UNIVERSIDAD POLITECNICA SALESIANA

SEDE GUAYAQUIL

CARRERA DE INGENIERIA EN SISTEMAS

PROYECTO TECNICO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO DE SISTEMAS CON MENCIÓN EN INFORMÁTICA PARA LA GESTIÓN.

TEMA:

“Sistema Inmunizador de Aplicaciones y Control Organizacional”

AUTOR:

ENRIQUE MACHADO CEDEÑO

TUTOR:

ING. MIGUEL QUIROZ

GUAYAQUIL - ECUADOR

JULIO - 2018

DECLARATORIA DE RESPONSABILIDAD Y AUTORIZACIÓN DE USO DELTRABAJO DE GRADO

Yo Enrique Machado Cedeño autorizo a la Universidad Politécnica Salesiana la publicación total o parcial de este trabajo de grado y su reproducción sin fines de lucro.

Además, declaro que los conceptos y análisis desarrollados y las conclusiones del presente trabajo son de exclusiva responsabilidad del autor.

Nombres y apellidos completos

CC

CESIÓN DE DERECHOS DE AUTOR

Yo ENRIQUE MACHADO CEDEÑO, con documento de identificación N° 0924282437, manifiesto mi voluntad y cedo a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que soy autor del trabajo de grado intitulado: "Sistema Inmunizador de Aplicaciones y Control Organizacional", mismo que ha sido desarrollado para optar por el título de: INGENIERO EN SISTEMAS CON MENCION EN INFORMATICA PARA LA GESTION, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En aplicación a lo determinado en la Ley de Propiedad Intelectual, en mi condición de autor me reservo los derechos morales de la obra antes citada. En concordancia, suscribo este documento en el momento que hago entrega del trabajo final en formato impreso y digital a la Biblioteca de la Universidad Politécnica Salesiana.

(Firma)

.....

Nombre: Enrique Machado Cedeño

Cédula: 0924282437

Fecha: 26/JULIO/ 2018

CERTIFICACIÓN DE TUTOR

Habiendo sido nombrado, **ING. Miguel Quiroz** tutor de grado como requisito para optar por el título de Ingeniero en Sistemas con mención en Informática para la Gestión presentado por el egresado:

ENRIQUE MACHADO CEDEÑO CON C.I # 0924282437

TEMA: "Sistema Inmunizador de Aplicaciones y Control Organizacional"

CERTIFICO QUE: He revisado y aprobado en todas sus partes, encontrándose apto para su sustentación.

_____

ING. MIGUEL QUIROZ

TUTOR

GUAYAQUIL, 26/JULIO/2018

AGRADECIMIENTO

Agradezco siempre y ante todo a Dios, por las bendiciones que ha enviado hacia mí para llegar a la meta deseada y haberme concedido la dicha de hacer realidad este sueño anhelado.

A la UNIVERSIDAD POLITECNICA SALESIANA, por darme la oportunidad de estudiar y ser profesional.

A mi tutor de tesis, ING. MIGUEL QUIROZ, por su esfuerzo y dedicación, quien con sus conocimientos, su experiencia, su paciencia y su motivación ha logrado en mí que pueda terminar mis estudios con éxito.

También me gustaría agradecer a mis profesores, que durante toda mi carrera profesional han aportado con un granito de arena a mi formación ahora como profesional.

Quiero agradecer profundamente también a mis padres quienes fueron un pilar muy fundamental en el camino de este gran recorrido.

Un agradecimiento muy especial a mi esposa e hijos que son el motor que me impulsaron para seguir adelante y culminar esta etapa de la vida.

Son muchas las personas que han formado parte de mi vida profesional a las que me encantaría agradecerles por su amistad, sus consejos, su apoyo, ánimo y compañía en los momentos más difíciles de mi vida.

Para ellos: Muchas gracias y que Dios los bendiga

Enrique Machado Cedeño.

DEDICATORIA

A mis padres quienes me han dado la existencia; y en ella la capacidad para superarme y desearme siempre lo mejor en cada paso que doy por este camino difícil y arduo de mi vida. Gracias por ser como son, porque con vuestra presencia han ayudado a construir y a forjar la persona que hoy en día soy.

A mis hijos, quienes han sido mi mayor motivación para nunca rendirme en mis estudios y poder llegar a ser ejemplo para ellos.

A mi esposa, quien me ha sabido apoyar día a día contribuyendo incondicionalmente a lograr metas y objetivos propuestos.

Nada de lo que expreso en este documento son suficientes para darle las gracias por el apoyo, comprensión y consejos en momentos difíciles.

Enrique Machado Cedeño.

RESUMEN

El control a accesos de las aplicaciones dentro de una empresa es algo muy importante porque permite controlar de manera eficaz el tiempo y actividades que realiza cada empleado de acuerdo a sus funciones, es importante contar con una aplicación que evite el contagio de los ordenadores con algún virus informático.

Por estas razones nace la idea de implementar una herramienta informática que me permita combatir los problemas acceso que existen en la empresa RESNORTE SA.

La herramienta propuesta es una aplicación desarrollada en C# que permite controlar que un usuario no ejecute programas sin la autorización del supervisor de redes, de esta manera la aplicación ayudará a la empresa a tener un control sobre sus empleados y alargar el tiempo de vida útil de los equipos informáticos.

El programa permite que solo se ejecuten programas que estén dentro de una “lista blanca”, si una aplicación sospechosa quiere ejecutarse, el programa avisará y bloqueará la ejecución.

ABSTRACT

Control access to applications within a company is very important because it allows to effectively control the time and activities performed by each employee according to their functions, it is important to have an application that prevents the transmission of computers with some computer virus.

For these reasons born the idea of implementing a computer tool that allows me to combat the access problems that exist in the company RESNORTE SA.

The proposed tool is an application developed in C # that allows you to control that a user does not run programs without the authorization of the network supervisor, in this way the application will help the company to have control over its employees and lengthen the useful life of the computer equipment.

The program allows only programs that are inside a "whitelist" to run, if a suspicious application wants to run, the program will warn and block the execution.

ÍNDICE GENERAL

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCIÓN

El Sistema Inmunizador de Aplicaciones y Control Organizacional es una herramienta que permite satisfacer la necesidad de la empresa Resnorte SA en cuanto al manejo de los permisos de acceso a Aplicaciones en los equipos informáticos.

El sistema sirve de control para los usuarios, éstos en ocasiones pierden su tiempo instalando programas para uso personal, o a su vez para prevenir la ejecución de software maligno, el sistema controlará lo que se ejecute en cada PC.

La parametrización se hará en base a listas blancas donde cada usuario tendrá acceso a una lista determinada de programas dependiendo del tipo de trabajo y la necesidad del usuario.

Si un programa que no esté dentro de la lista intenta ejecutarse, el programa no dejará que se abra, hasta pedir la autorización al administrador del sistema.

Básicamente el sistema ayudará a la empresa Resnorte SA a controlar y optimizar mejor el tiempo de sus empleados y ayudará a cuidar los equipos informáticos ante cualquier software malintencionado, por lo que el sistema bloqueará cualquier ejecución que no esté dentro de la lista permitida.

El siguiente documento presenta información del proyecto y de la empresa que se va a implementar el programa, los antecedentes, la problemática que tiene la empresa para controlar el uso de programas, el diseño de la base de datos, las funciones del programa, los casos en los que puede ser usada la aplicación, las pruebas, conclusiones y recomendaciones para el buen manejo de la aplicación.

CAPÍTULO I

EL PROBLEMA

1.1 Planteamiento del Problema

La empresa RESNORTE SA tiene problemas con infecciones de virus en sus ordenadores y no posee el control para restringir a los usuarios que no ejecuten programas basuras (juegos o software que no sea de uso exclusivo para su trabajo), por el tiempo que les toma y se distraen o por el peligro de infectar a la máquina y red de virus y malwares.

Muchos de los empleados quienes trabajan para la empresa RESNORTE SA, han olvidado cuán importante es enfocarse en optimizar el tiempo de trabajo y cuidar los equipos informáticos que tienen a su cargo como medio de trabajo.

1.2 Formulación del problema

¿Cómo puedo controlar el uso de aplicaciones y ejecución de programas para optimizar el tiempo de los empleados de la empresa RESNORTE SA?

1.3 Sistematización del problema

¿Cómo podrá un software optimizar el tiempo de uso de los ordenadores de cada usuario?

¿Cómo disminuir el número de infecciones por virus o malwares?

¿La empresa tendrá información más a tiempo si utilizamos el software?

1.4 Antecedentes

La empresa RESNORTE SA durante muchos años ha trabajado con programas gratuitos de antivirus y no ha tenido un plan de control de infecciones por el mal uso que los empleados le dan a los recursos informáticos, descargando del internet programas para uso personal, o para distracción dentro del horario de trabajo, trayendo éstos programas consigo virus o malwares que dañan los sistemas operativos y entorpecen su jornada laboral.

La compañía dentro del área administrativa, tiene a su cargo a 36 empleados, cada uno de ellos maneja un ordenador para sus tareas diarias.

En la actualidad no existe un control para cada uno de ellos que pueda garantizar si están haciendo bien o no su trabajo, o controlar en qué ocupan su tiempo de trabajo.

1.5 Importancia y Alcances

Los usuarios pierden mucho tiempo en descargar archivos que vienen en muchas ocasiones infectados de virus y lo instalan sin ninguna restricción, la aplicación ayudará a la empresa a que controle estas acciones por parte de los usuarios finales.

La causa más común de daños de ordenadores y/o archivos son los virus, por lo cual se necesita de una herramienta que ayude a la empresa a controlar las infecciones informáticas.

Tener un control del recurso humano dentro de la empresa es vital para el negocio, de esta manera se maximiza el rendimiento de cada empleado y puede producir más dentro de la jornada laboral.

Cada empleado es responsable de la información que procesa dentro de la empresa según el área en donde se desenvuelva.

Ante estas situaciones las empresas hoy en día son conscientes de la importancia que tiene el administrar bien los recursos humanos para tener al día información que es de suma importancia para la toma de decisiones.

Si la empresa desea maximizar la utilidad e importancia que posee la información, el negocio debe manejar de forma correcta y eficiente todos los recursos asociados al recurso humano como son los ordenadores y todo lo que ellos utilizan para su trabajo.

Actualmente la empresa cuenta con el 50% de los ordenadores con un rendimiento bajo, es decir, muy lentas, con archivos personales en los pc, y programas instalados considerados “basura” para la compañía, por el motivo de no contar con un programa que les permita tener el control sobre su recurso informático.

Con el Sistema Inmunizador de Aplicaciones y Control Organizacional que se está proponiendo para la empresa, busca minimizar el desperdicio de tiempo del empleado y mejorar el rendimiento de los ordenadores, restringiendo la instalación y ejecución de programas sean estos de manera manual (impulsada por el usuario/empleado) o automática (virus, malwares, etc.).

En la figura 1 se muestra el porcentaje de empleados que dedica tiempo a tareas no relacionadas con el trabajo y las horas que ocupan al día.

Figura 1: Tiempo en el Trabajo

Fuente: Elsupervisor.com, 2012

El proyecto tiene como grupo de beneficiarios a todos los empleados de las oficinas de la empresa RESNORTE SA en todas sus áreas, como son:

Usuarios finales: Gerencia General, Administración, RRHH, Operaciones, Control de Calidad, Importaciones, Mercadeo, Pagaduría, Auditoria, Financiero, Contabilidad.

Administrador del sistema: Sistemas

El área de sistemas tendrá controles de acceso de supervisor y se encargaría de dar soporte y ayuda a los demás usuarios dentro de la empresa.

1.6 Delimitación del Proyecto

1.6.1 Espacio:

- País: Ecuador
- Región: Costa
- Cantón: Guayaquil
- Sector: Empresa RESNORTE SA

La empresa está ubicada en la Av. Juan Tanca Marengo Km 1 1.5 C.C Dicentro Oficina 12.

Las áreas comprendidas para el uso del sistema están comprendidas en departamentos los cuales son:

- Contabilidad
- Auditoria
- Mercadeo
- Operaciones
- Pagaduría
- Financiero
- Calidad
- Importaciones
- Gerencia General
- Administración
- Sistemas

La empresa está dividida en 2 secciones planta baja y planta alta (mezanine) como lo muestran las figuras 2 y 3 a continuación.

Figura 2: División área planta baja

Elaborado por: Autor

Figura 3: División área planta alta

Elaborador por: Autor

1.6.2 Tiempo:

El proyecto tendrá una duración de 9 meses a partir de su aceptación y podrá utilizarse en la empresa indefinidamente.

1.7 Objetivos

1.7.1 Objetivo General

Desarrollar un software que permita controlar el acceso a las aplicaciones y la ejecución de programas para optimizar el tiempo de trabajo de cada usuario.

1.7.2 Objetivos Específicos

Implementar la aplicación en cada uno de los equipos de la empresa RESNORTE SA.

Capacitar sobre el uso de la aplicación y sus beneficios.

Ayudar a la empresa a optimizar el tiempo de los empleados.

CAPÍTULO II

MARCO TEÓRICO

2.1 Antecedentes

Los virus informáticos tienen su origen desde 1955, donde un grupo de programadores empezó como un juego denominado “COREWAR” que consistía en enviar instrucciones a modo de competencia a la memoria y el que consumía primero la memoria del ordenador era el ganador. (Alexander Pineda, 2016)

Pero en si del primer virus informático que se tiene constancia es “CREEPER” un malware que ataco a una maquina IBM en 1972. Este lanzaba un mensaje a la pantalla que para eliminarlo surgió la idea de crear un antivirus, el primero registrado llamado “Reaper”. (Ginnet Bello, 2016)

Ante estos conocimientos, los empleados de la empresa RESNORTE SA no logran identificar lo que es un virus informático, porque ejecutan cualquier archivo por curiosidad, tratándose muchas veces de malwares.

