UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA
Escuela de Ingeniería Informática

Trabajo de Fin de Grado

Aplicación para la configuración automática del certificado digital en un ordenador personal

Julio 2014 – Las Palmas de Gran Canaria

Alumno: Ravi Raj Khubchandani Khubchandani
Grado en Ingeniería Informática (Sistemas de Información)

Tutores: Dr. Francisco Javier Carreras Riudavets
Departamento de Informática y Sistemas

Ángel Sánchez de la Cruz
Oficina Técnica de la Administración Electrónica

Índice

Índice de ilustraciones

Introducción

En los tiempos modernos los equipos informáticos vienen configurados de fábrica para que el usuario pueda disfrutar de su producto desde el momento en que lo adquiere, pero existen situaciones en que se necesita cumplir con una serie requerimientos. Esta situación es común en empresas, donde muchos trabajadores necesitan sus ordenadores configurados de cierta manera.

Gracias a las nuevas tecnologías se han desarrollado elementos software que detecten el problema y lo solucionen automáticamente, adquiriendo archivos desde la red en caso necesario. Todo ello con una intervención mínima del usuario. Estas tareas se podrían hacer de forma manual, pero consumiría mayor cantidad de tiempo y por tanto, se resta productividad.

Este proyecto se ha creado como una solución a un problema real encontrado en la administración de la Universidad de Las Palmas de Gran Canaria (en adelante ULPGC), con el cual se pretende facilitar y automatizar el proceso de configuración de un ordenador personal de forma que un miembro de la comunidad universitaria pueda identificarse ante la página web institucional usando su certificado electrónico emitido por la Fábrica Nacional de Moneda y Timbre.

Estado actual y objetivos

Hoy en día existe diversidad de servicios tanto privados como públicos que pueden ser usados de forma telemática, éstos son totalmente seguros y aportan gran comodidad al usuario. Dependiendo del tipo de servicio que se ofrece se requiere de unos componentes u otros que tienen que ser instalados en el equipo del usuario. Si se usa más de un servicio, el hecho de tener que cumplir distintos requisitos puede causar rechazo a los mismos y optar por medios más tradicionales.

Situación actual

Con la idea de automatizar la configuración de un equipo para un fin concreto, distintos programas han sido creados, ejemplos de estos son el configurador de redes inalámbricas para acceder a universidades europeas Eduroam y el configurador de navegadores web de la Agencia Tributaria de España.

Ambos están disponibles como programas que se pueden descargar desde sus respectivas páginas web, la ejecución de estos programas modifica los archivos del equipo de la forma en que sea necesaria y de forma automática en cuestión de minutos para que el usuario siga con sus tareas.

Actualmente la configuración del funcionamiento de la firma electrónica de la ULPGC se realiza manualmente, no existe ninguna solución que automatice este proceso. En caso de problemas se contacta con la oficina técnica de la administración electrónica.

Objetivos

El objetivo de este proyecto es desarrollar un software que instale y configure automáticamente los componentes necesarios para que la firma electrónica funcione correctamente en al menos un sistema operativo y un navegador web concretos.

Se ha elegido el sistema operativo Windows 7 y navegador web Internet Explorer 11. Para tomar esta decisión se han tenido en cuenta las siguientes consideraciones:

- Microsoft ya no da soporte a Windows XP, por lo que en poco tiempo la administración debería migrar a otro sistema operativo.
- Internet Explorer se posiciona entre los navegadores más usados entre el personal de la administración.

Se ha ideado desde cero un sistema de configuración automática de dicho proceso pero que a la vez es flexible y escalable a otras necesidades. La idea es que el usuario sin ayuda técnica, usando el programa tenga listo su equipo con un mínimo esfuerzo y aproveche al máximo su tiempo.

Más adelante vemos en profundidad el funcionamiento del primer prototipo, pero los principales objetivos del proyecto son:

- Identificar los requerimientos del equipo en base al sistema operativo y navegadores web usados.
- Tras identificar las necesidades, el usuario puede elegir los navegadores web que quiere configurar para usar la firma electrónica.
- Instalación de los componentes, todos los componentes o cada uno individualmente bajo petición del usuario.
- Mantenimiento de un archivo de log para monitorizar todo cambio realizado para poder revertir el equipo al estado anterior.
- Desinstalar componentes instalados por el propio programa.

El sistema es escalable a otras necesidades que puedan aparecer en el futuro. Esto es así ya que los requerimientos para cada sistema operativo y navegador se establece mediante un archivo de configuración, de forma transparente al usuario. Esto permite que actualizando este archivo, el programa siga siendo útil sin realizar modificaciones.

La identificación mediante certificado electrónico es más segura que mediante usuario y contraseña. Además

permite a los miembros de la comunidad universitaria realizar tareas administrativas a través de la red.

Este proyecto ha permitido poner en práctica el conocimiento sobre diseño e ingeniería de software adquiridos en la carrera. En el desarrollo del proyecto se ha seguido la metodología ágil Scrum y Lean Software Development, se ha ido desarrollando de forma progresiva, manteniendo reuniones periódicas para la supervisión del tutor.

En cada reunión se ha mostrado una versión del programa cada vez más completa hasta llegar a una primera versión prototipo que cumple con los requisitos establecidos.

Además me ha dado la oportunidad de desarrollar el aprendizaje autónomo así como un conocimiento en cierta profundidad sobre las distintas versiones del sistema operativo Windows y su sistema de registros.

Competencias cubiertas

Trabajo Fin de Grado

TFG01

Ejercicio original a realizar individualmente y presentar y defender ante un tribunal universitario, consistente en un proyecto en el ámbito de las tecnologías específicas de la Ingeniería en Informática de naturaleza profesional en el que se sinteticen e integren las competencias adquiridas en las enseñanzas.

Este proyecto se diseñó desde cero, el desarrollo se ha llevado a cabo de forma modular aplicando arquitecturas de software y diseño de patrones estudiados en asignaturas de la carrera. Se ha elegido desarrollarlo usando herramientas modernas y de uso extendido. Al ser modular, al proyecto se pueden agregar componentes nuevos que podrían incrementar su funcionalidad a la vez que componentes de este proyecto se pueden usar en otros. Al usar herramientas modernas y de uso extendido se evita que se convierta en software obsoleto a la vez que la documentación disponible es amplia.

Comunes a la Ingeniería Informática

CII08

Capacidad para analizar, diseñar, construir y mantener aplicaciones de forma robusta, segura y eficiente, eligiendo el paradigma y los lenguajes de programación más adecuados.

Antes de iniciar el desarrollo de la aplicación se investigó distintos sistemas y lenguajes de programación disponibles para asegurarse de que la opción tomada es la correcta. Se planteó la posibilidad de que el proyecto fuese una aplicación web pero dada la naturaleza del software y sus requerimientos esta posibilidad no se pudo realizar y se optó por el lenguaje C# usando el framework .NET 4.5 desarrollado por Microsoft.

Gran parte del tiempo disponible para el desarrollo se ha invertido en la correcta programación de patrones de diseño de forma que el mantenimiento y ampliación del programa se pueda realizar de la forma más eficiente posible. Se ha optado por una arquitectura MVC por su sencilla implementación pero siendo a la vez eficiente y su capacidad para cumplir con los requisitos del programa, también se han usado más patrones de diseño para los componentes del programa.

CII010

Conocimiento de las características, funcionalidades y estructura de los Sistemas Operativos y diseñar e Implementar aplicaciones basadas en sus servicios.

El proyecto se ha desarrollado para sistemas Windows, dada la naturaleza del proyecto se ha realizado una importante investigación sobre su sistema de registros, las diferentes opciones que dispone en su sistema de

ficheros para archivos temporales, archivos de opciones personales individuales para cada usuario y sus variables de entorno. Un motivo por el cual se opta por usar C# y .NET es que este framework está desarrollado por Microsoft y facilita y agiliza el desarrollo de aplicaciones que necesiten conocimiento del hardware y sistema operativo del equipo en que se ejecuta.

CII012

Conocimiento y aplicación de las características, funcionalidades y estructura de las bases de datos, que permitan su adecuado uso, y el diseño y el análisis e implementación de aplicaciones basadas en ellos.

El proyecto ha permitido conocer en profundidad el funcionamiento, estructuración y uso del registro de Windows, el cual es una base de datos jerárquica así como su integración para ser accedida desde programas externos para su lectura y escritura.

CII014

Conocimiento y aplicación de los principios fundamentales y técnicas básicas de la programación paralela, concurrente, distribuida y de tiempo real.

El programa desarrollado se ejecuta en diferentes hilos de ejecución concurrentes que se comunican entre ellos. Se ha usado técnicas y estructuras propias de .NET para la programación de los mismos ya que permite la ejecución paralela de varios hilos sin que el trabajo de un hilo afecte al de otro.

CII016

Conocimiento y aplicación de los principios, metodologías y ciclos de vida de la ingeniería de software.

Se han seguido metodologías de desarrollo ágil durante el proceso de desarrollo, de forma que pequeños cambios no han supuesto grandes cambios en el proyecto. Se han mantenido reuniones periódicas con los tutores, como propone la metodología Scrum, en cada reunión se ha mostrado una versión cada vez más completa del programa. Al implementar un requisito se hacía de forma que el programa respondiese de la forma esperada y luego se refactorizaba para obtener una versión mejorada, como establece la metodología Lean Software Development.

En el desarrollo del programa se han usado los patrones de diseño más convenientes para cada componente, como el patrón 'Command' para programar la acción de la interfaz; 'SRP' o Single Responsibility Principle, de forma que se ha procurado que cada clase tenga un solo propósito y sea independiente del resto; y así otros patrones generales como el 'Principio de inversión de dependencias', clases 'Singleton' o 'Factory'.

Todo ello modularizado de forma que se respete el principio 'DRY' o Don't Repeat Yourself para evitar repetición de código.

CII017

Capacidad para diseñar y evaluar interfaces persona computador que garanticen la accesibilidad y usabilidad a los sistemas, servicios y aplicaciones informáticas.

Para el funcionamiento de este programa se necesita de un archivo de configuración, su formato se ha diseñado de forma que permita insertar la mayor cantidad de información de la forma más cómoda posible. Este archivo se verá en detalle más adelante.

La interfaz de usuario también se ha diseñado de forma que su uso sea intuitivo al usuario final, reduciendo el número de ventanas en la medida de lo posible. Más adelante veremos que la primera versión de la interfaz fue modificada de forma que resulte más familiar al estilo que el usuario está acostumbrado.

Ingeniería del software

IS01

Capacidad para desarrollar, mantener y evaluar servicios y sistemas software que satisfagan todos los requisitos del usuario y se comporten de forma fiable y eficiente, sean asequibles de desarrollar y mantener y cumplan normas de calidad, aplicando las teorías, principios, métodos y prácticas de la ingeniería del software.

En el desarrollo del proyecto se ha llevado a cabo una detallada planificación de todos sus aspectos funcionales. Se han puesto en práctica las diferentes etapas del ciclo de vida de un software. Desde la especificación de requisitos hasta su testeo en distintos equipos. Se ha puesto énfasis en control del estado de los datos en cada momento para evitar fallos en el programa, de esta forma es más robusto y fiable.

Se han empleado buenas prácticas de programación, se ha documentado toda acción realizada, se han realizado prototipos de la interfaz, el código es modular y legible aunque se han añadido algunos comentarios en el código en donde es necesario.

Se ha desarrollado aplicando arquitecturas y patrones de diseño ya consolidados, de esta forma es más fácil de mantener y realizar cambios cuando sea necesario.

IS05

Capacidad de identificar, evaluar y gestionar los riesgos potenciales asociados que pudieran presentarse.

El mayor riesgo para el proyecto era la variación en el tipo de requisitos que tenía el programa. Algunos requisitos consumieron más tiempo por el objetivo que tenían ya que requerían un conocimiento avanzado, se tuvo que investigar la solución al problema o la documentación para resolver el problema es escasa. Por este motivo se llegó a un consenso en los requisitos a cumplir en el tiempo disponible.

Sistemas de Información

SI01

Capacidad de integrar soluciones de Tecnologías de la información y las comunicaciones y procesos empresariales para satisfacer las necesidades de información de las organizaciones, permitiéndoles alcanzar sus objetivos de forma efectiva y eficiente, dándoles así ventajas competitivas.

El programa está pensado para su uso en red. Esto es así ya que el programa se comunica con los servidores remotos de la ULPGC y obtener archivos alojados en el mismo. El software genera archivos de log para cada usuario, en un principio estos logs estaban pensados para ser alojados también en los servidores de la ULPGC, por ello debían tener identificadores únicos, esto se logró usando la herramienta 'WMI', Windows Management Instrumentation, como veremos más adelante. En una de las iteraciones del software se decide que el log se aloja en el propio equipo del usuario pero la implementación ya está realizada, por lo que si en el futuro se decide alojar estos logs en la nube, el cambio no será muy costoso y la competencia y el conocimiento para ello ya han sido adquiridos.

Aportaciones

Aportación al medio socio-económico

El uso de esta aplicación aporta a sus usuarios la comodidad de automatizar el proceso de configuración de la firma electrónica en sus ordenadores. Afecta de manera económica ya que el propio usuario puede configurar con éxito su propio equipo sin ayuda y en menos tiempo. Por este motivo tanto el usuario como el personal técnico pueden emplear su tiempo en otras tareas.

Si el programa cumple con el nivel de calidad exigido, se pondrá a disposición de la comunidad universitaria.

Aportación al medio técnico

El proyecto está estructurado de forma que se pueden sustituir o añadir módulos, se pueden integrar nuevas funcionalidades si se desease.

Se introducen datos en el programa mediante un archivo de configuración. En este archivo se establecen todos los requerimientos del equipo para que funcione la firma electrónica. Cambiar este archivo supondría que el usuario tiene a su disposición las actualizaciones que hubiesen.

Por estos dos motivos se alarga la vida útil del software a bajo coste.

Normativa y legislación

Ley Orgánica de Protección de Datos (LOPD)

La Ley Orgánica 15/1999 de 13 de diciembre de Protección de Datos de Carácter Personal tiene el objetivo de garantizar y proteger las libertades públicas y derechos fundamentales de las personas físicas en lo concerniente al tratamiento y comunicación de sus datos personales independientemente del soporte usado.

Esta ley afecta a todos los datos que hacen referencia a personas físicas registradas sobre cualquier tipo de soporte (informático o no), aunque están excluidos datos recogidos para uso doméstico y material clasificado por el estado que tratan sobre delincuencia organizada y terrorismo.

Este proyecto trata datos del equipo local y almacena un archivo de log en el mismo, pero en ningún caso se envía información a servidores remotos ni almacena información personal del usuario, por lo que no es necesario cifrarlo.

Licencias del software usado

En la realización de este proyecto se ha usado tanto software libre como propietario, cada uno con sus respectivas licencias. A continuación vemos cada una de ellas:

Software propietario

Se trata de software que puede ser gratuito o no, pero se necesita de permiso para poder usarlo para ciertos fines y no otros. Estas pueden ser su modificación, su distribución o copia. El software propietario usado en el proyecto se ha conseguido por medio de la suscripción de la ULPGC al programa MSDN de Microsoft.

Éstos son:

- Microsoft Visual Studio 2012 Premium
- Microsoft Windows 7 Professional
- Microsoft Visio 2013

Licencia general de GNU

La licencia general de GNU o GNU GPL es la licencia más usada en el mundo del software. Garantiza a los usuarios la libertad de usar, estudiar, copiar, distribuir y modificar el software. Software usado con esta licencia GNU GPL es:

- Gimp 2

Licencia Apache

Es una licencia de software libre que permite que el software sea usado, estudiado, copiado, modificado y distribuido por el usuario, pero si se crea software a partir de otro software con esta licencia, hay que mostrar al público una nota que lo comunique. Software usado con esta licencia es:

- OpenOffice 4.0

Requisitos

La lista de requisitos contiene todas las funcionalidades que el software desarrollado debe cumplir. Estas funcionalidades no son finales ni invariables. Puede darse el caso de que un requisito se sustituya por otro o se añadan requisitos nuevos para funcionalidades nuevas.

Inicialmente hubo una reunión con los tutores en la que se decidió una lista inicial de requisitos funcionales. A partir de esta funcionalidad básica del programa se derivan requisitos no funcionales. Junto a estos requisitos aparecen otros, ideas de tutores o personales. Algunos se implementaron y otros no, atendiendo al tiempo disponible. Las ideas propias fueron consultadas a los tutores antes de ser realizadas.

Analizando las necesidades surgidas se analizaron los diferentes recursos disponibles para elegir las herramientas que serían de mayor utilidad para cumplir con el objetivo.

En las reuniones de cada iteración del proyecto hubo entrevistas abiertas para consultar dudas e identificar modificaciones a realizar mientras se testeaba el software.

