|
| ¿Es claro el ámbito de la aplicación? |
| **Explicación:** |

| Todo software debe tener unos objetivos específicos y a simple vista dar una |
|---|

| |
|---|
| descripción de su dominio, para que y a quien le sirve. |
| **Respuesta:** |
| Si_____No____ |
| **Comentarios:** |
| |
| **Pregunta 3 :** |
| ¿La aplicación permite administrar la información en tiempos aceptables para el buen curso de sus actividades? |
| **Explicación:** |
| Mantenimiento de información: Inserción, modificación o borrado de algún registro. |
| **Respuesta:** |
| Siempre____Casi siempre_____A veces_____ Nunca____ |
| **Comentarios:** |
| |

| |
|---|
| **Pregunta 4:** |
| ¿Es fácil el acceso a las diferentes secciones de la aplicación? |
| **Explicación:** |
| Desde el ingreso a la aplicación, la navegación por las diferentes secciones es clara y le es proporcionada información suficiente para tener seguridad acerca de su ubicación y de las opciones a los siguientes accesos. |
| **Respuesta:** |
| Siempre____Casi siempre_____A veces_____ Nunca____ |
| **Comentarios:** |
| |

| |
|---|
| **Pregunta 5:** |
| ¿La búsqueda en particular de algún dato es sencilla de realizar? |
| **Explicación:** |
| Cuando desea buscar un dato en particular, la aplicación le permite acceder fácilmente a la información que usted como usuario requiere. Además de contar con opciones de búsqueda rápida. |
| **Respuesta:** |
| Siempre____Casi siempre_____A veces_____ Nunca____ |
| **Comentarios:** |
| |

| |
|---|
| **Pregunta 6:** |
| ¿La información que se recopila en ciertos procesos es necesaria y suficiente? |
| **Explicación:** |
| Para ciertos procesos la aplicación solicita el llenado de cierta información que puede parecer exagerada, carente o necesaria. |
| **Respuesta:** |

| |
|---|
| Si____No_____ |

| **Comentarios:** |
| --- |
| |

| **Pregunta 7:** |
| --- |
| ¿Es fácil el acceso al software Libre y puede descargarse sin contratiempos? |
| **Explicación:** |
| Algunas aplicaciones software presentan la opción de descarga para que sean usadas por un período determinado de tiempo. |
| **Respuesta:** |
| Si____No_____ |
| **Comentarios:** |
| |

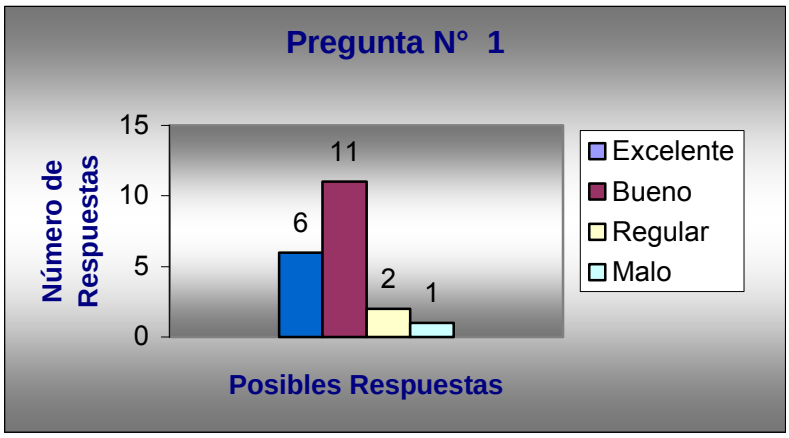
| **Pregunta 8:** |
| --- |
| ¿Es útil el correo de la aplicación para la comunicación con otros usuarios? |
| **Explicación:** |
| La aplicación cuenta con un correo interno para la recepción y envió de solicitudes o mensajes entre los usuarios. |
| **Respuesta:** |
| Si____No_____ |
| **Comentarios:** |
| |

Revisado por:_____

## 3.1 ANÁLISIS DE RESULTADOS

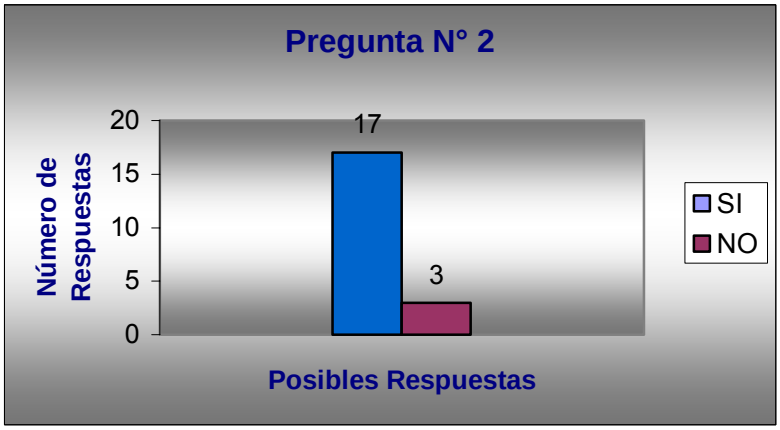
A continuación se mostrara el histograma de los resultados de cada pregunta, el cual se constituye en una herramienta para visualizar y analizar los datos obtenidos para la toma de decisiones.

- **Histograma Resultados Pregunta 1**



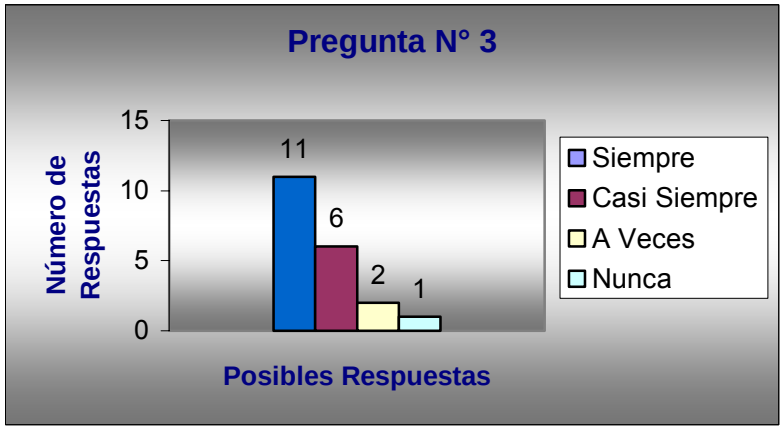

La figura anterior muestra claramente que los resultados obtenidos son bastantes satisfactorios porque 17 personas respondieron afirmativamente y tan solo 3 negativamente.

- **Histograma Resultados Pregunta 2**




17 personas respondieron que es claro el ámbito de la aplicación y solo 3 no lo entendieron.