El tiempo dentro de un área de trabajo es muy importante para la organización y para el dueño de la empresa, esto permite obtener información a tiempo y que muchas veces es importante para el proceso de toma de decisiones.

2.2 Marco Conceptual

Capacitación

Proceso educativo corto por el que se adquieren conocimientos y habilidades técnicas para lograr metas. (Definicionabc, 2007)

Ordenador

Es una máquina electrónica que recibe y procesa datos con la misión de transformarlos en información útil. (Definicionabc, 2007)

Hardware

Son los componentes físicos de la computadora, es decir lo que se puede ver y tocar. (Harold Ávila, 2016)

Software

Es un conjunto de instrucciones, programas y reglas usadas en la informática para ejecutar varias tareas en un ordenador. (Julián Pérez Porto, 2008)

Sistema Operativo

"Un Sistema Operativo es un "programa" que actúa como interface entre el usuario y el hardware de un ordenador, ofreciendo el entorno necesario para que el usuario pueda ejecutar programas." Stallings (2001)

Virus Informático

Los virus informáticos son programas informáticos que tienen como objetivo alterar el funcionamiento del ordenador, sin que el usuario final se dé cuenta. Estos, por lo general, infectan otros archivos del sistema con la intensión de modificarlos para destruir de manera intencionada archivos o datos almacenados en el ordenador. (GCF Community Foundation, 1998)

Archivo (fichero)

Un archivo o fichero informático es un conjunto de bits que son almacenados en un dispositivo. Un archivo es identificado por un

nombre y la descripción de la carpeta o directorio que lo contiene. (Fundación Wikipedia, 2016)

Base de datos (Database)

Es el conjunto de informaciones almacenadas en un soporte legible por ordenador y organizadas internamente por registros (formado por todos los campos referidos a una entidad u objeto almacenado) y campos (cada uno de los elementos que componen un registro). Permite recuperar cualquier clase de información: referencias, documentos textuales, imágenes, datos estadísticos, etc. (María Pinto, 2015)

Dato

Es el componente mínimo de una información mayor. Necesario para llegar al conocimiento exacto de una cosa o hecho, que describe objetos, situaciones, etc. (Patricia Dip, 2008)

Sistema

Es un conjunto de elementos relacionados entre sí que contribuyen a un determinado fin. Donde cada elemento tienen su propia función y en conjunto generan una nueva función. (Patricia Dip, 2016)

Aplicación

Aunque se suele utilizar indistintamente como sinónimo genérico de programa es necesario subrayar que se trata de un tipo de programa específicamente dedicado al proceso de una función concreta dentro de la empresa. (Pep Cardona, 1997)

SQL SERVER

SQL Server es un sistema de gestión de bases de datos relacionales (RDBMS) de Microsoft que está diseñado para el entorno empresarial. SQL Server se ejecuta en T-SQL (Transact -SQL), un conjunto de extensiones de programación de Sysbase y Microsoft que añaden varias características a SQL estándar, incluyendo control de transacciones, excepción y manejo de errores, procesamiento fila, así como variables declaradas. (Margaret Rouse, 2015)

Visual Studio

Es un conjunto completo de herramientas de desarrollo para la generación de aplicaciones web ASP.NET, Servicios Web XML, aplicaciones de escritorio y aplicaciones móviles, tales como son: Visual Basic, Visual C# y Visual C++. (Microsoft, 2016)

C#

C# es un lenguaje de programación que se ha diseñado para compilar diversas aplicaciones que se ejecutan en .NET Framework. C# es simple, eficaz, con seguridad de tipos y orientado a objetos. (Microsoft, 2016)

Lista Blanca

Una lista blanca, lista de aprobación ó whitelist en Inglés es una lista o registro de entidades que, por una razón u otra, pueden obtener algún privilegio particular, servicio, movilidad, acceso o reconocimiento, es decir todo lo que este contenido en esta lista podrá ejecutarse, permitirse o autorizarse. (Wikipedia, 2015)

Código Fuente

Texto escrito en un lenguaje de programación específico y que puede ser leído por un programador. (Leandro Alegsa, 2016)

Lenguaje de Programación

Es un sistema de comunicación que posee una determinada estructura, contenido y uso. (Julián Pérez Porto y María Merino, 2012)

Programación

Es el procedimiento de escritura del código fuente de un software, es decir la programación le indica al programa informático que acción debe llevar a cabo y cuál es el modo de concretarla. (Julián Pérez Porto y María Merino, 2012)

Levantamiento de información

Es un proceso mediante el cual el analista recopila datos e información de la situación actual de un sistema, con el propósito de identificar problemas y oportunidades de mejora. Se lleva a cabo mediante el uso de instrumentos y técnicas como: entrevista, encuesta, inspecciones. (José Miguel Machado, 2015)

Análisis de Requerimientos

Comprende todas las tareas relacionadas con la determinación de las necesidades o de las condiciones a satisfacer para un software nuevo o modificado, tomando en cuenta los diversos requisitos de las partes interesadas, que pueden entrar en conflicto entre ellos. (Wikipedia, 2016)

Scalar (Metodo)

Ejecuta una consulta y devuelve la primera columna de la primera fila del conjunto de resultados devuelto por la consulta. Las demás columnas o filas no se toman en cuenta. (Microsoft, 2000)

DataTable

Un objeto DataTable representa una tabla de datos relacionales de la memoria; los datos son locales de la aplicación basada en .NET en la que residen, pero se pueden llenar desde un origen de datos como Microsoft SQL Server. (Microsoft, 2017)

Entidad

Es la representación de un objeto o concepto del mundo real que se describe en una base de datos. (Leandro Alegsa, 2016)

CAPÍTULO III

MARCO METODOLÓGICO

Para la realización del proyecto la metodología a seguir incluirá los siguientes puntos:

3.1 Recopilación de Información

Para conocer las principales amenazas que existen hoy en día en la red, se ha buscado información en la web y así estar al tanto para poder armar una lista de los principales virus y sus variantes, además del lenguaje de programación a utilizar y el motor de base de datos.

3.2 Desarrollo del Proyecto

Para el desarrollo de este proyecto se usara como herramienta de programación Visual Studio Community en su lenguaje C# y un motor de base de datos SQL SERVER EXPRESS, en donde se almacenará una lista sobre las aplicaciones que tendrán permiso a ejecutar los usuarios.

3.3 Demostración del Proyecto (Factibilidad)

El proyecto se basa en el control de la ejecución de aplicaciones por parte de los usuarios a partir de una base de datos que almacenará una lista blanca, el cual se declara como un proyecto factible que se puede aplicar a todas las áreas de la empresa y que funcionará en sistemas operativos Windows 7 o posterior, según lo que se pretende demostrar.

CAPÍTULO IV

ANALISIS Y DISEÑO DEL SISTEMA

4.1 Nombre del Sistema

“Sistema Inmunizador de aplicación y Control Organizacional”

4.2 Logotipo del Sistema

Figura 4: Logotipo del Sistema

Elaborado por: Autor

4.3 Análisis

En primer lugar hay que tener en cuenta que la aplicación se ejecutará en una empresa donde los usuarios finales son un grupo totalmente heterogéneo, por lo que la usabilidad debe ser lo más fácil y clara que sea posible.

El Sistema que ayudará a la solución de la problemática de la empresa RESNORTE SA, se ha dividido en 2 componentes:

- Consola Administrativa
- Monitor de Ejecución

Cada componente o modulo cumplirá tareas específicas y tendrá roles de acuerdo a cada usuario como se lo resume en la siguiente tabla:

Tabla 1: Módulos del Sistema

MODULO	ROL	TAREAS
Consola Administrativa	Administrador	Configuración por usuarios de programas que se puedan usar y/o ejecutar por medio de una lista blanca definida. Revisión de peticiones de usuarios para ejecutar alguna aplicación.
Monitor de ejecución	Cliente	Monitorea los ordenadores y restringe la ejecución de programas según la lista blanca. Solicitar al administrador permisos para ejecutar alguna aplicación. Usuarios deberán logonearse al sistema para poder utilizar la pc.

Elaborador por: Autor

4.3.1 Análisis Técnico

El Sistema Inmunizador de Aplicaciones y Control Organizacional se ha desarrollado dentro de los parámetros que la empresa RESNORTE SA nos ha permitido trabajar.

Para ello se han utilizado las herramientas gratuitas como son:

- SQL SERVER EXPRESS EDITION
- VISUAL STUDIO COMMUNITY

La empresa RESNORTE SA, para la implementación del sistema, técnicamente ya cuenta con los demás requisitos para el funcionamiento del programa que son:

- Los ordenadores con sistema operativo Windows 7
- La red de datos

- Las divisiones o departamentos
- Usuarios

4.4 Requisitos Funcionales

En la siguiente lista se dan los requisitos funcionales que cumple la aplicación agrupado según el tipo de usuario que necesita dichas funcionalidades:

Administrador:

- Crea un nuevo usuario: Crea un usuario con su clave para la autenticación en el sistema.
- Edita un usuario existente: Modifica datos del usuario y lo puede inactivar cuando se necesite.
- Crea plantillas (listas blancas): Crea la base de programas se puede ejecutar y se asigna a un usuario existente.
- Solicitudes: Aprueba las solicitudes que envían los usuarios para ejecutar algún programa específico.

Usuario:

- Envía solicitud: Usuario puede enviar una solicitud al administrador del sistema para poder ejecutar un programa en específico.
- Ver la lista de programas cargados.
- Ver la lista de programas permitidos.

4.5 Perfiles de Usuarios

Es necesario hacer una distinción entre los tipos de usuarios que usaran la aplicación:

Administrador: Es el único que puede ingresar a la consola administrativa y el encargado crear las cuentas de usuarios que tengan que usar la aplicación.

Usuario Final: Son los usuario mayoritarios básicamente pasivos porque solo tienen la función de enviar solicitudes para poder ejecutar algún programa.

4.6 Documentación de tareas

Tabla 2: Tareas

TAREAS	USUARIOS	ENTRADAS	SALIDAS	POSIBLES PROBLEMAS
Login	Todos	Usuario y contraseña	Devuelve la autenticación de usuario	La entrada no coincida con la base de datos
Crear nuevo usuario	Administrador	Datos del nuevo usuario		El usuario ya este creado
Crear plantilla (lista blanca)	Administrador	Programas que puede ejecutar y asignación de usuario		La plantilla ya este creada
Enviar solicitud	Usuario Final	Ruta y nombre del programa que quiere ejecutar y el motivo	Mensaje del sistema de envío	
Aprobar Solicitud	Administrador	Solicitud el Usuario	Mensaje del sistema de aprobación o negación	

Elaborador por: Autor

4.7 Casos de Usos

Para el sistema se ha definido los siguientes casos de uso:

- Login
- Crear Usuario
- Crear plantilla
- Envío y aprobación de solicitud de ejecución de programas
- Logout

Caso de uso 1: Login

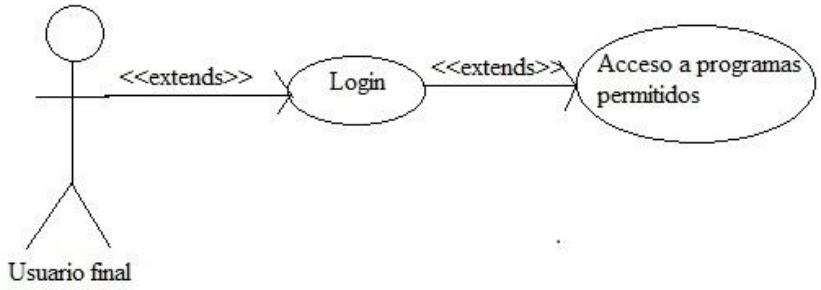


Figura 5: Login

Elaborador por: Autor

Tabla 3: Caso de Uso 1

Caso de Uso 1: Login
Actores: Usuario Final
1) El usuario final ingresa su nombre de usuario y contraseña.
2) Los datos son validados en la base de datos.
3) Accede al sistema y programas permitidos.

Elaborador por: Autor

Caso de uso 2: Crear Usuario

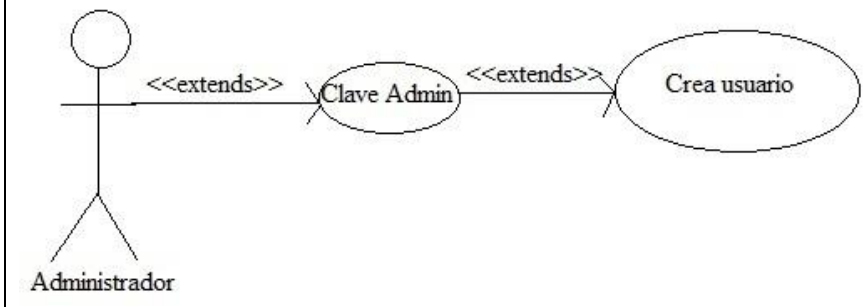


Figura 6: Login

Elaborador por: Autor

Tabla 4: Caso de Uso 2

Caso de Uso 2: Crear Usuario
Actores: Administrador
1) El administrador deberá introducir los datos como nombres, usuario, la contraseña y darle un estado activo o inactivo.
2) Los datos se guardan en la base de datos.
3) La contraseña será inicialmente el nombre del usuario.