Es en este tipo de reuniones donde más datos se han sacado en claro, sobre todo a la hora de diseñar la interfaz de usuario. Lo mejor, a la hora de diseñar la interfaz de usuario, es que lo pruebe una persona ajena al proyecto, que pueda ser un futuro usuario. Por este motivo el tutor Ángel Sánchez proporcionó una copia del ejecutable del prototipo a una compañera de la administración de la ULPGC.

Este feedback es muy valioso en proyectos software ya que permite saber si el trabajo realizado se ha hecho bien o se necesita modificar algún aspecto.

Planificación y estructuración del proyecto

Para el buen desarrollo del proyecto se ha hecho uso de las ideas y del marco de trabajo que proponen las metodologías ágiles. Concretamente se han seguido recomendaciones de la metodología Scrum y de Lean Software Development. No se han seguido formal y estrictamente sino se han adaptado a las necesidades del proyecto para su correcta evolución.

Siguiendo la metodología Scrum se han hecho varias iteraciones, en cada iteración se lograba incluir funcionalidades nuevas, también se marcaron directrices para realizar correcciones de iteraciones anteriores.

Aunque la primera reunión se realizó tras un mes de comenzar el proyecto, el resto se celebraron cada dos semanas aproximadamente. Scrum recomienda realizar una reunión diaria, este tipo de reuniones no se llevaron a cabo por cuestiones de tiempo y disponibilidad.

Las iteraciones han beneficiado al proyecto ya que han permitido ver el avance progresivo del mismo y facilitado la realización de cambios.

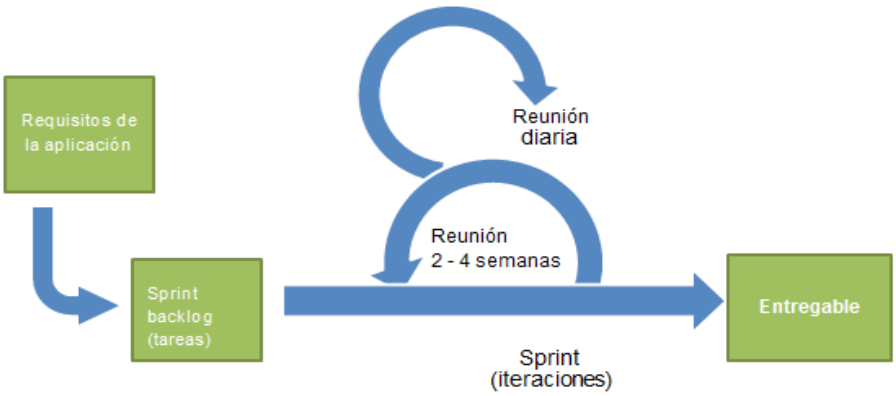


Ilustración 1: Marco de trabajo de Scrum

Siguiendo la metodología Lean Software Development, la funcionalidad de los módulos se ha limitado, de forma que se cumplen los requerimientos exigidos pero no incluyen funcionalidad que no se usa en el proyecto. Al incorporar funcionalidad para cumplir los requerimientos, primero se ha implementado de forma que el software realice su tarea correctamente y luego se ha refactorizado para obtener una versión mejorada.

Los requisitos se han logrado de acuerdo a, primero, su nivel prioridad ya que son imprescindibles para la siguiente fase del desarrollo y segundo, a su nivel de dificultad.

Tecnologías, herramientas y metodologías aplicadas

Tecnologías y herramientas

C#

Es un lenguaje de programación orientado a objetos desarrollado por Microsoft y estandarizado por ECMA (ECMA-334) e ISO (ISO/IEC 23270). Su desarrollo comenzó en el año 2000 y la versión actual es la 5.0. Deriva de C++ incluyendo mejoras derivadas de otros lenguajes, como Java.

La ventaja de usar C# para un software de escritorio para Windows es que podemos usar el framework .NET. C# es una buena alternativa frente a los otros lenguajes de .NET. Es mejor alternativa a J# porque éste está en desuso desde 2008 y, acorde a la página web de Microsoft (link disponible en la sección de fuentes de información), es una alternativa igual de válida que usar Visual Basic.NET.

.NET

Es un framework de Microsoft que permite un rápido desarrollo de aplicaciones ya que es aplicable tanto a software de escritorio, móvil y servicios web. Se lanzó en el año 2002 y su versión actual es 4.5. Está estandarizado por ECMA (ECMA-335) e ISO(ISO/IEC 23271). Incorpora potentes bibliotecas como funciones criptográficas, conexión a bases de datos ADO.NET, conexión a servidores remotos empleando los protocolos HTTP, HTTPS y FTP, herramientas de acceso a bajos niveles del sistema operativo con WMI y programación dirigida por eventos, entre otros.

UML

El Lenguaje Unificado de Modelado, UML por sus siglas en inglés, es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. Está estandarizado por ISO (ISO/IEC 19501:2005). Su versión actual es la 2.5. Es muy útil en los sistemas software, como en este caso, para describir la interacción entre las clases que componen el mismo.

WPF

Windows Presentation Foundation es una tecnología que permite el desarrollo de interfaces de usuario en Windows. Ofrece una mejora visual frente a Windows Forms pero el verdadero potencial de WPF radica en su sintaxis para crear interfaces, XAML, y la incorporación de elementos multimedia avanzados.

XAML

XAML es un lenguaje para crear interfaces de usuario para Windows y web en Microsoft Silverlight. Tiene una sintaxis inspirada en XML.

WMI

Windows Management Instrumentation es una herramienta desarrollada por Microsoft que fue creada para

tener acceso a bajos niveles del sistema operativo como detalles acerca de la CPU y el disco duro. Fue creado con el propósito de facilitar la realización de tareas administrativas técnicas en sistemas Windows remotos, pero también puede ser usado en equipos locales para obtener datos que de otra forma podrían ser más complejos de obtener.

Microsoft Visual Studio 2012 Premium

Es un Entorno de Desarrollo Itegrado, IDE por sus siglas en inglés, para Windows. Permite el desarrollo de software y servicios web en varios lenguajes, como C, C++, C#, Visual Basic, Java, ASP y ASP.NET. Fue lanzado por primera vez en 1997 y la versión actual es Visual Studio 2013. Tiene integración directa con el framework .NET aunque se pueden instalar otros frameworks y plugins disponibles en el equipo local y desde la red.

Algunas de sus características más notables es que tiene integración con programas de control de versiones y que permite ver la interfaz de usuario mientras se esta desarrollando, en pantalla dividida junto a su código equivalente.

Regedit

Es un programa incluido con los sistemas operativos de Windows, permite ver, modificar y crear contenido de la base de datos del registro de Windows.

En el registro de Windows se almacena información sobre configuración de los distintos programas instalados en el equipo, así como preferencias personales configuradas por los usuarios del sistema y otra información de bajo nivel sobre el hardware instalado y el propio sistema operativo. En esta base de datos jerárquica, existen varias ramas con distintos propósitos. Aunque existen más, las más usadas para el propósito que nos ocupa son:

- HKEY_LOCAL_MACHINE o HKLM: almacena configuraciones específicas del equipo local, como el hardware y el software instalado.
- HKEY_CURRENT_USER o HKCU: almacena configuraciones específicas para el usuario que tiene sesión iniciada en cada momento.
- HKEY_CLASSES_ROOT o HKCR: almacena información básica como asociación de extensiones de archivos y los programas con los que se usan por defecto, iconos asociados con extensiones, entre otros.
- HKEY_USERS o HKU: almacena información relacionada con el perfil del usuario que ha iniciado sesión, es un subárbol que forma parte de HKEY_CURRENT_USER.

Es necesario tener el permiso adecuado para realizar cada una de estas acciones en cada rama (y/o parte de ésta) del registro.

Cabe destacar que existen subárboles en versiones de 64 bits del sistema operativo que no están disponibles en sus respectivas versiones de 32 bits. Estos subárboles permiten la compatibilidad de software diseñado para sistemas de 32 bits en equipos de 64 pero no al contrario. Veremos ejemplos más adelante sobre el correcto tratamiento de éstas para evitar problemas.

Metodologías

En este proyecto se ha puesto en práctica dos métodos de desarrollo ágil de software. Estos son métodos caracterizados por el desarrollo creciente de un software, basados en iteraciones del proceso de desarrollo hasta lograr una versión que cumpla con los objetivos establecidos.

Scrum

Fue desarrollado en Japón de la década de 1980 hasta 1990. Es un modelo que define un conjunto de prácticas y roles que pueden tomarse como punto de partida para definir el proceso de desarrollo que se ejecutará durante un proyecto. Los roles principales son el 'Scrum Master' o 'Scrum Manager', que mantiene

los procesos y trabaja de forma similar al director de proyecto, el 'Product Owner', que representa a los interesados, y el Team, que es el equipo de desarrolladores.

Durante cada 'sprint', o iteración, tiempo entre una y cuatro semanas (es definido por el equipo) es el tiempo de desarrollo entre una reunión y la siguiente, el equipo desarrolla una versión de software potencialmente utilizable. El conjunto de características que forma parte de cada iteración viene del 'Product Backlog', que es una lista de requisitos priorizados que definen el trabajo a realizar.

Los requisitos a desarrollar durante una iteración se determinan durante la reunión anterior, 'Sprint Planning'. Durante esta reunión, el 'Product Owner' identifica los elementos del 'Product Backlog' que quiere ver completados y lo comunica al equipo. Entonces, el equipo determina la cantidad de ese trabajo que puede comprometerse a completar durante el siguiente sprint. Durante el sprint no se puede cambiar el 'Sprint Backlog'.

Un principio clave de Scrum es el reconocimiento de que durante un proyecto los clientes pueden cambiar de idea sobre lo que quieren y necesitan, por este motivo y por el buen desarrollo del proyecto en general, la comunicación entre desarrolladores entre sí y comunicación con el cliente es vital.

Lean Software Development

La metodología Lean fue desarrollada por Toyota pero fue adaptada a la ingeniería de software por Mary y Tom Popendieck. Se basa en siete principios:

- Eliminar los desperdicios: todo lo que no añade valor para que el proyecto sea más valioso al cliente, es un desperdicio. Esto incluye funcionalidad innecesaria, retraso en el proceso de desarrollo, requisitos poco claros, burocracia y comunicación interna lenta.
- Ampliar el aprendizaje: se refiere al proceso de generar nuevo conocimiento que aparece en un proyecto para poderlo aplicar en otro proyecto donde pueda ser necesario.
- Retrasar la toma de decisiones: no tomar decisiones basadas en suposiciones en caso en que un requisito sea poco claro para evitar errores que tienen que ser soluciones más tarde.
- Reaccionar tan rápido como sea posible: se refiere al proceso de desarrollo del software, comunicación al cliente y obtención de feedback para volver al desarrollo, si se hace sin retrasos el proyecto se realiza con mayor éxito.
- Potenciar el equipo: la toma de decisiones en equipo se incrementa, no es el directivo quien da órdenes para realizar un trabajo.
- Fomentar la integridad: desarrollar los diferentes componentes que formen parte del sistema simultáneamente en lugar de forma secuencial, de esta manera se desarrolla software de mayor calidad y mejor adaptado a la necesidad del cliente. Se evita creación de funcionalidad innecesaria.
- Ver el sistema en su conjunto: el software no es la suma de sus partes, sino también el producto de sus interacciones. Los defectos tienden a acumularse durante el proceso de desarrollo, éstas deben ser encontradas y eliminadas.

Principios, mecanismos y patrones de diseño usados

SRP

Single Responsibility Principle, según este principio un módulo debe tener una única función. De esta forma cada clase tiene una funcionalidad limitada y concreta, es preferible tener mayor cantidad de módulos con menor cantidad de código que menos módulos menos manejables. También aumenta la posibilidad de que estos módulos sean reutilizados en otro proyecto.

OCP

Open-Closed Principle, según este principio un módulo debe estar cerrado a cambios pero abierto a extensiones. Una excepción a esto es que puede estar abierto a corrección de errores. De esta forma módulos

escritos en el pasado son compatibles con nuevos proyectos mediante la inversión de dependencias.

Principio de Liskov

Este principio establece que si una clase B es heredera de A entonces el programa pueda seguir funcionando si las referencias a la clase A se cambia por B.

Inyección de dependencias

Según este principio, es preferible insertar dependencias en el código cuando una clase tenga que referencias a otra. Por ejemplo si la clase A tiene que hacer referencia a una funcionalidad de B, en lugar de instancias un objeto de B en A se debe insertar un objeto de B en A, como propiedad en A de B o parámetro de algún método.

Inversión de dependencias

Establece que si en un módulo C y D son clases herederas de B y A necesita hacer referencia a éstas, el módulo debe estar diseñado de tal forma que A pueda hacer referencia a B sin modificar el funcionamiento del programa.

MVC

Modelo – Vista – Controlador es un patrón de arquitectura de software de uso extendido que separa los datos y la lógica de negocio (modelo) de la interfaz de usuario (vista) del módulo encargado de gestionar los eventos a controlar (controlador) en una aplicación.

- Modelo: es la representación lógica interna con la que el programa opera. Formada por datos que representan la situación que se está tratando.
- Controlador: es el código encargado de recoger los datos del usuario a través de la vista, procesar los mismos y devolver el resultado de nuevo a la vista para que se muestren al usuario.
- Vista: es el componente encargado de mostrar al usuario los datos procesados por el controlador del programa.

De esta forma el usuario interacciona con el programa a través de la vista (interfaz de usuario), las acciones realizadas por este son recogidas por el controlador, quien procesa los datos recogidos haciendo uso del modelo. Tras procesar éstos, se envían datos actualizados a la interfaz de usuario.

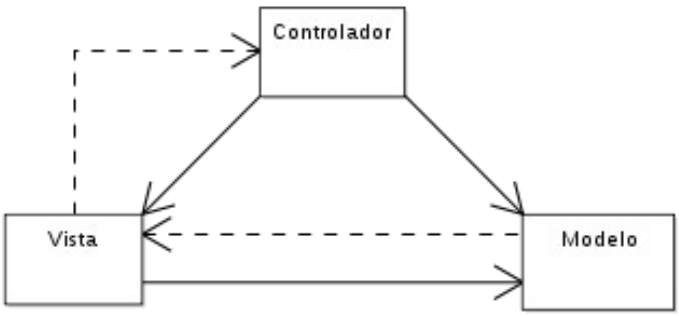


Ilustración 2: Marco de trabajo de la arquitectura MVC

Clases Singleton

Patrón de diseño de clases que permite que en cada momento exista una única instancia de esta clase.

Command

El patrón Command permite diseñar clases que encapsulan las acciones que se deben ejecutar como respuesta a un evento ocurrido en la interfaz de usuario.

Análisis y diseño

En este apartado se explica la funcionalidad de los módulos desarrollados, así como su integración y objetivo con el cuál fue ideado. Junto a cada módulo que se explique se muestra su diagrama UML parcial, el diagrama UML completo está disponible en el anexo.

Por último veremos cómo funcionan los módulos en conjunto para resolver un caso de uso de ejemplo.

Archivo de configuración

El configurador de firma electrónica se basa en dos archivos de configuración. Son archivos de texto que puede ser editado usando cualquier editor de texto compatible con la codificación UTF-8, como por ejemplo Notepad en Windows.

Al primero lo llamaremos 'archivo de configuración principal' y es el mismo para todos los usuarios. Para ser usado será descargado desde el servidor de la ULPGC por el configurador. El segundo es el archivo de log, es único por cada usuario, lo vemos más adelante en este apartado.

Su formato ha sido diseñado específicamente para este proyecto. Su estructura es sencilla, permite insertar líneas vacías y escribir comentarios. Está estructurado por bloques, existen cinco bloques definidos por sus etiquetas de apertura y cierre.

En estos bloques se especifican grupos de instrucciones, éstas tienen funciones diferentes, como leer un registro o un archivo y compararlo con un valor, escribir un valor en un registro o archivo o ejecutar un proceso nuevo en el equipo. Los comentarios comienzan con el símbolo // y abarcan una línea del archivo. Todas las instrucciones y sintaxis relacionadas con el archivo se describen en siguientes apartados.

En este documento se va a referir como 'sistema soportado' como un sistema que cumple ciertas características y para el cuál sabemos los requerimientos que debe cumplir para que la firma electrónica funcione correctamente.

Por cuestiones de compatibilidad de la configuración de la firma electrónica con diferentes sistemas operativos de Microsoft y diferentes navegadores disponibles para el mismo, los requerimientos para éstos pueden variar.

El primer bloque, el bloque 'NEEDS', comienza con la etiqueta de apertura #NEEDS y se cierra con #/NEEDS. Este bloque contiene grupos de instrucciones que sirven para identificar un sistema soportado, al mismo se le asigna una lista de requerimientos que el equipo debe cumplir. Existe un grupo de instrucciones por cada uno.