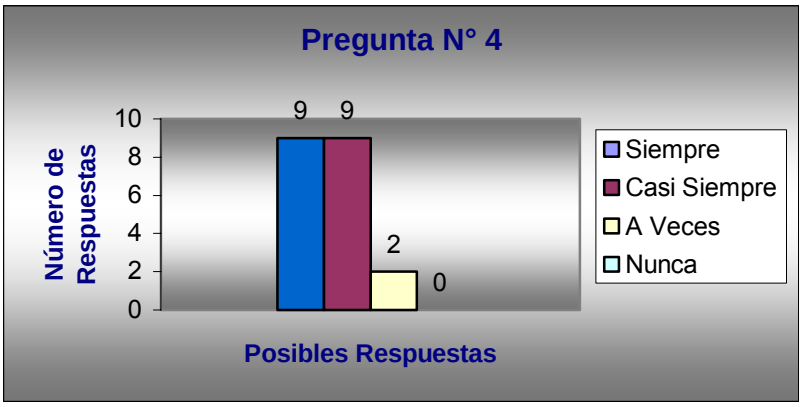
- **Histograma Resultados Pregunta 3**




El gráfico anterior muestra claramente que los resultados referentes a la administración de la información que maneja la aplicación son

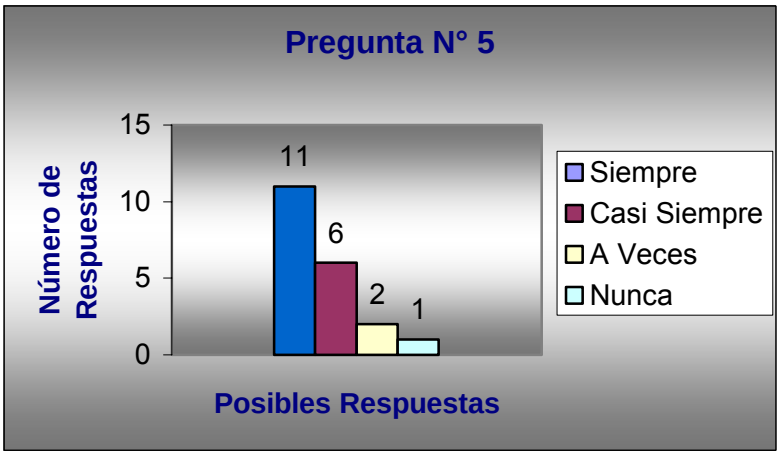
satisfactorios.

- **Histograma Resultados Pregunta 4**



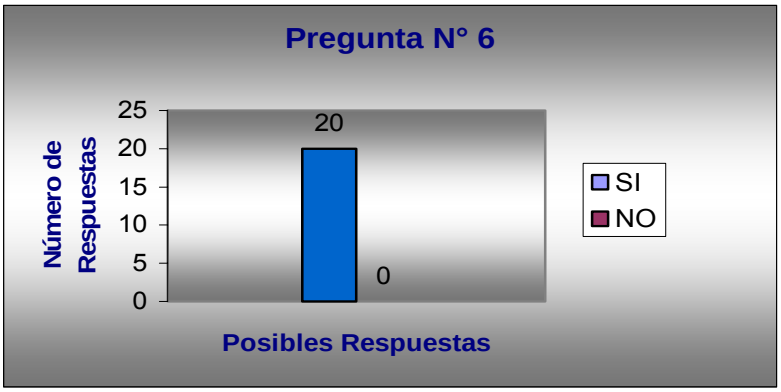

Cuando se pregunto si es fácil el acceso a las diferentes secciones de la aplicación se nota contundentemente que los resultados son altamente positivos.

- **Histograma Resultados Pregunta 5**



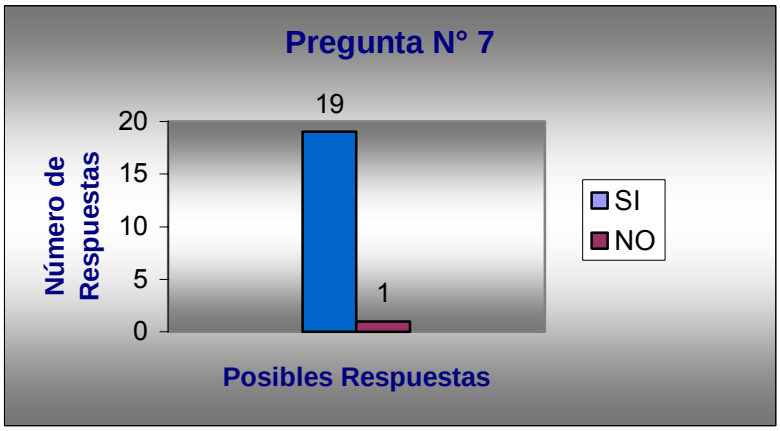

La grafica anterior muestra claramente la satisfacción de los usuarios cuando se indago sobre las búsquedas de información y las opciones con que cuenta la aplicación para esto.

- **Histograma Resultados Pregunta 6**



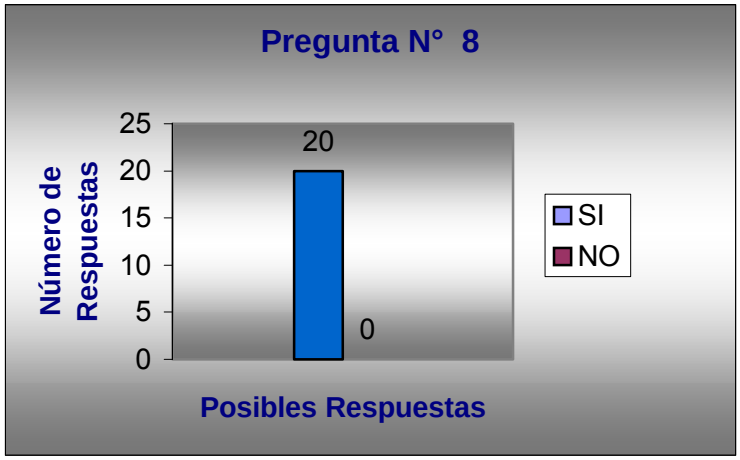

Como lo han venido mostrando los resultados cada vez más, se nota que la satisfacción de los usuarios por la aplicación es alta.

- **Histograma Resultados Pregunta 7**




Aunque para esta prueba se contó con archivos no protegidos y diferente extensión, el proceso de descarga funciona bien.

- **Histograma Resultados Pregunta 8**




Los resultados referente al correo interno de la aplicación demuestran la gran aceptación por parte de los usuarios.

**Tabulación de Datos:**

Para tabular los datos se tuvieron en cuenta las siguientes consideraciones:

- Excelente, Bueno, SI, Siempre, Casi siempre = Cumplió
- Regular, Malo, No, A veces, Nunca = No Cumplió
- Número total de Encuestados = 20.

| | Cumplió | % Cumplió | No Cumplió | No Cumplió |
|---|---|---|---|---|
| **Pregunta1** | 17 | 85% | 3 | 15% |
| **Pregunta2** | 17 | 85% | 3 | 15% |
| **Pregunta3** | 17 | 85% | 3 | 15% |
| **Pregunta4** | 19 | 95% | 1 | 5% |
| **Pregunta5** | 19 | 95% | 1 | 5% |
| **Pregunta6** | 20 | 100% | 0 | 0% |
| **Preguinta7** | 19 | 95% | 1 | 5% |
| **Pregunta8** | 20 | 100% | 0 | 0% |

Los resultados muestran un buen grado de aceptación por parte de los usuarios finales, quienes manifestaron un agrado positivo por la interfaz de la aplicación, muy buenas velocidades de consulta, facilidad en las búsquedas y en términos generales facilidad en el manejo completo de la aplicación.

La encuesta realizada nos permitió además observar la satisfacción de los usuarios frente al producto que se entrega, descrito en la calificación de cada una de las preguntas planteadas. Como respaldo de esta prueba y de lo dicho anteriormente se anexan las encuestas con sus resultados.

**ANEXO 4 DLL Y JNI**

## 4. 1 BIBLIOTECA DE VÍNCULOS DINÁMICOS

Las bibliotecas de vínculos dinámicos, más conocidas como archivos DLL (Dynamic Link Library), son la solución que presenta el sistema operativo Windows ante las desventajas del vínculo estático. Con el vinculo estático el enlace se realiza cuando se genera el archivo .EXE, es aquí cuando el ensamblador vincula el código objeto del programa compilado con el código objeto de la biblioteca temporal y los escribe como un único archivo ejecutable. Si se quiere saber cual es el verdadero código del programa y de la biblioteca por separado seria muy difícil.