Elaborador por: Autor

Caso de Uso 3: Crea Plantilla

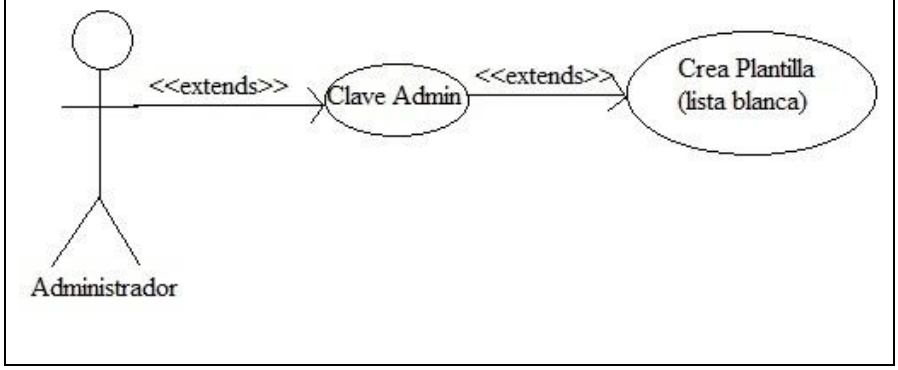


Figura 7: Login

Elaborador por: Autor

Tabla 5: Caso de Uso 3

Caso de Uso 3: Crear Plantilla
Actores: Administrador
1) El administrador deberá introducir el nombre de la lista como identificador, el estado activo o inactivo, el nombre o ruta de los programas que va a tener acceso esa lista.
2) El administrador deberá seleccionar según criterio o área definida por la empresa qué usuario accederá a la lista.
3) Los datos serán almacenados en la base de datos.

Elaborador por: Autor

Caso de Uso 4: Envío y aprobación de solicitud de Ejecución de Programas

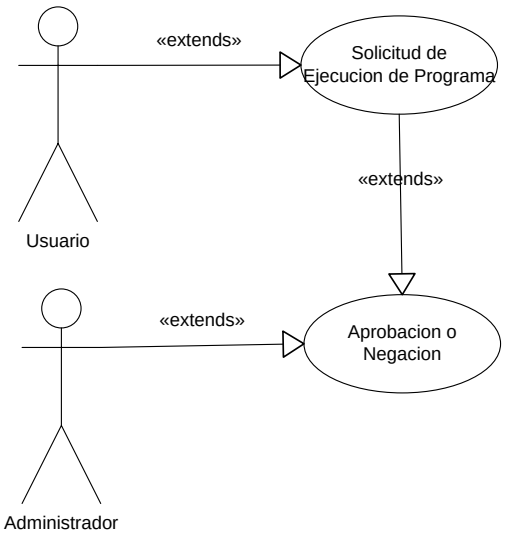


Figura 8: Envío y aprobación de solicitud de Ejecución de Programas

Elaborador por: Autor

Tabla 6: Caso de Uso 4

Caso de Uso 4: Solicitud de Ejecución de Programas
Actores: Usuario y Administrador
1) El usuario ejecuta un programa y la aplicación responde con una solicitud.
2) La solicitud es leída por el administrador.
3) EL Administrador Aprueba o Niega la solicitud según las políticas de la empresa y permisos necesarios.

Elaborador por: Autor

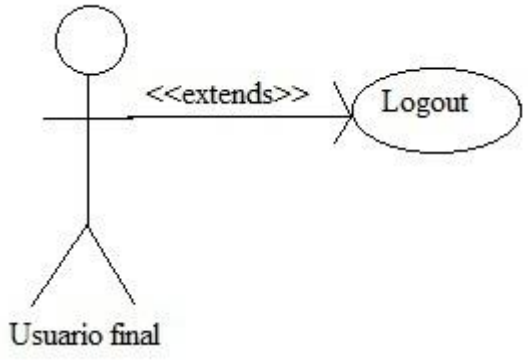
Caso de Uso 5: Logout


Figura 9: Login

Elaborador por: Autor

Tabla 7: Caso de Uso 5

Caso de Uso 5: Logout
Actores: Usuario
1) El usuario se desautentica.
2) Se retorna a la pantalla de Login.

Elaborador por: Autor

4.8 Modelo Entidad Relación

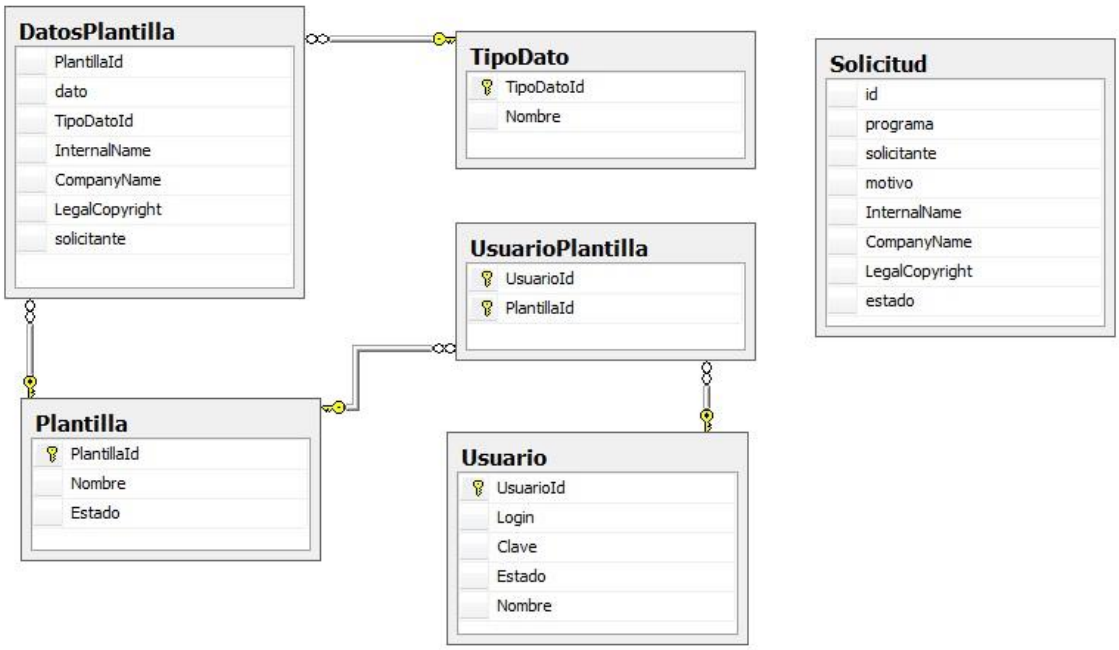


Figura 10: Diagrama General Entidad Relación

Elaborado por: Autor

4.9 Diccionario de datos

El Sistema Inmunizador de Aplicaciones y Control Organizacional está compuesto por 6 tablas que mencionaremos a continuación:

Tabla 8: Datos de Plantilla

Nombre de Tabla	DatosPalntilla		
Descripción	Se encuentran las plantillas de accesos a los programas		
Nombre	**Tipo**	**Descripción**	**PK**
Plantillaid	Int	Numero de plantilla	Si
Dato	Nvarchar(255)	Nombre o ruta del programa	
TipoDatoid	Int	Tipo de dato	

Internalname	Nvarchar(100)	Nombre del programa	
CompanyName	Nvarchar(100)	Nombre de la compañía del programa	
LegalCopyright	Nvarchar(150)	CopyRight del programa	
Solicitante	Nvarchar(50)	Usuario Solicitante	

Elaborador por: Autor

Tabla 9: Plantilla

Nombre de Tabla	Plantilla		
Descripción	Se encuentran los nombres de las plantillas		
Nombre	**Tipo**	**descripción**	**PK**
Plantillaid	Int	Numero de plantilla	Si
Nombre	Nvarchar(200)	Nombre de la plantilla	
Estado	Nchar(1)	Estado de la plantilla	

Elaborador por: Autor

Tabla 10: Solicitud

Nombre de Tabla	Solicitud		
Descripción	Se encuentra las solicitudes enviadas por los usuarios.		
Nombre	**Tipo**	**Descripción**	**PK**
Id	Int	Numero de	

		solicitud	
Programa	Nvarchar(500)	Nombre / ruta del programa	
Solicitante	Nvarvhar(50)	Usuario Solicitante	
Motivo	Nvarchar(600)	Motivo de la solicitud	
Internalname	Nvarchar(100)	Nombre del programa	
CompanyName	Nvarchar(100)	Nombre de la compañía que desarrollo el programa	
LegalCopyright	Nvarchar(150)	copyright del programa	
Estado	Nchar(1)	Estado de la solicitud	

Elaborador por: Autor

Tabla 11: Tipo de Dato

Nombre de Tabla	Tipodato		
Descripción	Se encuentra los nombres de los diferentes tipos de datos.		
Nombre	**Tipo**	**Descripción**	**PK**
TipoDatoId	Int	Numero de plantilla	Si
Nombre	Nvarchar(200)	Nombre del tipo de dato	

Elaborador por: Autor

Tabla 12: Usuario

Nombre de Tabla	Usuario		
Descripción	Se encuentran los usuarios del sistema		
Nombre	**Tipo**	**Descripción**	**PK**
UsuarioId	Int	Numero de usuario	Si
Nombre	Nvarchar(200)	Nombre de la plantilla	
Estado	Nchar(1)	Estado de la plantilla	

Elaborador por: Autor

Tabla 13: Usuario Plantilla

Nombre de Tabla	UsuarioPlantilla		
Descripción	Se encuentran las plantillas asignadas a los usuarios.		
Nombre	**Tipo**	**Descripción**	**PK**
UsuarioId	Int	Numero de usuario	Si
PlantillaId	Int	Numero de plantilla	Si

Elaborador por: Autor

4.10 Componentes del Sistema y Paquetes de la Base de Datos

- **DMTEM (Data Manager)**
    - Configuración

      Clase que guarda y lee el archivo de configuración donde se guarda la información necesaria para conexión a base de datos SQL Server.

    - DataAccess

Lee de la base de datos la lista de programas a las que tiene acceso el usuario, enviado por parámetros.

- DataTransport

  Asigna el contenido de un datatable a una entidad, si el datatable tiene varios registros, devolverá el contenido del último registro.

  Transforma el contenido de una colección GenericList en XMLDocument

- SqlDml

  Administrador de conexiones a la base de datos SQL Server y el manejo de archivos

- **Entidades**
  - Usuario

    Guarda la información de los usuarios del sistema.

  - DatosPlantilla

    Guarda la información acerca los permisos que tiene el usuario para ejecutar programas.

  - Plantilla

    Cabecera de datos de plantilla.

  - UsuarioPlantilla

    Guarda la relación del usuario con la plantilla.

  - Solicitud

    Guarda datos sobre las peticiones al supervisor del sistema para ejecutar un programa.

  - TipoDato

    Guarda información acerca del tipo de dato

- **MONITOR TEM**
  - **Formularios**

- FrmAcerca

  Información del Sistema.

- FrmClave

  Ingreso y validación de clave Administrador.

- FrmFondo

  Bloqueo de pantalla.

- Frminicio

  Login de usuario.

- FrmMensaje

  Alerta de mensaje de ejecución de programa y envió de solicitud de ejecución.

- FrmMonitor

  Opciones del monitor.

- FrmPreferencias

  Configuración de usuario y conexión a la base de datos.

- FrmSolicitud

  Solicitud de permisos de ejecución de programas.

- Program

  Inicio de ejecución del monitor.

- **TEM**
  - **Formularios**
    - frmAprobacion

      Aprobaciones de solicitudes de programas enviados desde el cliente.

    - frmEditPlantilla

      Edición de accesos por plantillas de usuarios.

- frmEditUsuario

  Creación, modificación de usuarios.

- frmPlantilla

  Creación y edición de plantillas de accesos de programas.

- frmSolicitudUsuario

  Listado de solicitudes de ejecución de programas de usuarios.

- frmUsuario

  Listado de usuarios.

- frmUsuarioPlantilla

  Lista de usuarios plantillas.

- **Procedimiento Almacenados**

  - ApruebaSolicitud

    Procedimiento donde se aprueba o niega las solicitudes enviadas al administrador desde el cliente.

  - GuardaPlantillasDatos

    Procedimiento donde se guardan las plantillas que dan accesos a los programas.

  - GuardaSolicitud

    Procedimiento donde se guardan las solicitudes de ejecución de programas realizadas por el cliente

  - GuardaUsuario

    Procedimiento donde se guardan los usuarios del sistema.

  - LogonUsuario.

    Procedimiento que valida el usuario y contraseña.

  - ObtenerListaDatos

Procedimiento que retorna la lista de los programas permitidos del usuario.