Vemos un ejemplo:

```
#NEEDS
  %CertificadosDigitales::Capicom::LectorPDF::Java::IE11_SafeSite::IE11_VistaCompatible::IE
    REGREAD::STRINGZ::HKEY_LOCAL_MACHINE/SOFTWARE/Microsoft/Windows NT/CurrentVersion/CurrentVersion::6.1
    REGREAD::STRINGZ::HKEY_LOCAL_MACHINE/SOFTWARE/Microsoft/Windows NT/CurrentVersion/ProductName::Windows 7 *
    REGREAD::STRINGZ::HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Internet Explorer\Version::9.11.*
#/NEEDS
```

Ilustración 3: Ejemplo de sistema soportado

En este ejemplo se leen tres valores del registro de Windows, se comprueba que la versión de Windows instalada en el equipo en que se ejecuta el configurador es la 6.1, el nombre de la versión comienza con Windows 7, '*' indica que la edición de Windows 7 es indiferente. Por último se comprueba que el equipo tiene instalado el navegador web Internet Explorer 11.

Si estas tres condiciones se cumplen, en el equipo deben estar instalados los requerimientos indicados por

'CertificadosDigitales', 'Capicom', 'LectorPDF', 'Java', 'IE_SafeSite' y 'IE11_VistaCompatible' para que la firma electrónica funcione correctamente en Internet Explorer 11 (indicado por IE).

Ya tenemos identificados los requerimientos que debe cumplir el sistema, ahora hay que especificar cómo se tiene que comprobar que estos requerimientos se cumplen o no.

La especificación de estas comprobaciones se realizan en el segundo bloque, llamado CONFIG, comienza con la etiqueta #CONFIG y se cierra con #/CONFIG. Este bloque contiene grupos de instrucciones con las que se puede determinar si el equipo cumple con los requerimientos identificados en el bloque NEEDS. Existe un grupo de instrucciones por cada requerimiento necesario.

Vemos un ejemplo:

```
#CONFIG
  %Java::SUCCEED_ONE::Java::Maquina virtual de Java::¿Desea instalar Java?
    REGREAD::STRINGZ::HKEY_LOCAL_MACHINE\SOFTWARE\JavaSoft\Java Runtime Environment\Java7FamilyVersion:: *.7.0_55
    REGREAD::STRINGZ::HKEY_CURRENT_USER\Software\JavaSoft\Java Runtime Environment\Java7FamilyVersion:: *.7.0_55
    REGREAD::STRINGZ::HKEY_LOCAL_MACHINE\SOFTWARE\JavaSoft\Java Runtime Environment\Java7FamilyVersion:: *.7.0_60
    REGREAD::STRINGZ::HKEY_CURRENT_USER\Software\JavaSoft\Java Runtime Environment\Java7FamilyVersion:: *.7.0_60
    REGREAD::STRINGZ::HKEY_LOCAL_MACHINE\SOFTWARE\JavaSoft\Java Runtime Environment\Java8FamilyVersion:: *.8.0_05
    REGREAD::STRINGZ::HKEY_CURRENT_USER\Software\JavaSoft\Java Runtime Environment\Java8FamilyVersion:: *.8.0_05
#/CONFIG
```

Ilustración 4: Ejemplo de comprobación de requerimiento

En este ejemplo se muestra cómo averiguar si la máquina virtual Java está instalada, el formato completo de la cabecera está explicada en el siguiente apartado, pero podemos ver que se indica 'SUCCEED_ONE', esto significa que para considerar que Java está instalado, como mínimo una de las instrucciones debe ejecutarse con éxito.

En este caso, sabiendo que la versión mínima compatible con la firma electrónica de la ULPGC es Java 7 update 55 y que la versión actual es Java 8 update 5, se considera que se cumple el requerimiento 'Java' si el equipo tiene instalado Java 7 update 55, Java 7 update 60 o Java 8 update 5.

En el bloque CONFIG también tienen que estar presentes las comprobaciones a realizar para verificar el cumplimiento del resto de requerimientos identificados.

Con los dos bloques presentados podemos comprobar qué requerimientos debe cumplir un equipo y cómo saber si éstos se cumplen. En el bloque ONFAIL, que comienza con la etiqueta #ONFAIL y termina con #/ONFAIL se especifica al programa, mediante grupos de instrucciones, las acciones que se deben realizar para instalar cada requerimiento si se ha determinado que este no se cumple.

Vemos un ejemplo:

```
#ONFAIL
  %Java::SUCCEED_ALL
    COPY::http://e-administracion.ulpgc.es/sites/default/files/java7_55.exe::java.exe
    THREAD::%COMSPEC%::/C "%CONFIG_TEMP%\java.exe"
#/ONFAIL
```

Ilustración 5: Ejemplo de instalación de un requerimiento

En este ejemplo vemos las acciones a realizar si el requerimiento 'Java' no está instalado, en este bloque se definen los grupos de instrucciones para el resto de requisitos. Primero se copia al equipo local el ejecutable del programa java7_55.exe desde el servidor de la ULPGC y luego se lanza su ejecución.

Uno de los requisitos funcionales de este proyecto es que el software debe ser capaz de deshacer los cambios realizados en el equipo. Para cumplir con este objetivo se introduce el bloque UNDO, comenzado y terminado por las etiquetas #UNDO y #/UNDO, respectivamente.

Vemos un ejemplo:

```
#UNDO
  %Java::SUCCEED_ONE::Java
    DELETE::SOFTWARE::HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall,
    HKEY_LOCAL_MACHINE/SOFTWARE/Wow6432Node/Microsoft/Windows/CurrentVersion/Uninstall::
    DisplayName::Java 7 Update 55::UninstallString
#/UNDO
```

Ilustración 6: Ejemplo de desinstalación de un requerimiento

En este ejemplo vemos las acciones a realizar para desinstalar el requerimiento 'Java', puede haber más grupos que indiquen cómo deshacer otros requerimientos.

En el bloque UNDO del archivo de configuración principal se define cómo desinstalar requerimientos pero teniendo en cuenta que estas acciones se realizan de la misma forma en todos los equipos, es decir, para desinstalar el requisito siempre se siguen las mismas instrucciones, independientemente del equipo.

Se genera un archivo de log individual para cada usuario. Este archivo tiene el mismo formato que el archivo de configuración principal pero contiene el bloque UNDO y el bloque INSTALLED que vemos a continuación.

Es necesario dividir el bloque UNDO ya que hay situaciones en las que un mismo requerimiento no se desinstala de la misma forma para dos equipos distintos.

Este tipo de situaciones se da, por ejemplo, al escribir un valor en un registro. Si este ya existe, hay que guardar este valor existente para poderlo restaurar cuando se vaya a deshacer el cambio realizado en el equipo. Si el registro no existe, la instrucción para deshacer el cambio será borrar el registro.

El archivo de log es generado por el programa y su contenido varía en función de las acciones del usuario.

Por último, el bloque INSTALLED contiene los requisitos instalados.

```
#INSTALLED
%Java::Capicom
#/INSTALLED
```

Ilustración 7: Ejemplo de identificación de requerimientos instalados

En el ejemplo vemos que se ha instalado Java y Capicom usando el configurador. A la hora de deshacer cambios se realizan las acciones del bloque UNDO pertenecientes a dichos requerimientos.

El modelo de negocio

Vamos a ver la estructura de datos diseñada para cargar en memoria principal los archivos de configuración y trabajar con ésta información. Como se ha visto, existen cinco bloques, en cuatro de éstos existen grupos de instrucciones. Cada bloque tiene su propia razón de ser y almacena unos datos u otros. Por este motivo debe existir más de un tipo de grupo de instrucciones. A su vez existen diferentes tipos de instrucciones, según la acción que se tenga que realizar. Por este motivo tiene que existir más de un tipo de instrucción.

Se ha identificado que los grupos de instrucciones en los bloques CONFIG, ONFAIL y UNDO tienen la misma estructura. El bloque NEEDS también tiene grupos de instrucciones pero con unas necesidades diferentes. El bloque INSTALLED almacena una lista de los nombres de los requerimientos instalados.

Teniendo esta idea en mente se ha creado la clase SpecList. Un SpecList define un grupo de instrucciones. Como se ha dicho, existen dos clases de grupos de instrucciones, existen dos clases herederas de SpecList, IdentificationSpecList, que define grupos de instrucciones del bloque NEEDS e InstallationSpecList, que define grupos de instrucciones de los bloques CONFIG, ONFAIL y UNDO.

El equivalente a una instrucción en el modelo de negocio es la clase Spec (de 'specification'). Como existen distintas clases de instrucciones, existen distintas clases de Specs, cada una con su propia finalidad.

El contenido del bloque INSTALLED se traduce como una lista de elementos de tipo ristra (List<string>).

Al igual que un SpecList es una colección de Specs, un SpecSet es una colección de SpecLists. Se ha definido de esta forma ya que un bloque puede tener varios grupos de instrucciones o lo que es lo mismo, un SpecSet tener varios SpecLists.

Para agrupar estos elementos existe la clase DataSet que almacena:

- cuatro SpecSet, correspondiente a los bloques NEEDS, CONFIG, ONFAIL y UNDO.
- un List<string>, correspondiente al bloque INSTALLED.

Estos son los tipos de Spec, que representan diferentes tipos de instrucciones:

- CopySpec: guarda información relativa sobre archivos que deben ser copiados al equipo local.
- DeleteSpec: guarda información relativa a elementos del equipo local que deben ser eliminados.
- DirectorySpec: guarda información relativa a operaciones de lectura o escritura de directorios.
- FileSpec: guarda información relativa a operaciones de lectura o escritura de archivos.
- InfoSpec: guarda información que será mostrada al usuario por pantalla.
- MSCertSpec: guarda información relativa a operaciones de lectura o escritura de certificados digitales.
- RegistrySpec: guarda información relativa a operaciones de lectura o escritura en valores de registro.
- SubKeySpec: guarda información relativa a operaciones de lectura o escritura de subclaves de registro.
- ThreadSpec: guarda información sobre hilos de ejecución que serán solicitados.

Dependiendo del bloque que se trate, se permiten un tipo de Spec u otros. Las clases IdentificationSpecList e InstallationSpecList heredan de la clase List<Spec>.

En el apartado anterior vimos que un SpecList del bloque NEEDS está destinado a determinar los requerimientos para un sistema soportado y además indica los navegadores a los que dichos requerimientos están destinados a configurar. En el modelo de negocio, cada SpecList tiene asociado un objeto Browsers, son tres variables booleanas que identifican si dicho SpecList está destinado, según el archivo de configuración, para Internet Explorer, Google Chrome y Firefox.

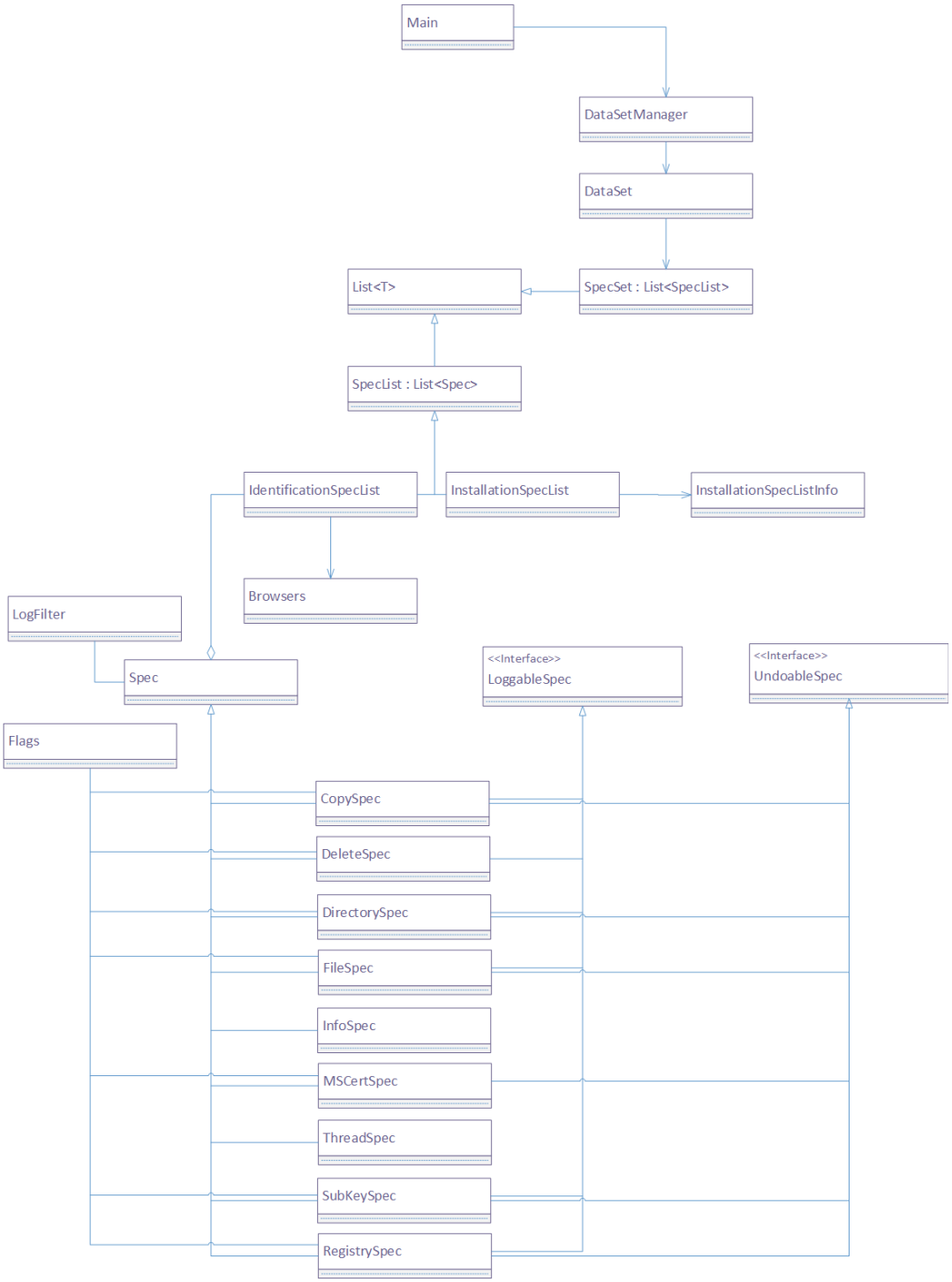


Ilustración 8: Diagrama de clases del modelo de negocio

Cabeceras de SpecLists

Los SpecLists son grupos de instrucciones. Existen dos tipos de SpecList. Un SpecList debe comenzar después de una etiqueta de apertura de un bloque. Los IdentificationSpecList pertenecen al bloque NEEDS e identifican sistemas soportados. Los InstallationSpecList pertenecen a los bloques CONFIG, ONFAIL y UNDO, definen grupos de instrucciones para comprobar, instalar o desinstalar un requerimiento, respectivamente. Para comenzar cada tipo de SpecList se especifican unos parámetros u otros.

Flags

Se puede ver este archivo en el modelo de negocio, esta es una clase estática en la que se han definido cuáles son todos los signos y palabras reconocidos por el intérprete a la hora instanciar objetos a partir de los archivos de configuración de forma correcta.

IdentificationSpecList

Identifican cada sistema soportado, para ello también tienen que definir sus requerimientos.

Comienzan con el símbolo % seguido de los nombres de los requerimientos, si hay más de un requerimiento, se separa cada uno por el símbolo :: Finalmente se especifican los navegadores para los cuáles están dirigidos estos requerimientos, los valores posibles son IE, FIREFOX y CHROME, en caso de que la misma sea aplicable a más de un navegador, se separan mediante el símbolo -.

Los nombres de los requerimientos escritos en los IdentificationSpecList serán identificadores únicos de cada requerimiento. Estos identificadores se usan en el resto de bloques para formar sus InstallationSpecList.

En la Ilustración 3 podemos ver un ejemplo, en este se necesitan hasta seis requerimientos para configurar Internet Explorer.

InstallationSpecList

Especifican cómo comprobar si un requerimiento está instalado, cómo instalarlo y desinstalarlo.

A la hora de ejecutar un InstallationSpecList para considerar que su ejecución es exitosa se especifica uno de los tres criterios siguientes, a este criterio se le llamará 'grado de satisfacción':

- SUCCEED_ALL: se considera que tiene éxito si todas sus instrucciones han tenido éxito.
- SUCCEED_ONE: se considera que tiene éxito si al menos una de sus instrucciones ha tenido éxito, por lo tanto se detiene su ejecución tras ejecutar su primera instrucción exitosa.
- SUCCEED_POSSIBLE: se ejecutan todas sus instrucciones indiferentemente de cuáles han tenido éxito y cuáles no.

Comienzan con el símbolo % seguido del identificador del requerimiento al que corresponde. A este dato le siguen cuatro parámetros más. El primero será el grado de satisfacción, seguido de su nombre (este nombre no es un identificador, será el nombre que se muestre al usuario a través de la interfaz).

Los dos últimos parámetros serán utilizados para cumplir con el objetivo de permitir la instalación de componentes de forma individual bajo petición del usuario. Es información ampliada sobre el requerimiento mostrada al usuario durante la instalación personalizada.

