Universidad de las Ciencias Informáticas

Front-End para el Sistema de Atención a Emergencias SAEM

TESIS EN OPCIÓN AL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS

Autores: Miguel Angel Martín Piloto
Rolando Lenzano Rodriguez
Tutores: MSc. Aurelio Antelo Collado
Ing. Valentín de León Torres
Ing. Gilberto E. González Hidalgo

La Habana, junio de 2012
"Año 54 de la Revolución"

DEDICATORIA

Miguel Angel

A mi abuelo Lázaro. Papi has sido para mí: amigo, hombre, y más que abuelo, padre ejemplar. Gracias por estar siempre cerquita cuando te he necesitado.

A mi abuela Juana. Para ti las palabras más bonitas: "cumplí contigo, hoy puedes verme graduado como un universitario, sé que este es el mejor regalo que puedo darte".

A mi tía Lisi. Quizás tus métodos no son los mejores, pero me han ayudado a convertirme en el hombre que soy. Sé que en este día no hice nada especial, como tú dices: "solo cumplí con mi deber".

A mi madre. Gracias mamá por brindarme los medios necesarios para llegar hasta aquí. La vida quiso que tus hijos crezcan lejos de ti, pero esa distancia es mínima comparada con el amor que siento por ti. Tu hijo mayor, hoy se convierte en un hombre hecho y derecho.

A todos ustedes hoy les digo que ha llegado la hora de retribuirles lo que han hecho por mí. Gracias.

Rolando

Dedico este trabajo de diploma a mis padres Ricardo Y Darlines, a mi hermano Ricardito, a toda mi familia en general y a mi novia Maricé, por ser todos tan importantes para mí.

AGRADECIMIENTOS

Miguel Angel

A mi tutor Aurelio, por confiar desde el inicio en el éxito de este trabajo y enfundarme confianza cuando más lo necesité.

A mis tutores, Gilberto y Valentín, por la paciencia mostrada y por ayudarme en todo lo que pudieron.

A todos los que de una forma u otra aportaron su granito de arena para que este trabajo saliera adelante. Gracias a Jorge Heriberto, Adrian, Lia, Lafourie, Dariel, Manfred, por aportar parte de su tiempo y conocimiento para ayudarme cuando fue necesario.

A Laritza, Rosita y Adilén. Antes de entrar a la universidad me hubiera gustado saber que las personas que te quieren y aprecian son realmente quienes hacen de este un lugar valioso e importante. Se les quiere, gracias.

A mis hermanos de 5 años de universidad: Angel, Chino, Tony, Alejandro, Karina, Yuliany y Yamila. Todos los días doy gracias por haberlos conocido, porque al pasar unos años tal vez no recuerde muy bien lo que aprendí en clases, pero siempre recordaré todo aquello que viví con ustedes, con quienes estudié, me desesperé, grité y festejé, con quienes compartí todo mi tiempo. Quien logra un amigo en la universidad, logra un amigo para toda la vida. Sepan que siempre tendrán un lugar especial en mi corazón.

A mis abuelos Juana y Lázaro por estos 23 hermosos años que me han regalado, por la excelente crianza que me han dado y por ayudarme a convertirme en el hombre que soy.

A mis tios Lisi y Sergio por quererme como un hijo más y velar por mi bienestar durante todo este tiempo.

A mi madre Ivedt por darme la vida, por quererme mucho y preocuparse por todo lo concerniente a mí en todo momento.

A mi hermano Leo y mis hermanas Elianny y Adianny, tomen experiencia de este día, dentro de muy poco les toca a ustedes.

A Dios y mi virgencita, por darme salud y fuerza para llegar hasta aquí y cumplir con mi familia.

A la Revolución y al Comandante, principal artífice de este proyecto, por brindarnos la posibilidad a muchos jóvenes, de hacer realidad nuestros sueños. Gracias.

Gracias por todo lo que han hecho por mí.

Rolando

A mis padres Ricardo y Darlines, por ser siempre mis ejemplos a seguir durante mi etapa de estudios y mi vida en general.

A mi hermano Ricardito, por estar siempre pendiente de mis problemas y dispuesto a darme una solución.

A todos mis familiares, por aportar siempre un granito de arena para hacer de mí una mejor persona.

A todos mis compañeros de grupo y de la residencia desde primer año hasta quinto, los cuales hicieron mi estancia en la universidad más agradable.

A mis profesores.

A mis compañeros de proyecto, que de alguna forma contribuyeron con la realización de este trabajo, especialmente a Gilbertus, Aurelio, Valentín.

A mi compañero de residencia, de grupo, de proyecto, de tesis, Miguel Angel.

A todos mis amigos y amigas en general.

A mis hermanos de siempre Daniel, Freddys, Yoevis y Miguel Enrique.

A mi novia Maricé, por estar siempre conmigo y apoyarme en los buenos y malos momentos por los que he pasado.

RESUMEN

La tendencia de desarrollo de software vigente en el mundo actual va dirigida a generar estrategias viables que incorporen actualizaciones. Esto se debe a la permanente evolución tecnológica, la velocidad con que suceden estos cambios, la dinámica de necesidades y los cambiantes requerimientos de los usuarios hacia las aplicaciones. La actualización se ha convertido en un proceso que garantiza que los clientes mantengan el uso de los productos.

El sistema informático del Centro de Información y Mando de la Unidad Provincial de Patrullas (UPP) es el encargado de contribuir al cumplimiento de las misiones asignadas a las fuerzas policiales de La Habana. Cuenta con un sistema distribuido que no posee un mecanismo de actualización central, lo que dificulta su mantenimiento al tener que realizarse el mismo en cada una de las computadoras que se encuentra instalado. Además, no es lo suficientemente flexible como para absorber todos los cambios a nivel organizacional que inducen nuevos requerimientos, lo que complejiza la realización de modificaciones sobre sus componentes.

El presente trabajo radica en el diseño e implementación de un Front-End para la aplicación informática Sistema de Atención a Emergencias (SAEM), basado en una arquitectura modular que facilite la actualización automática de los subsistemas del proyecto Perfeccionamiento de los Sistemas Informáticos del Centro de Información y Mando de la UPP.

Palabras claves: actualización de software, actualizaciones automáticas de software.

ÍNDICE

INTRODUCCIÓN

En la actualidad, el manejo operativo de muchas organizaciones está soportado en aplicaciones de software. Este creciente desarrollo de sistemas se corresponde a que, en las últimas décadas, la informática ocupa un papel fundamental en la mayoría de las esferas; debido a esto, se ha convertido en un elemento clave para orientarse con pensamiento propio, asimilación e innovación.

La permanente evolución tecnológica, la velocidad con que suceden estos cambios, la dinámica de necesidades y los cambiantes requerimientos de los usuarios hacia las aplicaciones contribuye a que este sector presente una problemática inicial de constante adaptación y mejora de sus servicios (1).

Históricamente, el software era una forma estática de tecnología. Se adquiría un programa, se cargaba en el equipo y se utilizaba, sin tener permitido realizarle ninguna modificación, hasta que aparecía la próxima versión. Sin embargo, ese modelo ya no es aplicable en su totalidad (2).

La tendencia de desarrollo de software vigente en el mundo, va dirigida a generar estrategias viables que incorporen actualizaciones que ofrezcan niveles aceptables de integración, interoperación y distribución, dado que la actualización de software se ha transformado en un indicador para medir su eficiencia (1).

La actualización se ha convertido en un proceso realizado por los propios proveedores para garantizar que los clientes mantengan el uso de sus productos, debido a que un sistema que se resiste a modificaciones y actualizaciones pierde competitividad aunque en un inicio fuera desarrollado aplicando adecuados principios de ingeniería y arquitectura.

El sistema informático encargado de contribuir al cumplimiento de las misiones asignadas a las fuerzas policiales Sistema Automatizado del Puesto de Mando de Seguridad Pública (PMSP) que presenta el Centro de Información y Mando de la UPP de La Habana es un software que, en estos momentos, en temas de actualización se ha quedado rezagado. El mismo, se encuentra en funcionamiento desde 2002, pero presenta las siguientes dificultades:

- Cuenta con un aplicación de escritorio que no posee un mecanismo de actualización central, lo que dificulta su mantenimiento al tener que realizarlo en cada una de las computadoras que se encuentra instalado, esto complejiza la realización de modificaciones sobre sus componentes.
- No es lo suficientemente flexible como para absorber todos los cambios a nivel organizacional que inducen nuevos requerimientos. Es por esto, que se ha convertido en un verdadero reto su escalabilidad, lo que dificulta el cumplimiento de sus objetivos y metas.

Por las razones descritas surge la necesidad de desarrollar una nueva versión del producto, denominada Sistema de Atención a Emergencias (SAEM), como parte del proyecto Perfeccionamiento de los Sistemas Informáticos del Centro de Información y Mando de la UPP de La Habana.

Esta solución se encuentra en fase de desarrollo en el Centro de Informatización de la Seguridad Ciudadana (ISEC) perteneciente a la Facultad 2 de la Universidad de las Ciencias Informáticas (UCI). La creación de este nuevo sistema debe cumplir con los siguientes objetivos: mejorar el intercambio de información con los sistemas ministeriales, responder a los cambios presentados en el negocio y agregar nuevas funcionalidades.

Debido a esto se necesita gestionar que la nueva solución sea desarrollada con una arquitectura modular, donde sus subsistemas puedan ser ejecutados en el momento preciso y actualizados de forma dinámica agregando mejoras y nuevas funcionalidades para disminuir el tiempo de mantenimiento del mismo.

Teniendo en cuenta los elementos antes mencionados, se definió el siguiente **problema científico**: ¿Cómo garantizar en la aplicación informática SAEM la actualización automática de su aplicación común y sus subsistemas?

El problema descrito tiene como **objeto de estudio**: el proceso de actualización de software y como **campo de acción**: el proceso de actualización de software de escritorio.

Para dar solución al problema antes mencionado se plantea, como **objetivo general**: desarrollar un Front-End como parte de la aplicación común de la solución informática SAEM que garantice su actualización y la de sus subsistemas de forma automática.

Con vista a cumplimentar el objetivo general se trazaron los siguientes **objetivos específicos**:

- Definir una arquitectura informacional para el Front-End.

- Desarrollar una arquitectura modular para el Front-End.
- Implementar un mecanismo de actualización para la aplicación informática SAEM.
- Realizar pruebas adecuadas al Front-End.

Se plantearon las siguientes **tareas de investigación** para dar cumplimiento al objetivo general:

- Estudio del lenguaje de etiquetado XML.
- Selección de librerías para el trabajo con el lenguaje de etiquetado XML.
- Selección de los patrones adecuados para la arquitectura modular.
- Definición de la estructura de los subsistemas en un formato definido por el proyecto.
- Selección de librerías para el trabajo visual del Front-End.
- Reutilización de nuevos componentes visuales para el Front-End.
- Diseño de la arquitectura modular a utilizar en el Front-End.
- Implementación de la arquitectura modular propuesta.
- Organización del Front-End.
- Rotulación del Front-End.
- Selección del mecanismo de actualización de aplicaciones a emplear en la aplicación informática SAEM.
- Implementación del mecanismo de actualización definido.
- Definición de los pasos necesarios para implementar un subsistema a añadir por el Front-End.
- Selección de criterios de evaluación y de validación del Front-End.
- Realización de las pruebas seleccionadas al Front-End.

En el desarrollo del presente trabajo se han utilizado un conjunto de métodos científicos para la obtención de la solución, procesamiento y arribo a conclusiones, dentro de los que se pueden mencionar los siguientes:

Métodos teóricos

- Analítico - sintético: permitió complementar el objeto de estudio mediante el análisis del estado del arte.

- Histórico – lógico: permitió analizar la tendencia del desarrollo del software de actualización.

La **significación práctica** de este trabajo radica en el diseño e implementación de un Front-End para la aplicación informática SAEM basado en una arquitectura modular que facilite la actualización automática de los subsistemas del proyecto Perfeccionamiento de los Sistemas Informáticos del Centro de Información y Mando de la UPP.

El presente trabajo de diploma se ha estructurado por capítulos de la siguiente manera:

Capítulo 1. Fundamentación teórica: aborda un estudio del estado del arte de las tecnologías informáticas desarrolladas para la actualización de software, su importancia y particularidades. Además de las características fundamentales de las tecnologías, las herramientas y la metodología de desarrollo utilizadas para el desarrollo del sistema.

Capítulo 2. Características del Front-End: describe la solución propuesta que utiliza un modelo de dominio para una mejor comprensión de los conceptos asociados al entorno. Se definen los requerimientos del sistema, tanto funcionales como no funcionales y a partir de estos se obtienen y describen los casos de uso del sistema (CUS) que guiarán la solución del mismo.

Capítulo 3. Diseño del Front-End: aborda la arquitectura modular del Front-End. Incluye una descripción sintáctica y semántica de la estructura de los archivos XML pertenecientes a los subsistemas. Se muestra el modelo de diseño y se abordan los temas de arquitectura informacional y usabilidad. Se presenta el diagrama de despliegue del sistema.

Capítulo 4. Implementación del Front-End: presenta los estilos de codificación y algunos códigos de ejemplos, así como el diagrama de componentes.

Capítulo 5. Prueba del Front-End: aborda las pruebas funcionales que se realizarán al Front-End.

FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

El presente capítulo tiene como objetivo fundamentar los principales conceptos relacionados con el proceso de actualización de software. Además, se analizan las características fundamentales de las tecnologías, las herramientas, la metodología de desarrollo, lenguaje de modelado y lenguaje de programación necesarias para el desarrollo de la solución.

1.2 Actualización de aplicaciones de software

La rápida evolución de las tecnologías de la información, la creciente complejidad de los requerimientos de negocio, los continuos cambios en el contexto de desarrollo, entre otros aspectos ocasionan una rápida obsolescencia del software (1).

Históricamente, el software informático era una forma estática de tecnología. Un cliente compraba un programa, lo instalaba y utilizaba en su equipo hasta que aparecía una nueva versión. Sin embargo, ese modelo ya no es aplicable (2).

El mundo digital actual está en constante cambio, y con la finalidad de que los últimos avances estén disponibles en el menor tiempo posible, el software ahora es mucho más dinámico. La actualización es generalmente un proceso realizado por los proveedores para garantizar que los clientes mantengan el uso de sus productos (3).

El proceso de actualización de un software puede ser visto como el cambio de una configuración a otra mediante adición, eliminación, reemplazo o reconfiguración de las funcionalidades de este.

Una actualización es una revisión de amplia distribución de un problema específico. Contiene alteraciones apropiadas de la funcionalidad y de la configuración, esta actividad constituye para muchas organizaciones un proceso mayoritariamente exento de errores para que se logre una actualización exitosa de los sistemas informáticos (4).

La administración de actualizaciones es una función importante para mantener cualquier SO o programa informático. La capacidad de identificar qué actualizaciones existen e implementarlas selectivamente de forma automatizada y oportuna, ayuda a mantener la seguridad y productividad de los sistemas informáticos.

Las actualizaciones de software incluyen revisiones y paquetes de servicios. Una revisión es un paquete acumulativo, que responde a una situación de un cliente específico, compuesto de uno o varios archivos que solucionan un problema de un producto (5). Un paquete de servicios es una actualización periódica que corrige los problemas con una versión en particular del producto (4).

La revisión y actualización, en su caso, de la tecnología de hardware y software que un usuario utiliza, deben de contemplar los posibles reajustes que se imponen y que deben sufrir sus aplicaciones. Para ello, deben tenerse en consideración fundamentalmente criterios relativos a la mayor eficacia y al máximo rendimiento posible, ambos en el ámbito de funcionalidad, tiempo y resultados positivos.

1.3 Importancia de las actualizaciones de productos de software

Los avances tecnológicos influyen constantemente en el campo de la informática. A medida que los usuarios dependen de sistemas cada vez más antiguos que no pueden reemplazarse o de nuevos sistemas que son complejos y modulares, se enfrentan a la necesidad de gestionar cambios en el software o de incorporar diferentes versiones (6).

En la actualidad existe una constante necesidad de actualizar el software, ya sea por cambios en las organizaciones en cuanto a sus procesos o por el constante desarrollo de la tecnología que afecta la infraestructura de las computadoras y dispositivos (6).

Cuando un usuario actualiza sus aplicaciones, logra dos objetivos (7):

- Que estas se encuentren más actualizadas, lo que debería mejorar tanto el rendimiento como la estabilidad.
- Que sean menos vulnerables a los ataques, y por lo tanto, más seguras.

El primer objetivo es simple: obtener las actualizaciones que estén diseñadas para las configuraciones actuales del equipamiento físico y de las aplicaciones instaladas en el ordenador del usuario. Por consiguiente, los programas que son actualizados deberían ejecutarse de manera más efectiva y sin problemas en su equipo. Este enfoque garantiza que existan menos errores y previene el mal funcionamiento de las aplicaciones (7).

Según Antonio[1] (8), el segundo objetivo es fácil de entender, pero difícil de medir. Décadas atrás, muchos especialistas sostenían que la creación de software era más un

[1] Antonio F. Zapico: Consultor de Tatum, empresa española de consultoría comercial, de marketing y personas.

arte que una técnica. Esto era especialmente cierto, pues se carecía de las herramientas tecnológicas y metodologías existentes en la actualidad para el desarrollo de software. En este llamado arte de crear software, la seguridad no figuraba como un objetivo a seguir y lo que se buscaba era una aplicación funcional que resolviera las necesidades del usuario (9).

Hoy en día, las metodologías y herramientas tecnológicas empiezan a considerar cada vez más el factor de seguridad, dado que para muchos desarrolladores de software no es suficiente codificar programas funcionales sino también seguros. Un aspecto esencial en la seguridad del ordenador de un usuario es mantener el software y el SO lo más actualizado posible, con los últimos parches instalados (10).