4.11 Proceso de la Aplicación

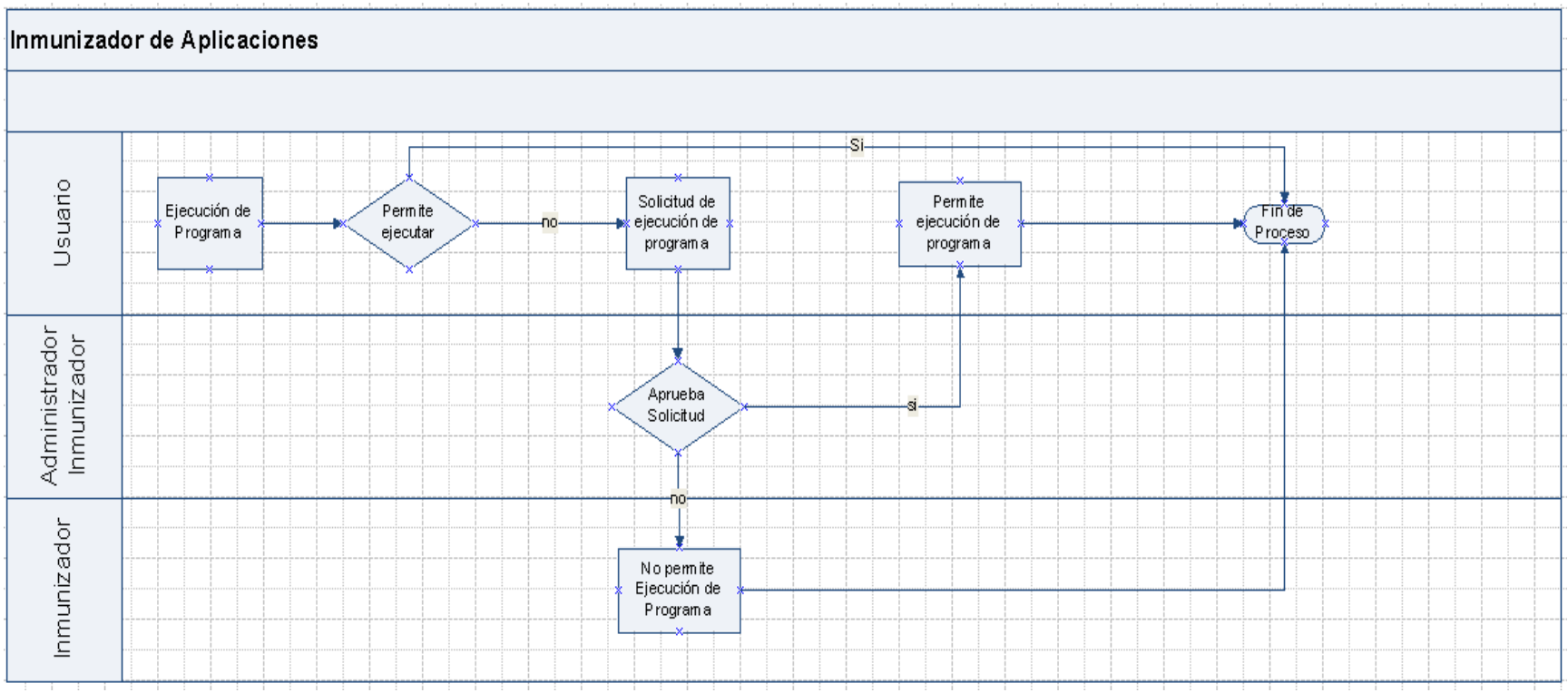


Figura 11: Proceso de la Aplicación

Elaborador por: Autor

El proceso de la aplicación consta de los siguientes pasos:

- Los usuarios deberán logonearse a la aplicación para poder utilizar la computadora con un nombre de usuario y clave previamente registrados en el sistema.
- El Usuario ejecutará alguna aplicación por ejemplo WORD, si el programa está cargado en la lista blanca se ejecutará sin problemas y termina el proceso, caso contrario el usuario enviara una solicitud al administrador para que éste la apruebe o niegue.
- Si la solicitud es aprobada por el administrador, el usuario volverá a ejecutar la aplicación y ésta ya podrá ejecutarse sin ningún bloqueo, caso contrario, si es negada, termina el proceso y el usuario no podrá ejecutar la aplicación.

4.12 Arquitectura de la Aplicación

La aplicación consta de 3 capas bien identificadas que son:

- Capa de Presentación: Es la capa donde interactúan los usuarios finales y el administrador del programa.
- Capa de Negocio: Esta capa comprende la parte lógica de la aplicación en donde se ejecuta la mayor parte de los comandos, consultas, etc.
- Capa de Datos: Es aquí donde el programa almacena la información, propiamente la Base de Datos en este caso SQL SERVER.

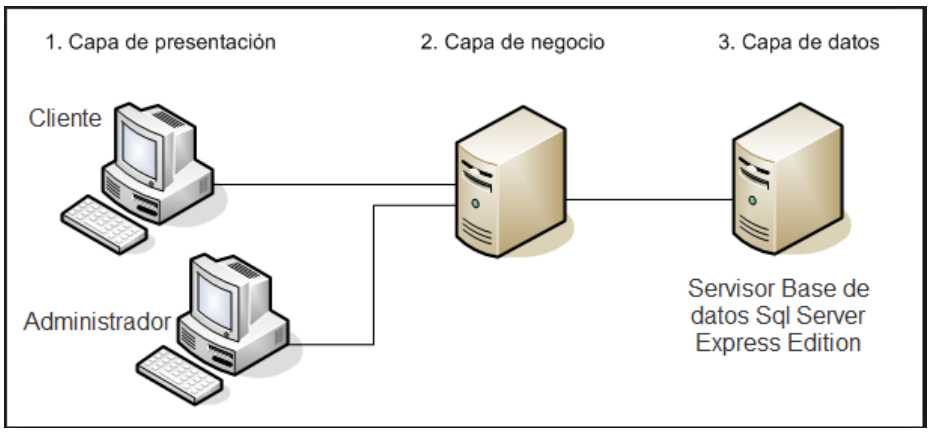


Figura 12: Arquitectura de la Aplicación

Elaborado por: Autor

4.13 Interfaces del Sistema

4.13.1 Modulo Consola Administrativa

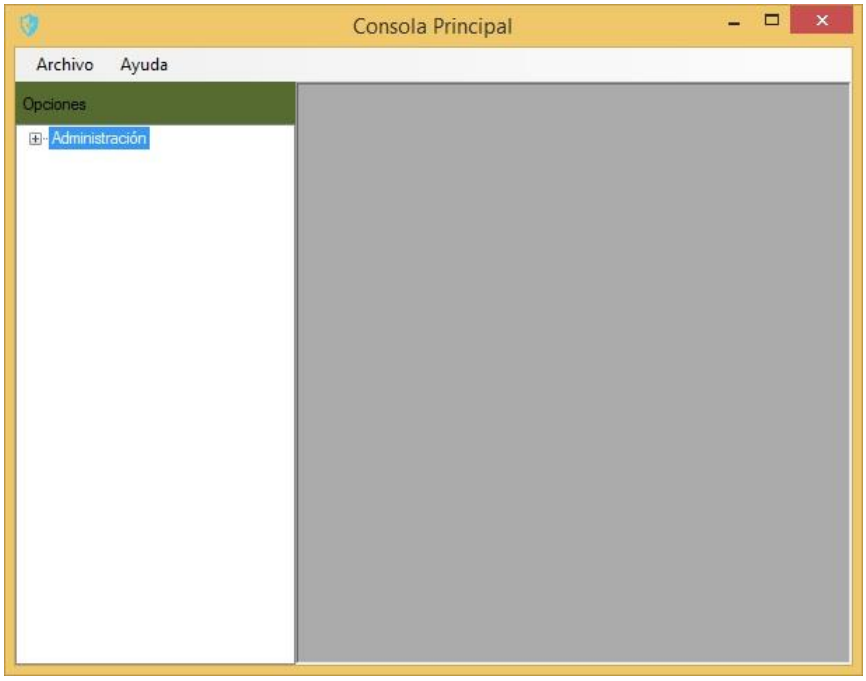


Figura 13: Consola Principal

Elaborado por: Autor

La pantalla que se muestra en la figura 13 es la consola principal, se ingresa mediante una clave predefinida de administrador, maneja todas las configuraciones del programa las cuales son:

Creación de usuarios: Aquí se crean los diversos usuarios que utilizaran el sistema, son los empleados que trabajan en la empresa RESNORTE SA, donde se le creara un nombre de usuario y clave con su respectivo estado activo o inactivo, como lo muestra la figura 14:

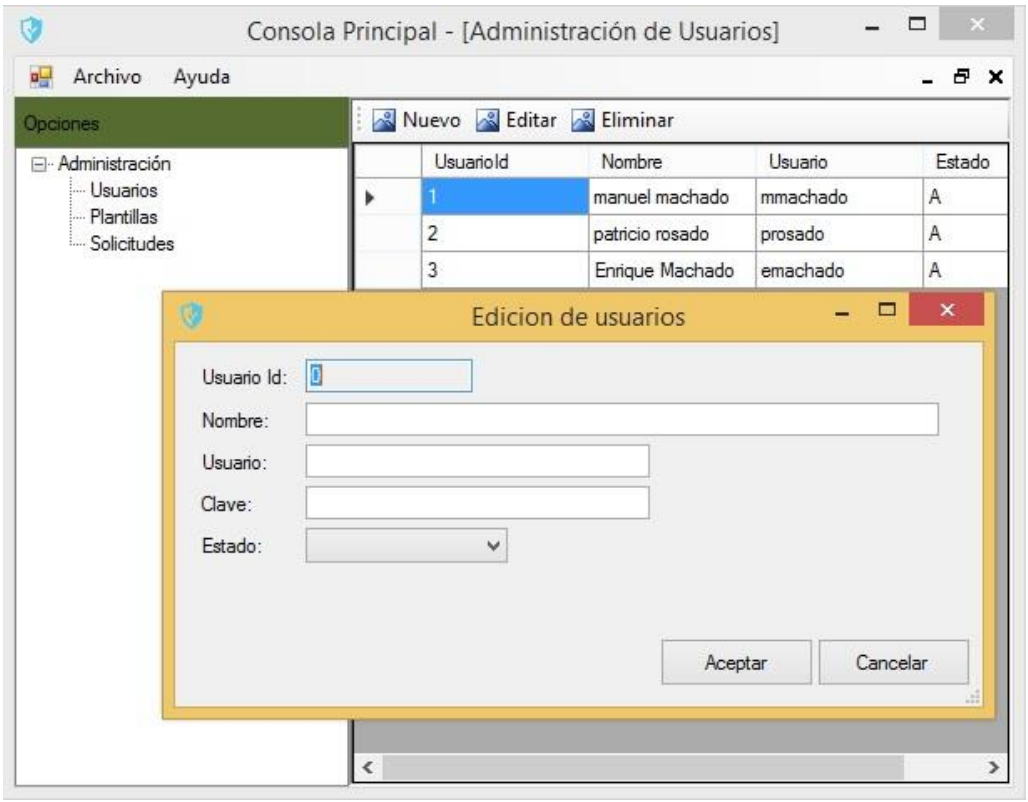


Figura 14: Consola Principal Creación de Usuarios

Elaborado por: Autor

Creación de Plantillas y Asignación de Usuarios: En esta pantalla se crea la "lista blanca" de programas a los que puede acceder cada usuario. Una vez creada la lista de programas permitidos, se asigna el o los usuarios que van a utilizar dicha plantilla. Cabe mencionar que pueden crearse N plantillas dependiendo de los perfiles que deseemos crear para determinados usuarios, como muestra la figura 15 y 16 a continuación.

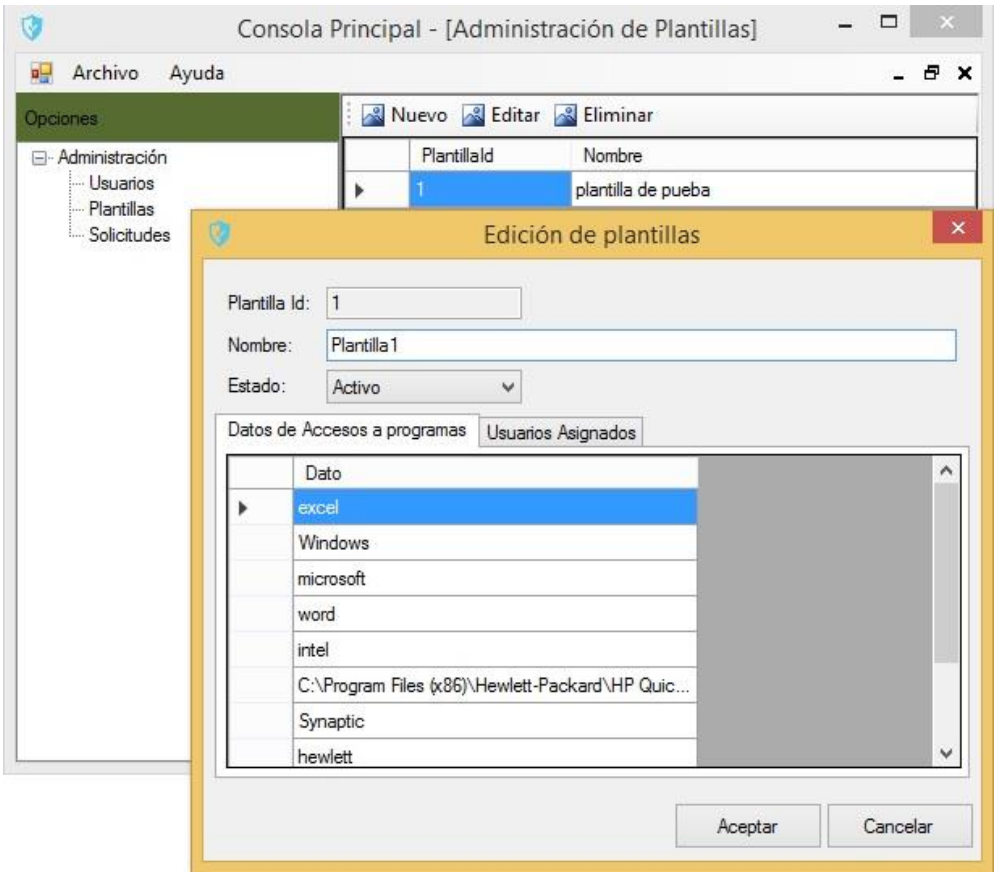


Figura 15: Consola Principal Creación y Asignación de Programas

Elaborado por: Autor

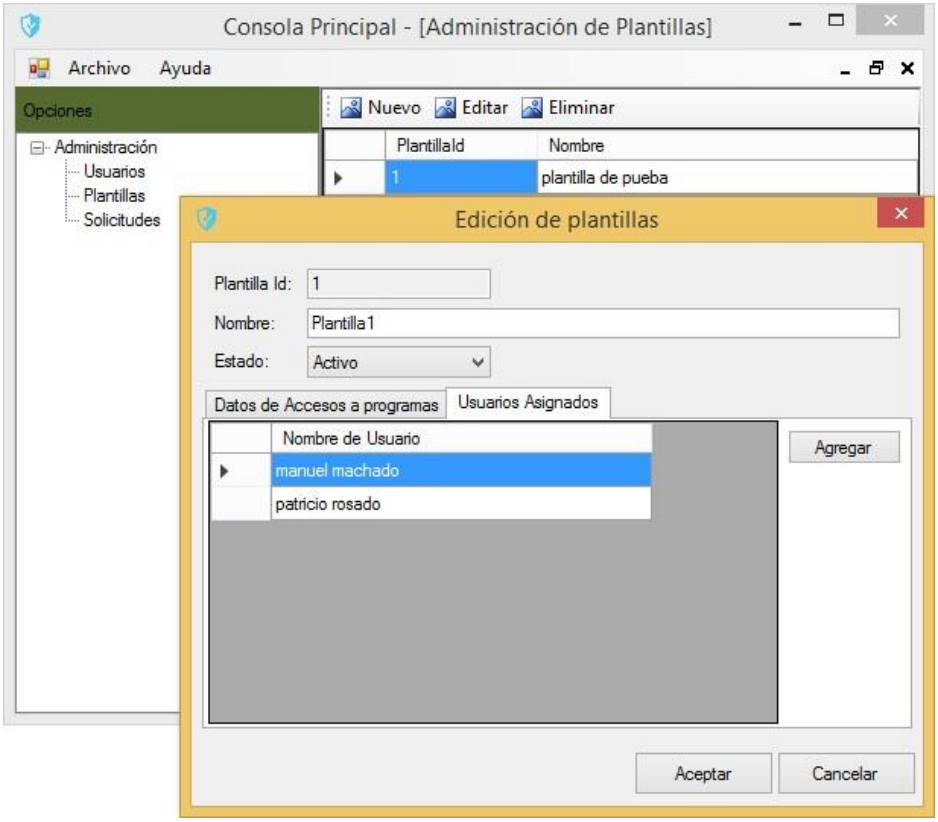


Figura 16: Consola Principal Asignación de Plantillas a Usuarios

Elaborado por: Autor

Aprobación de Solicitud de permisos: Muestra las distintas peticiones de los diferentes usuarios para acceder a determinado programa o aplicación. Desde aquí el administrador podrá aprobar o negar dichas peticiones según sea el caso. Como lo muestra la figura 17.