Todos los datos en esta cabecera se separan por el símbolo ::.

En el bloque CONFIG es necesario especificar todos los parámetros, en el bloque ONFAIL es suficiente con especificar el identificador único y el grado de satisfacción. Por último, en el bloque UNDO es necesario especificar el identificador único, el grado de satisfacción y el nombre del requerimiento que será mostrado al usuario.

En las ilustraciones 4, 5 y 6 podemos ver ejemplos de los tres usos de esta cabecera.

Tipos de instrucciones

Las instrucciones son órdenes que se especifican en el archivo de configuración. Existen once tipos de instrucciones diferentes. Una instrucción debe aparecer después de haber comenzado un SpecList. Cada uno de los tipos de instrucciones comienzan con una palabra reconocida por el programa. Dependiendo de la instrucción, puede tener unos parámetros u otros.

REGREAD

Lee un registro y lo compara con el valor deseado. Se pueden leer registros de tipo stringz, dword, qword y binarios. Si el registro es binario, se especifica su codificación.

Éstos son los formatos de la instrucción:

- REGREAD::(STRINGZ/DWORD/QWORD)::RUTA DEL REGISTRO::VALOR
- REGREAD::BINARY::CODIFICACIÓN::RUTA DEL REGISTRO::VALOR

FREAD

Lee un archivo del equipo local y lo compara con un valor. Si el contenido con el que se quiere comparar tiene más de una línea, éstas se especifican con {{jumpline}}.

Este es el formato de la instrucción:

- FREAD::RUTA DEL ARCHIVO LOCAL::VALOR

REGSUBKEY

Examina el contenido del registro en busca de una subclave dada. Permite crearla si ésta no existe.

Este es el formato de la instrucción:

- REGSUBKEY::RUTA DE SUBCLAVE EN REGISTRO::READ ONLY (TRUE/FALSE)

El campo READ ONLY es false crea la subclave si no existe ya.

MSCERTS

Permite acceder a la base de certificados de Windows e instalar un certificado digital pertenecientes al estándar PKCS12, en formatos p12 y pfx, o comprobar si existen certificados digitales emitidos por una organización.

Este es su formato:

- MSCERTS::*O=ORGANIZACIÓN*::TIPO::BASE::READ ONLY (TRUE / FALSE)

El campo TIPO puede ser (OTHERS / 3PARTYCA / CA / ROOTCA / MY / PEOPLE / PUBLISHERS)
El campo BASE puede ser (USER / MACHINE)

Si el campo READ ONLY es false, primero se busca un certificado con las características descritas y si no se encuentra ninguno se permite instalar un certificado, este puede ser emitido por cualquier organización.

THREAD

Crea un hilo de ejecución nuevo, si existe el programa ejecutable especificado en el equipo local. Este ejecutable puede tomar argumentos al iniciar.

Su formato es este:

- THREAD::RUTA EJECUTABLE::ARGUMENTOS

INFO

Muestra un mensaje al usuario en una ventana de diálogo.

Su formato es este:

- INFO::TÍTULO DE VENTANA::CONTENIDO DEL MENSAJE

REGWRITE

Escribe un valor en el registro.

Este es su formato:

- REGWRITE::TIPO::APPEND (TRUE / FALSE)::RUTA DEL REGISTRO:: VALOR

El campo TIPO puede ser (STRINGZ / DWORD / QWORD)

COPY

Descarga un archivo de la red al equipo local. Si existe un archivo con el mismo nombre, éste se sobrescribe.

Su formato es el siguiente:

- COPY::URL::RUTA LOCAL

FWRITE

Escribe un valor en un archivo. Si el valor tiene más de una línea, éste se especifica con {{jumpline}}.

Su formato es este:

- FWRITE::APPEND (TRUE / FALSE)::RUTA LOCAL::VALOR

MKDIR

Crea un directorio en el equipo local, si éste no existe.

Su formato es:

- MKDIR::RUTA LOCAL

DELETE

Elimina un archivo, directorio, registro, subclave de registro o desinstalar un certificado digital o un software del equipo local. Dependiendo del tipo de elemento que se quiere borrar se especifica unos parámetros u otros.

Formato para eliminar un archivo, directorio, registro o subclave

- DELETE::RUTA

Formato para desinstalar un certificado digital:

- DELETE::TIPO::BASE::(Número de serie del certificado)

Los campos TIPO y BASE son los mismos que en la instrucción MSCERTS.
Formato para eliminar un software:

- DELETE::BASE DE DESINSTALACIÓN::REGISTRO DE IDENTIFICACIÓN::VALOR DE IDENTIFICACIÓN::REGISTRO DE DESINSTALACIÓN

En la base de datos del registro de Windows existen subclaves destinadas a almacenar información sobre cómo desinstalar software, por ejemplo HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall, esta subclave (junto a otras separadas por coma) forma el campo BASE DE DESINSTALACIÓN.

Dentro de dichas subclaves existe una subclave por cada software instalado, para identificar el software podemos usar registros propios de dicho software que estén en esa subclave, por ejemplo 'DisplayName', este dato forma el campo REGISTRO DE IDENTIFICACIÓN. VALOR DE IDENTIFICACIÓN es el valor que toma REGISTRO DE IDENTIFICACIÓN para ese software concreto. En la misma subclave donde se encuentra el REGISTRO DE IDENTIFICACIÓN debe existir un registro que indica el comando a ejecutar

para desinstalar dicho software, por ejemplo 'UninstallString'.

Este tipo de instrucciones están disponibles únicamente en el bloque UNDO. Exceptuando el formato para desinstalar software, el resto se generan automáticamente para el archivo de log.

Windows dispone de un directorio donde programas pueden almacenar archivos de forma temporal. Los archivos de este directorio es eliminado periódicamente por el sistema operativo. El directorio es accesible mediante la variable de entorno %TEMP%.

Al iniciar el programa se crea un subdirectorio, 'ConfiguradorFirmaULPGC', dentro del directorio temporal.

En las instrucciones donde en alguno de los campos se deba especificar una ruta para un archivo en el equipo local, por ejemplo COPY, por defecto el archivo se guarda en el directorio 'ConfiguradorFirmaULPGC', a no ser que se especifique otra ruta, puede ser relativa (será relativa al directorio donde el usuario ha guardado el propio configurador de firma electrónica) o absoluta (haciendo uso de variables de entorno como %USERPROFILE%).

En todas las instrucciones, todas las apariciones de la cadena '%CONFIG_TEMP%' será sustituida por la ruta absoluta del directorio temporal creado por el programa.

Interpretar los archivos de configuración

Teniendo un archivo de configuración independiente del programa se necesita de un mecanismo que permita trabajar con estos datos, con esta finalidad se crean las clases Interpreter y FileInterpreter. FileInterpreter es una clase heredera de Interpreter. Lee archivos de configuración y genera una instancia de la clase DataSet, donde hay SpecList correspondientes a los distintos bloques.

Se ha usado el mecanismo de inversión de dependencias, de esta forma si en el futuro aparece otra forma de almacenar e interpretar la configuración, como en una base de datos, los cambios son más fáciles de introducir.

La clase FileInterpreter está implementada como una clase Singleton ya que no se necesita tener más de un intérprete en una instancia del programa. Esta clase devuelve una instancia de la clase DataSet, vista antes, que contiene los datos en memoria principal equivalentes a los almacenados en el archivo correspondiente.

Al iniciar el programa se interpreta tanto el archivo de configuración principal como el archivo de log personal (en caso de que exista información logueada con anterioridad), como tienen el mismo formato se puede usar el mismo intérprete.

Para interpretar cada Spec, FileInterpreter se basa en la clase SpecInterpreter, esta clase permite traducir instrucciones del archivo de configuración a objetos Spec de su tipo correspondiente.

Como se dijo en el apartado anterior, dependiendo del bloque (NEEDS, CONFIG, ONFAIL o UNDO), se permiten ciertos tipos de Spec u otros. Esta prohibición no viene dada por el modelo de negocio, es decir, el modelo de negocio no impide que una IdentificationSpecList sea capaz de almacenar un ThreadSpec, sino es el intérprete quien no lo permite. Si en un bloque aparece una instrucción no permitida, esta es ignorada.

En la siguiente tabla vemos los tipos de instrucciones permitidas en los diferentes bloques:

	NEEDS	CONFIG	ONFAIL	UNDO	INSTALLED
REGREAD	Sí	Sí	Sí	Sí	
FREAD	Sí	Sí	Sí	Sí	
REGSUBKEY	Sólo lectura	Sólo lectura	Sí	Sí	
MSCERTS		Sí	Sí	Sí	
THREAD		Sí	Sí	Sí	
INFO		Sí	Sí	Sí	
REGWRITE			Sí	Sí	

COPY			Sí	Sí	
FWRITE			Sí	Sí	
MKDIR			Sí	Sí	
DELETE				Sí	

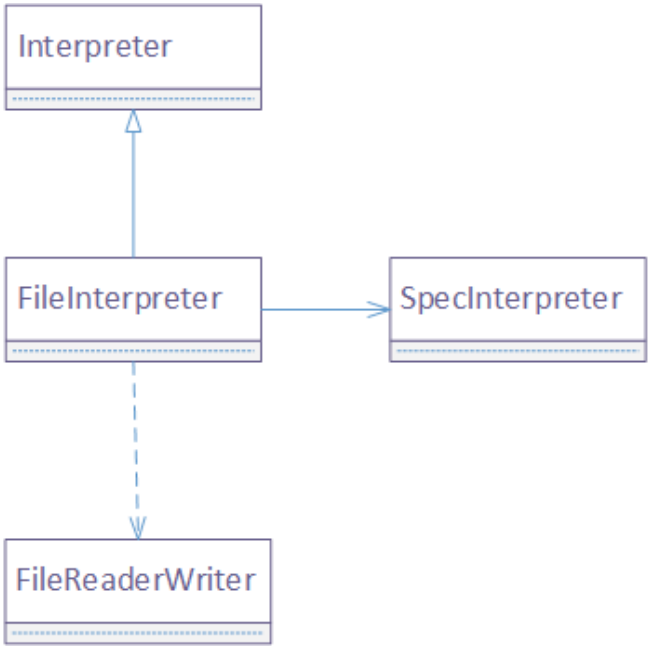


Ilustración 9: Diagrama de clases del intérprete

Identificación del sistema

Al iniciarse el programa, se crea en el directorio temporal de Windows un subdirectorio de nombre 'ConfiguradorFirmaULPGC'. Como el archivo de configuración principal está almacenado en un servidor de la ULPGC, para poder interpretar su contenido primero se descarga a este directorio temporal.

A no ser que en el archivo de configuración se especifique un directorio concreto, los archivos copiados al equipo local mediante la instrucción COPY se almacenan en este directorio. Los archivos copiados al directorio temporal no se incluyen en el log para ser eliminados, en cambio los que se guarden en un directorio diferente a éste, sí.

Una vez interpretado el archivo de configuración, tenemos los sistemas soportados y los navegadores web a los que dan soporte, que pueden ser Internet Explorer, Google Chrome o Firefox.

De la lista de NEEDS interpretada, hay que averiguar qué requerimientos son los necesarios para el equipo que ejecuta el programa. Para lograr este objetivo se ha ideado la clase SystemIdentifier. Esta clase determinará los nombres de los requerimientos aplicables para el equipo y los navegadores para los cuales están destinados a configurar estos requerimientos.

Mediante la interfaz de usuario, que veremos más adelante, se permite al usuario elegir los navegadores web que desea configurar. De esta forma se instalarán los requerimientos que elija el usuario.

Esta elección de navegadores realizada por el usuario se almacenan en una instancia de la clase Browsers, esta clase tiene como atributos tres variables booleanas, cada una corresponde a cada uno de los navegadores web soportados, que indican si el usuario quiere o no configurar el navegador web asociado a esta.

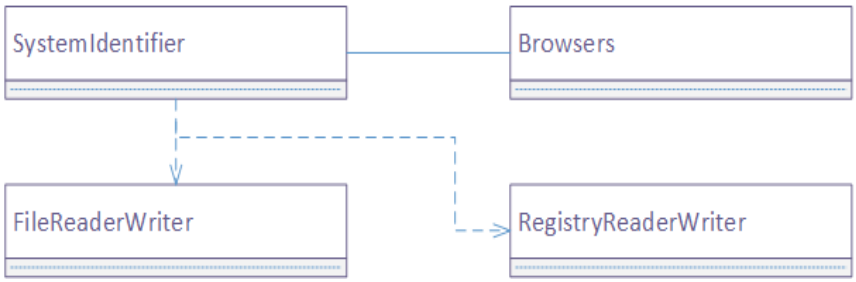


Ilustración 10: Diagrama de clases para identificar un sistema

Ejecución de SpecSets

En este punto tenemos las instrucciones cargadas en memoria, sabemos de qué tipo es cada una y la acción que puede realizar en el equipo. Se necesita de un sistema que permita llevarlas a cabo. Se ha ideado la clase SpecExecutor, de esta clase heredan otras clases, concretamente existe un SpecExecutor diferente para cada tipo de Spec ya que realizan acciones diferentes.

Para implementar esta funcionalidad se ha hecho uso del patrón Factory, de esta forma, existe una instancia de cada tipo de Executor que es llamado a la hora de ejecutar un Spec de su tipo correspondiente.

Teniendo las SpecLists, que definen las instrucciones a ejecutar y teniendo los SpecExecutor, que ejecutan Spec indivuduales, hay que ejecutar una o más listas de Spec, SpecList. Para ello se ha ideado la clase SpecSetExecutor.

Se hace uso de esta clase, SpecSetExecutor, tanto para instalar requerimientos como para desinstalar. Esto es posible ya que realmente ambos conceptos se abstraen y se codifican de la misma manera. Son secuencias de instrucciones a ejecutar una detrás de otra.

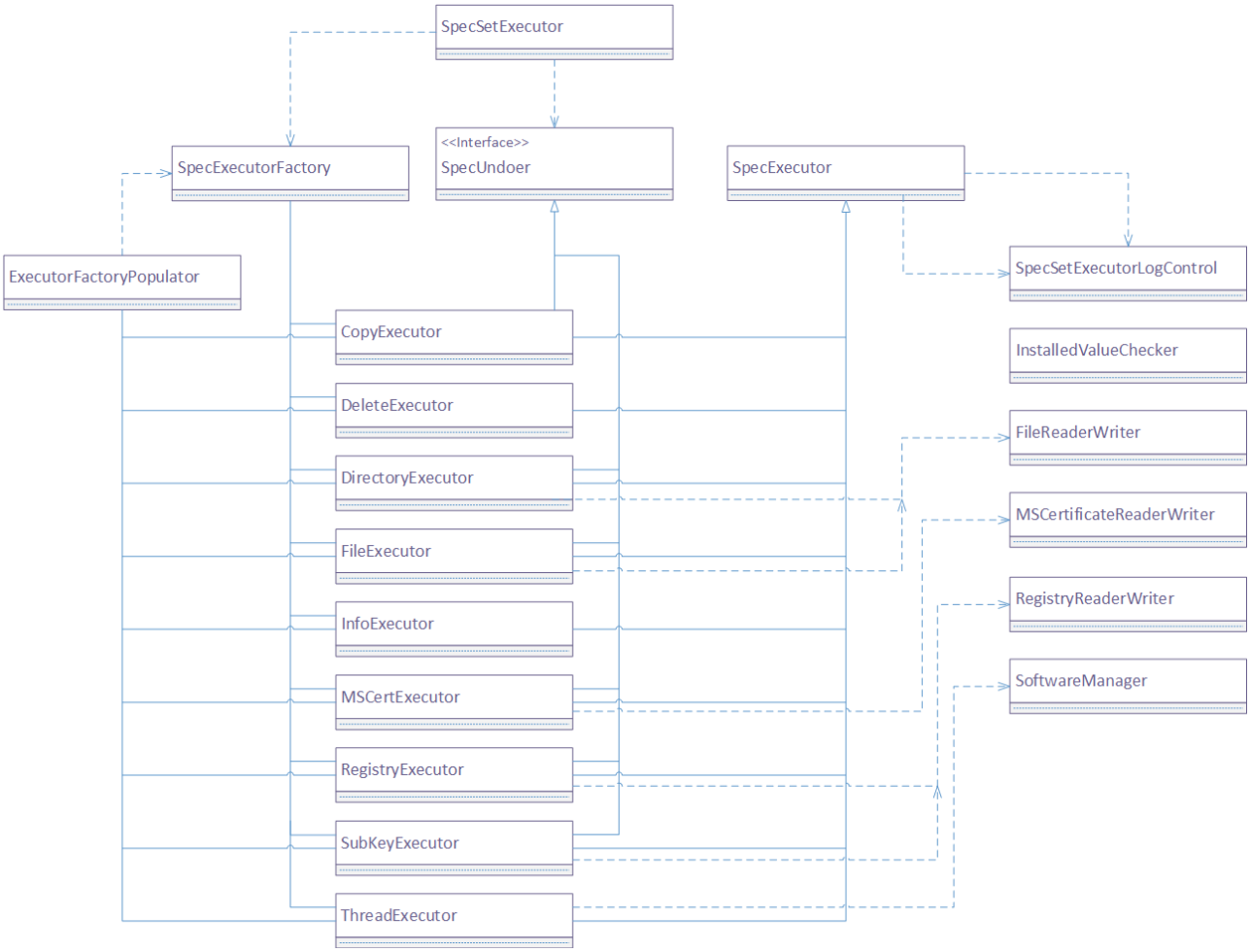


Ilustración 11: Diagrama de clases para ejecutar SpecSet

Sistema de logging

Tras la instalación exitosa de un requerimiento, se añade en el sistema de logging una entrada correspondiente para dejar constancia de la acción realizada. Se debe dejar constancia de que se ha realizado esta acción para permitir deshacerla si así lo desea el usuario. Lo mismo sucede al desinstalar un requerimiento, se eliminan del log entradas correspondientes al requerimiento.