Cuando un producto sale al mercado, ha pasado una batería de pruebas realizadas por el fabricante. Al lanzarse y hacerse de dominio público, los usuarios tienen la oportunidad de buscar y probar nuevas características del producto. Muchos usuarios buscan los puntos débiles del software, los que lo hacen susceptible de ser manipulado o dañado. La teoría indica que la actualización del software con los últimos parches, actualizaciones o versiones, cierra los agujeros existentes en los programas y elimina las vulnerabilidades inherentes, de forma tal que las aplicaciones se tornen más seguras y resistentes a las intrusiones y secuestros (8).

Las actualizaciones deben tenerse en cuenta también para el SO instalado en el ordenador. Por lo general, las descargas de actualización para SO suelen corregir problemas de seguridad o de mal funcionamiento, solucionan problemas con protocolos de conexión, reparan el Kernel[2] para que refleje los últimos estándares de seguridad y, además, aplican ajustes a la configuración actual de seguridad y a la del entorno (7). Es conveniente, para los usuarios, tener actualizado su equipo, pues en ocasiones depende del hardware para que puedan actualizarse los programas.

Para las organizaciones, las actualizaciones de software deben ser objeto de un tratamiento específico que incluya la definición de un entorno de pruebas, en el que se valide que las mismas permitan la ejecución de todas las aplicaciones inmersas en el entorno de producción. Establecer cauces de recepción de parches o actualizaciones por parte de los fabricantes, su revisión y prueba y la puesta en producción de una forma eficaz, minimiza la probabilidad de ser afectados por software dañino que utilice sus defectos.

[2] Kernel: Parte esencial de un sistema operativo que provee los servicios más básicos del sistema.

Las empresas más importantes de software actualizan sus productos por Internet, mediante una búsqueda automática o manual, dependiendo del fabricante y con el previo consentimiento del usuario. Por lo general, estas actualizaciones son mejoras esenciales para corregir los errores detectados en las aplicaciones y no mejoras de funcionalidad. Para algunos especialistas, en un futuro no muy lejano, se prevé que muchas aplicaciones funcionarán por Internet, y que no será necesario instalarlas en los ordenadores. Esto puede garantizar que las actualizaciones lleguen a los usuarios de forma inmediata y reduce muchos costes a los fabricantes, debido a que no tienen que emplear soporte físico para distribuir sus programas (8).

Las actualizaciones de aplicaciones de software constituyen, en general, procedimientos claves de mantenimiento que los usuarios deberían realizar regularmente en sus ordenadores. Los beneficios de contar con un sistema actualizado y bien configurado superan, en gran medida, el tiempo que lleva ejecutar las actualizaciones.

La realización de este trabajo se centrará en desarrollar una solución informática que permita mantener actualizada la aplicación SAEM, lo que permitirá mejorar tanto el rendimiento como la estabilidad de la misma, así como adicionar nuevas funcionalidades.

1.4 Características de los sistemas de actualización

Existen dos tendencias principales en la concepción de mecanismos de actualización de aplicaciones (ver Figura 1). La corriente A está orientada a la publicación de actualizaciones en el servidor, que deja la responsabilidad a los clientes de realizar el proceso de actualización, mientras que en la tendencia B, el servidor tiene conocimiento de sus clientes y es capaz de ejecutar operaciones de actualización en estos. También existe la posibilidad de que el cliente decida cuándo actualizarse, de forma manual o automática (11).

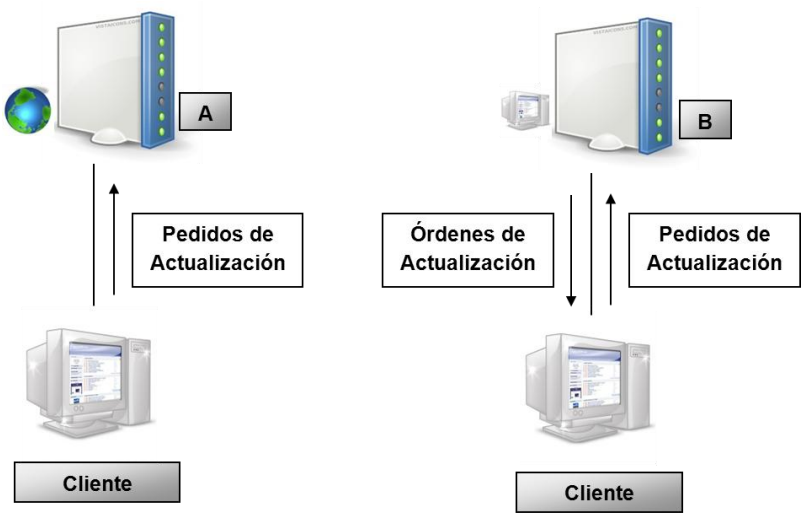

Figura 1 Tendencias en la concepción de mecanismos de actualización de aplicaciones

Cuando se analizan los mecanismos de actualización mencionados anteriormente, se identifican características y conceptos comunes dentro del proceso de actualización, que se muestran a continuación:

- Se identifican los pasos: detección, descarga y activación, en el proceso de actualización.
- Descripción de los procesos de actualización de las aplicaciones a través de archivos en formato XML.
- Publicación de los ficheros de actualización de manera centralizada.
- Empleo del modelo cliente-servidor a través de protocolos de comunicación ampliamente utilizados como HTTP y FTP.

De las tendencias expuestas, se decide utilizar la corriente A para la realización de este trabajo, debido a que el objetivo es almacenar las actualizaciones en un servidor y que las mismas sean consumidas por los clientes. Para este proceso solo es importante que los usuarios realicen peticiones de actualizaciones al servidor y que este permita la detección y descarga de las mismas.

1.5 Tecnologías que proporcionan actualizaciones de software

En la actualidad, los desarrolladores de software a nivel mundial son conscientes del beneficio de proporcionar actualizaciones de aplicaciones informáticas. La instalación y actualización son procesos fundamentales para la explotación de las potencialidades de un sistema informático. Con el objetivo de acelerar el desarrollo de estos procesos, se han desarrollado tecnologías que permiten la configuración de los mismos, todas orientadas a

resolver contratiempos de soporte, mantenimiento de sistemas y soluciones informáticas. Para el desarrollo de este trabajo se realizó un estudio de las tecnologías que tienen, entre sus objetivos fundamentales, la actualización de aplicaciones informáticas en aras de encontrar soluciones que puedan ser utilizadas para resolver el problema planteado.

1.5.1 ClickOnce

ClickOnce constituye una tecnología de implementación ofrecida por Microsoft, que proporciona actualizaciones automáticas de aplicaciones basadas en Windows a través de lecturas a su archivo manifiesto de implementación (12). El uso de esta tecnología facilita que no siempre se necesite descargar todos los archivos de una aplicación, a menos que hayan sido modificados.

Este mecanismo consiste en comparar las firmas hash[3] de cada fichero especificado en el manifiesto de programación de los archivos del cliente con las firmas del servidor, en caso de ser iguales no son descargados los archivos. Por otra parte, si el proyecto es compilado con Visual Studio, este genera una firma hash para cada fichero y ensamblado perteneciente a la aplicación, por lo que serán descargados todos los archivos de la aplicación aunque no se hayan modificado (12).

Al utilizar ClickOnce es posible establecer estrategias de actualización, las cuales consisten en determinar en qué momento de ejecución de la aplicación se deben realizar las actualizaciones, la frecuencia con que se realizan las mismas y si son de carácter obligatorio (13).

1.5.1.1 Estrategias de actualización con ClickOnce

Comprobar si hay actualizaciones antes del inicio de la aplicación

Con esta estrategia, la aplicación, intentará buscar y leer el archivo de manifiesto de implementación cada vez que sea ejecutada por el usuario. En caso de que no existan actualizaciones se ejecutará la aplicación con la versión actual.

Es recomendable usar esta estrategia antes del inicio de la aplicación, cuando se cuenta con conexiones de gran ancho de banda y las descargas se realizan de manera rápida, de lo contrario, el usuario tendría que esperar un largo tiempo, lo que resulta muy incómodo (13).

Comprobar si hay actualizaciones después del inicio de la aplicación

[3] Hash: Valores creados mediante funciones matemáticas.

Cuando se cuenta con conexiones de poco ancho de banda o aplicaciones de gran tamaño que requieren mucho tiempo de descarga se puede comprobar si hay actualizaciones después del inicio de la aplicación. Esta estrategia consiste en leer el archivo de manifiesto en segundo plano una vez ejecutada la aplicación y realizar las actualizaciones la próxima ejecución (13).

1.6 Estrategias de implementación de aplicaciones basadas en Windows

En la actualidad, Microsoft proporciona dos estrategias diferentes para implementar aplicaciones basadas en Windows: publicar una aplicación mediante ClickOnce o implementarla mediante un programa de instalación tradicional utilizando Windows Installer (14). Como ya se ha explicado, con la implementación ClickOnce, la aplicación se publica en una ubicación centralizada y el usuario la instala y ejecuta desde el sitio escogido. La implementación Windows Installer permite empaquetar la aplicación en un archivo *setup.exe* y distribuirlo entre los clientes, que lo ejecutan para instalar la aplicación.

Existen factores importantes a tener presentes al elegir una estrategia de implementación: el tipo de aplicación, el tipo y la ubicación de los usuarios, la frecuencia de actualizaciones de la aplicación y los requisitos de instalación.

1.6.1 Implementación de ClickOnce

ClickOnce es una tecnología de implementación que permite crear aplicaciones que pueden instalarse con una mínima interacción del usuario. Su ejecución supera tres problemas principales inherentes a la implementación de aplicaciones (12):

- **Dificultad para actualizar aplicaciones**: ClickOnce suministra actualizaciones automáticamente. Solo se descargan las partes de la aplicación que hayan cambiado, y a continuación, la aplicación completa y actualizada vuelve a instalarse desde una nueva carpeta simultánea.
- **Impacto en el equipo del usuario**: con ClickOnce, cada aplicación es independiente y no interfiere con otras aplicaciones.
- **Permisos de seguridad**: la implementación de ClickOnce habilita a los usuarios que no tienen derechos administrativos para realizar la instalación y solo concede los permisos de seguridad de acceso del código necesarios para la aplicación.

Una aplicación ClickOnce es cualquier aplicación de Windows Presentation Foundation (WPF), Windows Forms o de consola, publicada mediante esta tecnología. Para hacer efectiva la publicación existen tres vías fundamentales: desde una página web, desde un recurso compartido de archivos de red, o desde otros medios como un CD-ROM.

Las aplicaciones ClickOnce pueden ser instaladas en el ordenador del usuario final y ejecutarse localmente incluso cuando el equipo trabaja sin conexión, o bien, pueden ser ejecutadas solo en línea, sin instalar los recursos de manera permanente en el equipo del usuario final (15). Las aplicaciones realizadas con esta tecnología son la mejor elección para sistemas que requieran cambios con cierta frecuencia y deban actualizarse automáticamente.

La arquitectura central de la implementación ClickOnce se basa en dos archivos de manifiesto XML: un manifiesto de aplicación y un manifiesto de implementación. El manifiesto de aplicación describe la aplicación en sí. Incluye los ensamblados, las dependencias y los archivos que constituyen la aplicación, los permisos necesarios y la ubicación en que estarán disponibles las actualizaciones. El manifiesto de implementación describe cómo se implementa la aplicación. Esto incluye la ubicación del manifiesto de aplicación y la versión de la aplicación que debe ser ejecutada por los clientes (12).

Las aplicaciones ClickOnce, una vez instaladas, son agregadas al menú Inicio del usuario y al grupo Agregar o quitar programas del Panel de Control. A diferencia de otras tecnologías de implementación, no se agrega nada a la carpeta Archivos de Programas, al Registro ni al Escritorio, y no se requieren derechos de administración para realizar la instalación.

1.6.2 Implementación de Microsoft Windows Installer

Microsoft Windows Installer es un servicio de instalación y configuración que se incluye en el SO Windows a partir de la versión Windows 95. Está basado en un modelo controlado por datos que proporciona toda la información en instrucciones de instalación en un único paquete (16). Con Windows Installer, los ordenadores mantienen una base de datos (BD) de información sobre cada aplicación que se instala, incluyendo los archivos, las llaves del Registro y los componentes.

Es compatible con la reparación automática, es decir, la posibilidad de que una aplicación reinstale automáticamente los archivos que falten en el caso de que el usuario los haya borrado sin intención de hacerlo. Además, Windows Installer proporciona la posibilidad de dar marcha atrás en una instalación, de manera que, si una aplicación precisa una BD

determinada y esta no se encuentra, la instalación puede anularse y el ordenador volverá al estado que tenía antes de comenzar el proceso (17).

Los paquetes de Windows Installer se distribuyen mediante archivos de tipo MSI[4] y archivos MSP[5]. Los paquetes MSI se definen como instaladores de Microsoft, es decir, aquellos paquetes de software que contienen la información necesaria para automatizar su instalación sin necesidad de intervención manual del usuario en dicho proceso. Los paquetes MSP son paquetes de revisión de software, distribuidos como parches o actualizaciones para solución de problemas (17).

La instalación desatendida de una aplicación en un ordenador por medio de paquetes MSI, es sin duda un gran avance en la primera instalación de dicho software, pero a medida que el tiempo transcurre nuevas versiones de dicho producto salen al mercado, razón por la cual se debe actualizar la versión instalada previamente en la estación de trabajo del usuario. Esta tecnología permite especificar que un paquete MSI determinado es una actualización de otro anteriormente instalado. A diferencia de la implementación de ClickOnce, la implementación Windows Installer requiere permisos administrativos y solo permite una instalación de usuario limitada. Siempre que se actualiza una aplicación, el usuario debe volver a instalar todo el sistema, que puede utilizar componentes compartidos con otras aplicaciones, por lo que podrían existir posibles conflictos de versiones.

1.7 Transferencia eficaz de archivos

La transferencia de archivos garantiza que pueda realizarse el proceso de actualización, pues a través de ella los paquetes de versiones se desplazan desde el servidor de aplicaciones hacia las aplicaciones cliente.

La distribución de archivos constituye una tecnología versátil. En la actualidad, esta no se limita a un solo mecanismo debido a que las múltiples características de las redes a nivel mundial, exigen al hombre desarrollar nuevas técnicas de distribución de ficheros.

Un archivo o fichero es un sistema real o virtual de organización de la información mediante una clasificación determinada (18). En términos informáticos, es también un conjunto de información que se almacena en forma virtual para ser leído y/o accedido por medio de una computadora (19).

[4] MSI: Acrónimo de Microsoft Windows Installer.

[5] MSP: Acrónimo de Microsoft Windows Installer Patch.

El término transferencia se aplica, en el área de la tecnología, especialmente a la noción de transferencia de datos. Está representado por el traspaso de información de cualquier índole de un determinado tipo de sistema a otro similar (20). La transferencia de datos es hoy en día uno de los procesos más simples y útiles de realizar con los sistemas electrónicos y muchas de las posibilidades están especialmente adaptadas para facilitar la tarea al usuario promedio (Ver Figura 2).

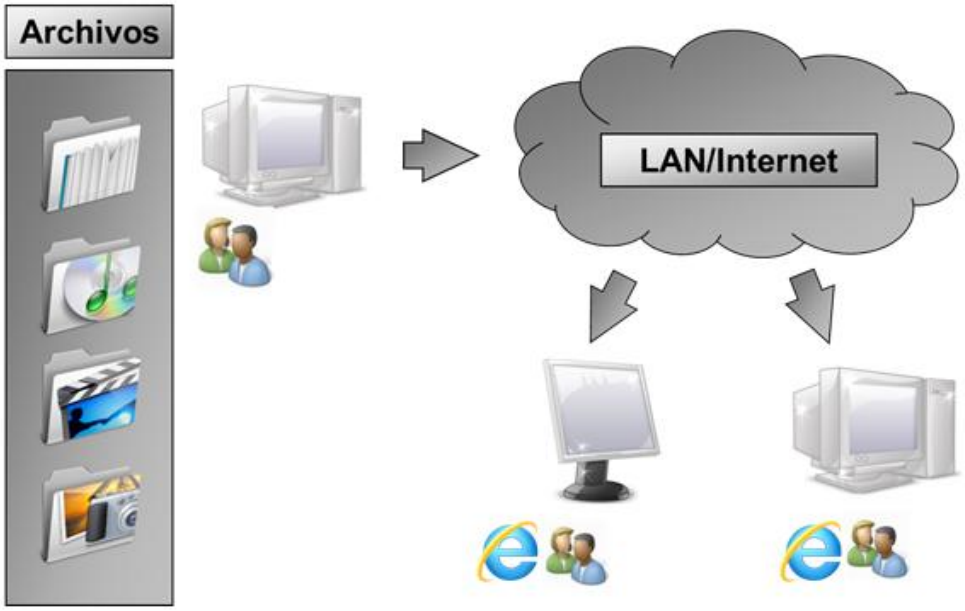

Figura 2 Transferencia de archivos

Este proceso puede realizarse de dos maneras básicas: a través de un sistema en red o a través de un puerto, siendo USB[6] el más conocido y utilizado, toda vez que los dispositivos involucrados deben contar con un lenguaje de protocolo que les haga compatibles.

En la medida en que los sistemas informáticos han evolucionado, han surgido nuevas y eficaces vías de transferencias de ficheros. Estudiar las tecnologías para la transferencia de archivos por la red se torna imprescindible para analizar ventajas y desventajas de la utilización de las mismas.

[6] USB: Acrónimo de Universal Serial Bus. Puerto USB es un puerto diseñado para conectar periféricos a un ordenador.

1.7.1 Protocolo de transferencia de archivos

Files Transfer Protocol (FTP, por sus siglas en inglés) es un protocolo de red, basado en la arquitectura cliente-servidor, que se utiliza para la transferencia de archivos entre sistemas conectados a una red TCP[7] (21).

Su función básica es la de conducir los datos entre dos ordenadores para facilitar el intercambio de ficheros. También permite crear y eliminar directorios, cambiar el nombre de archivos y realizar otras funciones relacionadas con el mantenimiento de estos.