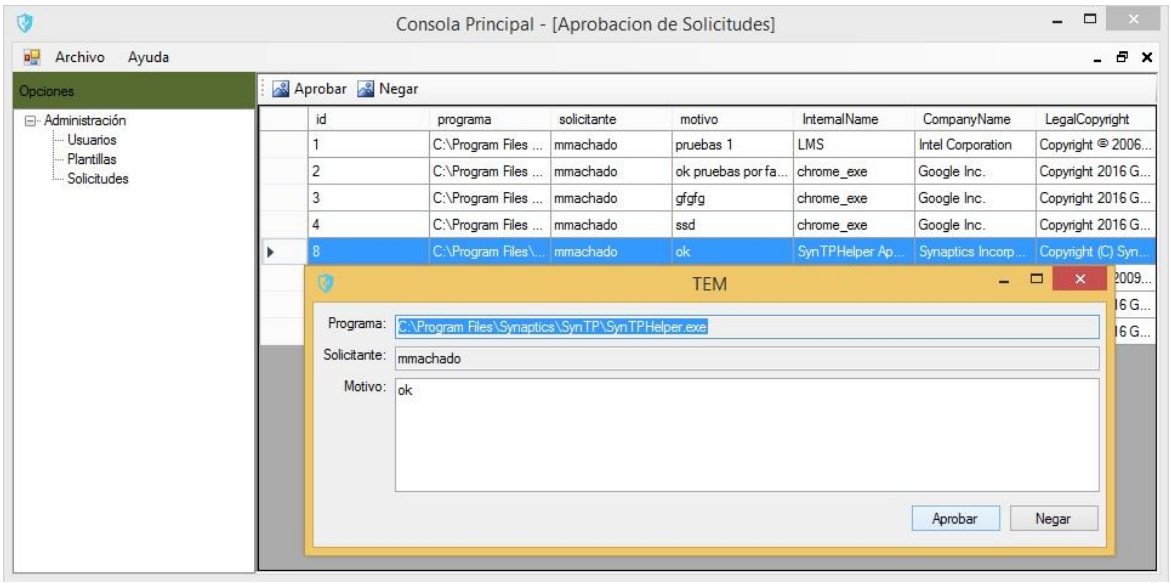


Figura 17: Consola Principal Aprobación de Solicitudes

Elaborado por: Autor

Orígenes de Datos SQL SERVER: En esta opción podemos definir los parámetros de conexión a la base de datos SQL SERVER como son:

1) Nombre o Identificador: Nombre para identificar la conexión.
2) Host o Instancia: Nombre de la instancia de SQL SERVER
3) Data Base o nombre de la base de datos: Nombre de la base de datos.
4) Tipo de Autenticación: Modo de autenticación con la base de datos, en este caso con autenticación de Windows.
5) Usuario: Usuario por defecto sa
6) Clave: Clave que se pone al momento de instalar el SQL SERVER.

En la figura 18 se observa el modelo de la pantalla de opciones.

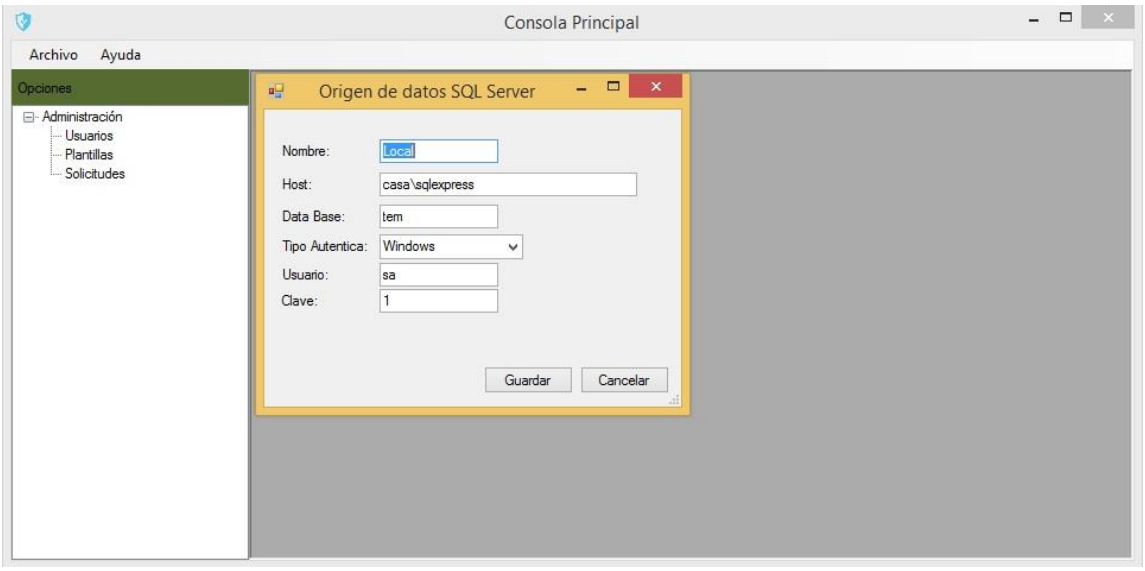


Figura 18: Consola Principal Parámetros de Conexión BD

Elaborado por: Autor

4.13.2 Monitor de Aplicaciones

Figura 19: Monitor de Aplicaciones

Elaborado por: Autor

Este es el programa de monitoreo de ejecución de programas, para poder utilizar la computadora el usuario deberá ingresar su usuario y clave anteriormente asignados desde la consola administrativa, una vez logoneado,

saldrá un mensaje de login exitoso y podrá acceder al computador, como lo muestra la figura 20 a continuación.

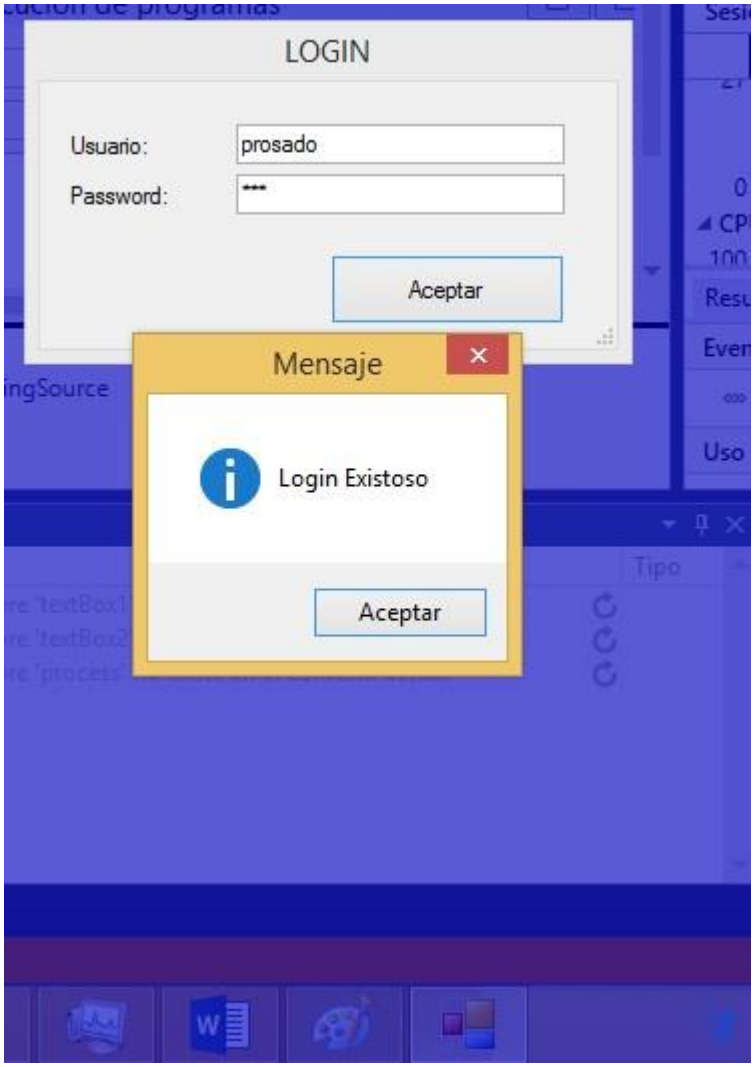


Figura 20: Login de la Aplicación

Elaborado por: Autor

El Programa de Monitor de Aplicaciones consta del siguiente Menú:

Solicitar Acceso de Programas: Esta opción sirve para que el usuario pueda pedir permiso de ejecución de X programa al administrador. El usuario deberá escoger la ruta del programa que quiere ejecutar y escribir un motivo por el cual quiere acceder a dicho programa, como lo muestra la figura 21.

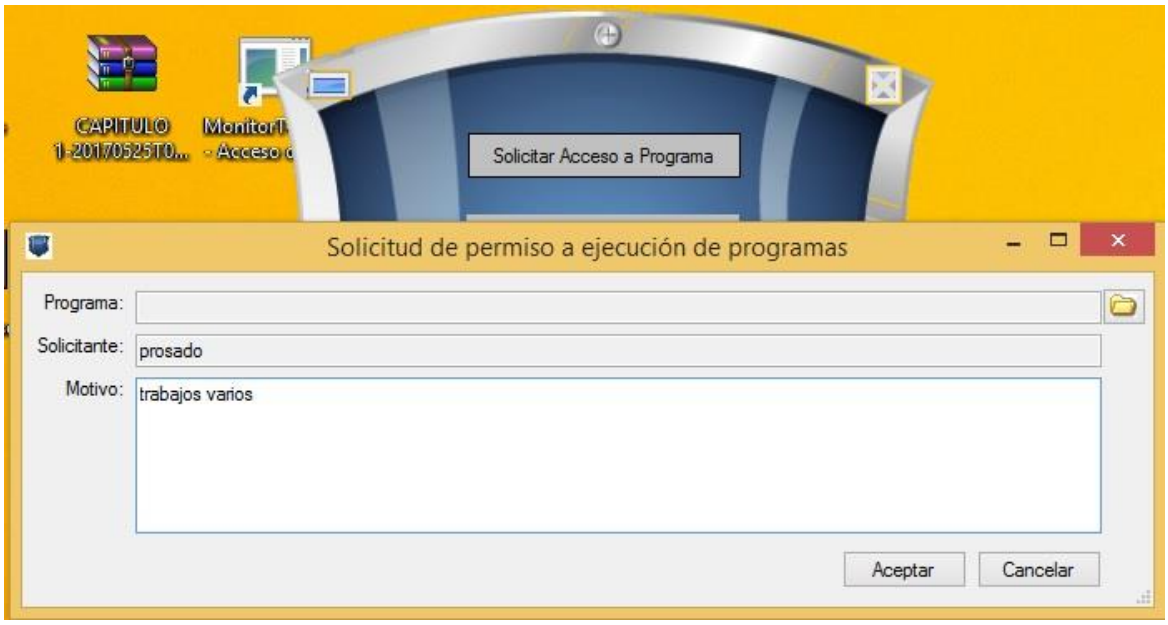


Figura 21: Solicitud de Permisos

Elaborado por: Autor

Actualizar Programas Cargados: Simplemente muestra la lista de programas que están cargados al momento de iniciar el sistema, es decir, muestra la "lista blanca" del usuario autenticado incluido los que son propios del sistema operativo. Ver figura 22.