Para cumplir con este objetivo se ha ideado la clase Logger, con esta clase se mantiene un SpecSet que simboliza los requerimientos instalados. Al iniciar el programa, los datos del archivo de log se copian en este SpecSet. En un principio, este estará vacío.

Al instalar un requerimiento, su SpecList se añadirá a este SpecSet, en el momento en que se desinstale, el mismo SpecList se elimina de este SpecSet. Cuando el usuario cierra el programa, antes de finalizar completamente el hilo de ejecución, este objeto SpecSet se almacena como parte del log, concretamente este SpecSet se traduce en el bloque UNDO del archivo de log.

La siguiente vez que el usuario use el programa, este archivo es interpretado y se recrea el objeto SpecSet, de esta forma se guardan las acciones para desinstalar los requerimientos instalados.

Windows tiene tres directorios por cada usuario destinados para que el software usado por el usuario guarde en estos sus archivos de preferencias y configuración. Estos directorios son:

- Roaming: contiene archivos de programas que permiten sincronizar información entre dispositivos.
- Local: contiene archivos de preferencias de programas locales que no compartidos entre dispositivos.
- LocalLow: tiene la misma función que Local pero para programas ejecutados bajo ciertas

condiciones, como 'modo seguro'.

Estos directorios están ocultos por defecto, se alojan en el directorio personal del usuario (C:\Users\Usuario).

En el caso de este proyecto, el directorio adecuado donde almacenar es Local, este subdirectorio es accesible por medio de la variable de entorno en Windows %localappdata%.

Dentro de %localappdata% se crea el directorio 'ConfiguradorFirmaULPGC', aquí es donde se guardará el log.

En principio el archivo de log estaba pensado para ser almacenado en un servidor remoto de la ULPGC. Por este motivo debía tener un nombre de archivo único. Para resolver este objetivo se decide que si la unión entre el nombre del fabricante del disco duro y el número de serie del mismo identifica inequívocamente a un equipo, entonces la unión de este código con el identificador el usuario en el sistema (UID) genera un token único para identificar únicamente un usuario en un equipo.

De este token generado se calcula su resumen con el algoritmo unidireccional SHA1, de esta forma se ha generado un token único totalmente seguro para llevar a la práctica.

Finalmente se decide almacenar el archivo en el equipo local, no obstante, esta funcionalidad ya está implementada y en el futuro se puede incluir sin ningún coste y sin emplear mucho tiempo.

Acceder a la información del disco duro y el UID del usuario se consiguió haciendo uso de la herramienta WMI, Windows Management Instrumentation integrada en .NET. WMI proporciona un API para facilitar tareas técnicas de administración.

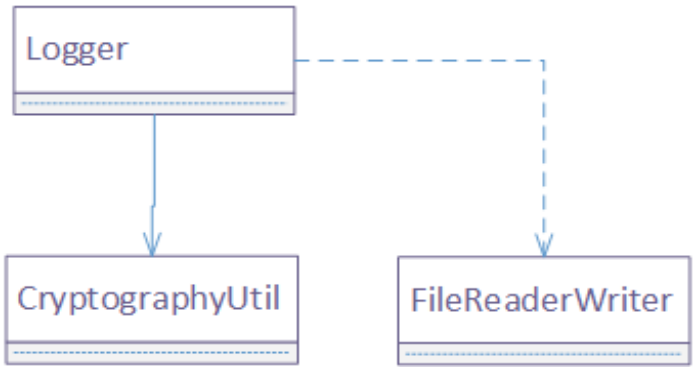


Ilustración 12: Diagrama de clases del sistema de logging

Acceso a funciones del sistema operativo y sistema de archivos

Se han creado clases de soporte que son usadas para ser usadas por otras. Se ha hecho de esta forma para 'centralizar' la comunicación del programa con diferentes características del sistema operativo. De esta forma operaciones comunes, como escribir o leer un archivo, se realizan desde una parte del código.

SystemInfo

Es una clase Singleton ya que no es necesario tener más de una instancia de la misma. Su objetivo es proporcionar al resto de los módulos información sobre el sistema: ruta del directorio temporal, URL de los archivos de configuración y datos sobre el disco duro (número de serie y fabricante) y sobre el usuario (UID) mediante consultas usando WMI.

RegistryReaderWriter

Es una clase que contiene métodos estáticos para la creación, lectura, escritura y eliminación de valores y subclaves de registro.

Esta clase permite leer registros binarios y de texto, a la hora de escribir se permite sobrescribir y crear nuevos registros de tipo texto. No se ha implementado la escritura de registros binarios porque no se ha dado su necesidad en el proyecto. Como se ha visto en las instrucciones, al leer un registro binario hay que especificar su codificación, sabiendo la codificación se puede convertir estos datos binarios en texto. Los caracteres de control, es decir, cualquier carácter no imprimible, no son incluidos como parte del resultado de conversión de binario a texto.

En la base de datos del registro de Windows, existen algunas subclaves en sistemas operativos de 64 bits que no existen en sistemas de 32 bits, como por ejemplo las que almacena información sobre software instalado:

HKLM\SOFTWARE

HKLM\SOFTWARE\Wow6432Node

Mientras que en sistemas de 32 bits existe únicamente la primera, en sistemas de 64 bits existen ambas. La segunda almacena información de software diseñado para sistemas de 32 bits. De esta forma sistemas de 64 bits son compatibles con software anterior. Sistemas operativos de 32 bits no soportan software diseñado para 64 bits.

Por defecto, a no ser que no se especifique lo contrario, un proceso de 32 bits en sistemas de 64 bits a la hora de acceder a una subclave determinada accede a través de los nodos de nombre Wow6432Node que haya disponibles en su recorrido.

Esta distinción es importante ya que este programa se ejecuta como un proceso de 32 bits y cuando es ejecutado en equipos de 64 bits si no se llevase a cabo este control, todo el contenido existente en HKLM\SOFTWARE (exceptuando la subclave Wow6432Node) no sería visible.

Por ejemplo, si queremos acceder al siguiente registro:

HKLM\SOFTWARE\JavaSoft\Java Runtime Environment\CurrentVersion

Si no se realiza el control descrito, se accedería a este otro registro, el cuál no tiene por qué existir:

HKLM\SOFTWARE\Wow6432Node\JavaSoft\Java Runtime Environment\CurrentVersion

FileReaderWriter

Es una clase que contiene métodos estáticos para la creación, lectura, escritura y eliminación de archivos y directorios nuevos y existentes.

Los archivos que esta clase puede manipular son archivos de texto. No se han implementado métodos para manipular archivos binarios porque no se ha dado su necesidad en el proyecto.

CryptographyUtil

Es una clase que proporciona servicios criptográficos, puede calcular resúmenes usando el algoritmo SHA1, cifrar texto (variables de tipo ristra) y archivos mediante clave pública usando el algoritmo AES-256.

MSCertificateReaderWriter

Es una clase que contiene métodos estáticos que sirven para leer y crear certificados de la base de certificados digitales de Windows. Estos certificados digitales luego pueden ser usados por otros servicios, como navegadores web.

Los certificados que esta clase puede leer del almacén de certificados de Windows son aquellos que están codificados según el estándar X509. A la hora de instalar, permite instalar certificados en formato p12 y pfx (pertenecientes al estándar X509).

Programas auxiliares

Instalación de DLL

Para realizar ciertas acciones que involucran al sistema operativo es necesario tener el permiso adecuado. Dependiendo de la acción que se intente realizar se necesita unos permisos u otros ya que se escriben datos que pueden involucrar a otros usuarios o al propio sistema operativo. Por ejemplo, se necesita permiso de administrador para instalar un software nuevo en el equipo.

El configurador de firma electrónica está pensado para que un usuario sin privilegios elevados pueda usarlo ya que no todas los requerimientos que se tienen que instalar necesitan este tipo de permiso. Por este motivo cuando el configurador llama a otro programa ejecutable por medio de una instrucción THREAD, si este necesita permisos especiales para funcionar, se piden al usuario antes de comenzar la ejecución de dicho programa.

Teniendo esta idea en mente, se ha desarrollado un programa auxiliar necesario para cumplir con los requisitos. Este programa se ha llamado 'DLLManager' y es independiente del proyecto principal, sirve para instalar y desinstalar librerías de enlace dinámico (DLL) en el equipo.

DLLManager requiere dos argumentos de entrada, 'instalar' o 'desinstalar' en función de lo que se deba realizar y la ruta absoluta del archivo DLL que se quiere instalar o desinstalar.

De esta forma, si se usan programas auxiliares, se pide permiso al usuario antes de ejecutarlos.

Analizador sintáctico de archivos de configuración

El archivo de configuración se almacena en un servidor remoto de la ULPGC y sería usado por todos los usuarios del configurador. Si en el futuro hay que modificar este archivo, sería un inconveniente no poder comprobar si el nuevo archivo tiene un formato válido. Por este motivo se ha desarrollado también junto al prototipo principal, un programa que analiza un archivo del equipo local y muestra los errores que hayan en caso de que éstos existan.

Dado el propósito de este programa auxiliar, no está dirigido al usuario final del configurador de firma electrónica sino a la persona que se encargada del mantenimiento del archivo de configuración público.

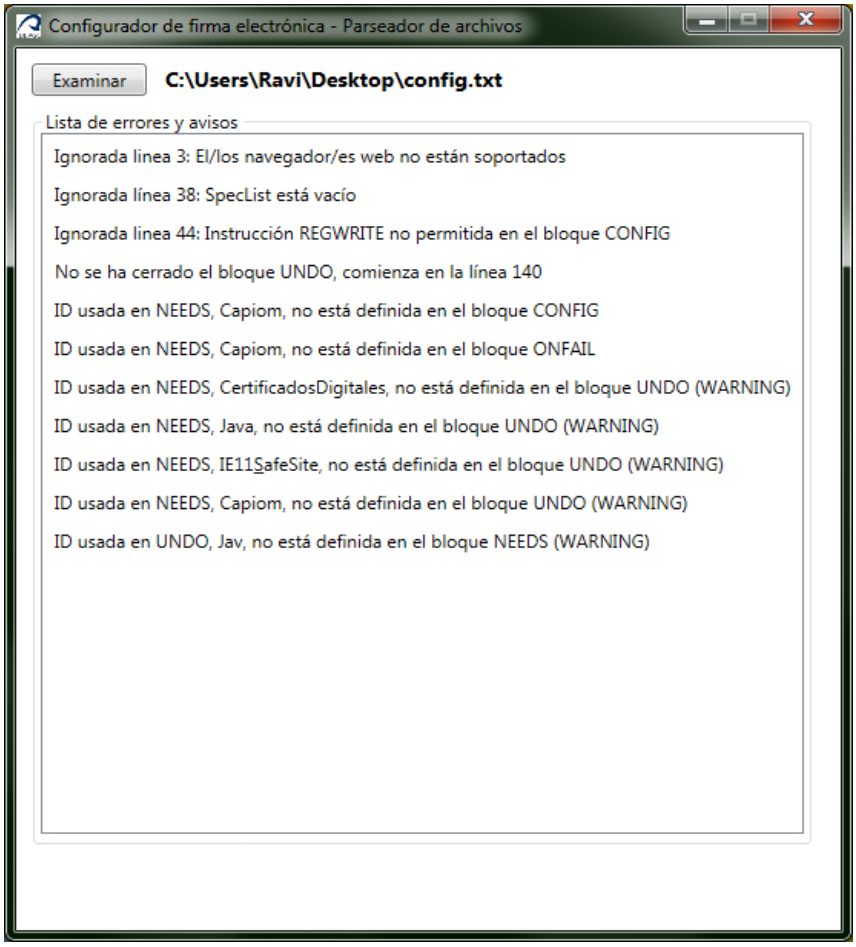


Ilustración 13: Ejemplo de análisis sintáctico de un archivo

Diseño e implementación de la interfaz de usuario

Diseño

Se ha procurado que el diseño sea sencillo pero efectivo a la hora de ser usado. A la hora de decidir la disposición de los distintos elementos en la interfaz se ha pensado que lo mejor es que usar estilos a los que el usuario ya está acostumbrado, de esta forma el uso del programa se hace más intuitivo.

Se han hecho dos diseños de la interfaz, el primero era en apariencia más sencillo pero para mostrar la información al usuario se generaban bastantes ventanas. Constaba de tan solo tres botones que generaban nuevas ventanas donde el usuario podía ver el proceso de configuración. Esta idea fue descartada y se diseñó otra.

Este es el diseño de la primera interfaz, que fue descartado:

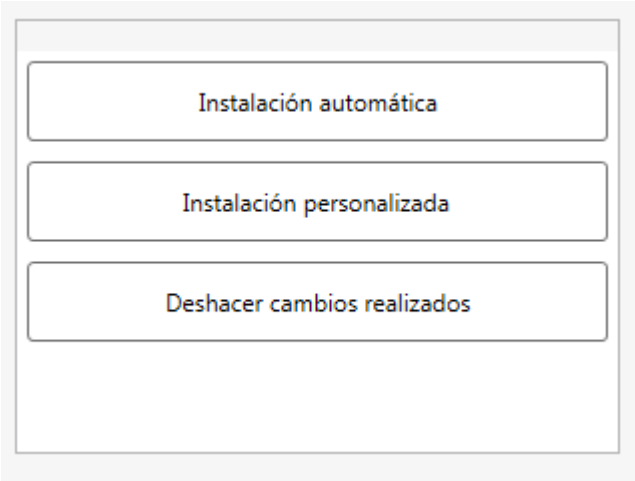


Ilustración 14: Diseño inicial de la interfaz de usuario

Este es el diseño de la interfaz actual del prototipo:

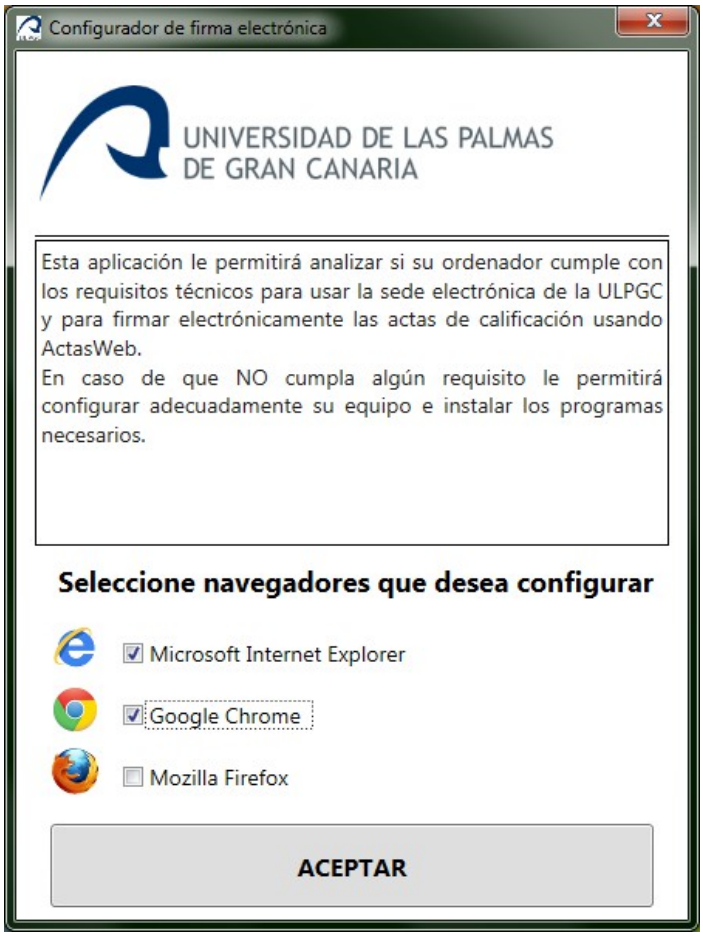


Ilustración 15: Diseño de la interfaz actual, inicio del programa

En primer lugar el usuario elige los navegadores que desea configurar, luego se muestra el siguiente panel:

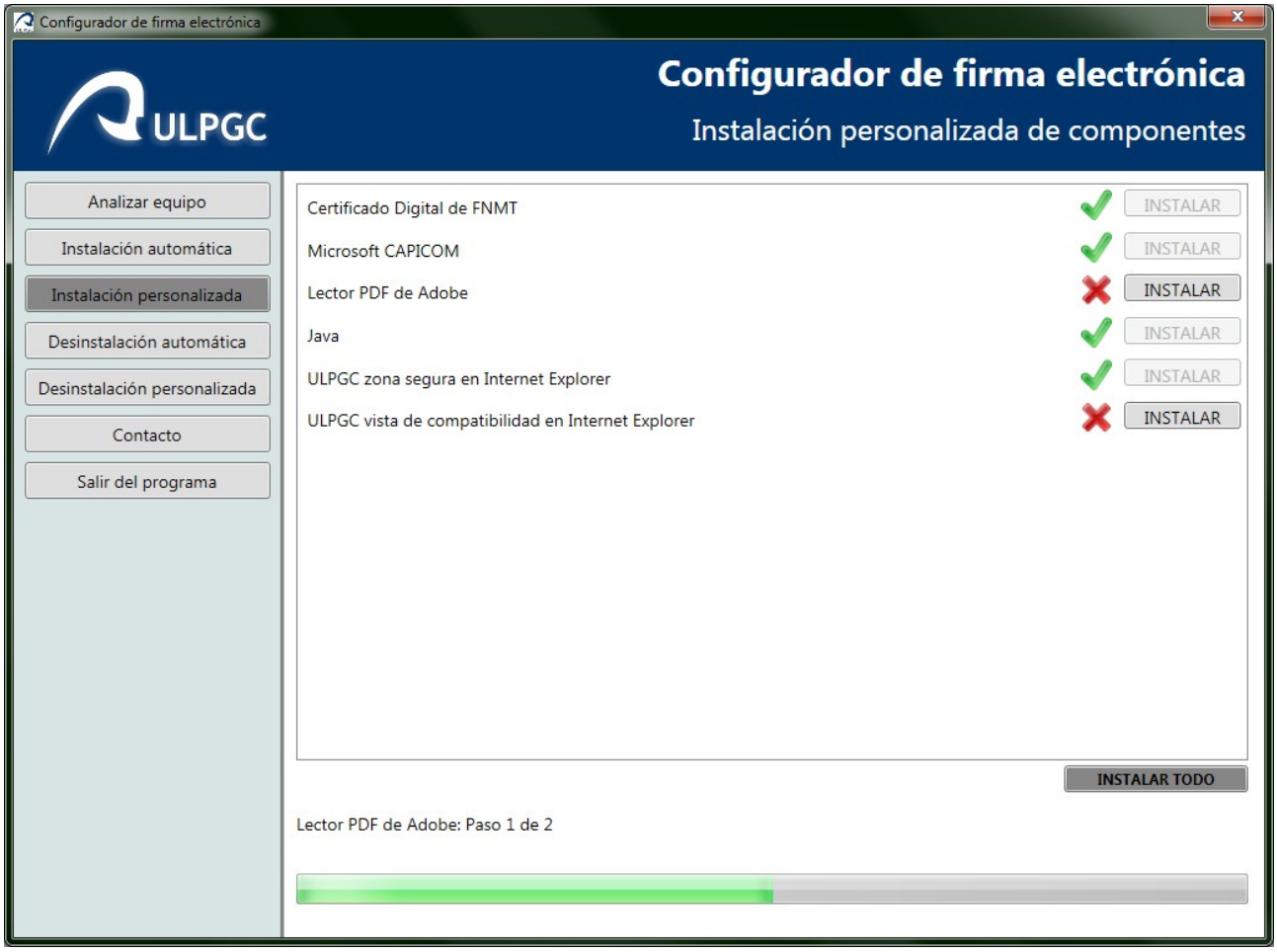


Ilustración 16: Diseño actual de la interfaz, menú de instalación personalizada

En el segundo diseño podemos ver que existe un panel central donde se muestra información al usuario y a la derecha está el menú principal de la aplicación.

Dependiendo de la opción elegida del panel derecho, el contenido del panel central varía. Como se ve en la imagen, en el panel inferior se muestran las acciones realizadas en el momento así como una barra de progreso que indica la cantidad realizada del proceso.

En el segundo diseño existe un panel central que no existe en el primero. De acuerdo al primer diseño esta información se habría mostrado en una ventana separada. De esta forma se reduce considerablemente el número de ventanas generadas.

Implementación

Para implementar la acción de los elementos de la interfaz de usuario se ha hecho uso del patrón Command. Con este patrón se ha creado una clase que simboliza una acción, a partir de esta 'acción' se definen 'acciones concretas'.

El comando a su vez se comunica con un objeto de clase Agent para realizar su tarea, es decir, se han abstraído las operaciones de instalación y desinstalación a esta clase. De esta forma se ha reducido el acoplamiento de las clases Command y la ejecución de SpecSets, es el agente quien ordena su ejecución.

El trabajo realizado en la ejecución programada para cada acción (cuando se pulsa un botón) tiene lugar en un hilo de ejecución separado del programa principal. De esta forma la interfaz no queda bloqueada, el usuario puede moverla por la pantalla, minimizarla, etc. El progreso de la ejecución se muestra en el panel

inferior.

Para ejecutar acciones en otros hilos se hace uso de la clase de .NET BackgroundWorker, esta clase permite, además de ejecutar procesos en segundo plano, generar eventos que refresquen la información en pantalla (usado para actualizar la barra de progreso) y realizar acciones cuando el hilo en segundo plano ha terminado.

Las acciones programadas para realizar usando la clase BackgroundWorker, incluyen análisis del equipo para comprobar que se cumplen los requerimientos, instalar requerimientos necesarios y la desinstalación de requerimientos, tanto de forma automática (instala todos los requerimientos necesarios) como de forma personalizada (el usuario instala individualmente los componentes que desea).

Pruebas y validación de uso

El prototipo ha sido probado en diferentes sistemas operativos, concretamente en Windows 8.1 Professional de 32 bits, Windows 7 Professional de 32 bits y en Windows 7 Professional de 64 bits.

Para validar el diseño de la interfaz de usuario, éste se ha puesto a prueba mediante el uso del prototipo de dos situaciones diferentes:

- En reuniones con los tutores, en los que se plantea un escenario que pudiese ser habitual entre los usuarios finales
- Uso en solitario de usuarios finales, donde el programa será usado sin ayuda.

El resultado de esta forma de validación es la más útil y productiva. El usuario final es quien mejor sabe cómo debe ser el diseño para que él mismo pueda usarlo de forma más cómoda.

El feedback recibido por usuarios finales ha generado la inserción de algunos requisitos nuevos pero factibles, de forma que se han podido cumplir en tiempo y forma.

Diagrama de secuencias

En este apartado se va a ver un diagrama de secuencias para describir las acciones que realiza el programa al realizar una instalación automática. De esta forma veremos de forma gráfica la interacción de los módulos descritos en el apartado de análisis y diseño.

En primer lugar se realizan algunas operaciones antes de mostrar la interfaz de usuario:

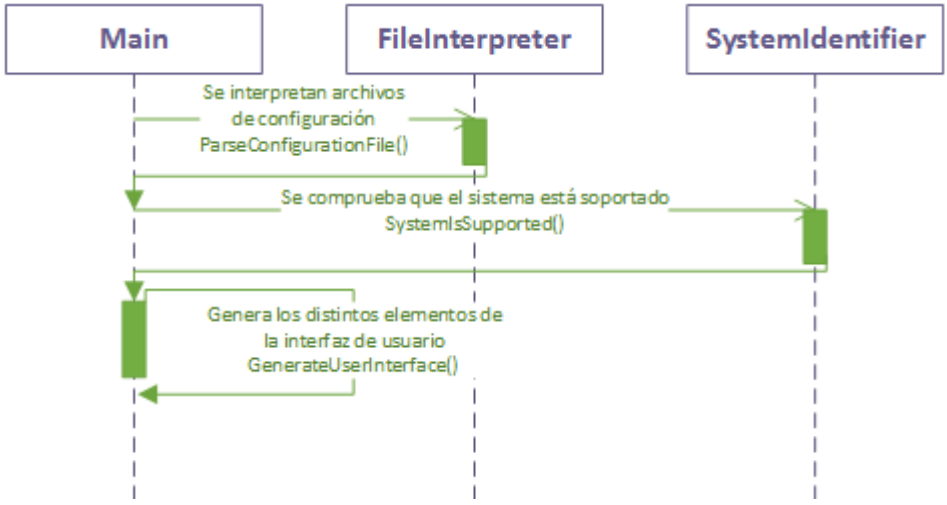


Ilustración 17: Diagrama de secuencias, carga inicial

Después de cargar en memoria principal estos datos iniciales, se muestra la interfaz.

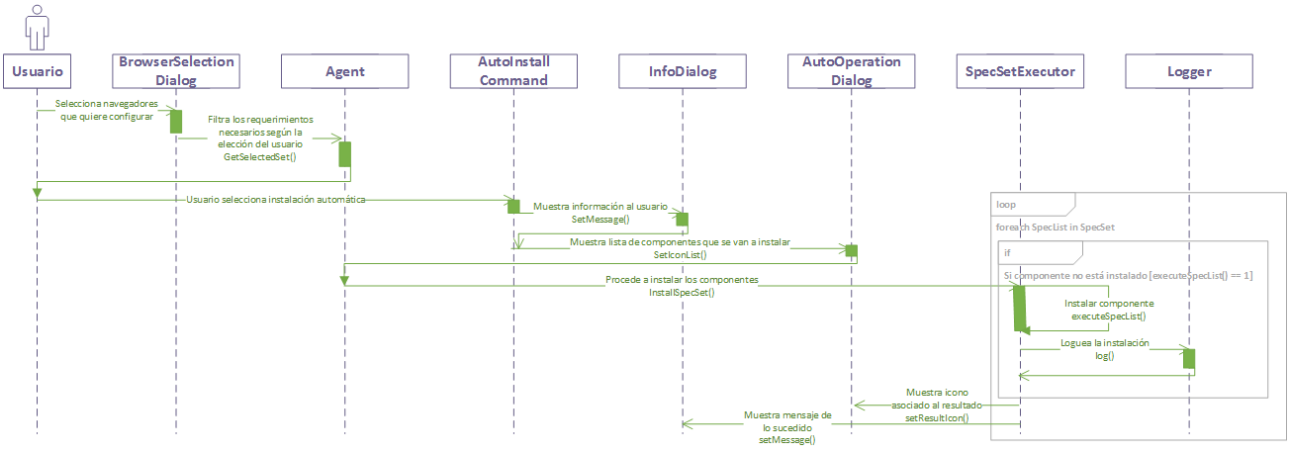


Ilustración 18: Diagrama de secuencias, instalación automática

En el anexo de este documento están disponibles más diagramas de secuencia.

Trabajos futuros

En el desarrollo de este proyecto se ha desarrollado un sistema que puede realizar la configuración de un equipo valiéndose de datos en su archivo de configuración.

Durante todo el trayecto se ha hecho énfasis en idearlo de forma que pueda seguir siendo útil en el futuro. Gracias a que el sistema de detección de requerimientos (en este caso configuración de la firma electrónica de la ULPGC) se ha independizado del programa, se ha hecho posible que si en un futuro estos requerimientos cambian, para reflejar estos cambios en el programa será suficiente con editar el archivo de configuración.

Siguiendo este planteamiento, editando el archivo de configuración el programa será capaz de configurar un equipo con los requerimientos que sean necesarios y no solamente la configuración para la firma electrónica, siempre y cuando estos requerimientos se puedan instalar de forma correcta con las instrucciones vistas en el apartado de 'Análisis y diseño'.

Teniendo en cuenta la duración que abarca el TFG, que son trescientas horas, se ha procurado implementar la mayor cantidad de requisitos posibles atendiendo a su prioridad. Por este motivo hay requisitos que se pueden seguir desarrollando, incluso se pueden plantear nuevas funcionalidades para incluir en el programa.

- El proyecto se ha desarrollado para sistemas Windows, pero los requerimientos identificados para que la firma electrónica de la ULPGC funcione correctamente son para sistemas Windows 7 y Windows 8.1. Se puede ampliar el número de sistemas soportados dentro de la familia Windows.
- El modo de proceder del programa puede transportarse a otros sistemas operativos y desarrollar configuradores para éstos. Sabiendo los requerimientos necesarios para que funcione la firma electrónica en sistemas operativos de Apple y Linux, se pueden desarrollar configuradores para ellos.
- Ampliar el número de navegadores web de la familia Internet Explorer que se pueden configurar, en el archivo de configuración se han definido los requerimientos para Internet Explorer 11, pero existen otras versiones que se siguen usando actualmente. Se pueden detectar sus requerimientos e incluirlos en el archivo de configuración.

Conclusiones

Se ha desarrollado un proyecto con muchas posibilidades de poder seguir creciendo en el futuro. Se han puesto en práctica diferentes áreas de la informática estudiadas a lo largo de la carrera, en especial la ingeniería de software.

Anteriormente había realizado algunos pequeños proyectos usando Windows Forms, que es otra tecnología para desarrollar software escrito con C# en .NET, pero no con Windows Presentation Foundation.

Al igual que WPF, existen otros aspectos que quería investigar pero no había tenido la ocasión de hacerlo. Estos son la programación concurrente, implementación de servicios criptográficos en C#. Por estos motivos el proyecto no solo ha posibilitado poner en práctica conocimiento ya adquirido sino también adquirir nuevas competencias.

Considero que se ha logrado cumplir con el objetivo con un buen grado de satisfacción ya que la configuración se realiza de forma automática en más de un sistema operativo y en más de un navegador web.

Fuentes de información

Para la realización de este proyecto se han consultado fuentes tanto virtuales como en soporte físico, se han consultado fuentes lo más actuales posibles.

Las fuentes de soporte físico han sido los siguientes libros:

- Beginning Visual C# 2012 Programming, escrito por Karli Watson, Jacob Vibe Hammer, Jon D. Reid, Morgan Skinner, Daniel Kemper y Christian Nigel. Publicado por Wrox, año 2013.
    - Se consultaron los capítulos 19 (información general sobre desarrollo de servicios web con ASP.NET) y 23 (Información básica sobre LINQ).
- Visual Studio 2012 And .NET 4.5 Expert Development Cookbook, escrito por Abhishek Sur. Publicado por PACKT Publishing, año 2013.
    - Se consultó el capítulo 3 (programación asíncrona en .NET).
- Codes And Cryptography, escrito por Dominic Welsh. Publicado por Oxford University Press, año 1988.
    - Se consultó el capítulo 11 (criptosistemas de clave pública).