FTP fue creado para cubrir los objetivos siguientes:

- Promover el intercambio de ficheros
- Aumentar el uso de ordenadores remotos
- Proteger al usuario de variaciones en los sistemas de almacenamiento usados por los distintos servidores
- Transmitir datos de manera eficiente

Este protocolo exige que un cliente FTP se conecte a un servidor FTP activo, que recibe e interpreta los comandos impartidos por el cliente. Para acceder a ficheros remotos, el usuario debe identificarse al servidor, que es responsable de autentificarlo antes de permitir la transferencia de datos.

Este estándar utiliza dos tipos de conexiones (22):

- **Conexión de control**: se establece por defecto en el puerto 21. A través de una conexión TELNET[8] se envían las órdenes al proveedor y se reciben los mensajes de respuesta.
- **Conexión de datos**: se establece por defecto en el puerto 20. A través de esta conexión se envían y reciben los ficheros que se transfieren.

Una de las ventajas de FTP frente a protocolos como HTTP[9] es que permite la descarga de archivos en modo binario directamente, sin necesidad de ser codificados a MIME, por ejemplo, donde el fichero es codificado como texto en la transmisión y se convierte nuevamente a binario al finalizar la transferencia, lo que incrementa las posibilidades de error en el envío y la recepción de datos. Por otra parte, FTP funciona sobre el protocolo

[7] TCP: Acrónimo de Transmission Control Protocol.

[8] TELNET: Acrónimo de Telecommunication Network.

[9] HTTP: Acrónimo de HyperText Transfer Protocol.

TCP/IP, que provee un sistema de control y corrección de paquetes una vez concluida la transferencia. Esto implica que si un paquete se pierde durante el envío, el ordenador que recibe los datos hace una petición de reenvío, lo que supone una ventaja de FTP al funcionar sobre un protocolo tan fiable como TCP/IP. El rendimiento y sencillez de FTP lo hace una opción conveniente para la transferencia de archivos a través de la red.

1.7.1.1 Servidor FTP

Un servidor FTP es un programa que se ejecuta en un equipo que funge como servidor conectado a diferentes tipos de redes (Internet, LAN[10], MAN[11], etc.), su función es permitir el intercambio de datos entre diferentes servidores y ordenadores (23).

Durante años el protocolo FTP estuvo pensado para ofrecer máxima velocidad en la conexión, pero no máxima seguridad. Los servidores FTP se comunicaban con los clientes sin que estos necesitaran iniciar una sesión en el host remoto o tener conocimientos sobre cómo utilizar el sistema remoto, lo que se traduce en que la información de la conexión y la contraseña eran vulnerables a la intercepción, debido a que no existía ningún tipo de cifrado.

Para contrarrestar este problema, en la actualidad, los servidores FTP controlan el acceso de los usuarios dentro de su sistema de carpetas, siendo responsabilidad del propietario del servidor FTP la política de creación y mantenimiento de usuarios y direcciones IP[12].

Soportan SSL[13]/TLS[14], lo que permite que puedan cifrar los comandos de control entre los clientes del FTP y el servidor, así como los datos del archivo. Algunos servidores, como es el caso de WS_FTP, se apoyan además en PGP[15] para asegurar los archivos con el cifrado público (24).

Dentro de las aplicaciones más comunes de los servidores FTP suelen estar el alojamiento web, en el que sus clientes utilizan sus servicios para almacenar sus páginas web; y su utilización como servidores de backup (copias de seguridad) de la información confidencial de organismos importantes.

[10] LAN: Acrónimo de Local Area Network

[11] MAN: Acrónimo de Metropolitan Area Network.

[12] IP: Etiqueta numérica que identifica, de manera lógica y jerárquica, a un dispositivo dentro de una red con protocolo IP.

[13] SSL: Acrónimo de Secure Socket Layer.

[14] TLS: Acrónimo de Transport Layer Security.

[15] PGP: Acrónimo de Pretty Good Privacy.

1.7.1.2 Cliente FTP

Un cliente FTP es un programa especial instalado en un ordenador de usuario, que emplea dicho protocolo para establecer una conexión con un servidor FTP y manejar los ficheros almacenados en este (21). Los clientes FTP clásicos, creados para acceder a los servidores FTP tradicionales (anónimos o privados) pueden subir/bajar archivos, es decir, transferirlos entre cliente y servidor FTP en cualquier sentido (25).

SO como Windows, DOS, GNU/Linux y Unix, disponen de clientes FTP básicos de modo texto. Aunque muy primitiva, esta aplicación permite conectarse y transferir información con cualquier servidor remoto.

Existen decenas de estos clientes FTP en la web, normalmente en forma de shareware[16]/freeware[17] para Windows y como software libre para Unix, con mayor o menor funcionalidades u opciones, pero todos comparten una estructura básica. Disponen generalmente de tres ventanas: una de ellas simboliza el ordenador del usuario (directorio de trabajo del equipo cliente); otra, el sistema de carpetas del servidor remoto (directorio actual en el servidor FTP). La tercera lleva un registro de los comandos utilizados, de manera que pueda verificarse lo ocurrido en la sesión (tanto los comandos enviados como las respuestas recibidas desde la máquina remota) (25).

Estos programas no solo realizan operaciones de transferencia de información, disponen también de un cierto número de utilidades que hacen más cómodo el trabajo. Pueden retomar descargas interrumpidas, guardar un directorio de los sitios más frecuentes con sus claves de acceso, directorio de inicio, etc.

Muchos navegadores recientes llevan integrados clientes FTP, aunque se recomienda la utilización de un cliente FTP para trabajar con un servidor FTP privado en lugar de un navegador. Entre los clientes FTP más utilizados en la actualidad se pueden citar: FileZilla, SmartFTP, CuteFTP, FlashXFP y Cyberduck (21).

1.8 Metodologías y herramientas utilizadas

En este epígrafe se analizan las características fundamentales de las tecnologías, herramientas y la metodología de desarrollo utilizadas para la construcción del sistema. Para la selección de las tecnologías y herramientas se tomó como referencia el manual de Lineamientos Tecnológicos del Ministerio de Interior (MININT). Este documento no puede

[16] Shareware: Modalidad de distribución de software.

[17] Freeware: Definición de un tipo de software que se distribuye sin costo alguno.

ser referenciado en el presente trabajo debido a que constituye un documento confidencial de este ministerio.

Se decide utilizar el Proceso Unificado de Desarrollo (RUP, por sus siglas en inglés) como metodología para desarrollar este trabajo, debido a que es más que un simple proceso. Es un marco de trabajo que puede ser usado para desarrollar una gran variedad de aplicaciones informáticas para diferentes áreas de aplicación, tipos de organización, niveles de aptitud y tamaños de proyecto.

Además, guía el diseño, implementación y prueba a través de CU, convirtiéndose en uno de los elementos fundamentales dentro del proceso de desarrollo. Exige que la arquitectura deba estar diseñada para que el sistema evolucione no solamente en su fase inicial, sino también en futuras generaciones.

Está basada en un conjunto de funcionalidades primordiales para el proyecto, debido a que es centrada en la arquitectura y permite dividir el trabajo en varias iteraciones o mini proyectos los cuales representan un incremento del producto, que permite en proyectos muy complejos que duran años, la entrega de pequeñas versiones, para satisfacer paulatinamente las necesidades de los clientes (26).

Se selecciona Visual Paradigm para UML (VP-UML) porque constituye una herramienta profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos (OO), construcción, pruebas y despliegue.

VP-UML es una suite de herramientas CASE, multiplataforma, utilizada para dar soporte al modelado UML siguiendo el estándar $UML_{2.1}$. Está diseñada para usuarios interesados en el diseño de software OO como: ingenieros de software, analistas y arquitectos de sistemas (27). Su entorno de trabajo es amigable y flexible, sus componentes están organizados de forma sencilla y lógica, lo que facilita su utilización y aprendizaje a los usuarios. Además, presenta licencia gratuita y comercial, su instalación y actualización son fáciles y avala la compatibilidad entre anteriores versiones de la herramienta. Además permite la integración con diferentes IDEs[18].

Teniendo en cuenta el documento que expone los Lineamientos Tecnológicos del MININT se aprecia que los lenguajes de programación avalados para implementar una aplicación informática son: C++, C#, Java, PHP y Python. Dado que existe como precedente que el

[18] IDE: Acrónimo de Integrated Development Environment.

framework base de la arquitectura de la aplicación SAEM se implementó en C#, se decide utilizar este lenguaje para el desarrollo del Front-End.

Como plataforma de desarrollo se decide emplear Microsoft .NET Framework 4.0, debido a su utilización previa en la implementación del framework base de la arquitectura del sistema. Además, se tuvieron presentes las numerosas ventajas que ofrece para el desarrollo de sistemas informáticos.

La misma está formada por una serie de herramientas y librerías con las que se pueden crear todo tipo de aplicaciones, como las tradicionales aplicaciones de escritorio (WPF o Windows Forms), aplicaciones para XBOX[19] (XNA[20]), sitios web (ASP.NET[21]), desarrollo para móviles (Compact Framework[22]), aplicaciones de servidor (WPF, WCF[23]), etc (28).

Es un componente integral de Windows que admite la compilación y ejecución de aplicaciones y servicios web. Proporciona un entorno de ejecución, un desarrollo e implementación simplificado y la integración con una gran variedad de lenguajes de programación (29).

Este framework conjuntamente con el SO conforma una plataforma de desarrollo que pone a disposición del programador las herramientas y servicios necesarios para crear desde grandes aplicaciones web hasta ligeras aplicaciones móviles a través de mecanismos robustos, seguros y eficientes para asegurar la ejecución de las mismas.

Se selecciona Visual Studio (VS) como IDE por ser un conjunto completo de herramientas de desarrollo para la generación de aplicaciones web ASP.NET, servicios web XML, aplicaciones de escritorio y aplicaciones móviles. Los lenguajes de programación Visual Basic, Visual C# y Visual C++ utilizan todos el mismo IDE, que habilita el uso compartido de herramientas y facilita la creación de soluciones en dichos lenguajes (30).

Este potente IDE está compuesto por un conjunto de herramientas de desarrollo basadas en otras tecnologías para compilar soluciones de alto rendimiento. Está optimizado para el diseño, desarrollo e implementación en equipo para soluciones de gran envergadura. El

[19] XBOX: Videoconsola de sobremesa de sexta generación producida por Microsoft.

[20] XNA: Conjunto de herramientas con un entorno de ejecución administrado proporcionado por Microsoft que facilita el desarrollo de juegos de ordenador.

[21] ASP.NET: Framework para aplicaciones web desarrollado por Microsoft.

[22] Compact Framework: Versión de .NET Framework, diseñado para funcionar en Windows CE (Windows Embedded Compact) basado en móviles/dispositivos embebidos.

[23] WCF: Acrónimo de Windows Communication Foundation.

mismo viene con un conjunto de mejoras de diferentes tipos que hacen más cómodo su uso (30):

- **Compatibilidad con varios monitores**: las ventanas de documentos como editor de código y vista de diseño pueden situarse fuera de la aplicación de VS lo que posibilita tener varias páginas abiertas en diferentes monitores.
- **Editor de código**: el nuevo editor de código brinda la posibilidad de ampliar el tamaño de la fuente para una mejor visión. Seleccionando cualquier símbolo de Visual C# el editor de código automáticamente resalta las demás instancias de dicho símbolo en todo el documento.
- **Jerarquía de llamadas**: permite navegar desde un miembro hasta los miembros que lo llaman, lo que resulta útil para explorar el código sin necesidad de ir abriendo clases por clases.
- **Depuración**: el IDE contiene una nueva ventana que agrupa un conjunto de funcionalidades como filtrado, búsqueda y expansión de pilas de llamadas. Además, se pueden compartir puntos de interrupción con otros programadores en el resto del equipo de desarrollo lo que facilita el trabajo en equipo y colaboración.
- **Pruebas**: cuenta con pruebas de interfaz de usuario (IU) automatizadas en aplicaciones basadas en web y en Windows, así como de pruebas manuales, de rendimiento web, de carga, cobertura de código y otras características completas que no se encuentran en otras ediciones de VS.
- **IntelliTrace**: incorpora la herramienta IntelliTrace, la cual que permite depurar la aplicación en tiempo de ejecución, archivando errores enriquecidos y modificables para que los desarrolladores puedan reproducir siempre el error del que se informe en el estado en el que se encontró. Además, cuenta con analizadores de código estático, métricas de código y creación de perfiles (30).
- **Multitargeting Support**: permite compilar las soluciones en diferentes versiones de Framework .NET.

Si se toman como referencias el lenguaje de programación, la plataforma de desarrollo y el IDE anteriormente seleccionados; se decide la utilización de la tecnología ClickOnce para publicar las actualizaciones paulatinas del Front-End, dado que constituye una estrategia de implementación proporcionada por la plataforma de desarrollo seleccionada y además soluciona importantes problemas innatos a la implementación de aplicaciones.

Para el almacenamiento de las actualizaciones de los subsistemas de la aplicación SAEM se define como servidor FTP a utilizar, el software Gene6FTP Server. Este es desarrollado específicamente para los requisitos de seguridad de alto nivel y de elevado rendimiento. Es un servidor profesional que ofrece velocidad, fiabilidad y personalización. Sus principales activos son la administración remota, conexión encriptada (SSL de 128 bits), y la facilidad de uso.

Algunas de las opciones que ofrece Gene6FTP Server se mencionan a continuación:

- SSL (explícita/implícita) puede ser utilizada para encriptación de archivos y datos.
- Restricciones de acceso basados en IP/Host.
- Archivos/Directorios manejados por permisos de acceso.
- Administración remota a través de conexiones seguras.
- Autenticación NT (utilización de la contraseña de Windows).
- Limitación máxima de conexión (IP, dominio, usuario).
- Expiración de cuentas de usuario.
- Construido en los comandos de verificación de integridad (CRC[24], MD5[25]).

1.9 Conclusiones

En el presente capítulo se realizó un estudio de la fundamentación teórica y del conjunto de metodologías, tecnologías y herramientas que formarán parte de la solución propuesta. Se analizaron dos tendencias actuales en la concepción de mecanismos de actualización y se decidió utilizar la corriente que describe la realización, por parte de los clientes, de peticiones al servidor.

Se profundizó acerca de tecnologías que proporcionan actualizaciones automáticas de aplicaciones. Se define como protocolo de transferencia de archivos a utilizar, FTP, y la herramienta Gene6FTP Server como servidor FTP que almacenará las actualizaciones de los subsistemas de la aplicación, dadas las numerosas ventajas que provee para la seguridad de archivos. Además, se puntualiza el uso de ClickOnce para publicar las actualizaciones paulatinas del Front-End.

[24] CRC: Acrónimo de Verificación de Redundancia Cíclica. Método de control de integridad de datos.

[25] MD5: Acrónimo de Message-Digest Algorithm 5. Algoritmo de reducción criptográfico de 128 bits.

Se decidió la utilización de la metodología de desarrollo RUP y la herramienta de modelado VP-UML para la realización y diseño del Front-End mediante la utilización del lenguaje de modelado UML. Se seleccionaron las tecnologías que brinda la plataforma .NET Framework 4.0, el lenguaje de programación C# para la implementación de la aplicación mediante el uso del entorno de desarrollo integrado VS 2010.

CARACTERÍSTICAS DEL FRONT-END

2.1 Introducción

En el presente capítulo se describe la solución propuesta, donde a través de un modelo de dominio se presentan, para una mejor comprensión, los conceptos asociados al entorno. Se definen los requerimientos del sistema, tanto funcionales como no funcionales y a partir de estos se obtienen y describen los CU que guiarán la solución.

2.2 Breve descripción del sistema

La solución que se propone consiste en el desarrollo de un Front-End para la aplicación SAEM, provisto de un mecanismo de actualización automática de los subsistemas que lo componen.

Para lograr una adecuada estructuración del menú del Front-End, se utilizó el componente visual Ribbon de Microsoft, en su versión 3.5. Además, se utilizaron otros componentes como el manejador de ventanas AvalonDock, una librería de código abierto escrita totalmente en C# y XAML para aplicaciones basadas en WPF y Windows Forms. Este componente posibilita la creación de ventanas, objetos y regiones acoplables o flotantes, que el usuario puede distribuir por la aplicación según sus preferencias.

Para el desarrollo del servicio de actualización se siguió el mecanismo de comunicación cliente-servidor, donde la información es publicada y administrada en un servidor y los nodos clientes interactúan con el mismo haciendo pedidos de recursos. En las actualizaciones del Front-End, las versiones son publicadas en el servidor de aplicaciones del sistema mediante la utilización de la tecnología ClickOnce, que proporcionará actualizaciones automáticas al sistema mediante lecturas de su archivo manifiesto de implementación.

La información relacionada con las actualizaciones de los subsistemas, que contiene el archivo manifiesto y el comprimido correspondientes a cada uno, es publicada en el servidor FTP del sistema. Las nuevas versiones de actualizaciones son comprobadas a través de los archivos manifiesto, que contienen además, información acerca del desarrollador responsable del subsistema, la firma hash del comprimido para una posterior comprobación de su integridad, y la fecha en que fueron publicadas las versiones en el servidor FTP. El Front-End utilizará las credenciales del usuario

autenticado para establecer la conexión con el servidor FTP y determinar los directorios a los que tiene acceso dentro del sistema de carpetas.

El comprimido perteneciente a cada subsistema contiene un paquete de ensamblados, así como dos archivos XML. Uno se utilizará para cargar dinámicamente los elementos que conformarán el Ribbon y estructurar, de esta manera, el menú del subsistema, y el otro, para especificar la dirección física de cada ensamblado y sus dependencias. La aplicación es capaz de determinar cuáles recursos son diferentes en el nodo cliente respecto a los que se encuentran publicados en el servidor FTP, y así poder decidir si lo debe descargar o no.

2.3 Modelo de dominio

Un modelo de dominio es una representación de los conceptos u objetos de la realidad física, presentado como uno o más diagramas de clases. Puede utilizarse para capturar y expresar el entendimiento ganado en un área bajo análisis como paso previo al diseño de un sistema, ya sea de software o de otro tipo. Tiene como objetivo fundamental la descripción de las clases más importantes del sistema y contiene, no conceptos propios de un sistema de software, sino de la propia realidad física (26).