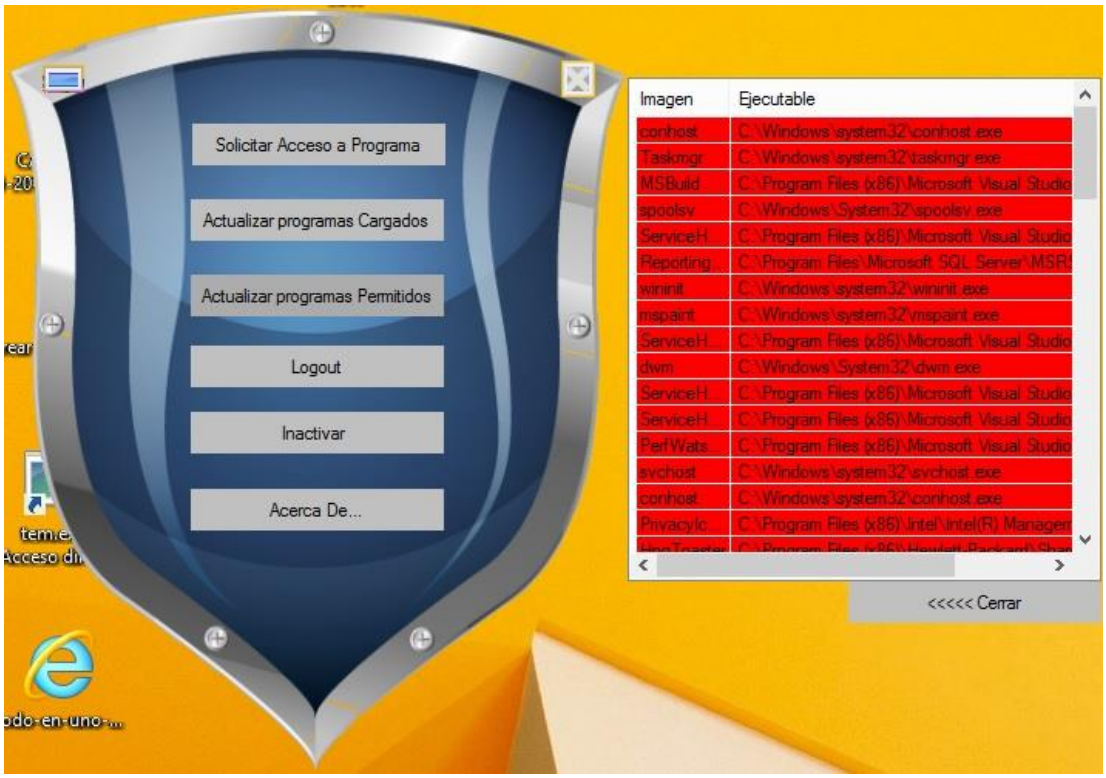


Figura 22: Programas Permitidos

Elaborado por: Autor

Actualizar Programas Permitidos: Con esta opción lo que hacemos es un refresh de los programas que tenemos acceso, en caso de haber solicitado un acceso nuevo, pulsando este botón actualizaremos el estado de la "lista blanca", en el caso que el administrador haya otorgado un nuevo acceso. Ver figura 23.

Figura 23: Actualizar Programas Permitidos

Elaborado por: Autor

Logout: Aquí únicamente el usuario al terminar su jornada o si tiene que salir, procederá a desconectarse del sistema, para que si algún otro usuario quiere utilizar esa computadora, pueda hacerlo sin ningún problema con su propio usuario, clave y accesos asignados.

Inactivar: Esta opción es solo de uso únicamente para el administrador del sistema, para alguna emergencia o si necesita hacer algo el usuario que aún no se ha aceptado o se hará solo por única vez. El administrador podrá desactivar la protección mediante su clave. Ver figura 24.

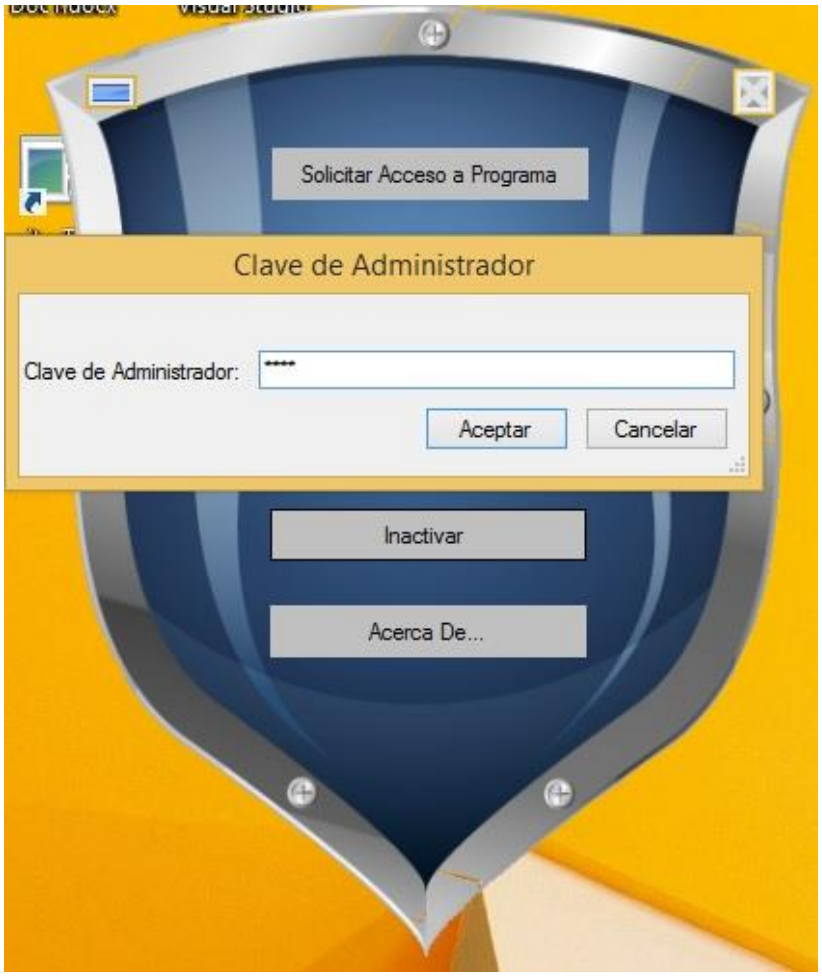


Figura 24: Inactivar Monitor

Elaborado por: Autor

CAPÍTULO V

PRUEBAS DEL SISTEMA

5.1 Pruebas de Funcionamiento

Se realizó una serie de pruebas al Sistema Inmunizador de Aplicaciones que permitieron determinar los requerimientos definidos inicialmente.

Se realizaron las siguientes pruebas tomando los 2 módulos elaborados:

- Prueba crear usuario
- Prueba crear plantilla
- Prueba de login
- Prueba de envío de solicitud de permiso de ejecución de programa
- Prueba de aprobación de solicitud

Tabla 14: Prueba 1: Crear Usuario

Prueba 1	Crear Usuario
Descripción	La consola administrativa permite crear nuevos usuarios
Usuario	Administrador
Pasos de Ejecución	1. El administrador ingresa a la aplicación. 2. El administrador selecciona la opción ADMINISTRACIÓN. 3. El administrador da doble clic sobre la opción USUARIOS. 4. El administrador da clic en NUEVO. 5. El administrador llena los datos de la pantalla. 6. El administrador da clic sobre

	el botón GUARDAR.
Resultado Esperado	La aplicación guarda correctamente los datos del usuario.
Evaluación	Prueba satisfactoria.

Elaborado por: Autor

Tabla 15: Prueba 2: Crear Plantilla

Prueba 2	Crear Plantilla
Descripción	La consola administrativa permite crear listas de acceso a programas permitidos para los usuarios.
Usuario	Administrador
Pasos de Ejecución	1. El administrador ingresa a la aplicación. 2. El administrador selecciona la opción ADMINISTRACIÓN. 3. El administrador da doble clic sobre la opción PLANTILLAS. 4. El administrador da clic en NUEVO. 5. El administrador llena los datos de la pantalla. 6. El administrador da clic sobre el botón GUARDAR.
Resultado Esperado	La aplicación guarda correctamente los datos del usuario.
Evaluación	Prueba satisfactoria.

Elaborado por: Autor

Tabla 16: Prueba 3: Login

Prueba 3	Login
Descripción	La consola de monitoreo pide usuario y contraseña para poder utilizar el pc y tener acceso a los programas permitidos para cada usuario.
Usuario	Usuario
Pasos de Ejecución	1. El usuario ingresa su nombre de usuario y contraseña. 2. El usuario da clic en el botón ACEPTAR. 3. El programa envía mensaje de confirmación de login.
Resultado Esperado	La aplicación carga los programas según el usuario autenticado.
Evaluación	Prueba satisfactoria.

Elaborado por: Autor

Tabla 17: Prueba 4: Envío de Solicitud

Prueba 4	Envío de solicitud de permiso de ejecución de programa.
Descripción	El usuario mediante la consola de monitoreo puede enviar solicitud de ejecución de programas al administrador del sistema.
Usuario	Usuario
Pasos de Ejecución	1. El usuario abre la aplicación. 2. El usuario da clic sobre el botón SOLICITAR ACCESO A PROGRAMA.

	3. El usuario deberá buscar la ruta del programa que desea ejecutar. 4. El usuario deberá llenar el motivo por el cual solicita la ejecución del programa. 5. El usuario deberá dar clic sobre el botón ACEPTAR.
Resultado Esperado	La aplicación guarda los datos correctamente.
Evaluación	Prueba satisfactoria.

Elaborado por: Autor

Tabla 18: Prueba 5: Aprobación de Solicitud

Prueba 5	Aprobación de solicitud de permiso de ejecución de programa.
Descripción	La consola de administración muestra las solicitudes de permiso de ejecución de programas enviada por los usuarios.
Usuario	Administrador
Pasos de Ejecución	6. El administrador ingresa a la aplicación. 7. El administrador selecciona la opción ADMINISTRACIÓN. 8. El administrador da doble clic sobre la opción SOLICITUDES. 9. El administrador da doble clic en la solicitud que desea aprobar. 10. La consola abre una pantalla donde muestra los datos de la

	solicitud. 11. El administrador da clic sobre el botón APROBAR O NEGAR.
Resultado Esperado	La aplicación guarda los datos correctamente.
Evaluación	Prueba satisfactoria.

Elaborado por: Autor

CAPÍTULO VI

RESULTADOS OBTENIDOS

La empresa Resnorte SA ha alcanzado con el Sistema Inmunizador de Aplicaciones y Control Organizacional, optimizar el tiempo de sus empleados, reduciendo el tiempo de respuesta en la entrega de información.

El resultado obtenido fue posible porque el sistema restringió el uso y abuso de aplicaciones como juegos, editores de imágenes, programas de reproducción de videos y música, etc, que antes de aplicar esta solución podían ejecutar sin ninguna restricción.

Además los equipos informáticos están menos cargados de procesos y hay más memoria liberada, algunos programas consumían recursos de memoria haciendo estas unidades lentas y en ocasiones se debía reiniciar los sistemas porque los computadores se quedaban inhibidos.

CAPÍTULO VII

CONCLUSIONES Y RECOMENDACIONES

Una vez concluido el presente proyecto del Sistema Inmunizador de aplicaciones y Control Organizacional, se ha llegado a las siguientes conclusiones las mismas que servirán de base para plantear las recomendaciones que a continuación se detallan:

7.1. Conclusiones

Una vez analizado los requerimientos de la empresa Resnorte SA, se comenzó a trabajar en la herramienta que controla la ejecución de programas mediante listas blancas, para lo cual se realizó el diseño de base de datos y el sistema propiamente dicho.

Para el sistema se diseñó una interfaz amigable que permite al usuario interactuar de manera rápida cuando necesite solicitar el acceso a algún programa específico, de la misma manera para el administrador que podrá administrar de manera ágil las diferentes opciones que tiene el sistema.

Los usuarios se familiarizaron rápidamente con la aplicación, pudiendo manejarla y hacer las solicitudes de permisos rápidamente y el administrador aprobarlas. Se logró mejorar el tiempo de entrega de información y tener un control de las tareas del personal.

7.2. Recomendaciones

a) Tener el software instalado siempre y en todo momento para no volver a incurrir en errores del pasado.

b) Es importante que los usuarios finales (empleados) se familiaricen con la herramienta y se acostumbren a ella para así mejorar de una manera eficaz el tiempo de trabajo y cuidar mejor de los recursos informáticos de la empresa.

c) Realizar un mantenimiento preventivo a cada ordenador, y monitorear el trabajo del sistema, para cerciorarse del buen manejo de la herramienta.

REFERENCIAS BIBLIOGRAFICAS

- Pérez Julián. (2008). Definición de software. Recuperado de http://definicion.de/software/
- Ávila Harold. (2016)¿Qué es Hardware? Recuperado de http://computaciona1.com/hardware/que-es-hardware/
- GCF Community Foundation International. (1998). Que es un Ordenador – definición de ordenador. Recuperado de http://www.masadelante.com/faqs/ordenador
- GCF Community Foundation International. (1998). Que es Virus Informático. Recuperado de https://www.gcfaprendelibre.org/tecnologia/curso/virus_informaticos_y_antivirus/los_virus_informaticos/1.do
- McConnell, Steve (1996). Rapid Development: Taming Wild Software Schedules, 1st ed., Redmond, WA: Microsoft Press. ISBN 1-55615-900-5.
- Patricia Dip. (2008). Dato e Información. Recuperado de http://latecnologiavirtual.blogspot.com/2008/04/dato-e-informacin.html
- Wiegers, Karl E. (2003). Software Requirements 2: Practical techniques for gathering and managing requirements throughout the product development cycle, 2nd ed., Redmond: Microsoft Press. ISBN 0-7356-1879-8.
- Landgraf, Katja. (2011). Requirement Management in Product Development, Symposion Publishing ISBN 978-3-939707-84-4
- Andrew Stellman y Jennifer Greene (2005). Applied Software Project Management. Cambridge, MA: O'Reilly Media. ISBN 0-596-00948-8.
- IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications –Description

- Julian Pérez. (2015). Definición de Requerimiento. Recuperado de http://definicion.de/requerimiento/#ixzz4HbYPJ6Rc)
- Marcelo Rivero. (2013). Que son los Malwares. Recuperado de https://www.infospyware.com/articulos/que-son-los-malwares/
- Margaret Rouse. (2015). SQL SERVER. Recuperado de http://searchdatacenter.techtarget.com/es/definicion/SQL-Server
- Microsoft. (2016). Introducción a Visual Studio. Recuperado de https://msdn.microsoft.com/es-es/library/fx6bk1f4%28v=vs.100%29.aspx?f=255&MSPPError=-2147217396
- José Machado. (2012). Levantamiento de Información. Recuperado de http://miguedt.blogspot.com/2012/12/levantamiento-de-informacion.html
- Pablo Tumero. (2016). Introducción a los Sistemas Operativos. Recuperado de http://www.monografias.com/trabajos107/introduccion-a-sistemas-operativos/introduccion-a-sistemas-operativos.shtml