La desventaja más grande que presenta el vínculo estático es que no se pueden alterar las funciones llamadas de la biblioteca temporal. Por lo cual si se detecta un error o si se pretende mejorar la eficiencia de la función habrá que generar un nuevo archivo EXE y distribuirlo entre los usuarios. Además si la biblioteca es utilizada por varios programas que se encuentran en ejecución, las funciones utilizadas de la biblioteca serán cargadas varias veces en memoria; estas situaciones presentadas van en contra de los principios planteados por un sistema multitarea, el cual exige un mínimo de desperdicio de espacio de memoria RAM.

El vinculo dinámico funciona de la siguiente manera: el código del programa no se ensambla con el código del programa de la biblioteca temporal.

La biblioteca temporal aparece como un archivo separado, el cual es vinculado durante la ejecución por el programa ejecutable, de forma que se puede llamar a las funciones contenidas. Esta acción permite actualizar las funciones sin necesidad de compilar de nuevo el programa completo, debido a que sustituyendo el archivo DLL y reiniciado el programa entrarían en ejecución las nuevas funciones.

De esta manera trabaja Windows por medio de las API's; cuenta con mas de mil funciones en diferentes archivos DLL, entre ellas:

☑ KERNEL32.DLL. Contiene funciones del sistema para el administrador de memoria, sirve para trabajar con procesos y subprocesos, etc.

☑ USER32. DLL. Rutinas para la interacción con el usuario, como lo son menús, cuadros de diálogos, etc.

☑ GDI32. DLL. Funciones necesarias para dibujar e imprimir.

### 4.1.1 FUNDAMENTOS DE LAS FUNCIONES DLL

En Windows se reconoce un solo tipo de funciones DLL y dos maneras que un programa acceda a ellas: durante el proceso de carga con el shell o durante la ejecución llamando directamente a las correspondientes funciones del API.
El primer caso se conoce como vinculo dinámico en tiempo de carga (Load Time Dynamic Linking), el segundo es un vínculo dinámico en tiempo de ejecución (Run Time Dynamic Linking).

A continuación se hará una breve comparación entre las dos maneras que puede utilizar un programa para acceder una función DLL.

| Vinculo dinámico en tiempo de carga. | Vinculo dinámico en tiempo de ejecución |
|---|---|
| En el código fuente del programa se indica cual función DLL se cargara. Las funciones son tratadas como externas de otros módulos | No se conoce el nombre de la biblioteca y de las funciones que se van a utilizar, sino hasta que el programa entra en ejecución. |

**Tabla 4.1** Comparación de tipos de vínculos de una DLL

### 4.1.2 LOCALIZACIÓN DE UNA FUNCIÓN DLL

Al momento de utilizar una función almacenada en una librería de vinculo dinámico se introduce el nombre del archivo DLL. Si el nombre del archivo no lleva extensión, se le asignara una .DLL. Si le falta la indicación explicita de un paquete de búsqueda se buscara el archivo en diferentes directorios, teniendo en cuenta el siguiente orden:

- El directorio del que se cargo el actual proceso.
- El directorio del sistema de Windows. En Windows XP este directorio es conocido como SYSTEM32, en versiones anteriores a este, el directorio es SYSTEM.
- El directorio de Windows (Windows).
- Los directorios incluidos dentro de la variable de entorno PATH.

De no encontrarse el archivo DLL aparecerá un mensaje de error ya que se retorna el llamado de la función NULL. Si se localiza la función, lo primero que se hará es verificar si ya se cargo al proceso algún archivo DLL del correspondiente directorio con el mismo nombre. Si la respuesta es afirmativa, no es necesario cargar de nuevo el archivo DLL indicado; lo único que se haría seria incrementar el contador interno de usuarios. De lo contrario se cargara el archivo DLL y se cargara en la zona de direcciones del proceso.

### CONSTRUCCIÓN DE UN ARCHIVO DLL

Los archivos DLL tienen su origen en módulos en los que se empaqueta el código de programas de las funciones DLL a exportar.

Las funciones DLL deben hacerse accesibles a otros programas, por lo cual se debe tener cuidado en el tipo de parámetros que recibe y envía la función, ya que si no son compatibles la información nunca llegará. La diferencia de un programa normal y una DLL es que esta última carece de impulso propio, es decir que tiene que ser invocada por otra aplicación alguna de las funciones que contiene para entrar en actividad.

### 4.1.4 EN QUE CASOS SE DEBE USAR UNA DLL

El uso de las DLL es muy útil porque permite aprovechar el mismo código desde diferentes aplicaciones, lo que supone un ahorro significativo de espacio en cada una de las nuevas aplicaciones; de la misma manera se da un ahorro de memoria ya que solo se cargan en memoria cuando se necesitan los recursos.

En el caso particular, se genera la necesidad de trabajar con un lenguaje como java, por la facilidad de interactuar con sistemas operativos como Windows y linux; actualmente se tienen las dlls compiladas en visual c++, en donde se implementaron los algoritmos para la protección (desfin.dll) y generación de claves públicas (mdtrio.dll), de esta manera la aplicación llevada a cabo en java interactúa con la base de datos de la aplicación que esta montada en un servidor linux, y a la vez llevar a cabo el proceso de protección invocando la librería dll, enviando a esta los parámetros necesarios para que se ejecute el proceso correctamente.

para llevar a cabo el proceso anterior (enviar parámetros a una dll desde java) Sun Microsystem creo una interfaz de código nativo denominado JNI; compilado para una determinada plataforma, generalmente escrito en C/C++; de esta forma se puede utilizar código especifico de una plataforma y aprovechar beneficios como velocidad y eficiencia.

## 4.2 JNI (Java Native Interface)

JNI es una interfaz de código nativo (código compilado para una determinada plataforma, ejemplo C) que emplea Sun. JNI permite ejecutar código Java y comunicarnos con librerías escritas en otros lenguajes. La mayor ventaja del JNI es que se puede programar una aplicación o librería nativa y trabajar en cualquier maquina virtual Java que soporte JNI.

### 4.2.1 ¿EN QUE CASOS SE USA JNI?

Siempre se debe tener la clara intención de crear el total de una aplicación en un solo lenguaje para así evitar incompatibilidades, pero existen casos particulares en los que se debe incorporar aplicaciones hechas en otros lenguajes y que son ineludibles para tener una aplicación completa. Este es el caso de: invocar las mencionadas dll's (desfin y mdtrio) desde java para poderlas incorporar al modulo de protección de aplicaciones, siguiendo estos pasos:

- Escritura del programa Java, invocando métodos nativos como native.
- Compilación del programa Java.
- Creación de un archivo de cabecera nativo (.h)
- Escritura de los métodos nativos.
- Creación de una biblioteca con esos métodos nativos.
- Ejecución del programa Java

### 4.2.2 EJEMPLO COMPLETO SOBRE COMO IMPLEMENTAR UNA DLL EN C++ E INVOCARLA CON JAVA MEDIANTE JNI

Para mostrar cómo utilizar los métodos nativos, se va a crear un pequeño programa, escrito con métodos nativos, que lo único que hace es imprimir un comentario *"¡esto es una implementación de jni!"*. Para ello se utilizara el JDK y un compilador de C.