En el modelo de dominio que se muestra en la Figura 3, se aprecia que el usuario interactúa de forma directa con el Front-End, una vez haya ganado en funcionalidades a través de la inserción de los subsistemas desarrollados. Además de tener permitido configurar las preferencias generales del Front-End.

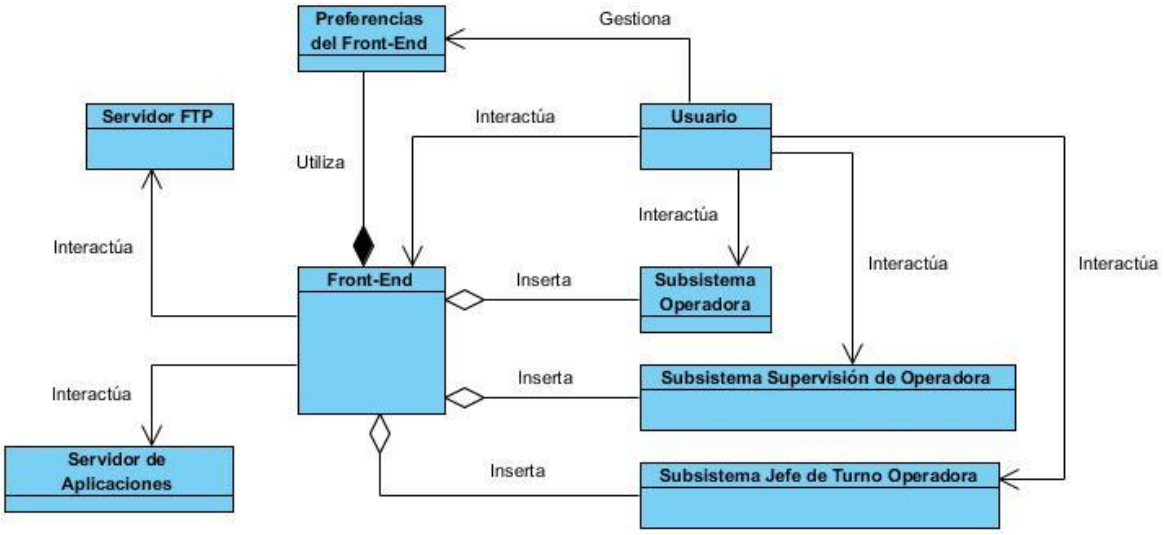

Figura 3 Modelo de dominio del Front-End

A continuación, para una mejor comprensión, se describen cada uno de los conceptos asociados al dominio que integran el diagrama anterior.

Front-End: parte visual que integra los datos que serán procesados por el Back-End.

Usuario: persona capacitada para interactuar con la aplicación.

Servidor FTP: servidor que almacenará las actualizaciones de los subsistemas desarrollados por el equipo de desarrollo del sistema.

Servidor de aplicaciones: servidor donde serán publicadas las actualizaciones paulatinas del Front-End.

Preferencias de la plataforma: permite configurar las preferencias del Front-End.

2.4 Especificación de los requisitos del sistema

Un requisito se define como una condición o capacidad que el sistema debe ser capaz de realizar, sin tomar en consideración ningún tipo de restricción física (26). Esta definición puede extenderse y ser aplicada a las condiciones que debe cumplir o poseer un sistema o uno de sus componentes para satisfacer un contrato, una norma o una especificación.

2.4.1 Requisitos funcionales

Los requisitos funcionales de un sistema describen la funcionalidad o los servicios que se espera que este provea, así como la manera en que reaccionará a entradas particulares (26). A continuación se muestran los requisitos funcionales del Front-End para la aplicación SAEM:

Requisitos funcionales del Front-End para la aplicación SAEM

RF1. Autenticar usuario
RF2. Actualizar subsistema
RF3. Adicionar subsistema
RF4. Actualizar el Front-End
RF5. Manipular ventanas
- RF5.1. Mover hacia arriba
- RF5.2. Mover hacia abajo
- RF5.3. Mover hacia la derecha
- RF5.4. Mover hacia la izquierda
- RF5.5. Embeber ventanas en el Front-End
- RF5.6. Extraer ventana del Front-End

RF6. Mostrar ventanas en forma de pestañas.

Requisitos funcionales del subsistema de administración

RF7. Publicar subsistema

RF7.1. Generar manifiesto del subsistema

2.4.2 Requisitos no funcionales

Los requisitos no funcionales son propiedades o cualidades que el producto debe poseer (26). Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable. Normalmente están vinculados a requerimientos funcionales, es decir, una vez que se conozca lo que el sistema debe hacer, se puede determinar cómo ha de comportarse, qué cualidades debe tener o cuán rápido o grande debe ser.

A continuación se presentan los requisitos no funcionales definidos para el desarrollo del Front-End para la aplicación SAEM:

Requisitos de software

Servidor FTP

RNF1. SO XP Service Pack (SP) 3 o superior y el servidor FTP Gene6FTP Server v3.8.0.

Servidor de Aplicaciones

RNF2. SO Windows Server 2008 R2 x64 Español, .NET Framework 4.0, IIS[26] 7.0, Cliente de Oracle.

Estaciones de trabajo

RNF3. SO Windows XP SP3, .NET Framework 4.0, WPFToolKit.

Requisitos de hardware

Servidor de Aplicaciones

RNF4. Microprocesador Intel Core 2 Duo, 4GB de memoria RAM y disco duro SATA con capacidad de almacenamiento mínima de 40GB.

Estaciones de trabajo

RNF5. Microprocesador Intel Pentium 4, 2GB de memoria RAM y disco duro SATA con capacidad de almacenamiento mínima de 40GB.

[26] IIS: Acrónimo de Internet Information Server.

Usabilidad

RNF6. El Front-End será diseñado para el Sistema de Atención a Emergencias SAEM, que será utilizado por los usuarios del Centro de Informatización y Mando de la UPP.

RNF7. Se debe lograr un diseño flexible, con capacidad de adicionar funcionalidades nuevas, sin que esto impacte de manera negativa en el sistema.

Fiabilidad

RNF8. Los subsistemas deben ser adicionados a la plataforma de forma dinámica. Por otra parte, deberá ser garantizada la funcionalidad independiente de cada uno, aunque existan dependencias entre ellos.

Soporte

RNF9. Los subsistemas y servicios que proporciona la aplicación deben estar bien documentados para que el equipo de desarrollo pueda darles mantenimiento en el menor tiempo posible.

RNF10. Los subsistemas de la aplicación deben funcionar sobre un SO sobre el cual se haya instalado previamente Microsoft .NET Framework 4.

RNF11. Las actualizaciones de los subsistemas y aplicación común deben estar de forma centralizada.

RNF12. Los subsistemas de la plataforma deben ser archivos comprimidos, que contienen un paquete de ensamblados y dos archivos XML.

Portabilidad

RNF13. La aplicación debe funcionar sobre el SO Microsoft Windows XP SP3 o superior.

Interfaz

Interfaces del sistema

RNF14. La interfaz externa debe ser fácil de utilizar, clara, sencilla y profesional. Debe ayudar a aumentar la efectividad y satisfacción del usuario.

RNF15. La interfaz externa debe proporcionar la funcionalidad de cambiar la apariencia en cuanto a la posición de las ventanas, según las necesidades del usuario.

RNF16. La navegación y distribución de las interfaces de cada subsistema deben estar diseñadas de forma tal que propongan escenarios familiares e intuitivos al usuario, centradas en las necesidades de este.

Seguridad

Confidencialidad

RNF17. La información que se transfiera será transmitida a través de un canal seguro, mediante el empleo del protocolo SSL/TLS, utilizado implícita o explícitamente por el servidor FTP del sistema, para cifrar el canal de control y/o datos.

Integridad

RNF18. Se manejarán mecanismos que comprueben la integridad de los archivos descargados. En este caso, se utilizará la función hash suma de verificación para detectar cambios accidentales o intencionados en los comprimidos que se descarguen, verificando que no existan discrepancias.

Disponibilidad

RNF19. Ante cualquier fallo en la conexión con el servidor FTP, el sistema trabajará con las versiones de los subsistemas contenidas en la carpeta local que almacena los módulos de la aplicación.

2.5 Definición de casos de uso del sistema

Los CU son fragmentos de funcionalidades que el sistema ofrece para aportar un resultado observable de valor para sus actores. Especifican secuencias de acciones, incluyendo variantes dentro de estas, que el sistema puede llevar a cabo al interactuar con sus actores (26).

2.5.1 Actores del sistema

Se denomina actor a una persona (usuario) o sistema que interactúa con la aplicación en desarrollo, con el propósito de lograr algún beneficio. Cada tipo de usuario, así como cada sistema externo con el que interactuará el sistema en desarrollo se representará mediante uno o más actores (26).

Actor	Descripción
Usuario	Persona que interactúa con las funcionalidades que el sistema ofrece.
Administrador	Tipo de usuario que interactúa específicamente con las funcionalidades del subsistema de administración.

2.5.2 Diagrama de casos de uso del sistema

Un diagrama de CUS especifica la comunicación y el comportamiento de un sistema, a la vez que muestra las relaciones entre los CUS y sus actores (26). Tanto actores como CU representan no las instancias particulares, sino los conjuntos de todos los actores de un tipo y de todos los escenarios posibles.

La Figura 4 representa el diagrama de CUS del presente trabajo, mientras que la Figura 5 muestra el diagrama de CUS perteneciente al subsistema Administración.

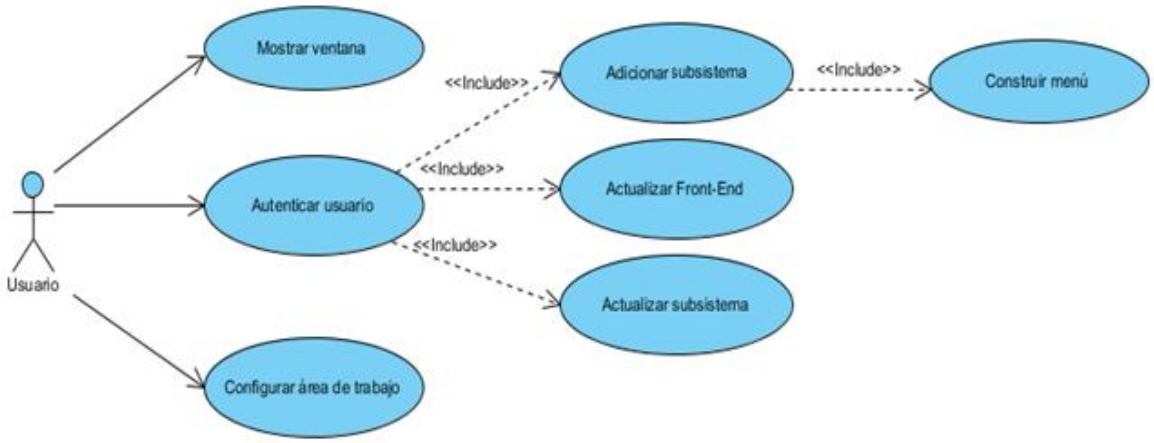

Figura 4 Diagrama de CUS del Front-End

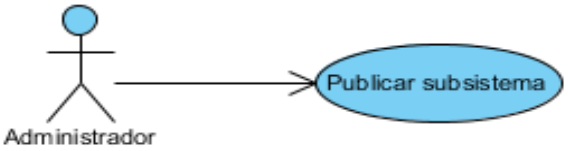

Figura 5 Diagrama de CUS del subsistema Administración

2.5.3 Descripción de los casos de uso del sistema

Un CU es una forma en que un actor utiliza el sistema (26). Especifica y describe comportamientos semejantes que suceden entre un tipo de usuario y el sistema al intentar realizar determinada funcionalidad. Proporciona uno o varios escenarios que indican la forma en la que debería interactuar el sistema con un usuario, o un sistema externo, para conseguir un resultado específico. A continuación se describen los principales CU del Front-End para la aplicación SAEM.

2.5.4 Descripción de los casos de uso del Front-End para la aplicación SAEM

CU Autenticar usuario

Objetivo	Permitir el acceso de un usuario al Front-End.

Actores	Usuario.
Resumen	El CU se inicia cuando el usuario para acceder al Front-End introduce su usuario y contraseña en los campos correspondientes. Concluye cuando accede al sistema.
Complejidad	Alta.
Prioridad	Crítica.
Precondiciones	No aplica.
Postcondiciones	El usuario se autenticó en el Front-End de la aplicación.
Flujo de eventos	
Flujo básico	

Actor	**Sistema**
1. El usuario introduce su usuario y contraseña.	
2. El usuario selecciona la opción que le permite autenticarse en el sistema.	3. Verifica que existe conexión con la BD.
	4. Verifica que los datos introducidos sean correctos.
	5. Verifica que el usuario no presente otra sesión abierta.
	6. Actualiza el estado del usuario a conectado.
	7. Se ejecuta el **CU Actualizar Front-End**.
	8. Se ejecuta el **CU Actualizar subsistema**.
	9. Se ejecuta el **CU Adicionar subsistema**.
	10. Verifica si existe conexión con el servidor de comunicación del sistema.
	11. Autentica al usuario en el servidor de comunicación de la aplicación.
	12. Crea el perfil del usuario autenticado con los siguientes datos:

	• Nombre completo. • Id de usuario. • Credenciales. • Roles. • Usuario. • Foto. • Jefes superiores. • Valores de sesión. • Id del turno de trabajo.
	13. Muestra en la región de datos del usuario los siguientes datos: • Nombre completo. • Usuario. • Rol. • Puesto de Trabajo.
Flujos alternos 3ª	
Nº Evento No existe conexión con la BD	
Actor	**Sistema**
	3ª.1 Muestra un mensaje de error.
Flujos alternos 4ª	
Nº Evento Datos incorrectos	
Actor	**Sistema**
	4ª.1 Muestra un mensaje de error.
	4ª.2 Realiza el ***flujo #1***.
Flujos alternos 5ª	
Nº Evento El usuario presenta una sesión de trabajo abierta	
Actor	**Sistema**
	5ª.1 Muestra un mensaje indicando al usuario que ya se encuentra utilizando el sistema.
Flujos alternos 10ª	
Nº Evento No existe conexión con el servidor de comunicación del sistema	
Actor	**Sistema**

	10ª.1 Muestra un mensaje de error.
Prototipo funcional	
	

CU Actualizar subsistema

Objetivo	Actualizar los subsistemas a los que tiene acceso el usuario autenticado en el Front-End.
Actores	Usuario.
Resumen	El CU se inicia cuando el sistema obtiene el listado de manifiestos de los subsistemas a los que tiene permiso el usuario autenticado y a través de una conexión al servidor FTP verifica que existen versiones superiores de los mismos. Concluye cuando los subsistemas son actualizados.
Complejidad	Alta.
Prioridad	Crítico.
Precondiciones	Deben existir en el servidor FTP versiones superiores de los subsistemas actualmente en uso por el usuario autenticado. El usuario debe estar autenticado en la aplicación.
Postcondiciones	Se actualizaron los subsistemas correspondientes.
Flujo de eventos	
Flujo básico	

Actor	**Sistema**
	1. Obtiene de la BD el listado de

	subsistemas a los que tiene acceso el usuario autenticado.
	2. Crea una instancia del servidor FTP con los siguientes datos: • Usuario. • Contraseña. • Url donde se encuentra el servidor FTP. • Carpeta local que contiene los subsistemas en uso.
	3. Verifica que existen, en la carpeta local que contiene los subsistemas de la aplicación, los archivos manifiestos de los subsistemas en uso por el usuario autenticado.
	4. Verifica que existe la carpeta temporal *SAEMTemporaryFolder* ubicada en la carpeta especial *Mis documentos*.
	5. Elimina la carpeta temporal *SAEMTemporaryFolder* y su contenido.
	6. Crea en la misma dirección la carpeta temporal *SAEMTemporaryFolder*, con las subcarpetas *Manifest* y *SubSystems*.
	7. Establece una conexión con el servidor FTP para obtener el listado de los directorios a los que tiene acceso el usuario autenticado.
	8. Establece una conexión con un directorio obtenido anteriormente.
	9. Obtiene, para el directorio obtenido, el archivo manifiesto del subsistema correspondiente, descargándolo en la

	carpeta *Manifest*, contenida en la carpeta temporal *SAEMTemporaryFolder*.
	10. Verifica que exista, en la carpeta local que contiene los subsistemas, una versión correspondiente al archivo manifiesto descargado.
	11. Guarda en dos variables temporales la versión del subsistema contenida en el archivo manifiesto descargado y la versión correspondiente existente en la carpeta local que contiene los subsistemas.
	12. Elimina el archivo manifiesto descargado.
	13. Compara las versiones del subsistema almacenadas en las variables temporales.
	14. Verifica que la versión del subsistema contenida en el archivo manifiesto descargado es superior a la versión almacenada en el archivo manifiesto existente en la carpeta local que contiene los subsistemas.
	15. Elimina el archivo manifiesto descargado.
	16. Descarga para la carpeta *SubSystems*, contenida en la carpeta temporal *SAEMTemporaryFolder*, el comprimido correspondiente al subsistema y su archivo manifiesto.
	17. Verifica que el usuario autenticado no tiene acceso a más directorios.