ANEXOS

MANUAL DE INSTALACION DEL SISTEMA

Para poder utilizar el sistema debemos de realizar las siguientes acciones:

Servidor

1. Tener instalado SQL SERVER EXPRESS
2. Instalar NET FRAMEWORK 4.5 o posterior
3. Crear una base de Datos llamada TEAM
4. Dejar el ejecutable de la consola en el escritorio TEM.exe

Usuarios

1. Instalar NETFRAMEWORK 4.5 o posterior.
2. Copiar la carpeta team del proyecto en la unidad C: de cada PC.
3. Verificar que exista la siguiente clave de registro:

   HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Policies\System\DisableTaskMgr

   Si no existe la clave de registro SYSTEM, crearla como muestra la figura 25 a continuación:

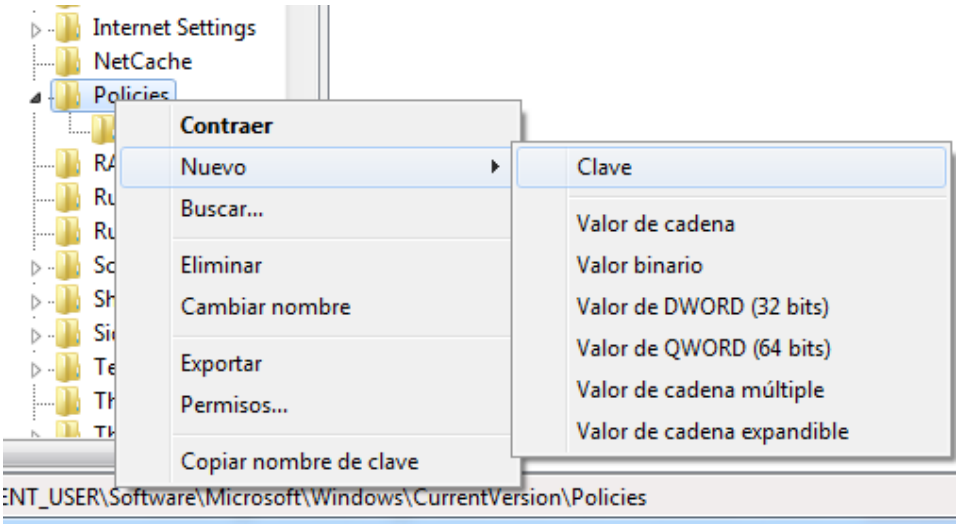


Figura 25 Crear clave de registro

Elaborado por: Autor

4. Para ejecutar el programa MONITORTEM.exe automáticamente al iniciar WINDOWS, se puede hacer de 2 maneras:

- Creando clave de registro
- Agregando el acceso directo de la aplicación MONITORTEAM.EXE dentro de la carpeta inicio del sistema operativo.

La ruta en la que se va a agregar la clave es la siguiente:

HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run

Damos clic derecho sobre la ventana, elegimos Nuevo y luego Valor de Cadena. Como lo muestra la figura 26.

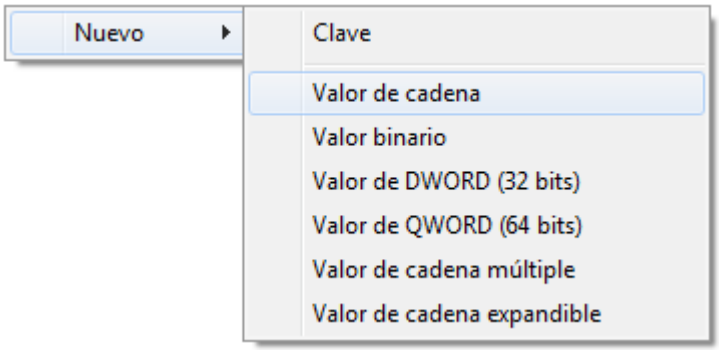


Figura 26 Agregar Cadena a Registro

Elaborado por: Autor

En este caso el programa se llama MonitorTeam, con este mismo nombre, se crea la entrada.

Luego se parametriza la cadena de entrada poniendo la ruta donde está localizado el programa en información de valor. Tal como lo muestra la figura 27.

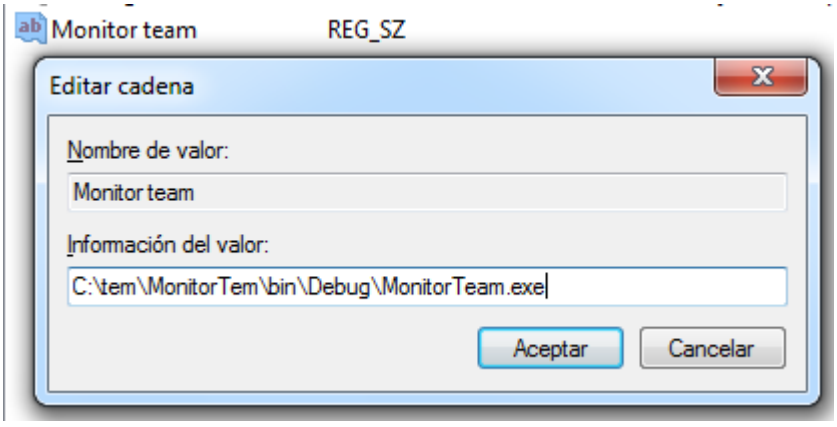


Figura 27 Parametrizar Valor de Cadena de Registro

Elaborado por: Autor

Ahora si se da clic en aceptar cerramos el registro y reiniciamos la PC.

Al iniciar Windows arrancará el sistema y pedirá usuario y clave para poder usar el PC con los permisos que tiene registrados en la consola administrativa, como lo muestra la figura 28.

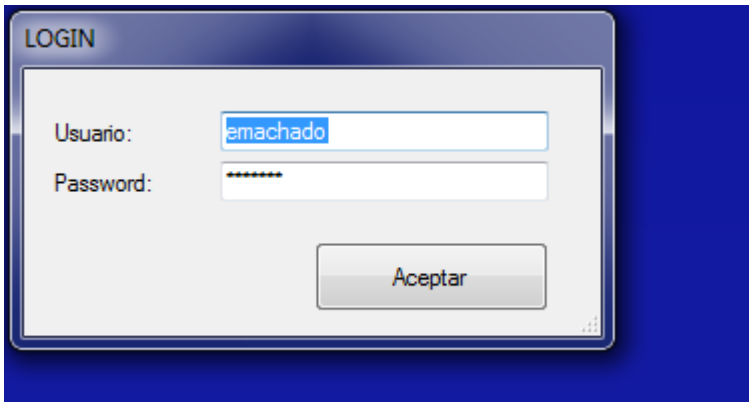


Figura 28 Inicio del Sistema

Elaborado por: Autor

PAQUETES DE LA BASE DE DATOS

Procedimiento ApruebaSolicitud

Guarda los datos de la solicitud de ejecución de programas en la plantilla correspondiente a cada usuario.

```
CREATE procedure [dbo].[ApruebaSolicitud] (@ids as int,@sestado as nvarchar) as
begin
	declare
		@Id int
		,@programa nvarchar(500)
		,@solicitante nvarchar(50)
		,@motivo nvarchar(600)
		,@InternalName nvarchar(100)
		,@CompanyName nvarchar(100)
		,@LegalCopyright nvarchar(150)
		,@Error varchar(200)
		,@estado nvarchar(1);

	BEGIN TRY
	declare dato cursor
	for SELECT [id]
,[programa]
,[solicitante]
,[motivo]
,[InternalName]
,[CompanyName]
,[LegalCopyright]
	,[estado]
FROM [tem].[dbo].[Solicitud] where id=@ids;
	open dato;
	fetch  dato  into
		@id
		,@programa
		,@solicitante
```

```
        ,@motivo
        ,@InternalName
        ,@CompanyName
        ,@LegalCopyright
        ,@estado ;
IF @sestado='A'
begin
        insert into DatosPlantilla
                (
                PlantillaId,
                dato,
                TipoDatoId
                ,InternalName
                ,CompanyName
                ,LegalCopyright,solicitante)
                values
                (
                1,
                @programa,
                1
                ,@InternalName
                ,@CompanyName
                ,@LegalCopyright
                ,@solicitante);
                update Solicitud set estado=@sestado where id=@ids;
end
else
        begin
                update Solicitud set
                estado=@sestado
                where id=@ids;
        end
close dato
deallocate dato
```

```
    END TRY
    BEGIN CATCH
        close dato
        deallocate dato
        SELECT @Error=
        CONVERT(VARCHAR,ERROR_LINE())+ERROR_MESSAGE()
        RAISERROR (@error,11,1)
    END CATCH
end
```

Procedimiento GuardaPlantillaDatos

Guarda los datos referentes a los premisos de ejecución de cada usuario.

```
CREATE procedure [dbo].[GuardaPlantillaDatos] (@DataCab as Xml,@DataDet as
Xml,@DataUsr as Xml) as
begin
    declare
        @PlantillaId int
        ,@Nombre nvarchar(200)
        ,@Estado nchar(2)
        ,@Datos nvarchar(255)
        ,@UsuarioId int
        ,@Conteo int
        ,@Error varchar(200)
        ,@InternalNames nvarchar(100)
        ,@CompanyNames nvarchar(100)
        ,@LegalCopyrights nvarchar(150)
        ,@solicitantes nvarchar(50);
    BEGIN TRY
    declare dato cursor
        for select
            CAST(CAST(y.item.query('data(PLANTILLAID)') as
            varchar(40)) as int) as PlantillaId,
            CAST(y.item.query('data(NOMBRE)') as nvarchar(200)) as
            Nombre,
```

```
                CAST(y.item.query('data(ESTADO)') as nchar(1)) as Estado
                from @DataCab .nodes('root/PLANTILLA')
                y(item);
open dato;
fetch  dato  into
        @PlantillaId
        ,@Nombre
        ,@Estado;
if @Nombre='' SELECT @Error='Debe poner un nombre a la plantilla';
 RAISERROR ('Debe poner un nombre a la plantilla ',11,1);
select @Conteo= COUNT(*) from Plantilla where PlantillaId=@PlantillaId;
if @Conteo=0 or @PlantillaId=0
        begin
                Select @PlantillaId=isnull(MAX(PlantillaId),0)+1 from
                Plantilla;
                insert into Plantilla
                                        (
                                        PlantillaId,
                                        Nombre,
                                        Estado)
                                        values
                                        (
                                        @PlantillaId,
                                        @Nombre,
                                        @Estado);
        end
else
        begin
                update Plantilla set
                        Nombre=@Nombre,
                        Estado=@Estado
                        where PlantillaId=@PlantillaId;
        end
close dato
```

```
deallocate dato
declare dato1 cursor
for select
        CAST(CAST(y.item.query('data(USUARIOID)') as varchar(40)) as
        int) as UsuarioId from
        @DataUsr.nodes('root/USUARIOPLANTILLA') y(item);
open dato1;
fetch  dato1  into
         @UsuarioId;
delete from UsuarioPlantilla where PlantillaId=@PlantillaId;
WHILE @@FETCH_STATUS = 0
BEGIN
        if @UsuarioId>0
        begin
                insert into UsuarioPlantilla values (@UsuarioId,@PlantillaId);
                FETCH NEXT FROM dato1  into @UsuarioId;
        end;
END;
declare dato2 cursor
for select
                CAST(y.item.query('data(DATO)') as varchar(200)) as Datos,
                CAST(y.item.query('data(INTERNALNAME)') as
                nvarchar(100)) as InternalNames,
                CAST(y.item.query('data(COMPANYNAME)') as
                nvarchar(100)) as CompanyNames,
                CAST(y.item.query('data(LEGALCOPYRIGHT)') as
                nvarchar(150)) as LegalCopyrights,
                CAST(y.item.query('data(SOLICITANTE)') as nvarchar(50))
                as solicitantes from
                @DataDet.nodes('root/DATOSPLANTILLA') y(item);
open dato2;
fetch  dato2  into
         @Datos,
         @InternalNames,
```

```
        @CompanyNames,
        @LegalCopyrights,
        @solicitantes;
    delete from DatosPlantilla where PlantillaId=@PlantillaId;
    WHILE @@FETCH_STATUS = 0
    BEGIN
        insert into DatosPlantilla values
(@PlantillaId,@Datos,1,@InternalNames,@CompanyNames,@LegalCopyrig
hts,@solicitantes);
    FETCH NEXT FROM dato2 into @Datos,
@InternalNames,@CompanyNames,@LegalCopyrights,@solicitantes;
    END;
    END TRY
    BEGIN CATCH
        close dato
        deallocate dato
    End
```