#### 4.2.2.1 ESCRITURA DEL PROGRAMA JAVA.

Se escribe el código en jcrator y se denominara *HolaNativo.java*, este archivo lo guardamos en el directorio correspondiente a la dirección donde están configuradas las variables de entorno del path ejemplo G:\j2sdk1.4.2_03\bin.
El código Java a utilizar es:

```
public class HolaNativo
        {

        public native void diHola();

                static {

                System.loadLibrary("pedro");

                        }

  public static void main( String[] args ) {

    new HolaNativo().diHola();

  }

}
```

El método nativo *diHola()* no tiene cuerpo porque será añadido

mediante una biblioteca nativa denominada *pedro*. Dicha biblioteca nativa es cargada mediante la sentencia *loadLibrary()*, sentencia que ha sido incluida como *static* para que sea ejecutada cada vez que se cree una instancia (objeto) de esta clase.
El programa principal tan sólo instancia un objeto de esta clase y utiliza el método nativo que imprime la cadena de *"¡esto es una implementación de jni!!"*.

#### 4.2.2.2 COMPILACIÓN DEL PROGRAMA JAVA.

Ahora ya es posible compilar la clase Java *HolaNativo* que fue creada mediante la utilizando el JDK) y apoyados en Jcreator.

#### 4.2.2.3 CREACIÓN DE UN ARCHIVO DE CABECERA NATIVO (.H).

Un fichero de cabecera nativo es un fichero que habitualmente tiene la extensión *".h"*. En un fichero de este tipo se definen en C/C++ las interfaces públicas (clases, métodos o funciones, variables globales, constantes).

La herramienta *javah* incluida en el JDK es capaz de crear automáticamente un fichero de cabecera para métodos nativos Java, con sólo invocarla indicando el nombre de la clase de la que extraer las cabeceras nativas:

javah HolaNativo

Esta operación crea un fichero *HolaNativo.h* que será útil para crear los métodos nativos. Hay dos líneas importantes dentro de este fichero:

```
#include <jni.h>

JNIEXPORT void JNICALL Java_HolaNativo_diHola(JNIEnv*,jobject);
```

La primera línea importa una biblioteca JNI que valdrá a C/C++ para saber cómo crear los métodos nativos.

La segunda línea corresponde al método nativo que definimos.

Los métodos nativos suelen denominarse siempre como Java_Paquete_Clase_Metodo() aunque en este caso al no haber paquete esta parte se ha omitido.

Así mismo los métodos nativos reciben como parámetros *JNIEnv** y *jobject* que permitirán al método nativo comunicarse con su entorno.

#### 4.2.2.4 ESCRITURA DE LOS MÉTODOS NATIVOS.

Se ha de crear un fichero fuente nativo, en el que se defina el cuerpo de la función que actuará como método nativo. Este fichero será denominado

*HolaNativo.c*:

```
#include <jni.h>

#include "HolaNativo.h"

#include <stdio.h>

JNIEXPORT void JNICALL Java_HolaNativo_DiHola( JNIEnv* e, jobject o ){

printf("¡esto es una implementacion de jni!!\n");

}
```

En este programa se incluyen tres bibliotecas: La de JNI (*jni.h*), el fichero de cabecera ya creado (*HolaNativo.h*) y una biblioteca de C para imprimir (*stdio.h*). Ser puede observar que el cuerpo del método nativo lo que hace es invocar a la función de C *printf()* que imprimirá por pantalla la cadena *"*¡esto es una implementacion de jni*!"*.

#### 4.2.2.5 CREACIÓN DE UNA BIBLIOTECA CON ESOS MÉTODOS NATIVOS.

Cada compilador de C o C++ tiene su propia herramienta para crear. Para esto cada compilador tiene su propia sintaxis.Para c++ esta la opción Dinamic-link Library, y la aplicación se guarda según la configuración de las variables de entorno de java. En cualquier caso se debe estar seguro de que la biblioteca creada tiene el mismo nombre con que se la invoca desde el archivo de clase Java en el método *loadLibrary()*, de la misma forma, el archivo con extensión .dll generado, se debe colocar en la dirección correspondiente a la configuración del path o en c:\windows\system32.

#### 4.2.2.6 EJECUCIÓN DEL PROGRAMA JAVA

El archivo HolaNativo.java ejecuta mediante jcreator para mayor facilidad, o mediante la instrucción java HolaNativo en una ventana de DOS.

Con lo que se muestra por pantalla:
esto es una implementacion de jni

**ANEXO 5** **BANCO DE**

## DATOS

## TABLAS DE LA BASE DE DATOS

| TABLA | INFORMACIÓN |
|---|---|
| *actualizaciones* | Información del proyecto luego de que se produce la renovación, además de los permisos que se le han otorgado. |
| *clientes* | Clientes que usan aplicaciones protegidas. |
| *usuarios* | Usuarios del sistema, personas propietarias o responsables de aplicaciones protegidas. |
| *permisos* | Identificación de usuario y niveles de accesos. |
| *proyectos* | Información de las aplicaciones Protegidas. |
| *clientes_has_proyectos* | Proyectos que usa un determinado cliente. |
| *clientes_has_usuarios* | Clientes que pertenecen a un determinado usuario. |
| *mensajes* | Repositorio de mensajes enviados entre los actores del sistema. |
| *integrantes* | Información de los integrantes del grupo software “TRIONIX”. |
| *estudios* | Información académica de los integrantes del grupo software “TRIONIX”. |
| *experiencia* | Experiencia laboral de los integrantes del grupo software “TRIONIX”. |
| *eventos* | Noticias y publicaciones del grupo software “TRIONIX”. |

## INFORMACION DE LAS TABLAS

### ✓ TABLA ACTUALIZACIONES

| Campo | Tipo | Longitud | Requerido | Llave primaria | Descripción |
|---|---|---|---|---|---|
| ID_ACTUALIZACION | INT | 10 | SI | SI | identificador de la actualización. |
| FEC_CREACION | DATE | | SI | NO | Fecha de inicio del periodo de la actualización. |
| FEC_EXPIRACION | DATE | | SI | NO | Fecha de fin del periodo de la actualización. |
| T_INICIAL | INT | 5 | SI | NO | Tiempo máximo de uso para el proyecto. |
| T_FINAL | INT | 5 | | NO | Tiempo real de uso de un proyecto. |
| A_INICIAL | INT | 5 | SI | NO | Número máximo de ejecuciones permitidas de un proyecto. |
| A_FINAL | INT | 5 | | NO | Número de ejecuciones reales del proyecto. |
| ESTADO_PROYECTO | INT | 2 | SI | NO | Permiso para que un determinado cliente renueve un proyecto (cliente Automático-No Automático). |
| CLAVE | VARCHAR | 50 | NO | | Clave de la aplicación protegida |

| Llave foránea | Referencia a: | |
|---|---|---|
| | Tabla | Llave Primaria |
| ID_PROYECTO | proyectos | ID_PROYECTO |
| ID-CLIENTE | clientes | ID_CLIENTE |

- ✓ **TABLA CLIENTES**

| Campo | Tipo | Longitud | Requerido | Llave primaria | Descripción |
|---|---|---|---|---|---|
| ID_CLIENTE | INT | 10 | SI | SI | Identificador del cliente. |
| CEDULA | INT | 11 | SI | NO | Número de identificación del cliente. |
| CLIENTEUSUARIO | INT | 11 | SI | NO | Identificador de un cliente como usuario. |
| NOM_CLIENTE | VARCHAR | 50 | SI | NO | Nombre del cliente que usara la aplicación protegida. |
| DIR_CLIENTE | VARCHAR | 30 | SI | NO | Dirección del domicilio del cliente. |
| TEL_CLIENTE | VARCHAR | 20 | SI | NO | Teléfono del cliente. |
| COR_CLIENTE | VARCHAR | 25 | SI | NO | Correo electrónico externo del cliente |
| CORTRIONIX_CLIENTE | VARCHAR | 25 | SI | NO | Correo electrónico del cliente en la aplicación Web. |
| ACTIVO | INT | 2 | SI | NO | Estado del cliente en la aplicación. |

| Llave foránea | Referencia a: | |
|---|---|---|
| | Tabla | Llave Primaria |
| ID_PERMISO | permisos | ID_PERMISO |