	18. Obtiene la firma hash del comprimido correspondiente al subsistema.
	19. Verifica la integridad del comprimido, a partir de la comparación de la firma hash obtenida con la almacenada en el archivo manifiesto correspondiente.
	20. Descompacta el comprimido y copia el archivo manifiesto en la carpeta local que contiene los subsistemas, sobrescribiendo su contenido.
	21. Verifica que no existan otros comprimidos descargados.
	22. Elimina la carpeta temporal *SAEMTemporaryFolder* y su contenido.
Flujos alternos 1ª	
Nº Evento No obtiene de la BD el listado de subsistemas a los que tiene acceso el usuario autenticado	
Actor	**Sistema**
	1ª.1. Muestra un mensaje informando que no existe conexión con la BD.
Flujos alternos 4ª	
Nº Evento No existe la carpeta temporal SAEMTemporaryFolder	
Actor	**Sistema**
	4ª.1 Realiza el ***flujo #6***.
Flujos alternos 7ª	
Nº Evento No establece una conexión con el servidor FTP	
Actor	**Sistema**
	7ª.1 Muestra un mensaje de error.
Flujos alternos 10ª	
Nº Evento No contiene una versión de los archivos manifiestos descargados	
Actor	**Sistema**
	10ª.1 Realiza el ***flujo #15***.
Flujos alternos 14ª	
Nº Evento El archivo manifiesto descargado no contiene una versión superior	

Actor	Sistema
	14ª.1 Elimina el archivo manifiesto descargado.
	14ª.2 Realiza el ***flujo #17***.
Flujos alternos 17ª	
Nº Evento El usuario autenticado tiene acceso a más de un directorio	
Actor	**Sistema**
	17ª.1 Realiza el ***flujo #8***.
Flujos alternos 19ª	
Nº Evento La firma hash del comprimido descargado no concuerda con la almacenada en el archivo manifiesto	
Actor	**Sistema**
	19ª.1 Realiza el ***flujo #21***.
Flujos alternos 21ª	
Nº Evento Existen otros comprimidos descargados	
Actor	**Sistema**
	21ª.1 Realiza el ***flujo #18***.
Prototipo funcional	
No aplica.	

CU Adicionar subsistema

Objetivo	Adicionar al Front-End una versión de un subsistema.
Actores	Usuario.
Resumen	El CU se inicia cuando el sistema obtiene los subsistemas a los que tiene acceso el usuario autenticado. Concluye cuando se adicionan los subsistemas al Front-End.
Complejidad	Alta.
Prioridad	Crítico.
Precondiciones	Debe existir en la carpeta local una versión de los subsistemas a los que tiene acceso el usuario autenticado. El usuario debe estar autenticado en la aplicación.
Postcondiciones	Se adicionó al Front-End una versión del subsistema al que tiene acceso el usuario autenticado.
Flujo de eventos	

Flujo básico	
Actor	**Sistema**
	1. Obtiene el listado de subsistemas a los que tiene permiso el usuario autenticado.
	2. Obtiene el ensamblado correspondiente al módulo *SAEM.Seguridad*.
	3. Obtiene el archivo *subsistemas.xml*, perteneciente al módulo *SAEM.Seguridad*, que guarda información sobre las direcciones físicas de los ensamblados de cada subsistema añadido a la aplicación.
	4. Verifica la integridad del archivo *subsistemas.xml*.
	5. Guarda en una variable temporal el contenido del archivo *subsistemas.xml.*
	6. Obtiene de las carpetas correspondientes a los subsistemas a los que tiene acceso el usuario autenticado los archivos denominados "*subsistema.xml*", que contienen información sobre las direcciones físicas de los ensamblados.
	7. Agrega el contenido de cada archivo denominado "*subsistema.xml*", a la variable temporal que guarda el contenido del archivo *subsistemas.xml*, perteneciente al módulo *SAEM.Seguridad*.
	8. Verifica que al menos se haya agregado información referente a un

	subsistema al archivo *subsistemas.xml*, perteneciente al módulo *SAEM.Seguridad*.
	9. Parsea el archivo *subsistemas.xml* y guarda en una variable temporal la información particular de los ensamblados pertenecientes a los subsistemas.
	10. Verifica la integridad de cada ensamblado perteneciente a los subsistemas agregados.
	11. Construye un catálogo con la información referente a cada ensamblado.
	12. Lanza un evento que indica que se ha cargado el catálogo de subsistemas.
	13. Se ejecuta el **CU Construir menú**.
	14. Inicializa las clases necesarias para cargar los subsistemas.
	15. Inicializa los subsistemas.
	16. Registra el proveedor de auditoría, que almacenará un registro de todas las acciones realizadas por el usuario.
	17. Cierra la ventana Splash de la aplicación.
	18. Muestra la ventana principal de la aplicación.
Flujos alternos 1ª	
Nº Evento No obtiene el listado de subsistemas	
Actor	**Sistema**
	1ª.1 Muestra un mensaje de error.
Flujos alternos 3ª	
Nº Evento No obtiene el archivo *subsistemas.xml*	

Actor	**Sistema**
	3ª.1 No construye el catálogo con la información referente a cada ensamblado perteneciente a los subsistemas a los que tiene acceso el usuario autenticado.
	3ª.2 Realiza el ***flujo #13***.
Flujos alternos 4ª	
Nº Evento El archivo *subsistemas.xml* presenta problemas de integridad	
Actor	**Sistema**
	4ª.1 Muestra un mensaje de error.
Flujos alternos 8ª	
Nº Evento No se agregó información referente a un subsistema al archivo *subsistemas.xml*	
Actor	**Sistema**
	8ª.1 Muestra un mensaje de error.
	8ª.2 Realiza el ***flujo #13***.
Flujos alternos 10ª	
Nº Evento Al menos un ensamblado presenta problemas de integridad	
Actor	**Sistema**
	10ª.1 Muestra un mensaje de error.
	10ª.2 Realiza el ***flujo #13***.
Prototipo funcional	
No aplica.	

CU Actualizar Front-End

Objetivo	Actualizar el Front-End de la aplicación.
Actores	Usuario.
Resumen	El CU se inicia cuando el sistema comprueba los requisitos previos de instalación del sistema. Concluye cuando se actualiza el Front-End del sistema.
Complejidad	Media.
Prioridad	Crítica.
Precondiciones	Debe existir en el servidor de aplicaciones una versión actualizada del Front-End.

	Debe existir conectividad con el servidor de aplicaciones.
Postcondiciones	Se actualizó el Front-End de la aplicación.
Flujo de eventos	
Flujo básico	

Actor	**Sistema**
	1. Comprueba que se encuentre instalado .NET framework 4.0 en la estación de trabajo.
	2. Verifica que existe una nueva versión del Front-End en el servidor de aplicaciones.
	3. Establece una conexión con el servidor de aplicaciones.
	4. Descarga las versiones de los archivos modificados.
	5. Actualiza los archivos modificados.

Flujos alternos 3ª	
Nº Evento No establece una conexión con el servidor de aplicaciones	
Actor	**Sistema**
	3ª.1 Muestra un mensaje de error.
	3ª.2 Pospone la actualización de los archivos modificados para el próximo inicio del sistema.

Prototipo funcional
No aplica.

2.6 Conclusiones

En el presente capítulo se presentó el modelo de dominio como punto de partida para la comprensión del sistema, toda vez que se describieron brevemente las clases que aportan un mejor entendimiento de la solución propuesta. Se describen los requerimientos funcionales y no funcionales que indican las capacidades o condiciones que el sistema debe cumplir y las cualidades o propiedades que el producto debe poseer. Se conforman los CUS, y estos son descritos al igual que sus actores.

DISEÑO DEL FRONT-END

3.1 Introducción

En este capítulo se describe la representación arquitectónica del sistema, haciendo énfasis en los patrones arquitectónicos empleados. Se desarrolla, además, el diseño del sistema.

3.2 Representación arquitectónica

La necesidad del manejo de la arquitectura de un sistema de software nace con los sistemas de mediana o gran envergadura, que se proponen como solución para un problema determinado. En la medida que un sistema crece en complejidad, bien sea por el número de requerimientos o por el impacto de los mismos, se hace necesario establecer medios para el manejo de esta complejidad (31).

Al hablar de arquitectura de software, se hace alusión a la estructura del sistema, entendida como la organización de componentes –módulos o piezas de código- y relaciones entre ellos; los requerimientos que debe satisfacer el sistema y las restricciones a las que está sujeto, así como las propiedades no funcionales del sistema y su impacto sobre la calidad del mismo; las reglas y decisiones de diseño que gobiernan esta estructura y los argumentos que justifican las decisiones tomadas (31).

Un patrón es una solución probada que puede ser aplicada con éxito a un determinado tipo de problemas que aparecen repetidamente en algún campo. El establecimiento de patrones comunes es lo que posibilita el aprovechamiento de la experiencia acumulada en el diseño de aplicaciones (31). A continuación se presentan los patrones utilizados para la implementación del sistema.

3.2.1 Patrón arquitectónico empleado

Los patrones arquitectónicos expresan el esquema de organización estructural fundamental para sistemas de software, toda vez que proveen un conjunto de subsistemas predefinidos, especifican sus responsabilidades e incluyen reglas y pautas para la organización de las relaciones entre ellos (31).

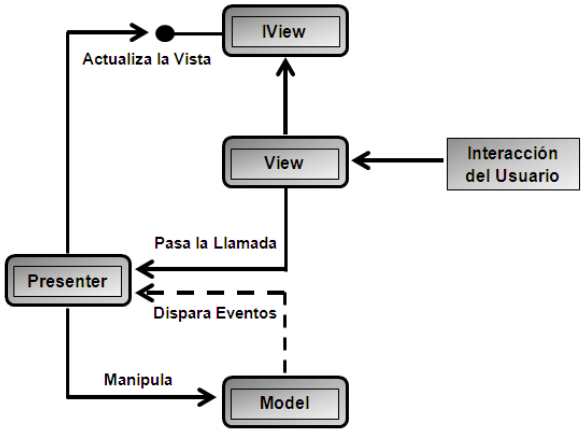

Figura 6 Descripción del patrón MVP

El patrón Modelo Vista Presentador, mostrado en la Figura 6, aísla el modelo del dominio, la presentación y las acciones basadas en la interacción con el usuario en tres clases separadas. La vista le delega a su presenter toda la responsabilidad del manejo de los eventos del usuario. El presenter se encarga de actualizar el modelo cuando surge un evento en la vista, también es responsable de actualizarla cuando el modelo le indica que ha cambiado. El modelo no conoce la existencia del presenter, por lo tanto, si el modelo cambia por acción de otro componente que no sea el presenter, debe disparar un evento para notificárselo.

A la hora de implementar este patrón, se identifican los siguientes componentes:

- **IView**: interfaz que permite la comunicación entre el presenter y la vista.
- **View**: vista que implementa la interfaz IView y se encarga de manejar los componentes visuales.
- **Presenter**: contiene la lógica para responder a los eventos y manipula el estado de la vista mediante una referencia a la interfaz IView. Utiliza el modelo para saber cómo responder a los eventos. Es responsable de establecer y administrar el estado de una vista.
- **Model**: compuesto por objetos que conocen y manejan los datos dentro de la aplicación.

3.2.2 Patrones de diseño empleados

Los patrones de diseño son soluciones concretas que proponen dar respuesta a problemas que ocurren frecuentemente en el desarrollo de aplicaciones, principalmente

en la fase de implementación. Para ello describen y clasifican diferentes formas para solucionar determinado problema (31).

A continuación se abordarán los patrones de diseño utilizados en el desarrollo del Front-End:

Command

Dos de los principales elementos de una interfaz gráfica de usuario (GUI) son los menús desplegables y las barras de botones. En la mayoría de los casos, un botón sirve para ejecutar alguna de las funciones más usuales de la aplicación, por lo cual coincide, en la función que desempeña, con alguna entrada de menú. Esto provoca un serio contratiempo, pues la implementación de la operación que ejecutan, tanto la entrada de menú como el botón, es duplicada. La utilización de este patrón evita esta redundancia definiendo la acción que se quiere ejecutar como un objeto, a la cual pueden referirse todos los elementos gráficos necesarios.

Este patrón se utilizó en la implementación de cada acción que realizan los elementos del menú de la aplicación, dígase por ejemplo, las pestañas y botones del Ribbon.

Facade

Este patrón, que también es conocido como fachada, ofrece un punto de acceso a un conjunto de clases o interfaces, de manera que se reducen las dependencias entre estas. De esta forma si es necesario modificar dichas clases solamente se actualiza la fachada. El mismo garantiza que se oculte la complejidad de las aplicaciones y disminuye el acoplamiento entre los subsistemas y el cliente. En la Figura 7 se muestra un ejemplo de la utilización de este patrón en el desarrollo del sistema.

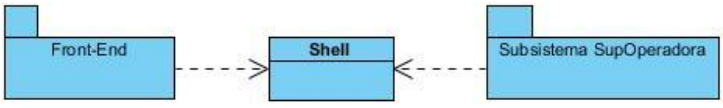

Figura 7 Aplicación del patrón Facade en la implementación del Front-End

Bajo acoplamiento

Asigna responsabilidades para mantener el bajo acoplamiento y aumentar la reutilización. Soporta el diseño de clases más independientes, que reducen el impacto de los cambios y también clases más reutilizables que aumentan la productividad. En la Figura 8 se muestra un ejemplo de la utilización de este patrón en el desarrollo del sistema.

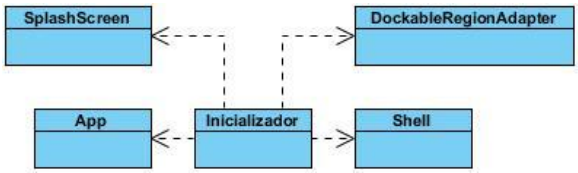

Figura 8 Aplicación del patrón Bajo Acoplamiento en la implementación del Front-End

Alta cohesión

Asigna una responsabilidad de modo que la cohesión siga siendo alta. La cohesión funcional es la medida de qué tan relacionadas están las responsabilidades de una clase. Una clase de alta cohesión posee un número relativamente pequeño de operaciones con una importante funcionalidad relacionada y poco trabajo por hacer. Colabora con otros objetos para compartir esfuerzo si la tarea es grande. En la Figura 9 se muestra un ejemplo de la utilización de este patrón en el desarrollo del sistema.

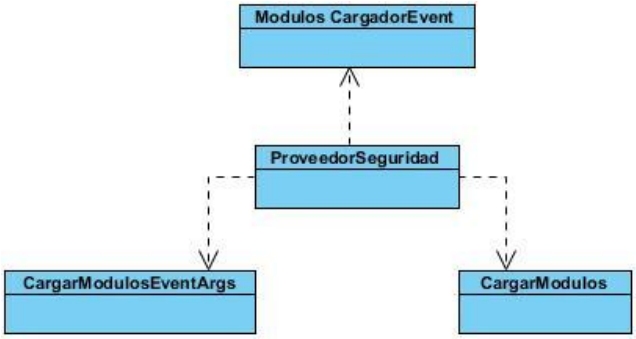

Figura 9 Aplicación del patrón Alta Cohesión en la implementación del Front-End

Creador

Asigna a la clase B la responsabilidad de crear una instancia de la clase A en uno de los siguientes casos:

- B agrega los objetos de A
- B contiene los objetos de A
- B registra las instancias de los objetos de A
- B utiliza específicamente los objetos de A
- B posee los datos de inicialización de que serán transmitidos a A cuando sea creado

Este patrón se utilizó en la implementación de varias clases del Front-End, como por ejemplo: *DockableRegionAdapter*, *Inicializador* (paquete SAEM.Shell) y *SaemFtp* (paquete Externals).

3.2.3 Arquitectura modular

Para el diseño e implementación de la arquitectura modular propuesta, el equipo de desarrollo del Front-End se apoyó en PRISM, la guía conocida formalmente como Composite Applications Guidance para WPF y Silverlight, que ayudó a diseñar y construir una aplicación de WPF flexible y fácil de mantener.

La Figura 10 muestra una vista vertical del Sistema de Atención a Emergencias SAEM y todos los componentes que colaboran para su funcionamiento, entre ellos, el Front-End desarrollado. Puede observarse, además, la utilización de patrones de diseño que incorporan principios de diseño arquitectónico, como el bajo acoplamiento, que PRISM utiliza para ayudar a diseñar y construir aplicaciones a través de la utilización de componentes débilmente acoplados que pueden evolucionar de forma independiente, pero que pueden ser fácil y perfectamente integrados en la aplicación.

Se desarrolló una aplicación WPF, que por lo general, contará con múltiples pantallas, visualización de datos, y que necesitará una importante presentación y lógica empresarial. El Front-End interactuará con sistemas back-end y servicios y, utilizando una arquitectura de capas, podrá ser físicamente desplegado en varios niveles. Se espera que la aplicación en general evolucione significativamente a lo largo de su vida en respuesta a las nuevas necesidades y oportunidades de negocio.

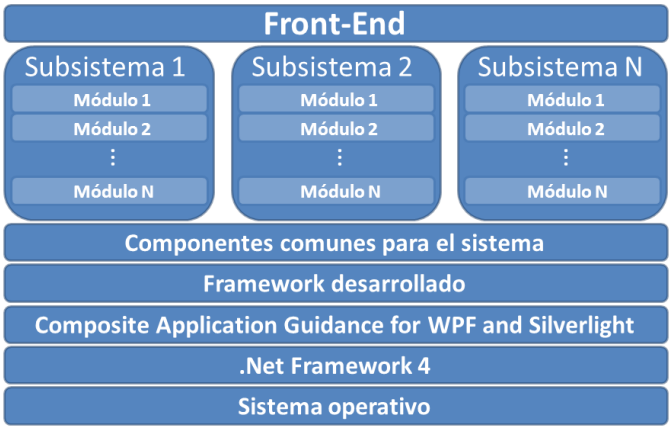

Figura 10 Vista vertical de la arquitectura de la solución SAEM

3.2.3.1 Ficheros XML

Como se explicó anteriormente, cada actualización de un subsistema contendrá un comprimido que incluirá -además de un paquete de ensamblados- dos archivos XML. La

Figura 11 muestra el XSD[27] del archivo XML utilizado para la estructuración del menú del subsistema correspondiente, mientras que en la Figura 12 se observa la estructura del archivo XML que especifica las direcciones físicas de cada ensamblado -perteneciente a un subsistema en específico- y sus dependencias.