Procedimiento GuardaSolicitud

Guarda los datos del programa y usuario que se desea ejecutar.

```
CREATE procedure [dbo].[GuardaSolicitud] (@Data as Xml) as
begin
    declare
        @Id int
        ,@programa nvarchar(500)
        ,@solicitante nvarchar(100)
        ,@motivo nvarchar(600)
        ,@InternalName nvarchar(100)
        ,@CompanyName nvarchar(100)
        ,@LegalCopyright nvarchar(150)
        ,@Error varchar(200);

    BEGIN TRY
```

```
declare dato cursor for select CAST(CAST(y.item.query('data(ID)') as
nvarchar(40)) as int) as id,CAST(y.item.query('data(PROGRAMA)') as
nvarchar(500)) as programa,CAST(y.item.query('data(SOLICITANTE)') as
nvarchar(100)) as solicitante,CAST(y.item.query('data(MOTIVO)') as
varchar(600)) as motivo,CAST(y.item.query('data(INTERNALNAME)') as
nvarchar(100)) as
InternalName,CAST(y.item.query('data(COMPANYNAME)') as
nvarchar(100)) as
CompanyName,CAST(y.item.query('data(LEGALCOPYRIGHT)') as
nvarchar(150)) as LegalCopyright from @Data.nodes('root/SOLICITUD')
y(item);
open dato;
fetch  dato  into
        @id
        ,@programa
        ,@solicitante
        ,@motivo
        ,@InternalName
        ,@CompanyName
        ,@LegalCopyright ;
        begin
                Select @id=isnull(MAX(id),0)+1 from Solicitud ;
                insert into Solicitud
                (
                id,
                programa,
                solicitante,
                motivo
                ,InternalName
                ,CompanyName
                ,LegalCopyright)
                values
                (
                @id,
```

```
                @programa,
                @solicitante,
                @motivo
                ,@InternalName
                ,@CompanyName
                ,@LegalCopyright);
        end
    close dato
    deallocate dato
    END TRY
    BEGIN CATCH
        close dato
        deallocate dato
        sELECT                                                  @Error=
        CONVERT(VARCHAR,ERROR_LINE())+ERROR_MESSAGE()
        RAISERROR (@error,11,1)
    END CATCH
End
```

Procedimiento GuardaUsuario

Guarda los datos del usuario como son: id, nombre, login, clave, estado.

```
CREATE procedure [dbo].[GuardaUsuario] (@Data as Xml) as
begin
    declare
        @UsuarioId int
        ,@Nombre nvarchar(200)
        ,@Login nvarchar(50)
        ,@Clave nvarchar(200)
        ,@Estado nchar(2)
        ,@Conteo int
        ,@Error varchar(200) ;
    BEGIN TRY
    declare dato cursor
```

```
for select CAST(CAST(y.item.query('data(USUARIOID)') as varchar(40)) as int) as UsuarioId,CAST(y.item.query('data(NOMBRE)') as nvarchar(200)) as Nombre,CAST(y.item.query('data(LOGIN)') as nvarchar(50)) as Login,CAST(y.item.query('data(CLAVE)') as nvarchar(200)) as Clave,CAST(y.item.query('data(ESTADO)') as nchar(1)) as Estado from @Data.nodes('root/USUARIO') y(item);
open dato;
fetch  dato  into
        @UsuarioId
        ,@Nombre
        ,@Login
        ,@Clave
        ,@Estado;
        select @Conteo= COUNT(*) from Usuario where UsuarioId=@UsuarioId
        if @Conteo=0 or @UsuarioId=0
        begin
                Select @UsuarioId=isnull(MAX(UsuarioId),0)+1 from Usuario ;
                insert into Usuario
                (
                UsuarioId,
                Nombre,
                Login,
                Clave,
                Estado)
                values
                (
                @UsuarioId,
                @Nombre,
                @Login,
                @Clave,
                @Estado);
        end
```

```
            else
            begin
                  update Usuario set
                        Nombre=@Nombre,
                        Login=@Login,
                        Clave=@Clave,
                        Estado=@Estado
                        where UsuarioId=@UsuarioId;
            end
      close dato
      deallocate dato
      END TRY
      BEGIN CATCH
            close dato
            deallocate dato
            sELECT

      @Error=CONVERT(VARCHAR,ERROR_LINE())+ERROR_MESSAGE()
      RAISERROR (@error,11,1)
      END CATCH
End
```

Procedimiento LogonUsuario

Verifica que el usario exista y coincida con su clave para iniciar el sistema.

```
CREATE procedure [dbo].[LogonUsuario] (@Login as nvarchar(30),@Clave as nvarchar(20)) as
begin
declare
      @existe int,
      @Error varchar(200);
      BEGIN TRY
            declare dato cursor
            for SELECT COUNT(*) existe FROM Usuario where Login=@Login and Clave = @Clave
```

```
        open dato;
            fetch  dato  into @existe ;
            if @existe=0
            begin
            RAISERROR ('Usuario o Contrasena Incorrecta ',11,1);
            end;
        close dato
        deallocate dato
    END TRY
    BEGIN CATCH
        close dato
        deallocate dato
        SELECT                                                    @Error=
        CONVERT(VARCHAR,ERROR_LINE())+ERROR_MESSAGE()+
        @Clave+@Login
        RAISERROR (@error,11,1)
    END CATCH
end
```

Procedimiento ObtenerListaDatos

Obtiene la lista de permisos para cada usuario, guardados en la plantilla de datos.

```
CREATE procedure [dbo].[ObtenerListaDatos] (@UsuarioId as integer) as
begin
    declare
    @solicitante nvarchar(50);
    declare dato cursor for SELECT [login]FROM [tem].[dbo].Usuario where
    UsuarioId=@UsuarioId;
    open dato;
    fetch  dato  into
    @solicitante ;
    Select d.dato,isnull(d.InternalName,' ') as InternalName,
    isnull( d.CompanyName,' ') as CompanyName,
    isnull(d.LegalCopyright,' ') as LegalCopyright from
    plantilla p inner join datosplantilla d on
```

```
        p.plantillaid=d.plantillaid where p.estado='A'          and
        p.plantillaid in (select plantillaid from usuarioplantilla u where
        usuarioid=@UsuarioId)and
        (d.solicitante=@solicitante or d.solicitante is null  ) ;
end
```

Codigo Fuente Funciones Principales

Login de Usuario:

Valida el ingreso al programa.

```
private void Btnaceptar_Click(object sender, EventArgs e)
    {
    try
    {
       da.LogonUsuario (textBox1.Text.Trim(), textBox2.Text.Trim());
       MessageBox.Show(this, "Login Existoso", "Mensaje",
       MessageBoxButtons.OK, MessageBoxIcon.Information);
       sql.PasswordSistema = txtpass.Text.Trim();
       sql.UsuarioSistema = txtusr.Text.Trim(); DMTem.Configuracion.Save(sql);
       fondo.Close();
       Close();
    }
    catch (Exception Error)
    {
       MessageBox.Show(this, Error.Message, "Error", MessageBoxButtons.OK,
       MessageBoxIcon.Error);
    }
    }
```

Monitoreo de las aplicaciones:

```
private void timer1_Tick(object sender, EventArgs e)
    {
       timer1.Enabled = false;
```

```
   System.Diagnostics.Process[] processes =
   System.Diagnostics.Process.GetProcesses();
   foreach (System.Diagnostics.Process process in processes)
{
   int existe = 0;
   try
{
 //Esta parte del código solo es para confirmar si tiene o no acceso al programa
   existe = (from x in BaseDatosPermitidos.Lista where
   process.MainModule.FileName.ToUpper().Contains(x.Dato.ToUpper()) ||
   process.MainModule.FileName.ToUpper().Contains("MONITORTEM")
   select x).Count();
}
   catch (Exception ex)
{ existe = 0; }
if (existe == 0 )
{
try
{
//se obtiene información del programa para la solicitud y no deja que se ejecute
string archivo = process.MainModule.FileName;
string company = process.MainModule.FileVersionInfo.CompanyName;
string copyrigth = process.MainModule.FileVersionInfo.LegalCopyright;
string internaln =  process.MainModule.FileVersionInfo.InternalName;
process.Kill();
frmMensaje msg = new frmMensaje();
msg.SetProgramFile(archivo, company, copyrigth, internaln);
msg.ShowDialog();
}
catch (Exception ex)
{ existe = 0; }
} else
{
```

```
existe = (from x in BaseDatosPermitidos.Lista where
process.MainModule.FileName.ToUpper().Contains(x.Dato.ToUpper()) &&
!string.IsNullOrEmpty(x.CompanyName.Trim()) select x).Count();
if (existe > 0)
{
bool encuentro = false;
foreach (DMTem.ListaDatos dat in BaseDatosPermitidos.Lista.Where(x =>
process.MainModule.FileName.ToUpper().Contains(x.Dato.ToUpper()) &&
!string.IsNullOrEmpty(x.CompanyName.Trim())))
{
if (process.MainModule.FileVersionInfo.CompanyName ==
dat.CompanyName && process.MainModule.FileVersionInfo.InternalName
== dat.InternalName &&
process.MainModule.FileVersionInfo.LegalCopyright ==
dat.LegalCopyright)
encuentro = true;
}
if (!encuentro)
{
process.Kill();
}
}
}
}
timer1.Enabled = true;
}
```

Actualiza programas permitidos:

```
public void ActualizaPermitidos()
{
ltwPermitidos.Visible = true;
btncerrarpermitido.Visible = true;
try
{
```

```
db2file.ActualizaDB();
ltwPermitidos.Items.Clear();
foreach (DMTem.ListaDatos dato in BaseDatosPermitidos.Lista)
{
   ltwPermitidos.Items.Add(dato.Dato);
}
}
catch (Exception e)
{
  MessageBox.Show(e.Message);
}
}
```

Guardar solicitud de programas:

```
private void btnAceptar_Click(object sender, EventArgs e)
    {
      //Validaciones sobre solicitud de ejecucion de programas
      if (txtPrograma.Text == string.Empty)
      {
         MessageBox.Show("Debe especificar un programa");
         return;
      }
      if (txtMotivo.Text == string.Empty)
      {
         MessageBox.Show("Debe especificar un Motivo");
         return;
      }
      if (txtSolicitante.Text == string.Empty)
      {
         MessageBox.Show("Debe especificar un Solicitante");
         return;
      }
      try
      {
```

```
        DMTem.Solicitud ent = (DMTem.Solicitud)SolicitudbindingSource.Current;
        da.GuardaSolicitud(ent);
        MessageBox.Show(this, "Guardado Existosamente", "Mensaje",
        MessageBoxButtons.OK, MessageBoxIcon.Information);
           Close();
        }
        catch (Exception Error)
        {
        MessageBox.Show(this, Error.Message, "Error", MessageBoxButtons.OK,
        MessageBoxIcon.Error);
        }
        }

public void GuardaSolicitud(DMTem.Solicitud Entidad)
      {
        //Despues de verificar
        SqlCommand cmd = new SqlCommand();
        cmd.CommandText = "GuardaSolicitud";
        cmd.CommandType = CommandType.StoredProcedure;
        cmd.Parameters.Add("@Data", SqlDbType.Xml).Value =
        DataTransport.EntidadToXmlString(Entidad);
        Datos.ExecuteNonQuery(cmd);
      }
```

Obtener datos del programa

Se Agrega un nuevo programa a la Lista de programas permitidos.

```
private void btnVerProgramas_Click(object sender, EventArgs e)
      {
        string FileFullPath, ImageName;
        openFileDialog1.FileName = "";
        openFileDialog1.ShowDialog();
        FileFullPath = openFileDialog1.FileName;
        ImageName = openFileDialog1.SafeFileName;
```

```
        ((DMTem.Solicitud)SolicitudbindingSource.Current).programa =
FileFullPath;
        SolicitudbindingSource.ResetBindings(false);
        System.Diagnostics.FileVersionInfo arc =
        System.Diagnostics.FileVersionInfo.GetVersionInfo(FileFullPath);
        ((DMTem.Solicitud)SolicitudbindingSource.Current).CompanyName =
        arc.CompanyName;
        ((DMTem.Solicitud)SolicitudbindingSource.Current).InternalName =
        arc.InternalName;
        ((DMTem.Solicitud)SolicitudbindingSource.Current).LegalCopyright =
        arc.LegalCopyright;
    }
```

Actualiza datos al servidor SQL Server:

Actualiza la base de datos local desde SQL server para uso de programas.

```
public static class db2file
  {
    /// <summary>
    public static void ActualizaDB()
    {
      DMTem.ServidorSQL data = DMTem.Configuracion.ReadFile();
      DMTem.DataAcess da = new DMTem.DataAcess();
      DMTem.BaseDatos db = new DMTem.BaseDatos();
      db.Lista= da.ActualizaPermisos(data.UsuarioSistema,
data.PasswordSistema);
      DMTem.LocalStore.Save(db);
    }
  }

public static BaseDatos ReadFile()
    {
      return ReadFile(ArchivoConfiguracion());
    }
public static BaseDatos ReadFile(string cFileName)
```

```
        {
            BaseDatos retorno;
            if (string.IsNullOrEmpty(cFileName))
            {
            throw new System.ArgumentException("No se ha especificado el Archivo",
            "PC - SIAP");
            }
            try
            {
            FileStream s = new FileStream(cFileName, FileMode.Open,
            FileAccess.Read);
            BinaryFormatter b = new BinaryFormatter();
            retorno = (BaseDatos)b.Deserialize(s);
            s.Close();
            }
            catch (Exception e)
            {
             throw new Exception("Error Al intentar Abrir el Archivo, " + e.Message,
            null);
            }
            return retorno;
            }
public static void Save(BaseDatos CLFile)
        {
            SaveAs(ArchivoConfiguracion(), CLFile);
        }
        /// <summary>
        /// Guarda archivo en el formato de
        /// </summary>
        /// <param name="NewFileName"></param>
        /// <param name="CLFile"></param>
        public static void SaveAs(string NewFileName, BaseDatos CLFile)
        {
            try
```

```
        {
            if (File.Exists(NewFileName))
            {
                //si existe lo borra para crear uno nuevo
                File.Delete(NewFileName);
            }
                FileStream stream = new FileStream(NewFileName, FileMode.Create,
                FileAccess.Write);
                BinaryFormatter b = new BinaryFormatter();
                b.Serialize(stream, CLFile);
                stream.Close();
        }
        catch (Exception e)
        {
            throw new Exception("Error al intentar Guardar Archivo, " + e.Message,
null);
        }
    }
  }
  [Serializable]
public class BaseDatos
  {
    public List<ListaDatos> Lista;
  }
  [Serializable]
  public class ListaDatos
  {
    public string Dato { get; set; }
    public string InternalName { get; set; }
    public string CompanyName { get; set; }
    public string LegalCopyright { get; set; }
  }
```