- ✓ **TABLA PROYECTOS**

| Campo | Tipo | Longitud | Requerido | Llave primaria | Descripción |
|---|---|---|---|---|---|
| ID_PROYECTO | INT | 10 | SI | SI | Identificador del proyecto |
| NOM_PROYECTO | VARCHAR | 45 | SI | NO | Nombre del proyecto protegido. |
| FEC_CREACION__PRO | DATE | | SI | NO | Fecha de creación del proyecto. |
| MODALIDAD | VARCHAR | 25 | SI | NO | Modalidad de desarrollo del proyecto. |
| URL_PROYECTO | VARCHAR | 150 | NO | NO | Dirección URL del proyecto en el servidor, caso opción de descarga. |
| DESCRIPCION | TEXT | | SI | NO | Descripción del proyecto protegido. |
| LONGITUD | VARCHAR | 10 | NO | NO | Longitud en Kb del archivo del ejecutable de la aplicación protegida. |
| DESARROLLADOR | MEDIUMTEXT | | SI | NO | Desarrollador (es) del proyecto protegido. |
| PERMISO | INT | 2 | SI | NO | Permiso o no de descarga de un proyecto protegido. |
| DIAS_USO | INT | 3 | SI | NO | Número días máximo de uso del proyecto que se descarga. |
| ACCESOS_PERMITIDOS | INT | 3 | SI | NO | Número máximo de ejecuciones del proyecto que se descarga. |

| Llave foránea | Referencia a: | |
|---|---|---|
| | Tabla | Llave Primaria |
| ID_USUARIO | usuarios | ID_USUARIO |

- ✓ **TABLA USUARIOS**

| Campo | Tipo | Longitud | Requerido | Llave primaria | Descripción |
|---|---|---|---|---|---|
| ID_USUARIO | INT | 10 | SI | SI | Identificador de usuario. |
| USUARIOCLIENTE | INT | 10 | SI | NO | Identificador del usuario como cliente. |
| NOM_USUARIO | VARCHAR | 50 | SI | NO | Nombre de Usuario o empresa. |
| APEL_USUARIO | VARCHAR | 20 | NO | NO | Apellidos del usuario. |
| DOCUMENTO_USUARIO | VARCHAR | 20 | SI | NO | Identificación del usuario como persona natural o jurídica. |
| TIPO_DOCUMENTO | INT | 5 | SI | NO | Tipo de Identificación. |
| TEL_USUARIO | VARCHAR | 15 | NO | NO | Teléfono fijo o móvil del usuario. |
| COR_USUARIO | VARCHAR | 25 | SI | NO | Correo electrónico del usuario. |
| CORTRIONIX_USUARIO | VARCHAR | 25 | SI | NO | Correo interno del usuario en la aplicación Web. |
| INST_USUARIO | VARCHAR | 30 | SI | NO | Institución u organización al cual pertenece el usuario. |
| TIEMPO_ADMINIST | DATE | | SI | NO | Periodo de administración de clientes y aplicaciones protegidas por parte del usuario. |
| ACTIVO | INT | 3 | SI | NO | Estado del usuario en la aplicación: Activo o Inactivo. |

| Llave foránea | Referencia a: | |
|---|---|---|
| | Tabla | Llave Primaria |
| ID_PERMISO | permisos | ID_PERMISO |

- ✓ **TABLA CLIENTES_HAS_PROYECTOS**

| Campo | Tipo | Longitud | Requerido | Llave primaria | Descripción |
|---|---|---|---|---|---|
| ID_RELA | INT | 4 | SI | SI | Identificador relación cliente-proyecto. |
| TIPO_PRO | INT | 4 | SI | NO | Tipo de protección del proyecto que usa un cliente. |

| Llave foránea | Referencia a: | |
|---|---|---|
| | Tabla | Llave Primaria |
| ID_CLIENTE | clientes | ID_CLIENTE |
| ID_PROYECTO | proyectos | ID_PROYECTO |

- ✓ **TABLA CLIENTES_HAS_USUARIOS**

| Campo | Tipo | Longitud | Requerido | Llave primaria | Descripción |
|---|---|---|---|---|---|
| ID_RELACU | INT | 4 | SI | SI | Identificador relación cliente-usuario. |

| Llave foránea | Referencia a: | |
|---|---|---|
| | Tabla | Llave Primaria |
| ID_CLIENTE | clientes | ID_CLIENTE |
| ID_USUARIO | usuarios | ID_USUARIO |

- ✓ **TABLA PERMISOS**

| Campo | Tipo | Longitud | Requerido | Llave primaria | Descripción |
|---|---|---|---|---|---|
| ID_PERMISO | INT | 10 | SI | SI | Identificador del permiso. |
| LOGIN | VARCHAR | 20 | SI | NO | Nombre del usuario para efecto de identificación en la aplicación. |
| PASSWORD | VARCHAR | 40 | SI | NO | Password para garantizar la integridad. |
| NIVEL | INT | 2 | SI | NO | Tipo de usuario. |
| FECHA_ACCESO | DATE | | SI | NO | Fecha más actual de ingreso de un usuario o cliente. |
| HISTORIAL_INGRESOS | TEXT | | SI | NO | Fechas de ingreso de un usuario a la aplicación. |
| PREGUNTA_SECRETA | VARCHAR | 255 | NO | NO | Mecanismo Para que un usuario recuerde su password. |
| RESPUESTA | VARCHAR | 150 | NO | NO | Mecanismo Para que un usuario recuerde su password. |
| INGRESOS | INT | 11 | SI | NO | Número de ingresos de un usuario a la aplicación. |

| Llave foránea | Referencia a: | |
|---|---|---|
| | Tabla | Llave Primaria |
| ID_CLIENTE | clientes | ID_CLIENTE |
| ID_USUARIO | usuarios | ID_USUARIO |

- ✓ **TABLA MENSAJES**

| Campo | Tipo | Longitud | Requerido | Llave primaria | Descripción |
|---|---|---|---|---|---|
| ID_MENSAJE | INT | 10 | SI | SI | Identificador del mensaje. |
| USUARIO_REC | INT | 10 | SI | NO | Identificador del usuario que recibe el mensaje. |
| CLIENTE_REC | INT | 10 | SI | NO | Identificador del cliente que recibe el mensaje. |
| NOCLIENTEUSUARIO | VARCHAR | 20 | SI | NO | Correo electrónico del navegante Web que envía un correo a un usuario. |
| ASUNTO | VARCHAR | 20 | SI | NO | Asunto o titulo del mensaje. |
| DESCRIPCIÓN | TEXT | | SI | NO | Descripción del mensaje |
| FECHA | DATE | | SI | NO | Fecha de envío del mensaje |

| Llave foránea | Referencia a: | |
|---|---|---|
| | Tabla | Llave Primaria |
| ID_USUARIO | usuarios | ID_USUARIO |
| ID_CLIENTE | clientes | ID_CLIENTE |

- ✓ **TABLA EVENTOS**

| Campo | Tipo | Longitud | Requerido | Llave primaria | Descripción |
|---|---|---|---|---|---|
| ID_EVENTO | INT | 10 | SI | SI | Identificador del evento. |
| TITULO | VARCHAR | 50 | SI | NO | Titulo del evento o publicación. |
| FECHA | DATE | | SI | NO | Fecha del evento. |
| TIPO_EVENTO | INT | 4 | SI | | Tipo de evento: Noticia o publicación. |