Figura 11 XSD del archivo menú.xml

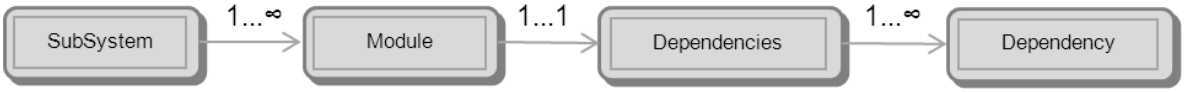

Figura 12 XSD del archivo subsistema.xml

3.2.3.2 Arquitectura informacional del Front-End

La organización de la información en una aplicación es un factor que incide de manera directa en el éxito o fracaso de esta. Es precisamente de esta organización que se ocupa la denominada Arquitectura de Información (AI). Con respecto a su definición, Wurman[28] (32) afirma que: se considera el estudio de la organización de la información con el objetivo de permitir al usuario encontrar su vía de navegación hacia el conocimiento y la comprensión de la información.

Para la creación de la AI del Front-End se tuvieron en cuenta los siguientes aspectos:

Organización de la información

Se mostrarán los subsistemas adicionados al Front-End en forma de pestañas, donde el título de esta coincidirá con el nombre del subsistema. Cada pestaña estará integrada por grupos, que a su vez contendrán botones; que garantizarán los requerimientos funcionales definidos por el equipo de desarrollo del sistema. En la Figura 13 se muestra un ejemplo de cómo debe ser diseñada una pestaña.

[27] XSD: Lenguaje de esquema utilizado para describir la estructura y restricciones de los contenidos de documentos XML.

[28] Richard Saul Wurman: Arquitecto y diseñador gráfico estadounidense que acuñó el término "Arquitectura de Información" en el año 1975.

Figura 13 Diseño de una pestaña

Ubicación de las opciones en el menú del subsistema

Se organizarán los componentes del menú de un subsistema según las semejanzas entre las funcionalidades correspondientes, además de establecerse una jerarquía entre ellos, para facilitar la localización de la información relevante. Los requerimientos de mayor importancia serán representados mediante botones con máxima prioridad, con un mayor tamaño e identificados con sus nombres.

Los requisitos secundarios tendrán prioridad media y serán representados con botones más pequeños identificados con los respectivos nombres. Los requerimientos de menor importancia para el subsistema implementado tendrán mínima prioridad y serán representados por botones pequeños que carecerán de sus nombres correspondientes. La Figura 14 muestra un ejemplo de la prioridad establecida para algunos componentes del menú de un subsistema.

Figura 14 Prioridad de los componentes del menú de un subsistema

Las categorías creadas deben estar de acuerdo con las necesidades del usuario

Las categorías que se definan estarán estrechamente relacionadas con el grupo donde se muestren dentro de la pestaña del subsistema. Cada nombre de grupo deberá reflejar las necesidades de los usuarios finales del Front-End de manera general.

Estructura de las pestañas

En la pestaña perteneciente al subsistema existen estructuras que soportan cambios de crecimiento. Estos componentes estarán diseñados e implementados de forma tal que si el equipo de desarrollo debiera agregar una nueva funcionalidad, contenida dentro de un componente en específico, solamente tendrían que agregarla al archivo XML que construye la pestaña y será visualizado el componente sin dificultades, con la nueva funcionalidad incluida. La Figura 15 muestra un ejemplo de estos componentes.

Figura 15 Componentes de una pestaña

Comportamiento del sistema ante la adición o supresión de funcionalidades

La navegación será asegurada a través de la utilización de un archivo XML que especifica las conformaciones del menú del subsistema. Si el equipo de desarrollo desea suprimir o adicionar una funcionalidad, solo debe eliminarla o agregarla respetando el esquema definido para el archivo XML. El sistema cargará el resto de los componentes sin afectaciones. La Figura 16 muestra un ejemplo de adición de una funcionalidad al menú de un subsistema.

Figura 16 Ejemplo de adición de una nueva funcionalidad al menú de un subsistema

Organización de los subsistemas cargados en el Front-End

Los subsistemas cargados en el Front-End serán organizados como se aprecia en la Figura 13. Además, en el manipulador de ventanas del Front-End se creará una pestaña principal cuyo título coincidirá con el nombre del subsistema, que contendrá el área de trabajo del subsistema cargado con la opción de cerrar desactivada. Es importante señalar que si un usuario tiene acceso a varios subsistemas, cuando seleccione la pestaña correspondiente a un subsistema en el Ribbon, la pestaña del manipulador de ventanas que incluya el área de trabajo del subsistema debe ser seleccionada.

Dentro del área de trabajo del subsistema, cada ventana que se desee mostrar debe abrirse en el manipulador de ventanas del subsistema. Es significativo acotar que si el usuario selecciona una funcionalidad cuya interfaz de usuario correspondiente ya se encuentra abierta, el sistema remplazará la existente por la nueva ventana.

3.3 Modelo de diseño

El modelo de diseño es un modelo de objetos que describe la realización física de los CU centrándose en cómo los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de implementación, tienen impacto en el sistema (26). Sirve de abstracción de la implementación de la aplicación y es, de esa manera, utilizado como una entrada fundamental de las actividades de implementación. A continuación se abordarán los temas relacionados con el diseño del Front-End.

3.3.1 Diagramas de clases

Un diagrama de clases es un tipo de diagrama estático que describe la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos (26). Son utilizados durante el proceso de análisis y diseño de los sistemas, donde se crea el diseño conceptual de la información que se manejará en el sistema, de manera conjunta con los componentes que se encargarán del funcionamiento y las relaciones entre unos y otros. En los siguientes epígrafes se muestran los diagramas de clases del Front-End.

3.3.1.1 Diagrama de clases del Front-End

A continuación se muestra, en la Figura 17, el diagrama de clases del diseño del Front-End, que por su complejidad y para un mejor entendimiento, se agrupó por paquetes.

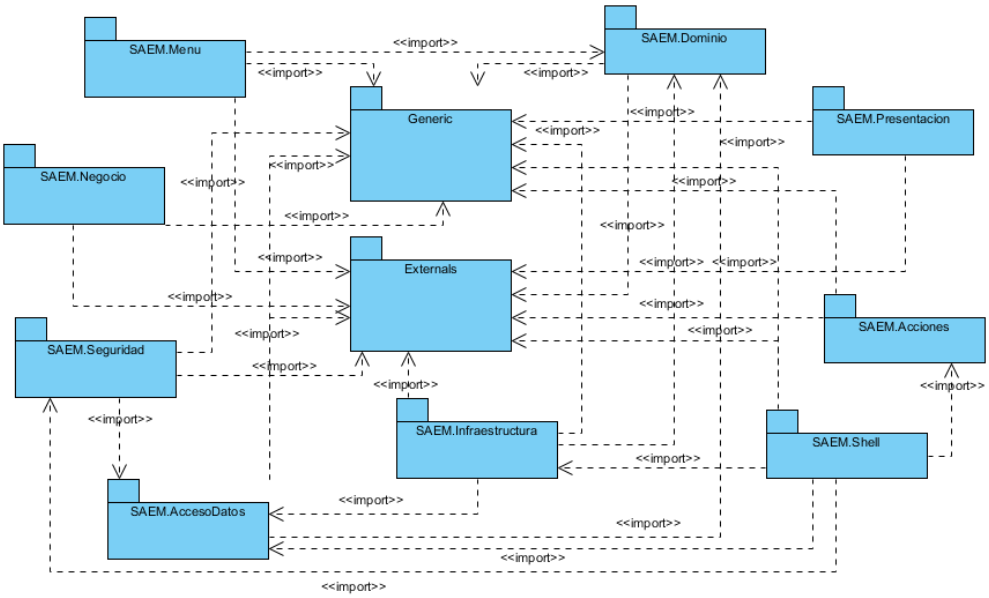

Figura 17 Diagrama de paquetes del Front-End para la aplicación SAEM

A continuación, para una mejor comprensión, se hará una breve explicación del mismo:

A través de la clase *SaemFtp*, contenida en el paquete Externals, el Front-End se conecta al servidor FTP para descargar las actualizaciones de los subsistemas de la aplicación, utilizando las credenciales de un usuario autenticado, validadas por la clase *ProveedorSeguridad*, agrupada dentro del paquete SAEM.Seguridad. Una vez descargadas las nuevas versiones de los subsistemas, las clases contenidas en los paquetes SAEM.Menu y SAEM.Seguridad validan los archivos XML y los ensamblados correspondientes a cada subsistema. Con la información obtenida, el paquete SAEM.Shell se encarga de la construcción de las regiones del menú y manejador de ventanas de la aplicación.

3.3.2 Descripción de clases del diseño

Una clase de diseño es una abstracción sin costuras de una clase o construcción similar en la implementación del sistema. Se utiliza un lenguaje equivalente al lenguaje de programación para especificarla, toda vez que las operaciones, parámetros, atributos, tipos y demás son especificados utilizando la sintaxis del lenguaje de programación elegido (26). A continuación, se describe en la Tabla 1 la clase *SaemFtp*, perteneciente al framework base del sistema.

3.3.2.1 Clase SaemFtp

Nombre de la Clase: SaemFtp

Atributos	**Tipo**
-UrlServer	string
-ServerUser	string
-UserPassword	string
Operaciones	**Descripción**
+GetSaemDirectoriesManifests(): void	Método principal que se encarga de preparar las condiciones para descargar los archivos del servidor Ftp, y ponerlos a disposición del sistema.
+DeleteSaemSubsystemsFiles(): void	Método que elimina la carpeta temporal SAEMTemporaryFolder y todo su contenido.
+DownLoadManifest(): void	Método que descarga un archivo manifiesto perteneciente a un subsistema del servidor

	FTP.
+ExistingSaemDirectory(): bool	Método que comprueba si existe algún directorio en el servidor FTP.
+ListFtpDirectoryFiles(): void	Método que construye una lista con los archivos existentes en un determinado directorio en el servidor FTP.
+DownloadFtpFiles(): void	Método que descarga todos los archivos existentes en un directorio en el servidor FTP.
+GetMd5HashFromFile(): string	Método que obtiene el valor hash de un comprimido descargado del servidor FTP.
+VerifySubsystemHash(): bool	Método que compara el valor hash de un comprimido, almacenado en el archivo manifiesto, con el obtenido cuando este es descargado del servidor FTP.
+CopyToSaemSystemFolder(): void	Método que copia los ensamblados y los archivos XML del subsistema en la carpeta que contiene los subsistemas de la aplicación.
+ExtractSaemSubsystemsFiles(): void	Método que descompacta los ensamblados y archivos XML del subsistema en la carpeta que almacena los subsistemas de la aplicación, sobrescribiendo la versión anterior del mismo.
+ListSaemFtpDirectories(): List<string>	Método que lista los directorios existentes en el servidor FTP.

Tabla 1 Descripción de la clase SaemFtp

3.3.3 Diagrama de despliegue

Un diagrama de despliegue muestra las relaciones físicas de los distintos nodos que componen un sistema. Un nodo constituye un recurso de ejecución que puede contener instancias de componentes de software, objetos o procesos (caso particular de un objeto).

De forma general, un diagrama de despliegue es un grafo de nodos unidos por conexiones de comunicación que muestra las relaciones físicas entre los componentes hardware y software en el sistema final, es decir, presenta cómo están configurados los elementos de procesamiento en tiempo de ejecución y los componentes software (33).

En la Figura 18 se muestra el diagrama de despliegue del Front-End para la aplicación SAEM:

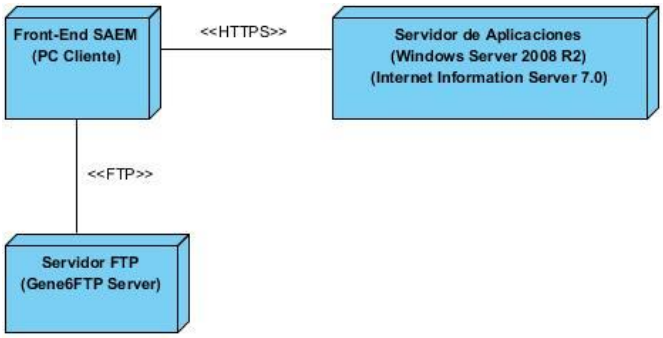

Figura 18 Diagrama de despliegue del Front-End para la aplicación SAEM

3.4 Conclusiones

En el presente capítulo se describió la representación arquitectónica del sistema propuesto. Se utilizó el patrón arquitectónico Modelo-Vista-Presentador para realizar la implementación de la aplicación. Se utilizaron los patrones de diseño Command, Facade, Bajo acoplamiento, Alta cohesión y Creador para diseñar el Front-End. Además se definieron los aspectos de arquitectura informacional y usabilidad que la aplicación debe poseer y por último se definió el diagrama de despliegue del sistema.

IMPLEMENTACIÓN DEL FRONT-END

4.1 Introducción

En este capítulo se describe la implementación del Front-End. Se modela el sistema en términos de componentes y se organiza de acuerdo a los nodos específicos que conforman el modelo de despliegue. Se definen además, las restricciones de nomenclatura y estándares de codificación a utilizar en la construcción de la solución. En el desarrollo del capítulo se incluye el diagrama de componentes.

4.2 Modelo de implementación

El modelo de implementación describe cómo los elementos del modelo de diseño, como las clases, se implementan en términos de componentes, como ficheros de código fuente, ejecutables, etc. Describe además cómo se organizan los componentes de acuerdo con los mecanismos de estructuración y modulación disponibles en el entorno de implementación y en el lenguaje o lenguajes de programación utilizados, y cómo dependen los componentes unos de otros (26).

4.2.1 Diagrama de componentes

El diagrama de componentes muestra las dependencias lógicas entre componentes de software. Describe la vista de implementación estática de cualquier sistema, sin importar tamaño o complejidad. Los componentes son unidades autónomas dentro de un sistema o subsistema, que representan tipos de elementos de software que forman parte del desarrollo de aplicaciones informáticas.

En la Figura 19 se muestra el diagrama de componentes del Front-End para la aplicación SAEM:

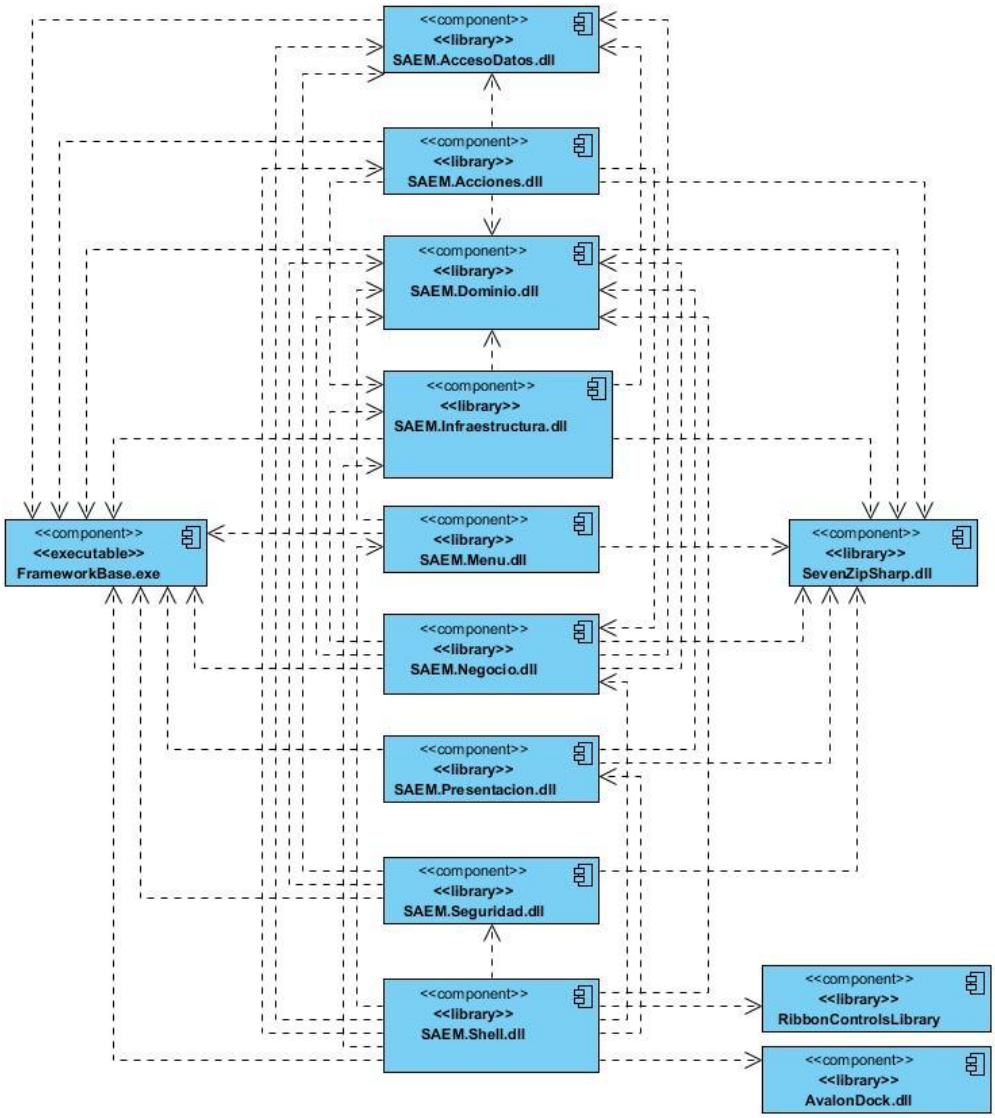

Figura 19 Diagrama de componentes del Front-End

A continuación se explicará brevemente la utilización de varios componentes para el desarrollo del Front-End:

El componente *SevenZipSharp.dll* es una librería escrita en lenguaje C# empleada para la compresión y descompresión de datos en aplicaciones desarrolladas .NET Framework 2.0 o superior. Contiene un conjunto de clases y objetos que permiten el manejo de los archivos para realizar operaciones hacia diferentes formatos, tales como .RAR, .ZIP, .TAR entre otros. Durante el proceso de actualización de los subsistemas en el Front-End, el componente *SAEM.Seguridad.dll* hace uso de la clase *SaemFtp,* perteneciente al componente *FrameworkBase.exe,* que utiliza el método *ExtractArchive()*, de la clase

SevenZipExtractor, para descomprimir el paquete de ensamblados perteneciente a la actualización del subsistema publicada en el servidor FTP.