- ✓ **TABLA INTEGRANTES**

| Campo | Tipo | Longitud | Requerido | Llave primaria | Descripción |
|---|---|---|---|---|---|
| ID_INTEGRANTES | INT | 10 | SI | SI | Identificador del Integrante. |
| CEDULA | INT | 10 | SI | NO | Número de identificación del Integrante. |
| NOMBRE | VARCHAR | 50 | SI | NO | Nombre y apellidos del integrante. |
| NACIONALIDAD | VARCHAR | 50 | SI | NO | Nacionalidad del integrante. |
| FECHA_NACIMIENTO | DATE | | SI | NO | Fecha de nacimiento del integrante. |
| CIUDAD | VARCHAR | 50 | SI | NO | Lugar de nacimiento del integrante. |
| SEXO | CHAR | 1 | SI | NO | Sexo del integrante. |
| EMAIL | VARCHAR | 50 | SI | NO | Correo electrónico del Integrante. |
| PERFIL_PROFESIONAL | VARCHAR | 50 | SI | NO | Cargo desempeñado del integrante en el grupo. |
| DESC_PERFIL | VARCHAR | 100 | SI | NO | Descripción del perfil profesional del integrante. |
| FOTOURL | VARCHAR | 50 | NO | NO | Dirección URL en el servidor donde se encuentra el archivo de imagen del integrante(foto). |

- ✓ **TABLA ESTUDIOS**

| Campo | Tipo | Longitud | Requerido | Llave primaria | Descripción |
|---|---|---|---|---|---|
| ID_ESTUDIOS | INT | 10 | SI | SI | Identificador hoja de vida académica del integrante. |
| IDIOMAS | MEDIUMTEXT | | SI | NO | Idiomas manejados por el integrante. |
| | | | | | |
| | | | | | |
| INSTITUCION_PRIMARIA | VARCHAR | 100 | SI | NO | Nombre de la institución donde el integrante finalizo sus estudios primarios. |
| INSTITUCION_SECUNDARIA | VARCHAR | 100 | SI | NO | Nombre de la institución donde el integrante finalizo sus estudios secundarios. |
| AÑO_PRIMARIA | VARCHAR | 6 | SI | NO | Año de finalización de los estudios primarios del integrante. |
| AÑO_SECUNDARIA | VARCHAR | 6 | SI | NO | Año de finalización de los estudios secundarios del integrante. |
| INSTITUCION_SUPERIOR | MEDIUMTEXT | | SI | NO | Institución(es) de educación superior donde el integrante obtuvo un determinado titulo. |
| TITULOS | MEDIUMTEXT | | SI | NO | Título(s) obtenido(s) por el integrante. |
| AÑO_SUPERIOR | MEDIUMTEXT | | SI | NO | Año(s) de obtención del (los) titulo(s). |

| Llave foránea | Referencia a: | |
|---|---|---|
| | Tabla | Llave Primaria |
| ID_INTEGRANTES | integrantes | ID_INTEGRANTES |

- ✓ **TABLA EXPERIENCIA**

| Campo | Tipo | Longitud | Requerido | Llave primaria | Descripción |
|---|---|---|---|---|---|
| ID_EXPERIENCIA | INT | 10 | SI | SI | Identificador hoja de vida laboral del integrante. |
| EMPRESA | MEDIUMTEXT | | SI | NO | Empresa(s) donde ha laborado el integrante. |
| CARGO | MEDIUMTEXT | | SI | NO | Cargo(s) desempeñado(s) por el integrante. |

| Llave foránea | Referencia a: | |
|---|---|---|
| | Tabla | Llave Primaria |
| ID_INTEGRANTES | integrantes | ID_INTEGRANTES |

## ANEXO 6 CONFIGURACION MODULO DE PROTECCIÓN

Estos son los pasos que debe seguir el administrador del sistema para instalar en su PC el modulo de protección de aplicaciones software. Todo el software necesario para la instalación se encuentra en el CD adjunto.

1. Instalar correctamente el j2sdk (Maquina Virtual de Java), SIGUIENDO LOS PASOS DE LA INSTALACIÓN.

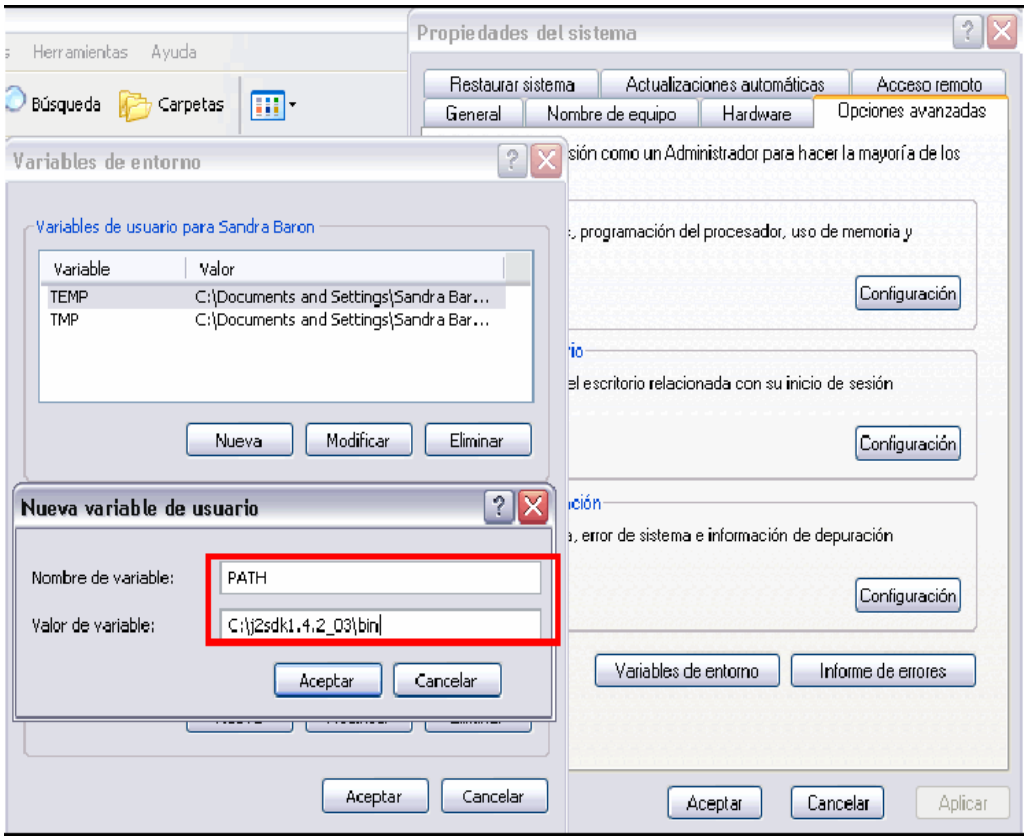
2. Configurar las variables del sistema CLASSPATH y PATH necesarias para que se reconozcan los comandos de JAVA desde cualquier ubicación en el sistema de archivos de Windows.

   Ejemplo de configuración de la variable de entorno PATH en Windows XP:

   Siga los siguiente Pasos:

✓ Vaya a: inicio >> Panel de Control >> Sistema >> Opciones Avanzadas >> Variables de Entorno

✓ Seleccionar NUEVA y digitar en variable de entorno PATH y en valor de la variable C:\j2sdk1.4.2_03\bin



Con lo anterior se logra que el sistema operativo reconozca los comandos (archivos ejecutables de java como JAVA, JAVAC y APPLEVIEWER) desde cualquier directorio del sistema.