RibbonControlsLibrary.dll es la librería propuesta por Microsoft para la creación de menús en aplicaciones de WPF. Contiene un paquete de clases y temas que conforman el control visual utilizado para construir el menú de la aplicación, que se encuentra en la interfaz principal del Front-End de la aplicación SAEM, concerniente al componente *SAEM.Shell.dll*.

El componente *AvalonDock.dll*, librería de código abierto implementado en el lenguaje C# para aplicaciones basadas en WPF y Windows Forms, es utilizado también por el módulo *SAEM.Shell.dll*. Este control visual consiste en un manejador de ventanas que permite crear regiones flotantes dentro de una interfaz, así como personalizar las posiciones de las mismas. Los subsistemas cargados en el Front-End de la aplicación SAEM, como se explicó anteriormente, son organizados por pestañas que se agrupan dentro del manejador de ventanas y contienen un conjunto de pestañas que responden a las funcionalidades de cada subsistema.

4.2.2 Trabajo con las librerías RibbonControlsLibrary y AvalonDock

Para el trabajo con las librerías RibbonControlsLibrary y AvalonDock fue necesaria la implementación de clases que permitieran aprovechar todas las funcionalidades que proporcionan.

4.2.2.1 Clase MainMenu

Como se definió con anterioridad, para la estructuración del menú general del sistema se utilizó el Ribbon de Microsoft versión 3.5. Este componente visual puede ser implementado de manera estática desde el diseñador de código XAML de una solución, o puede ser cargado dinámicamente desde un archivo XML. Para el desarrollo del presente trabajo se utilizó la segunda vía, obtener desde un archivo en formato XML y con una estructura XSD definida, los componentes necesarios para conformar el menú de cada subsistema de la aplicación. Para ello fue necesaria la implementación de una clase para parsear el archivo XML y adquirir la información necesaria sobre cada elemento almacenado en este. La clase *MainMenu* contiene métodos que permiten la creación de los componentes del Ribbon, dígase pestañas, grupos, controles, imágenes, etc.

4.2.2.2 Clase DockableRegionAdapter

Como se ha explicado con anterioridad, para la implementación de la arquitectura modular propuesta, el equipo de desarrollo del Front-End se basó en PRISM.

PRISM utiliza el concepto de “regiones” para dividir el espacio de la ventana de la aplicación, y en cada “región” se puede incrustar una o más vistas (objetos que se representan visualmente). Por su lado, AvalonDock se basa en proporcionar un contenedor especial, denominado DockingManager, dentro del cual se insertan otros contenedores especiales que almacenan el contenido a mostrar, que debe ser una instancia de dos clases especiales, denominadas DockableContent y DocumentContent.

La utilización de AvalonDock en conjunto con la guía PRISM requiere de la creación de un adaptador para conformar un contenedor especial del manejador de ventanas cuando se asocie una vista a una región. Este adaptador, de forma transparente, crea dicho contenedor especial, de tipo DockableContent o DocumentContent, y lo agrega al contenido del DocumentPane, que está mapeado en la región de PRISM. Esta guía provee el concepto de RegionAdapter, que como su nombre indica, es una clase que “adapta” los contenidos de una región de PRISM a un contenedor especial.

4.2.3 Restricciones de nomenclatura y estándares de codificación

Las técnicas y estándares de codificación incorporan muchos aspectos del desarrollo de software. Aunque generalmente no afectan a la funcionalidad de una aplicación, sí contribuyen a una mejor comprensión del código fuente. Para esto se tienen en cuenta los diferentes tipos de código fuente, como son los lenguajes de programación, secuencias de comandos, de marcado o de consultas (34).

Existen varios estándares para codificar de manera limpia y entendible durante el desarrollo de software, algunos de los más comunes han sido utilizados durante la implementación del Front-End del SAEM y se describen a continuación:

Notación Pascal

Consiste en escribir con mayúscula el primer carácter de cada palabra para los nombres de las clases y métodos (34).

Ejemplos: *SaemFtp*, *DownloadManifest()*

Notación Camel

El primer caracter de todas las palabras se escribe con mayúscula, exceptuando la primera palabra, el resto de las letras con minúscula. Esta notación se usa para los parámetros y las variables locales (35).

Ejemplos: *saemFtpUser*, *saemFtpPassword*

Además de las notaciones antes mencionadas, se han seguido una serie de buenas prácticas que ayudan a un mejor entendimiento del código empleado:

- Usar el prefijo “I” para nombrar las interfaces.
- Usar el prefijo “T” para estructuras de tipos de datos.
- El uso de palabras descriptivas y entendibles para nombrar las variables.
- Usar los prefijos “Es” o “Is” para las variables de tipo bool, ya que es más apropiado y se asemeja a los nombres de variables en .NET.
  Ejemplo: *IsValidFile*
- Los nombres de los espacios de nombre siguen el siguiente patrón <NombreDeProducto><MóduloSuperior><MóduloInferior>
  Ejemplo: *SAEM.Administracion.Acciones*
- Los nombres de los clases o métodos tienen siguen el patrón <Acción/Descripción>
- En la interfaz gráfica se nombraron los controles de forma uniforme para identificarlos rápidamente. En la Tabla 2 se describe la codificación de componentes visuales utilizados:

Control	**Prefijo**
Label	lbl
TextBox	txt
DataGrid	dtg
Button	btn
ListBox	lst
CheckBox	chk
RadioButton	rbt
Image	img

Tabla 2 Codificación de los controles visuales

4.3 Implementación de un subsistema para la aplicación SAEM

Para la implementación de un nuevo subsistema para el Front-End de la aplicación SAEM, es necesario cumplir los siguientes pasos:

1. Conformar dos archivos XML que cumplan con el XSD desarrollado por el grupo de arquitectura del Front-End (ver Ficheros XML). El primero se nombra "Menú.xml" y conforma el menú del subsistema. Contendrá datos como: nombre del subsistema, la acción principal del módulo de acciones perteneciente al subsistema, los componentes que formarán el menú y las acciones que realizan.

```
<MenuItem Priority="1" Text="Administración" Tipo="Tab"
            FontFamily="Calibri" FontSize="13"
            PrincipalAction="Administracion.CargarAdministracionPrincipal">
  <MenuItems>
    <MenuItem Priority="1" Text="Administrar" Tipo="Group"
            Width="800" FontFamily="Calibri" FontSize="13" >
      <MenuItems>
        <MenuItem Priority="1" Text="Generar Manifiesto"
          Tipo="Command" Width="32" FontFamily="Trebuchet MS"
          ImageSize="Large" ImgSource="Operadora\Llamada_Asociada.png"
          FontSize="12"
          ActionId="Administracion.CargarGenerarManifiesto"/>
        <MenuItem Priority="1" Text="Asignar Permisos" Tipo="Command"
          Width="32" FontFamily="Trebuchet MS" ImageSize="Large"
          ImgSource="Operadora\Llamada_Asociada.png" FontSize="12"
          ActionId="Administracion.CargarAsignarPermisos"/>
      </MenuItems>
    </MenuItem>
  </MenuItems>
</MenuItem>
```

El segundo, nombrado "Subsistemas.xml", contiene información acerca de la dirección física de cada ensamblado perteneciente al subsistema y sus dependencias. Por último, para adicionar nuevos componentes estos archivos, debe respetarse la jerarquía que se presenta en el XSD.

```
<subistema nombre="Administracion">
  <modulo tipo="SAEM.Administracion.AccesoDatos.InicializadorAccesoDatos,
        SAEM.Administracion.AccesoDatos"
        nombre="SAEM.Administracion.AccesoDatos"
        ensamblado =
        "Subsistemas/Administracion/SAEM.Administracion.AccesoDatos.dll"/>
  <modulo tipo="SAEM.Administracion.Negocio.InicializadorNegocio,
        SAEM.Administracion.Negocio" nombre="SAEM.Administracion.Negocio"
        ensamblado =
        "Subsistemas/Administracion/SAEM.Administracion.Negocio.dll">
    <dependencias>
      <dependencia modulo="SAEM.Administracion.AccesoDatos"
        subsistema="Administracion" />
    </dependencias>
  </modulo>
  <modulo tipo="SAEM.Administracion.Acciones.InicializadorAcciones,
        SAEM.Administracion.Acciones" nombre =
```

```
18.          "SAEM.Administracion.Acciones"
19.          ensamblado =
20.          "Subsistemas/Administracion/SAEM.Administracion.Acciones.dll">
21.    </modulo>
22.    <modulo tipo="SAEM.Administracion.Presentacion.InicializadorPresentacion,
23.          SAEM.Administracion.Presentacion" nombre =
24.          "SAEM.Administracion.Presentacion" ensamblado =
25.          "Subsistemas/Administracion/SAEM.Administracion.Presentacion.dll">
26.    </modulo>
27. </subistema>
```

2. Implementar el subsistema estrictamente con la estructura de paquetes definida por el equipo de desarrollo responsable de la arquitectura del sistema. Una vez que sea generado, se obtienen las librerías correspondientes y se almacenan junto a los archivos XML correspondientes en un comprimido cuyo nombre coincidirá con el nombre del subsistema, que se entregará al administrador de la aplicación para ser publicado en el servidor FTP del sistema.

En la Figura 20 se observa la estructura que muestra cómo deben agruparse las clases para implementar un nuevo subsistema. Un ejemplo de los elementos que componen un comprimido perteneciente a un subsistema se presenta en la Figura 21.

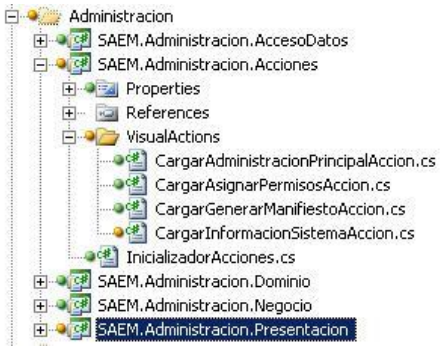

Figura 20 Ejemplo de la agrupación de clases de un subsistema

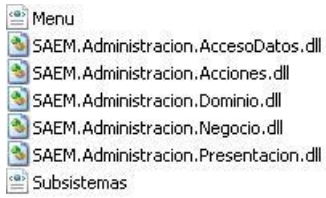

Figura 21 Ejemplo de estructura de un comprimido

3. El archivo manifiesto, que será publicado junto al comprimido correspondiente en el servidor FTP, contendrá información acerca de la versión del subsistema, la firma hash del comprimido, el nombre del desarrollador responsable del subsistema y la fecha en que fue publicado el comprimido en el servidor FTP.

```
1. <VersionConfig>
```

```
2.    <AvailableVersion>1.0.0.2</AvailableVersion>
3.    <SubSystemHash>C1-BA-7C-CE-A8-23-43-6F-CC-83-A8-BF-F5-53-10-
4.                   FB</SubSystemHash>
5.    <Author>Rolando Lenzano Rodriguez</Author>
6.    <Date>22/05/2012</Date>
7.  </VersionConfig>
```

4.3.1 Estructura de los directorios del servidor FTP

El sistema de carpetas del servidor FTP estará estructurado de la siguiente manera:

- La carpeta Subsistemas será la carpeta root del servidor FTP y contendrá los directorios propios de los módulos de la aplicación.
- El nombre de cada directorio coincidirá con el nombre del subsistema al que corresponde.
- El usuario autenticado en el sistema tendrá una cuenta en el servidor FTP con permisos de solo lectura sobre cada directorio al que tiene acceso.
- El usuario administrador es el único con acceso a todos los directorios del servidor FTP.
- El administrador del servidor FTP deberá confirmar que al crear una nueva cuenta de usuario, esta posea solamente los permisos requeridos.

La Figura 22 muestra un ejemplo de un subsistema publicado en el directorio correspondiente.

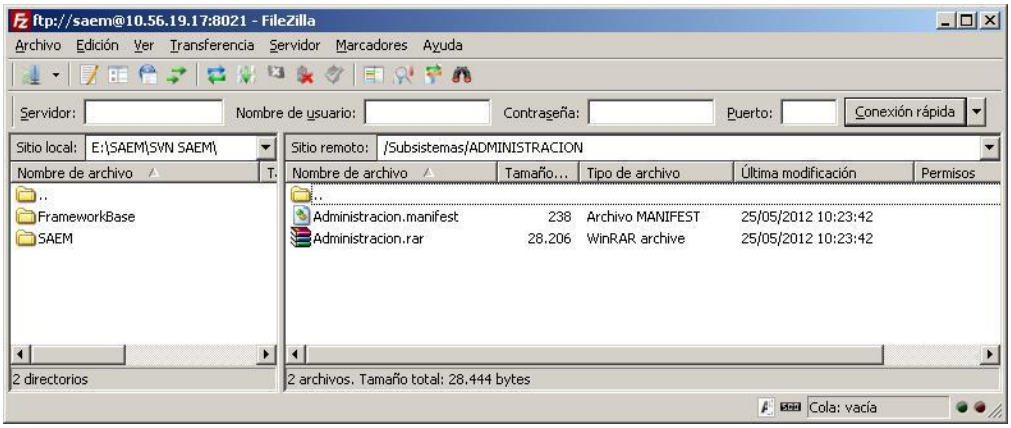

Figura 22 Ejemplo de subsistema publicado en el servidor FTP

4.4 Prototipo funcional del Front-End

La Figura 23 muestra un prototipo funcional del Front-End una vez que se le ha adicionado un subsistema.

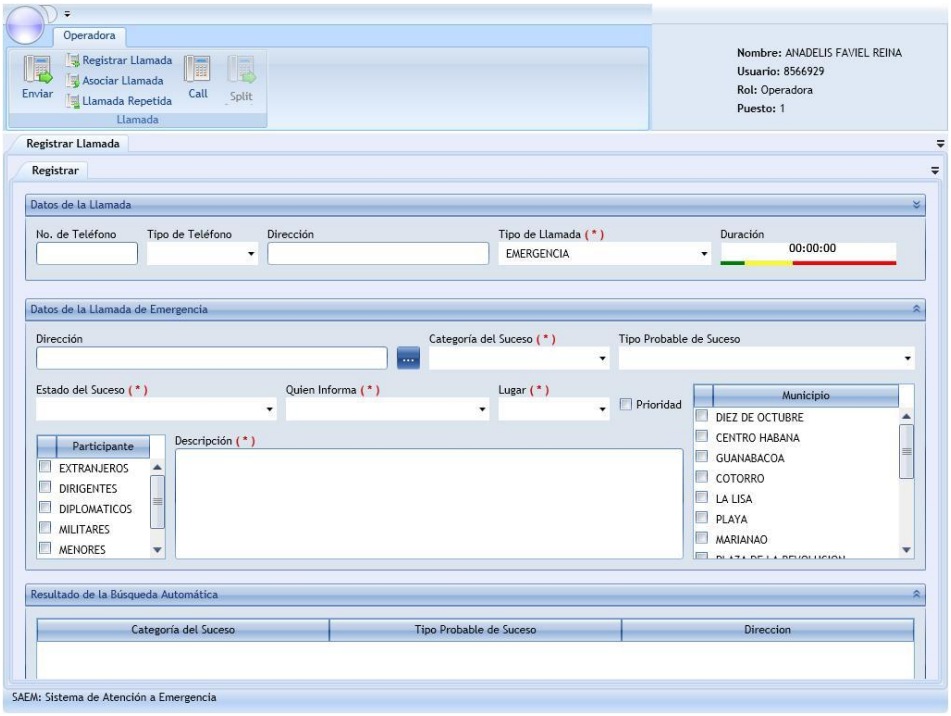

Figura 23 Subsistema Operadora adicionado al Front-End para la aplicación SAEM

4.5 Conclusiones

En el presente capítulo se presentó la implementación exitosa de la solución Front-End para la aplicación SAEM. Se definieron y elaboraron los pasos necesarios para implementar un nuevo subsistema para la aplicación, los mismos servirán de guía a los desarrolladores de los subsistemas. Además, se presentó la estructura de carpetas del servidor FTP del sistema y las restricciones de seguridad necesarias que fueron implementadas. Por último se muestra el diagrama de componentes y los prototipos funcionales del Front-End.

PRUEBA DEL FRONT-END

5.1 Introducción

En este capítulo se realizará la validación de los requisitos del Front-End, como resultado del establecimiento de una línea base por la que se rige el equipo de calidad para realizarle pruebas al sistema en un determinado tiempo.

5.2 Pruebas

En el flujo de trabajo de pruebas se verifican los resultados de la implementación probando la construcción, tanto interna como intermedia, así como las versiones finales del sistemas a ser entregadas a terceros (26).

Las pruebas de software consisten en un proceso que, una vez generado el código fuente de una aplicación, se lleva a cabo para descubrir la mayor cantidad de errores antes de entregarlo al cliente (36), y que por ende se obtenga la mayor calidad posible. La eficiencia de una prueba viene dada por los siguientes aspectos:

- Las pruebas deben tener una elevada probabilidad de encontrar errores. Para esto se requiere que la persona que aplica la prueba comprenda el software y trate de desarrollar una imagen mental de la manera en que puede fallar.
- Las pruebas no deben ser redundante, de manera que ninguna prueba debe tener el mismo propósito que otra.
- Las pruebas no deben ser ni muy simples ni demasiado complejas.

Aunque no hay una clasificación oficial acerca de los diversos tipos de pruebas de software, existen dos vertientes fundamentales. Si se conoce el funcionamiento interno de la aplicación, se llevan a cabo pruebas que demuestren que las operaciones internas funcionen de acuerdo con los requerimientos, denominadas pruebas de Caja Blanca. Por otra parte, si se conoce la función específica para la que se diseñó el producto, se realizan pruebas de Caja Negra, que demuestran que cada función es plenamente operacional.