3. Configurar el controlador JDBC siga los siguientes pasos:

   - Coloque el controlador JDBC en el lugar correspondiente, para que se pueda acceder a la Base de Datos:

   - Vaya a: **C:\j2sdk1.4.2_03\jre\lib\ext** Y guarde alli el *.jar del controlador que se encuentra en el CD.

     mysql-connector-java-3.0.10-stable-bin.jar

4. Guarde en C:\WINDOWS\SYSTEM32 o SYSTEM los archivos: pprotector.exe, protector.exe, MFC42D.dll,MSVCRTD.dll,MFCO42D.dll,MSVCP60D.dll

5. Guarde el archivo "TRIONIX.exe" en el disco y ejecútelo, Tenga en cuenta usar la misma versión del jdk, para este caso j2sdk1.4.2_03, de lo contrario puede tener problemas de incompatibilidad

6. Digite el Login y Password asignado.

7. Ingrese los datos del Usuario y la aplicación a proteger.

## ANEXO 7

## LA PIRATERÍA Y LA PROTECCIÓN DE LA INFORMACIÓN

El principal problema que se presenta a la hora de desarrollar y distribuir un producto software es la incertidumbre que acarrea su divulgación; por un lado el beneficio y por otro el riesgo asumido por exponer su trabajo a que terceros distribuyan ilegalmente su aplicación y desconozcan su esfuerzo y dedicación.

Las cifras en el ámbito mundial no mienten y desnudan el problema de la piratería del software, según la Alianza de Programas Nacionales (Bussines Software Alliance) informó en su primer estudio Anual Mundial, que la industria del software perdió cerca de 29.000 millones de dólares en ventas por piratería en el 2003, reflejando un aumento acelerado anual en perdidas; sin embargo datos más reveladores es que importantes industrias han pasado de adquirir software del mercado legal al mercado negro.

Luego de analizar brevemente las cifras de la BSA[25], surge el interrogante como empezar a combatir este flagelo?. Primero se debe entrar a analizar las causas, entre las cuales se destaca el desempleo, la inadecuada legislación y la carencia o desconocimiento de productos Hardware y Software para proteger las aplicaciones desarrolladas. Si miramos la principal causa le compete al estado, la segunda a la industria del software y estado y la tercera a la industria de software y a profesionales afines a esta.

Como Ingenieros de sistemas nos vemos en la tarea de enfrentar esta problemática, primero desarrollando calidad y no cantidad; segundo proponiendo alternativas de protección para la pequeña y mediana empresa productora de software. Una de estas alternativas es el uso de la herramienta para la protección de aplicaciones TRIONIX[26], que permite proteger archivos EXE Windows 32-bit. usando técnicas criptografícas y mecanismos de seguridad como clave publica y clave privada.

Herramienta Software “TRIONIX”

TRIONIX es un sistema que permite proteger aplicaciones software utilizando algoritmos criptográficos como el DES (Data Encryption Standard) y el RSA (Rivest, Shamir y Adleman). Se envuelve alrededor de un programa como una coraza, para protegerlo contra piratas y programas crackers permitiendo su uso durante un periodo pactado con los usuarios. Este software funciona con archivos EXE de Windows 32-bit, independientemente del lenguaje en que fueron desarrollados.

El sistema cuenta con dos formas de protección: Por número de accesos y por limite de tiempo, las cuales se seleccionan a la hora de proteger la aplicación por primera vez o a la hora de renovarla.

- **Protección por límite de tiempo:** El uso de la aplicación expira en un cierto número de días después que el usuario lo ejecuta por primera vez o en un número de días establecido, previo acuerdo al contrato.
- **Protección por uso:** El uso de la aplicación expira después de cierto número de ejecuciones.

### ¿Cómo funciona?

La herramienta Trionix se basa en conceptos de manejo de archivos, manejo de procesos y criptografía;

[25] Alianza de Programas Nacionales (Bussines Software Alliance) Primer estudio mundial de piratería Junio de 2004

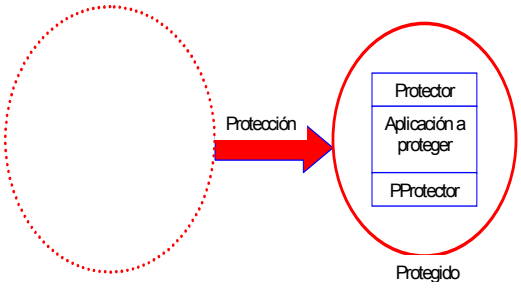
[26] Herramienta para la protección de aplicaciones software

cuenta con los módulos de protección, renovación y consulta.

***Protección:***

Para proteger una aplicación inicialmente se requiere:

1. **Aplicación a proteger**. Archivo ejecutable de la aplicación que el usuario quiere brindar protección.
2. **Protector**. Encargada de la verificación de permisos.
3. **Pprotector**. Monitorear el correcto flujo de eventos de las operaciones a realizar entre el protector y la aplicación a proteger.




*Descripción general TRIONIX*

Como se ilustra en la figura anterior, inicialmente todas las aplicaciones se encuentran separadas, el proceso comienza cuando se tiene una aplicación a proteger y se invoca al algoritmo DES para proteger y el RSA para generar los permisos de acceso(clave privada).

Finalmente luego de realizada la protección, la aplicación protegida se encontrara dentro de un nuevo ejecutable embebida entre las aplicaciones *Protector y Pprotector.*

***Renovación:***

Una vez el Pprotector verifica que la protección de una aplicación ha expirado, invoca el RSA para generar la clave publica y un mensaje solicitando la nueva clave de uso particular(clave privada).

El proceso de renovación comienza cuando el cliente (persona que usa la aplicación) contacta al usuario (dueño de la aplicación) para solicitarle la clave que le permita el ingreso a la aplicación. Luego de este primer contacto, el usuario debe contactar al Administrador Trionix para que le suministre la clave ya que este es la única persona que tiene acceso al modulo de renovación de aplicaciones.

***Consulta:***

Conjunto de consultas generales o especificas de acuerdo al perfil de usuario asignado.

**El proceso de Protección:**

Cuando se desea proteger una aplicación software el sistema solicita el tipo de protección, en este caso la aplicación se protegerá por tiempo y accesos, es decir después del quinto día al igual que a la tercera ejecución del mismo.

El sistema durante el proceso de protección garantiza que la aplicación a proteger no sea copiada cuando se encuentra descifrada en el disco; solo seria posible copiarla cuando se termina de grabar (fin del proceso descifrar), pero inmediatamente comienza su ejecución, lo cual por seguridad el sistema operativo impide copiar un archivo en ejecución. Cuando la aplicación a proteger deja de estar en ejecución es borrada de manera que en ningún momento puede ser copiada. Además de esto se toman medidas que mejoran la seguridad de la aplicación a proteger cuando se encuentra en disco descifrada, como son:

- El nombre de este archivo en el disco es generado de manera aleatoria. Su longitud es de mínimo cuatro (4) caracteres y máximo diez caracteres (10) que comprender mayúsculas y minúsculas sin incluir signos.

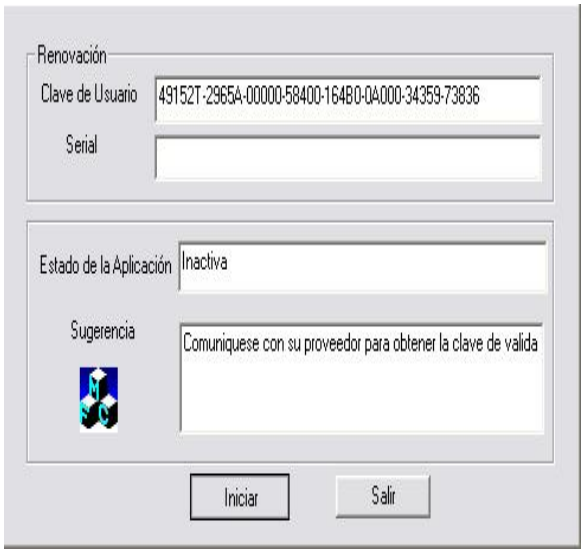
- Al terminar de descifrar el archivo, se coloca invisible al usuario.

Se espera dificultar la labor de cualquier persona que intente violar el esquema de seguridad aplicado.

### Primera ejecución

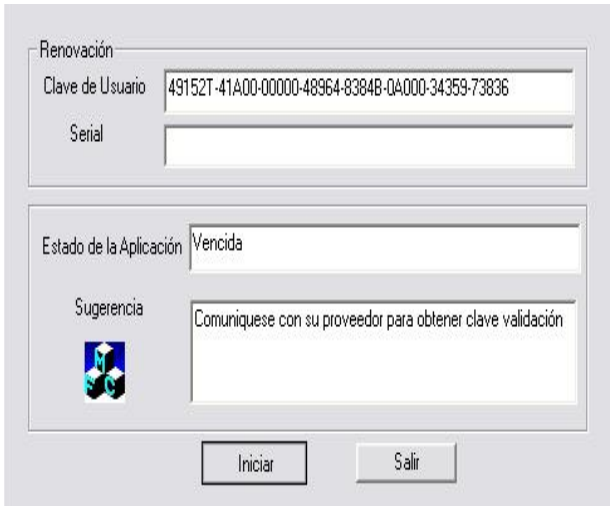
La primera vez que se ejecuta la aplicación protegida aparece en un estado inactiva, mostrando un serial y solicitando otro para su comparación. El usuario de la aplicación debe suministrar este serial al administrador TRIONIX para que le sea devuelto el serial solicitado (clave privada).

El proceso de generación de la clave debe ser autorizado por el creador de la aplicación. Una vez se introduzca el serial y se presione iniciar, la aplicación debe empezar su ejecución y seguir el flujo de eventos.




### Vencimiento

Cuando una aplicación se vence (por tiempo o accesos) la aplicación protector invoca funciones que le permiten generar la clave pública y la trama que le será suministrada al usuario al momento de iniciar la ejecución de la aplicación protegida. La generación de la clave (serial) se realiza mediante el algoritmo RSA que esta implementado en una librería dll[3]. Esta librería es invocada por la aplicación protector cuando alguno de los permisos otorgados se vence.



**Renovación**

Cuando la aplicación se vence se genera una clave (publica) que debe ser suministrada al administrador TRIONIX, quien procesa la información de actualización y genera la clave privada (serial).

Cuando la aplicación se vence se debe solicitar la renovación sin cerrar la aplicación vencida, ya que cada vez que se ejecuta, se genera una clave diferente. Esto se hace para evitar que la clave pública a suministrar sea alterada por medio de la copia o reemplazo.

**Ámbito de Trionix**

Como se ha venido mencionando "Trionix" es una herramienta que protege ejecutables EXE en ambiente WIN 32, generando un nuevo paquete ejecutable.

A pesar de que no todo el software esta compuesto por los programas ejecutables, si es un buen punto de partida para empezar a combatir el fenómeno de la piratería; pues son los ejecutables el resultado final del desarrollo software que va ser visto y utilizado por los usuarios.

**Acerca de los Algoritmos Criptográficos**

**DES**(Data Encryption Standard)

DES (Data Encryption Standard) es un esquema de encriptación simétrico desarrollado en 1977 por el Departamento de Comercio y la Oficina Nacional de Estándares de EEUU en colaboración con la empresa IBM, que se creó con objeto de proporcionar al público en general un algoritmo de cifrado normalizado para redes de ordenadores. Estaba basado en la aplicación de todas las teorías criptográficas existentes hasta el momento, y fué sometido a las leyes de USA.

Posteriormente se sacó una versión de DES implementada por hardware, que entró a formar parte de los estándares de la ISO con el nombre de DEA.

Se basa en un sistema monoalfabético, con un algoritmo de cifrado consistente en la aplicación sucesiva de varias permutaciones y sustituciones. Inicialmente el texto en claro a cifrar se somete a una permutación, con bloque de entrada de 64 bits (o múltiplo de 64), para posteriormente ser sometido a la acción de dos funciones principales, una función de permutación con entrada de 8 bits y otra de sustitución con entrada de 5 bits, en un proceso que consta de 16 etapas de cifrado.

En general, DES utiliza una clave simétrica de 64 bits, de los cuales 56 son usados para la encriptación, mientras que los 8 restantes son de paridad, y se usan para la detección de errores en el proceso.

**Acerca del RSA**

El algoritmo de clave pública RSA fue creado en 1978 por Rivest, Shamir y Adlman, y es el sistema criptográfico asimétrico más conocido y usado. Estos señores se basaron en el artículo de Diffie-Hellman sobre sistemas de llave pública, crearon su algoritmo y fundaron la empresa RSA Data Security Inc., que es actualmente una de las más prestigiosas en el entorno de la protección de datos.

El sistema RSA se basa en el hecho matemático de la dificultad de factorizar números muy grandes. Para factorizar un número el sistema más lógico consiste en empezar a dividir sucesivamente éste entre 2, entre 3, entre 4,..., y así sucesivamente, buscando que el resultado de la división sea exacto, es decir, de resto 0, con lo

que ya tendremos un divisor del número.

Ahora bien, si el número considerado es un número primo (el que sólo es divisible por 1 y por él mismo), tendremos que para factorizarlo habría que empezar por 1, 2, 3,..... hasta llegar a él mismo, ya que por ser primo ninguno de los números anteriores es divisor suyo. Y si el número primo es lo suficientemente grande, el proceso de factorización es complicado y lleva mucho tiempo.

Basado en la exponenciación modular de exponente y módulo fijos, el sistema RSA crea sus claves de la siguiente forma:

1. Se buscan dos números primos lo suficientemente grandes: **p** y **q** (de entre 100 y 300 dígitos).
2. Se obtienen los números **n**=p* q y **Ø** = (p-1) * (q-1).
3. Se busca un número **e** tal que no tenga múltiplos comunes con Ø.
4. Se calcula **d** = $e^{-1}$ mod Ø, con mod = resto de la división de números enteros.

Y ya con estos números obtenidos, **n es la clave pública y d es la clave privada**. Los números p, q y Ø se destruyen. También se hace público el número e, necesario para alimentar el algoritmo.

### *¿ Papel del RSA en Trionix?*

El algoritmo RSA es de vital importancia en los procesos de protección y renovación, pues es el que genera y valida los permisos de uso de la aplicación protegida aprovechando su fortalezas en los cálculos matemáticos, tornando dificultoso la obtención de alguna clave.

### Posibles Mejoras

Aunque los algoritmos criptográficos usados son seguros, existen mecanismos y expertos en la materia dedicados al negocio de la piratería; por esto la herramienta continua en proceso de mejoramiento de aquellos aspectos relevantes e importantes en el funcionamiento de la herramienta.

Consientes de esto el grupo software a propuesto la implementación de una aplicación web que permita la administración de usuarios y la automatización de los procesos de renovación y consulta, además de contar con un canal de divulgación de aplicaciones desarrolladas por la EISI, para que personas de cualquier parte del mundo sin costo alguno usen el producto durante cierto periodo de tiempo y de esta manera se pueda mostrar el potencial en el desarrollo software de la escuela.

**Grupo Software "Trionix"**

Ing. William Villamizar Vera
Pedro Erlendis González P.
Darwin Walter Rey C.

*Codirector.*
Ing. Juan Gabriel Quintero P.

*Director.*
Msc. Luis Ignacio González R.

Febrero de 2005