5.2.1 Pruebas de caja negra

Se refiere a las pruebas que se llevan a cabo sobre la interfaz de un software, por lo que se realizan casos de pruebas que pretendan demostrar que las funciones del software son

operativas; la entrada se acepta de forma adecuada, se produce una salida correcta y la integridad de la información externa se mantiene. Las pruebas de caja negra examinan el modelo, fundamentalmente de sistema, sin tener mucho en cuenta la estructura interna de software (37). Quien realiza las pruebas sólo conoce las entradas apropiadas que deberá recibir la aplicación, así como las correspondientes salidas, sin llegar a conocer cómo se realiza este proceso.

5.3 Diseño de casos de prueba

Un caso de prueba (CP) especifica una forma de probar el sistema, incluyendo la entrada o resultado con la que se ha de probar y las condiciones bajo las que ha de probarse. Suele describir la funcionalidad que se probará, las acciones que se realizan, así como los valores esperados y los resultados obtenidos (26).

A continuación se muestra el CP que corresponde al CU Actualizar subsistema:

CP **Actualizar subsistema**

Descripción general

El CP se inicia cuando el sistema obtiene el listado de manifiestos de los subsistemas que se encuentran en uso y mediante una conexión al servidor FTP verifica que existen versiones superiores de los mismos. Concluye cuando los módulos son actualizados.

Condiciones de ejecución

Deben existir en el servidor FTP versiones superiores de los módulos actualmente en uso por el usuario autenticado.

El usuario debe estar autenticado en la aplicación.

SC Actualizar subsistema

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1. Actualizar subsistema.	Se actualizan los subsistemas a los que tiene permiso el usuario autenticado, ubicados en la carpeta local que contiene los subsistemas de la aplicación.	Descarga para la carpeta SubSystems, contenida en la carpeta temporal SAEMTemporaryFolder, el comprimido y archivo manifiesto correspondiente a cada	Autenticarse en la aplicación.

		subsistema al que tiene acceso el usuario autenticado. Descompacta los comprimidos y copia los archivos manifiesto de los subsistemas en la carpeta local que contiene los subsistemas de la aplicación.	
EC 1.2. No se obtienen usuario y contraseña.	No se obtiene de la BD el usuario y contraseña, del usuario autenticado.	Muestra el mensaje de información: “No existe conexión con la BD”.	
EC 1.3. No existe la carpeta temporal SAEMTemporary Folder.	Se verifica que no existan la carpeta temporal y las subcarpetas contenidas, donde se descargan los comprimidos y archivos manifiesto pertenecientes a los subsistemas a los que tiene acceso el usuario autenticado.	Crea la carpeta temporal SAEMTemporaryFolder, con las subcarpetas Manifest y SubSystems, en la carpeta especial Mis Documentos.	
EC 1.4. No establece una conexión con el servidor FTP.	No se establece una conexión con el servidor FTP para obtener un listado de los directorios a los que tiene permiso el usuario autenticado.	Muestra los siguientes mensajes de error: “No es posible conectar con el servidor remoto” “Error 503: Not login”	
EC 1.5. No contiene una versión de los	Se verifica que no existen en la carpeta local que contiene los	Elimina los archivos manifiestos descargados temporalmente y salva	

archivos manifiestos descargados en la carpeta local que contiene los subsistemas de la aplicación.	subsistemas de la aplicación, versiones de los subsistemas a los que tiene permiso el usuario autenticado.	para la carpeta SubSystems, contenida en la carpeta temporal SAEMTemporaryFolder, los comprimidos y archivos manifiesto correspondientes a los subsistemas a los que tiene acceso el usuario autenticado.	
EC 1.6. El archivo manifiesto descargado no contiene una versión superior.	Se verifica que los archivos manifiesto descargados no contengan versiones superiores de los subsistemas a los que tiene acceso el usuario autenticado y que están contenidos en la carpeta local que almacena los subsistemas de la aplicación.	Elimina los archivos manifiestos descargados y verifica que el usuario autenticado no tiene acceso a más directorios en el servidor FTP.	
EC 1.7. El usuario tiene acceso a más de un directorio en el servidor FTP.	Se verifica si el usuario autenticado tiene acceso a varios directorios en el servidor FTP.	Establece una conexión con cada directorio al que tiene acceso el usuario autenticado.	
EC 1.8. La firma hash del comprimido descargado no concuerda con la almacenada en	Se verifica la integridad de los comprimidos descargados que pertenecen a los subsistemas a los que el usuario autenticado tiene	Verifica que no existan otros comprimidos descargados.	

el archivo manifiesto.	acceso.		
EC 1.9. Existen otros comprimidos descargados.	Se verifica si existen varios comprimidos descargados.	Obtiene las firma hash de los comprimidos descargados correspondientes a los subsistemas a los que tiene acceso el usuario autenticado.	

Tabla 3 Diseño de CP correspondiente al CU Actualizar subsistema

Descripción de las variables

No aplica.

5.3.1 Evaluación de los casos de prueba

Se probaron los CP diseñados para comprobar en qué medida se cumplen los requerimientos funcionales definidos. Los CP que se tuvieron en cuenta son los siguientes:

- **Autenticar usuario**: se comprueba que se autentique un usuario en el sistema, que este lo certifique en el servidor de comunicación de la aplicación y que se muestren los datos del usuario en la región de datos del usuario, una vez inicializado el sistema. De un total de once posibles no conformidades, el CP arrojó una no conformidad real.
- **Actualizar subsistema:** se comprueba que el sistema actualice automáticamente los subsistemas locales a los que tiene acceso el usuario autenticado, descargando las nuevas versiones de estos del servidor FTP de la aplicación. De un total de trece posibles no conformidades, el CP arrojó dos no conformidades detectadas.
- **Adicionar subsistema:** se comprueba que el sistema adicione una versión de cada subsistema a los que el usuario tenga acceso al Front-End. De un total de nueve posibles no conformidades, el CP arrojó una no conformidad detectada.
- **Actualizar Front-End:** se comprueba que los archivos pertenecientes al Front-End del sistema que sean modificados, se actualicen automáticamente mediante la

tecnología ClickOnce. De un total de nueve posibles no conformidades, el CP no arrojó alguna no conformidad.

- **Publicar subsistemas**: se comprueba que las nuevas versiones de los subsistemas sean publicadas en el servidor FTP de la aplicación. De un total de siete posibles no conformidades, el CP arrojó una no conformidad real.

5.3.1.1 Resultados de los casos de prueba

Total	Con peso crítico	Total de posibles no conformidades	Total real de no conformidades
4	4	49	5

Criterios de evaluación para los resultados de los casos de prueba

Positivos: 72% - 100%

Con problemas: 36% - 71%

Negativos: 0 - 35%

De un total de 49 posibles no conformidades, después de realizadas las pruebas se detectó un 10.2% de no conformidades reales, lo cual ubica al sistema dentro del rango de aceptación.

5.4 Conclusiones

En el presente capítulo se describió el proceso de realización de pruebas al Front-End implementado. Para ello, se seleccionaron las pruebas de caja negra, que permitieron el estudio de las entradas y salidas o respuestas que produjo el sistema sin necesidad de conocer su funcionamiento interno. Se confeccionó y aprobó un plan de prueba, que incluyó la preparación minuciosa de un entorno de pruebas adecuado, en aras de probar el sistema en condiciones adecuadas. Por último, se realizaron los diseños y pruebas de CP correspondientes a cada CU, cuyos resultados obtenidos ubican al Front-End implementado dentro de un rango de aceptación.

CONCLUSIONES

Con el desarrollo del presente trabajo y el cumplimiento de las tareas propuestas se plantean las siguientes conclusiones:

1. Se desarrolló una nueva arquitectura modular para el Front-End de la aplicación SAEM, que permite la adición y actualización de sus subsistemas dinámicamente, logrando aumentar funcionalidades al sistema.
2. Se implementó un mecanismo de actualización para el Front-End, que garantizará las actualizaciones automáticas de los subsistemas y del propio Front-End.
3. Se desarrolló un Front-End dinámico, encargado de la portabilidad de los subsistemas de la aplicación SAEM, que garantiza:
    - Realizar la carga de los subsistemas a través de archivos XML.
    - Ganar en organización del área de trabajo controlando los componentes visuales desde un mismo manipulador de ventana.
4. Se probó la arquitectura desarrollada a través de pruebas unitarias de caja negra, detectándose solamente 5 no conformidades en la primera iteración, hecho que demuestra la calidad de la aplicación desarrollada.

FUTUROS TRABAJOS

Por la experiencia adquirida durante el desarrollo de la presente investigación, los autores recomiendan:

1. Implementar un mecanismo de rollback, necesario para que la aplicación mantenga su funcionamiento en caso de que las actualizaciones publicadas contengan errores.
2. Definir un manual de usuarios para el Front-End que brinde información sobre las instrucciones necesarias para que los usuarios puedan utilizar las funcionalidades que ofrece.
3. Trabajar en la optimización de los componentes de WPF utilizados en el desarrollo del Front-End con el objetivo de mejorar el rendimiento del mismo.

BIBLIOGRAFÍA

1. **Pedraza García, Gilberto.** *Evolución e Integración de Aplicaciones Legadas: Comenzar de Nuevo o Actualizar?* Programa de Ingeniería de Sistemas, Universidad Piloto de Colombia. Bogotá : s.n., 2008. pág. 1.

2. **Symantec Corporation.** Norton. *Norton by Symantec.* [En línea] Symantec, 12 de Junio de 2011. [Citado el: 15 de Marzo de 2012.] http://es.norton.com/vital-security/article.

3. *Sitio web de Norton by Symantec.* [En línea] 12 de junio de 2011. [Citado el: 3 de mayo de 2012.] http://es.norton.com/vital-security/article.

4. *Sitio Web del Instituto Nacional de Tecnologías Educativas y de Formación del Profesorado.* [En línea] Secretaría de Estado de Educación, Formación Profesional y Universidades. [Citado el: 30 de abril de 2012.] http://sauce.pntic.mec.es/crer0052/paquetes/definicion.htm.

5. *Soporte Microsoft.* [En línea] 13.0, 14 de mayo de 2010. [Citado el: 5 de mayo de 2012.] http://support.microsoft.com/kb/824684/es.

6. **Salmerón Rubio, José Jaime.** *Control de Software.* Nuevo León : s.n., 2001. págs. 12-13.

7. *Agnitum.* [En línea] [Citado el: 12 de mayo de 2012.] ontinet.com.

8. **F. Zapico, Antonio.** *Tatum.* [En línea] [Citado el: 12 de mayo de 2012.] http://www.tatum.es.

9. **Cepeda, Fausto.** *Seguridad Descifrada.* [En línea] 27 de febrero de 2008. [Citado el: 12 de mayo de 2012.] http://seguridaddescifrada.blogspot.com/2008/02/la-importancia-de-actualizar.html.

10. *Vera's Soul.* [En línea] 2 de marzo de 2010. [Citado el: 12 de mayo de 2012.] http://www.verasoul.com/2010/02/por-qu-he-de-actualizar-mi-software.html.

11. **Kaspersky Lab.** *Kaspersky Lab.* [En línea] 14 de septiembre de 2009. [Citado el: 25 de mayo de 2012.] http://utils.kaspersky.com/docs/english/remote_adm/kasp8.0_ak_admguideen.pdf.

12. **Microsoft Corporation.** *MSDN.* [En línea] [Citado el: 28 de abril de 2012.] http://msdn.microsoft.com/es-es/library/142dbbz4(v=vs.80).aspx.

13. —. *MSDN.* [En línea] [Citado el: 27 de abril de 2012.] http://msdn.microsoft.com/es-es/library/s22azw1e(v=vs.80).aspx.

14. —. *MSDN.* [En línea] [Citado el: 29 de mayo de 2012.] http://msdn.microsoft.com/es-es/library/e2444w33(v=vs.80).

15. —. *MSDN.* [En línea] [Citado el: 28 de abril de 2012.] http://msdn.microsoft.com/es-es/library/71baz9ah(v=vs.80).aspx.

16. —. *MSDN.* [En línea] [Citado el: 28 de mayo de 2012.] http://msdn.microsoft.com/es-es/library/h2zwd6bw(v=vs.90).aspx.

17. **Instituto Nacional de Tecnologías Educativas y de Formación del Profesorado.** *Intef.* [En línea] [Citado el: 1 de junio de 2012.] http://www.ite.educacion.es/formacion/materiales/85/cd/REDES_W2000/pdf/11.pdf.

18. *Definición.de.* [En línea] 2008. [Citado el: 6 de mayo de 2012.] http://definicion.de/archivo/.

19. *Definición ABC.* [En línea] 2007. [Citado el: 6 de mayo de 2012.] http://www.definicionabc.com/historia/archivo.php.

20. *MASTERMAGAZINE.* [En línea] 2004. [Citado el: 6 de mayo de 2012.] http://www.mastermagazine.info/termino/6939.php.

21. **Quinodóz, Carolina.** *Sitio web de Profesora Carolina Quinodóz.* [En línea] 2012. [Citado el: 25 de abril de 2012.] http://www.profecarolinaquinodoz.com/alumnos/colegio/protocoloftp.pdf.

22. **Universidad de Málaga.** *NEO Networking and Emerging Optimization.* [En línea] 2008. [Citado el: 26 de abril de 2012.] http://neo.lcc.uma.es/evirtual/cdd/tutorial/aplicacion/ftp.html.

23. **Horat Flotats, David Jesús y Fernández Perdomo, Enrique.** *David Horas.* [En línea] [Citado el: 25 de abril de 2012.] http://es.davidhorat.com/publicaciones/descarga/redes-ftp.pdf.

24. **Gene6 SARL.** *Gene6FTP Server v3 User's Guide.*

25. *Zator Systems.* [En línea] 2012. [Citado el: 29 de abril de 2012.] http://www.zator.com.

26. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James.** *El proceso unificado de desarrollo de software.* [trad.] Salvador Sánchez, Miguel Angel Sicilia y Francisco Javier Durán. Madrid : Pearson Educación S.A., 2000. ISBN: 84-7829-036-2.

27. **Visual Paradigm International.** *UML, BPMN and Database for Software Development.* [En línea] http://www.visual-paradigm.com/product/vpuml/.

28. **Microsoft Corporation.** *MSDN.* [En línea] [Citado el: 10 de enero de 2012.] http://msdn.microsoft.com/es-es/netframework/aa496123.

29. —. *Inicio de .NET Framework.* [En línea] 2012. [Citado el: 13 de enero de 2012.] http://msdn.microsoft.com/es-es/netframework/aa496123.

30. *MSDN.* [En línea] [Citado el: 24 de Febrero de 2012.] http://msdn.microsoft.com/es-es/library/fx6bk1f4(v=vs.80).

31. **Camacho, Erika, Cardeso, Fabio y Nuñez, Gabriel.** *Arquitecturas de Software.* 2004. págs. 6-8.

32. **Wurman, Richard Saul.** *Information Architects.* Los Angeles : Watson-Guptill Pubis, 1997. págs. 10-11.

33. **Marca Huallpara, Hugo Michael y Quisbert Limachi, Nancy Susana.** *ANALISIS Y DISEÑO DE SISTEMAS II.*

34. **Microsoft Corporation.** Técnicas de codificación. *MSDN.* [En línea] 2012. [Citado el: 23 de Mayo de 2012.] http://msdn.microsoft.com/es-es/library/aa291593(v=vs.71).

35. **Fernández, Alberto y Ribal, Josep.** *Estándares de Codificación en C# y Buenas Prácticas de Programación.*

36. **S. Pressman, Roger.** Estrategias de Pruebas de Software. [aut. libro] Roger Pressman. *Ingeniería de Software.* 13, págs. 382-385.

37. Curso de Ingenieria de Software 2. *Entorno Virtual de Aprendizaje.* [En línea] 2012. [Citado el: 23 de Mayo de 2012.] http://eva.uci.cu/file.php/158/Documentos/Recursos_bibliograficos/Libros_y_articulos_UD_2/Comun/Material_de_caja_b_y_caja_n.pdf.

38. *Definición.de.* [En línea] [Citado el: 30 de abril de 2012.] http://www.definicion.de/aplicacion/.

39. **Lanzillota, Analía.** *MASTERMAGAZINE.* [En línea] [Citado el: 30 de abril de 2012.] http://www.mastermagazine.info/termino/3874.php.

40. **Santamaría Martínez, Kaory.** *Aplicaciones Web.* [En línea] 4 de febrero de 2011. [Citado el: 30 de abril de 2012.] http://kaory-sm.blogspot.com/2011/02/unidad-1-panorama-general-de-las.html.

41. **Pattinson, Ted.** *Programming Distributed Applications with COM+ and Visual Basic 6.0.* Segunda. ISBN 0-7356-1010-X.

42. **Jansen, Slinger y Ballintijn, Gerco.** *A Process Model and Topology for Software Product Updaters.* s.l. : The College of Information Sciences and Technology, 2007-2010.

43. *MSDN.* [En línea] [Citado el: 10 de abril de 2012.] http://msdn.microsoft.com/es-es/library/s22azw1e(v=vs.80).aspx.

44. *Definición ABC.* [En línea] 2007. [Citado el: 8 de mayo de 2012.] http://www.definicionabc.com/tecnologia/transferencia.php.

45. *Sitio web de Java.* [En línea] [Citado el: 16 de mayo de 2012.] http://www.java.com/es/download/faq/java_webstart.xml.

46. **Pérez Mariñán, Martín.** *javaHispano.* [En línea] [Citado el: 17 de mayo de 2012.] http://www.javahispano.org/storage/contenidos/jws2.pdf.

47. **Microsoft Corporation.** *MSDN.* [En línea] [Citado el: 28 de abril de 2012.] http://msdn.microsoft.com/es-es/library/76e4d2xw.aspx.

48. **González Seco, José Antonio.** *El lenguaje de programación C#.* págs. 10-23.

49. *Definición ABC.* [En línea] [Citado el: 30 de abril de 2012.] www.definicionabc.com/tecnologia/proveedor.php.

50. *Definición ABC.* [En línea] [Citado el: 30 de abril de 2012.] www.definicionabc.com/general/cliente.php.