Las fuentes de información en la red han sido las siguientes:

- Información general sobre Windows Presentation Foundation (WPF)
    - http://msdn.microsoft.com/es-es/library/ms754130%28v=vs.110%29.aspx
- Información general sobre las distintas ramas del registro de Windows
    - http://msdn.microsoft.com/en-us/library/windows/desktop/ms724836%28v=vs.85%29.aspx
- Cómo instalar y desinstalar librerías de enlace dinámico (DLL) en Windows
    - http://www.sophos.com/es-es/support/knowledgebase/14343.aspx
    - http://stackoverflow.com/questions/3474988/what-does-regsvr32-filename-ax-actually-do
- Información sobre implementación de cifrado simétrico en C#
    - http://www.technical-recipes.com/2013/using-rsa-to-encrypt-large-data-files-in-c/
- Información sobre implementación del algoritmo AES para cifrado simétrico en C#
    - http://www.gutgames.com/post/AES-Encryption-in-C.aspx
- Generación avanzada de números aleatorios para servicios criptográficos
    - http://www.dotnetperls.com/rngcryptoserviceprovider
- Información general sobre cómo usar Windows Management Instrumentation (WMI)
    - http://msdn.microsoft.com/en-us/library/aa389763%28v=vs.85%29.aspx
    - http://msdn.microsoft.com/es-es/library/system.management.managementobjectsearcher%28v=vs.110%29.aspx
    - http://msdn.microsoft.com/en-us/library/aa394346%28v=vs.85%29.aspx
    - http://www.dreamincode.net/forums/topic/288487-wmi-win32-useraccount-exception/
- Uso de WMI para obtener el número ID del usuario con sesión iniciada
    - http://social.msdn.microsoft.com/Forums/vstudio/en-US/dc973f8c-4fd9-41f3-82a9-3646721e9dbd/how-to-retrieve-a-remote-users-sid-using-wmi?forum=csharpgeneral

- Información sobre la clase Win32_UserAccount para obtener datos del usuario mediante WMI
  - http://msdn.microsoft.com/en-us/library/aa394507%28v=vs.85%29.aspx
- Información sobre obtención de valores del registro en C#
  - http://msdn.microsoft.com/en-us/library/microsoft.win32.registrykey%28v=vs.110%29.aspx
- Información sobre registros de desinstalación de software
  - http://msdn.microsoft.com/en-us/library/aa372105%28v=vs.85%29.aspx
  - http://msdn.microsoft.com/en-us/library/ms954376.aspx
- Cómo saber si una instancia de una clase de C# tiene una propiedad o un método
  - http://stackoverflow.com/questions/5114469/how-to-check-whether-an-object-has-certain-method-property
- Cómo desinstalar software instalado en Windows usando C#
  - http://stackoverflow.com/questions/9126104/how-to-uninstall-software-using-c-sharp-by-calling-software-uninstallstring-list
- Consulta sobre existencia de una subclave en el registro
  - http://stackoverflow.com/questions/13728491/opensubkey-returns-null-for-a-registry-key-that-i-can-see-in-regedit-exe
- Cómo saber desde código C# si el sistema operativo usado es de 64 bits o de 32 bits.
  - http://stackoverflow.com/questions/14423057/how-to-check-if-os-is-32-bit-os-or-64-bit
- Cómo acceder a controles de usuario en una interfaz matricial en WPF
  - http://stackoverflow.com/questions/1511722/how-to-programmatically-access-control-in-wpf-grid-by-row-and-column-index
- Información general sobre implementación de barras de progreso en WPF
  - http://www.wpf-tutorial.com/misc-controls/the-progressbar-control/
- Información general sobre programación concurrente aplicada a actualización de interfaz de usuario
  - http://stackoverflow.com/questions/4253088/c-updating-gui-wpf-using-a-different-thread
- Tabla de caracteres en codificación Unicode UTF-8
  - http://utf8-chartable.de/unicode-utf8-table.pl?utf8=0x&unicodeinhtml=hex
- Traducción de ristras binarias a texto, eliminando caracteres de control
  - http://stackoverflow.com/questions/4500870/how-to-remove-control-chars-from-utf8-string
- Información sobre la inclusión de sitios web en modo compatible de Internet Explorer 11 en su registro de Windows
  - http://jeffgraves.me/2014/02/19/modifying-ie-compatibility-view-settings-with-powershell/
- Iconos de uso libre usados en el proyecto
  - https://www.iconfinder.com/icons/27831/add_blue_minus_new_plus_icon#size=128
- Imágenes de ULPGC usadas en el proyecto
  - http://www.ulpgc.es/index.php?pagina=identidadgrafica&ver=inicio&
- Comparación de arrays usando LINQ

- http://stackoverflow.com/questions/43289/comparing-two-byte-arrays-in-net
- LOPD de la Agencia Española de Protección de Datos
  - http://www.agpd.es/portalwebAGPD/canaldocumentacion/legislacion/estatal/index-ides-idphp.php
- Instalación de certificados digitales del formato X509 en C#
  - http://stackoverflow.com/questions/16276213/issue-installing-x509certificate2-from-c-sharp-code
- Información sobre el directorio personal AppData en Windows
  - http://windows.microsoft.com/en-us/windows-8/what-appdata-folder
- Manual de Adobe del administrador, apartado 1.6
  - http://eddiejackson.net/web_documents/Acrobat_Enterprise_Administration.pdf
- Información sobre el patrón de pensamiento Lean en el marco de trabajo de Scrum
  - http://msdn.microsoft.com/es-es/library/jj161049.aspx
- Información sobre Scrum y Lean Software Development
  - http://es.wikipedia.org/wiki/Scrum
  - http://es.wikipedia.org/wiki/Lean_software_development
  - http://msdn.microsoft.com/es-es/library/hh533841.aspx
- Diferencias entre C# y Visual Basic aplicados en .NET
  - http://support.microsoft.com/?kbid=308470
- Información sobre hallar versión de Internet Explorer via registro de Windows
  - http://social.msdn.microsoft.com/Forums/windowsazure/en-US/b5f47548-69d5-499c-be60-de6ec78ce182/how-to-get-the-version-of-ie-programmatically-and-reliably
- Ejemplo de programación concurrente en C#
  - http://www.albahari.com/threading/part3.aspx
- Información sobre diferentes estrategias para obtener un UID de usuario
  - http://en.wikipedia.org/wiki/Universally_unique_identifier
  - http://tools.ietf.org/html/rfc4122
- Manipulación de certificados digitales en C#
  - http://www.programandoamedianoche.com/2009/08/utilizar-certificados-digitales-desde-net/
  - http://vyktor5k.blogspot.com.es/2010/06/howto-leer-certificados-digitales-en-c.html
  - http://programacion.com/articulo/trabajar_con_certificados_digitales_desde_net_i_838
  - http://www.programacion.com/articulo/trabajar_con_certificados_digitales_en_net_ii_847
- Uso de comandos de consola desde C#
  - http://stackoverflow.com/questions/437419/execute-multiple-command-lines-with-the-same-process-using-net
  - http://stackoverflow.com/questions/1469764/run-command-prompt-commands
- Información y ejemplos de uso de variables de entorno en Windows
  - http://ss64.com/nt/syntax-variables.html

Anexo

Diagramas de secuencias

En este apartado vamos a ver las distintas situaciones que se generan dependiendo de las acciones realizadas por el usuario en el programa. Primero vemos el diagrama correspondiente al arranque del prototipo, se interpretan los archivos de configuración y se comprueba si el equipo es sistema soportado.

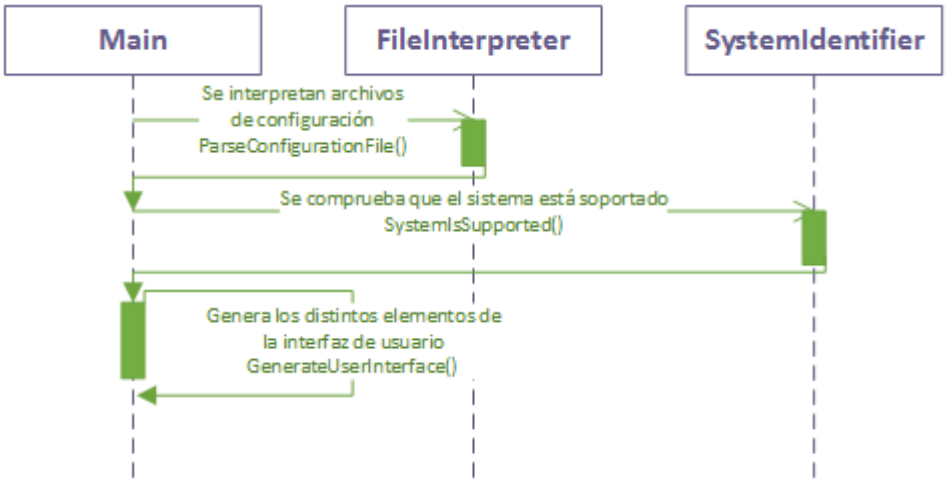


Ilustración 19: Diagrama de secuencias, carga inicial

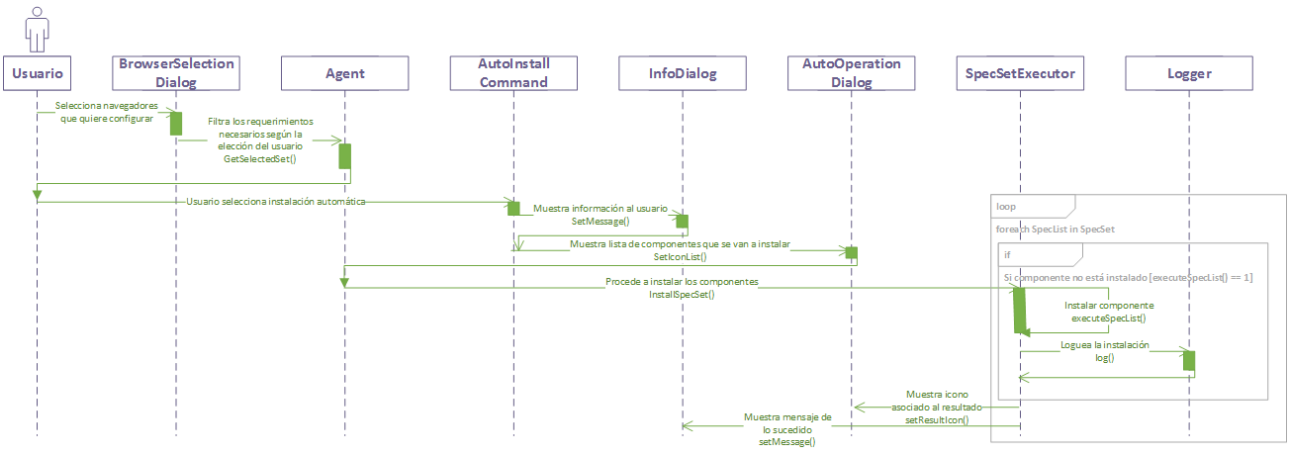
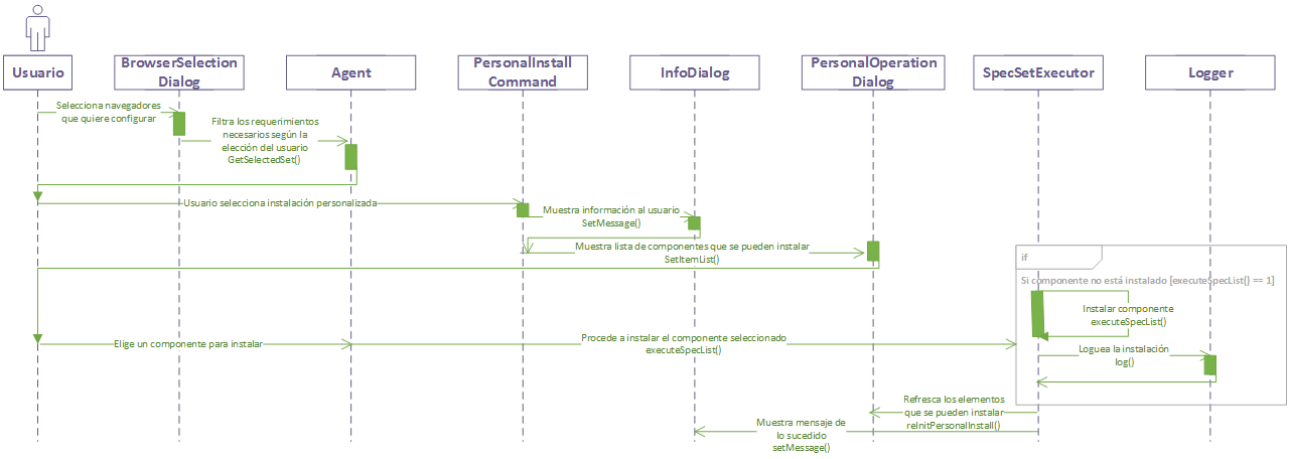


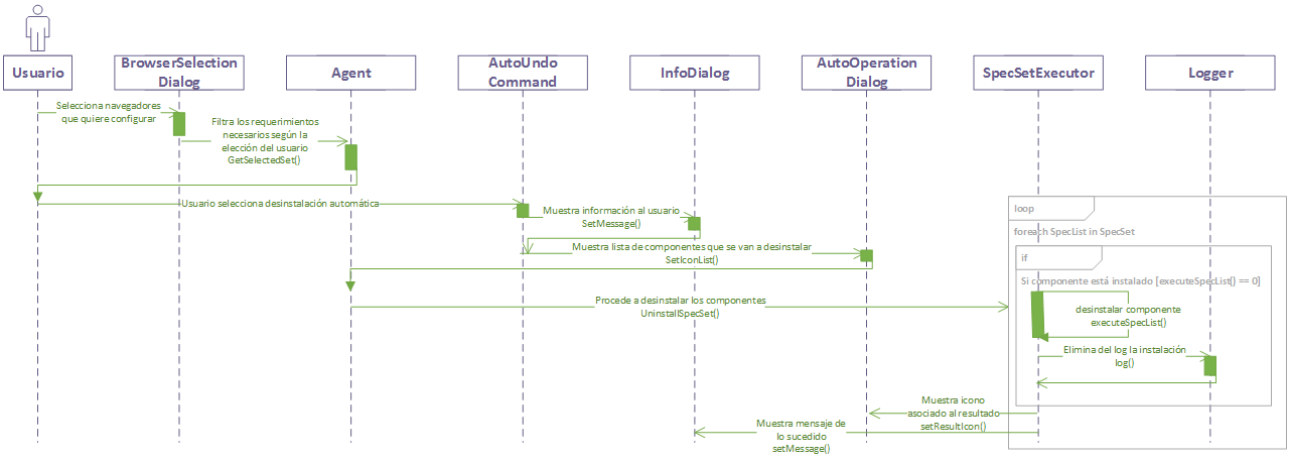
Ilustración 20: Diagrama de secuencias, instalación automática

Este diagrama representa la interacción de los módulos al realizar una instalación automática

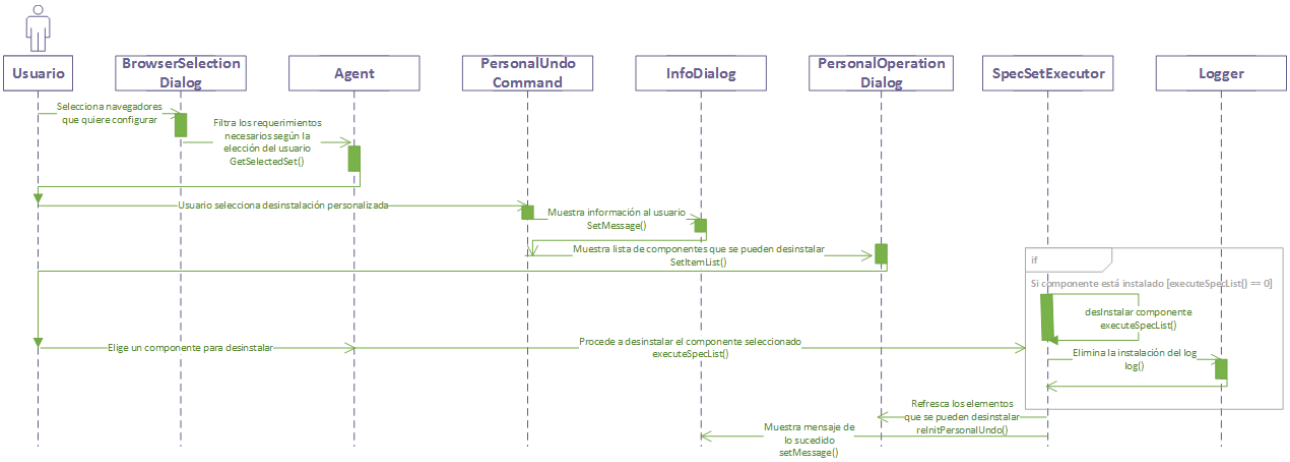
Este diagrama representa la interacción de los módulos al realizar una instalación personalizada:



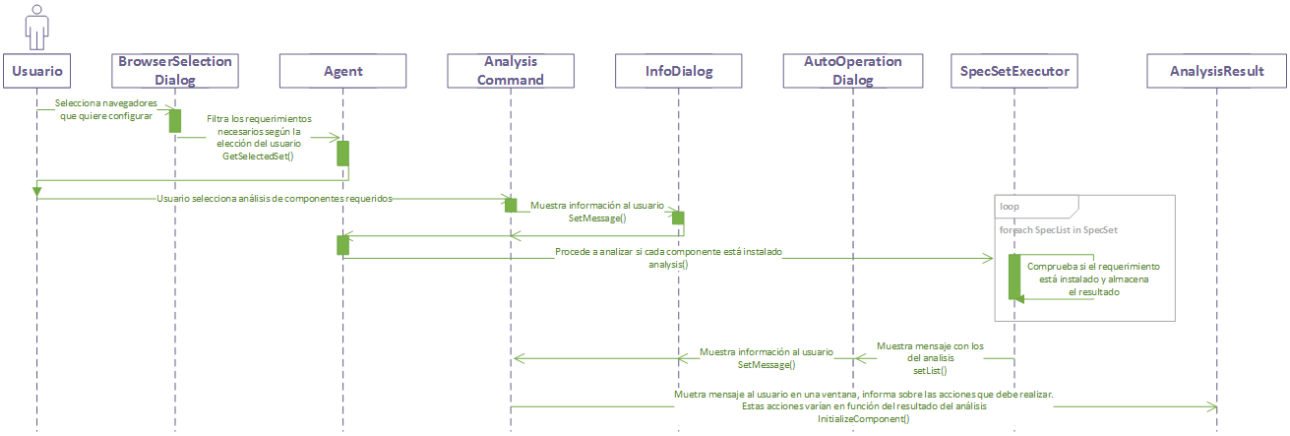
Este diagrama representa la interacción de los módulos al realizar una desinstalación automática:



Este diagrama representa la interacción de los módulos al realizar una desinstalación personalizada:



Este diagrama representa la interacción de los módulos al realizar un análisis del equipo:



Este diagrama representa la interacción de los módulos cuando el usuario solicita ver información de contacto:

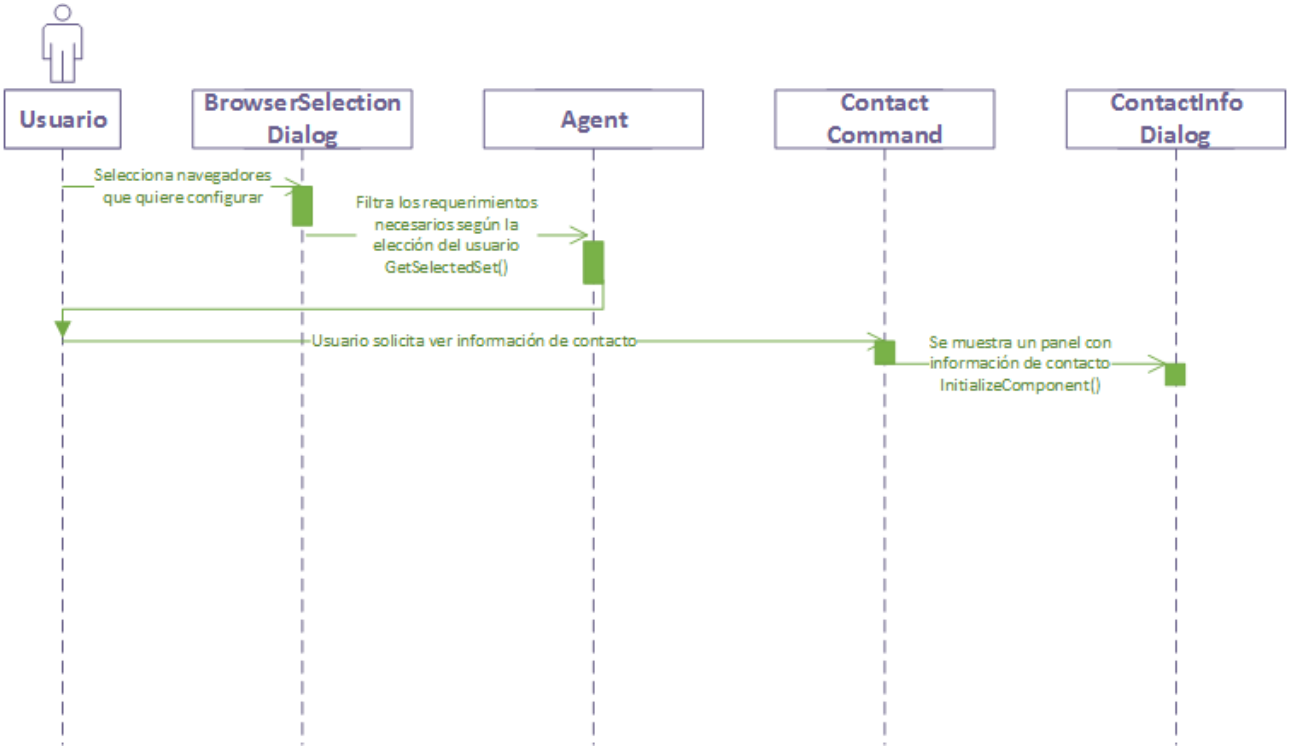
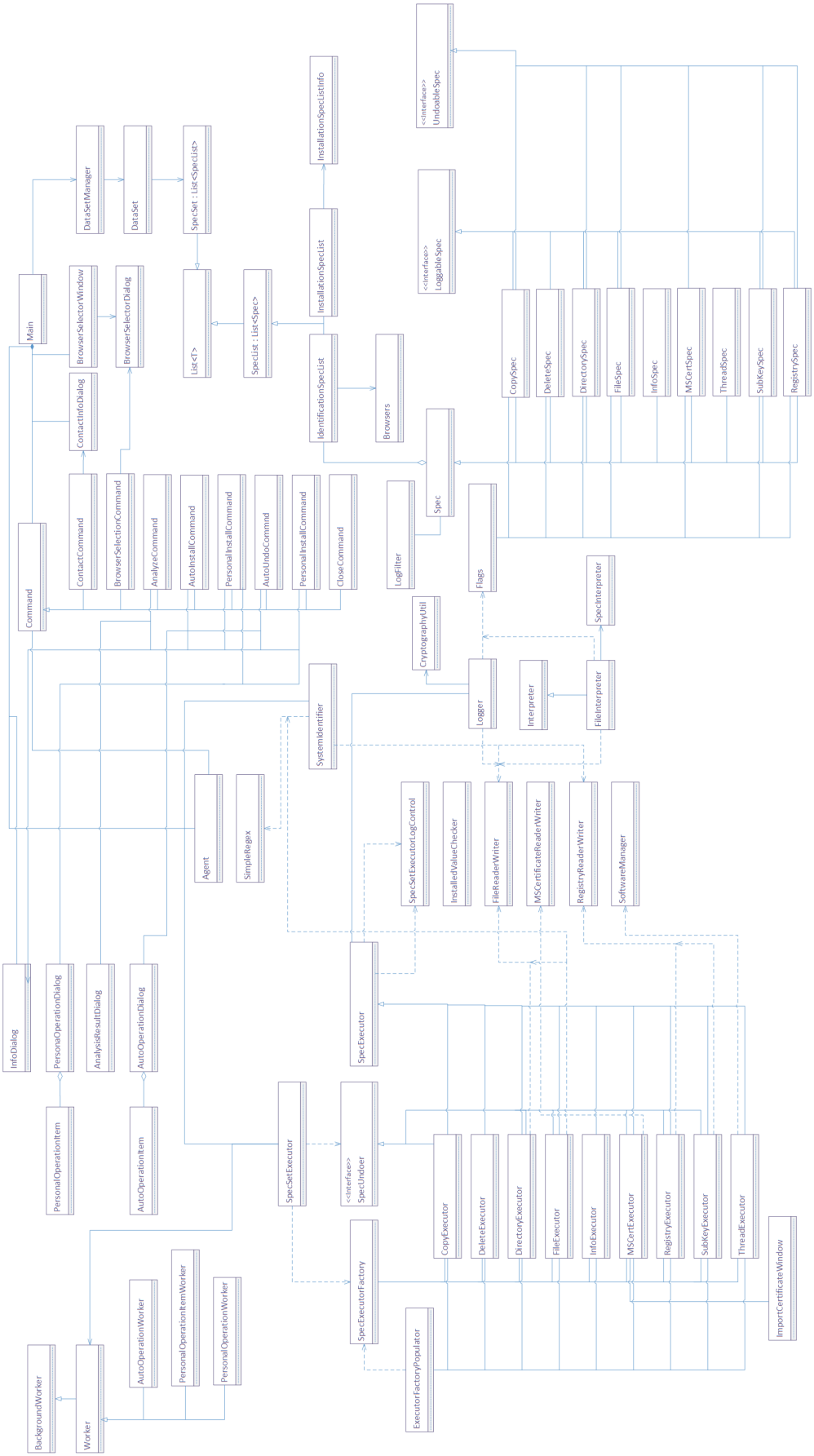


Diagrama de clases UML completo



Manual de usuario

Esta aplicación configura el ordenador personal del usuario para que la firma electrónica de la ULPGC funcione correctamente en los navegadores que desee configurar.

Necesita conexión a Internet durante el proceso de configuración ya que necesita acceder a archivos a servidores remotos de la ULPGC. Al iniciar el programa debe marcar los navegadores que desee configurar de la lista que se muestra en pantalla y pulsar el botón 'Aceptar':

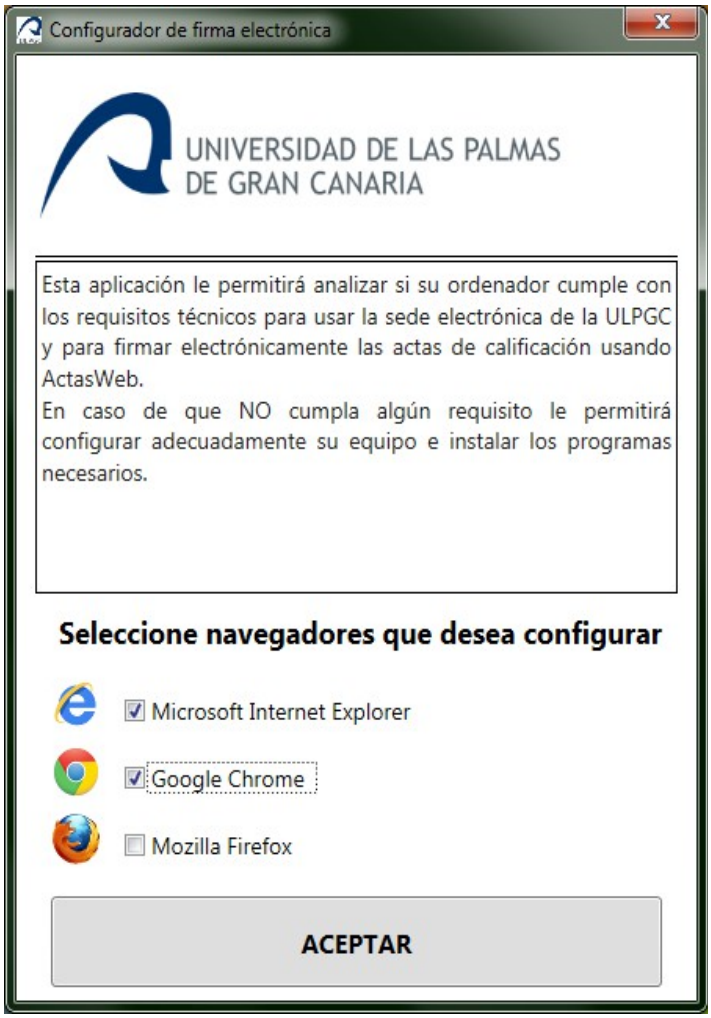


Ilustración 21: Pantalla de inicio de la aplicación

El programa recogerá las preferencias del usuario y mostrará el menú principal. En el menú principal se puede ver cuatro áreas principales, la primera es el área superior, donde junto al escudo de la ULPGC se mostrará el menú seleccionado actualmente. A la izquierda se podrá ver el menú de opciones, desde aquí se eligen las acciones que se deseen ejecutar. El área principal muestra diferente información al usuario dependiendo de la acción que se esté realizando, abarca la parte central y derecha de la pantalla. La última área es el inferior, en este panel se muestran mensajes mientras se realiza la configuración y en general durante la ejecución de las distintas acciones del menú de opciones.

Analizar el equipo

Si solamente se quiere analizar el equipo para averiguar si se cumplen con los requerimientos, hay que pulsar el botón 'Analizar equipo'. Vamos a ver el resultado del análisis cuando no se cumplen todos los requerimientos. En la imagen vemos que se cumplen algunos de los requerimientos pero no todos, por lo que la firma electrónica no funcionará, hay que configurar el equipo.

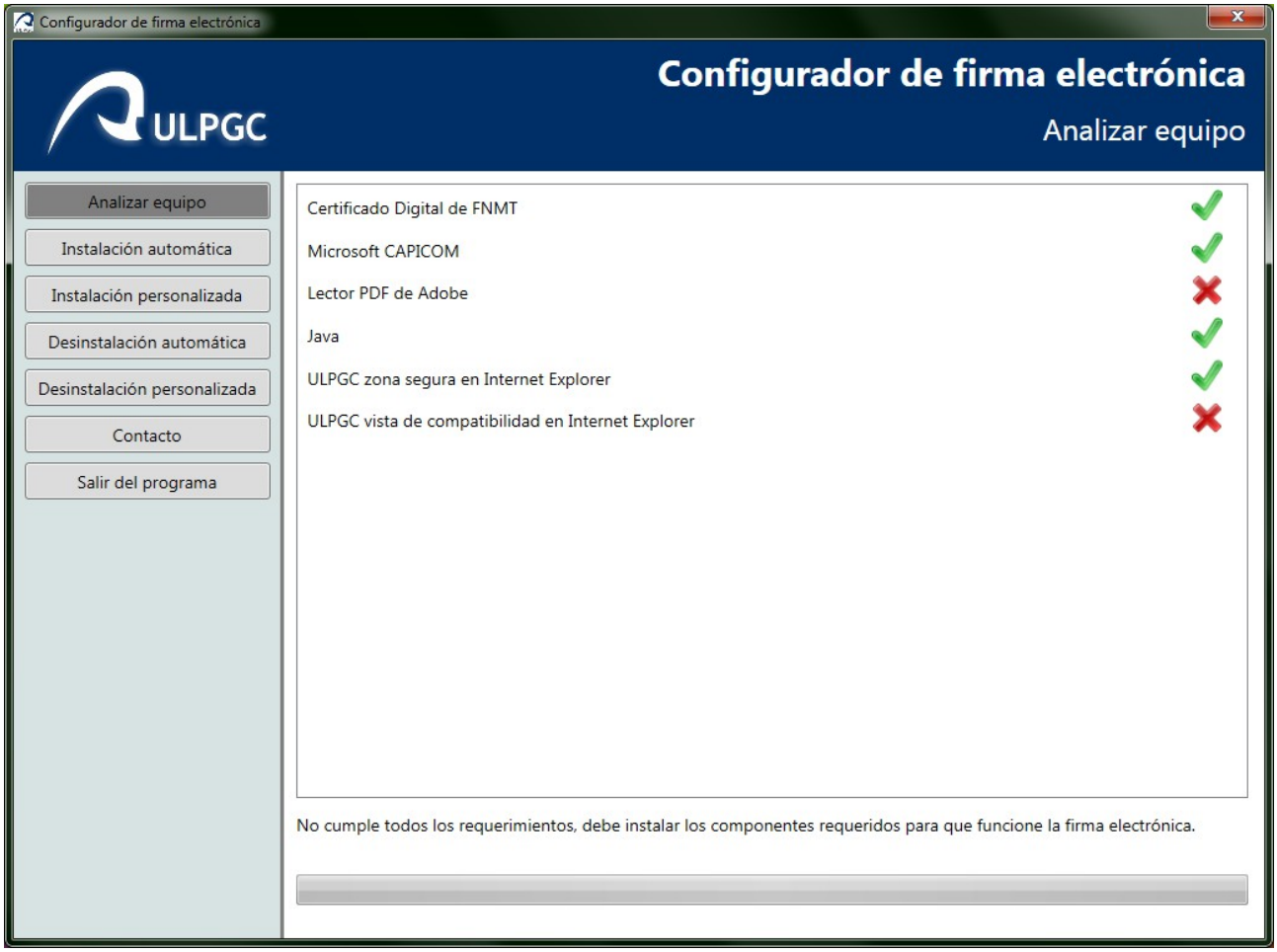


Ilustración 22: Menú de análisis del equipo, análisis negativo

Se muestra una lista de requerimientos, las marcas rojas indican que el requerimiento correspondiente no se cumple. Junto a este menú se muestra también el siguiente diálogo:

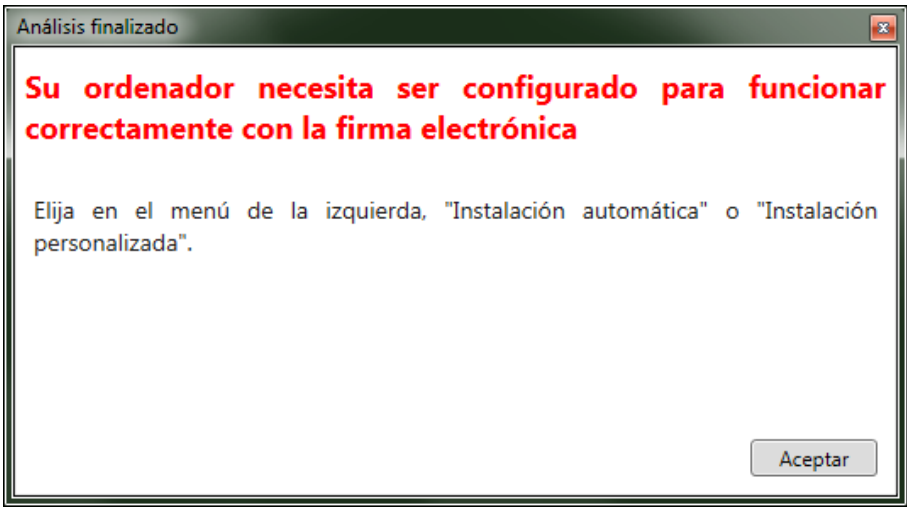


Ilustración 23: Diálogo de análisis negativo

Instalación de componentes

Vamos a proceder a instalar los requerimientos que no se cumplen. Como nos muestra el mensaje, se pueden instalar todos automáticamente o el usuario puede elegir individualmente cada uno.

Vamos a ver las opciones que nos muestra el menú de instalación personalizada, nos mostrará los requerimientos instalados con marcas verdes y los no instalados con marcas rojas y nos da la opción de instalar éstos:

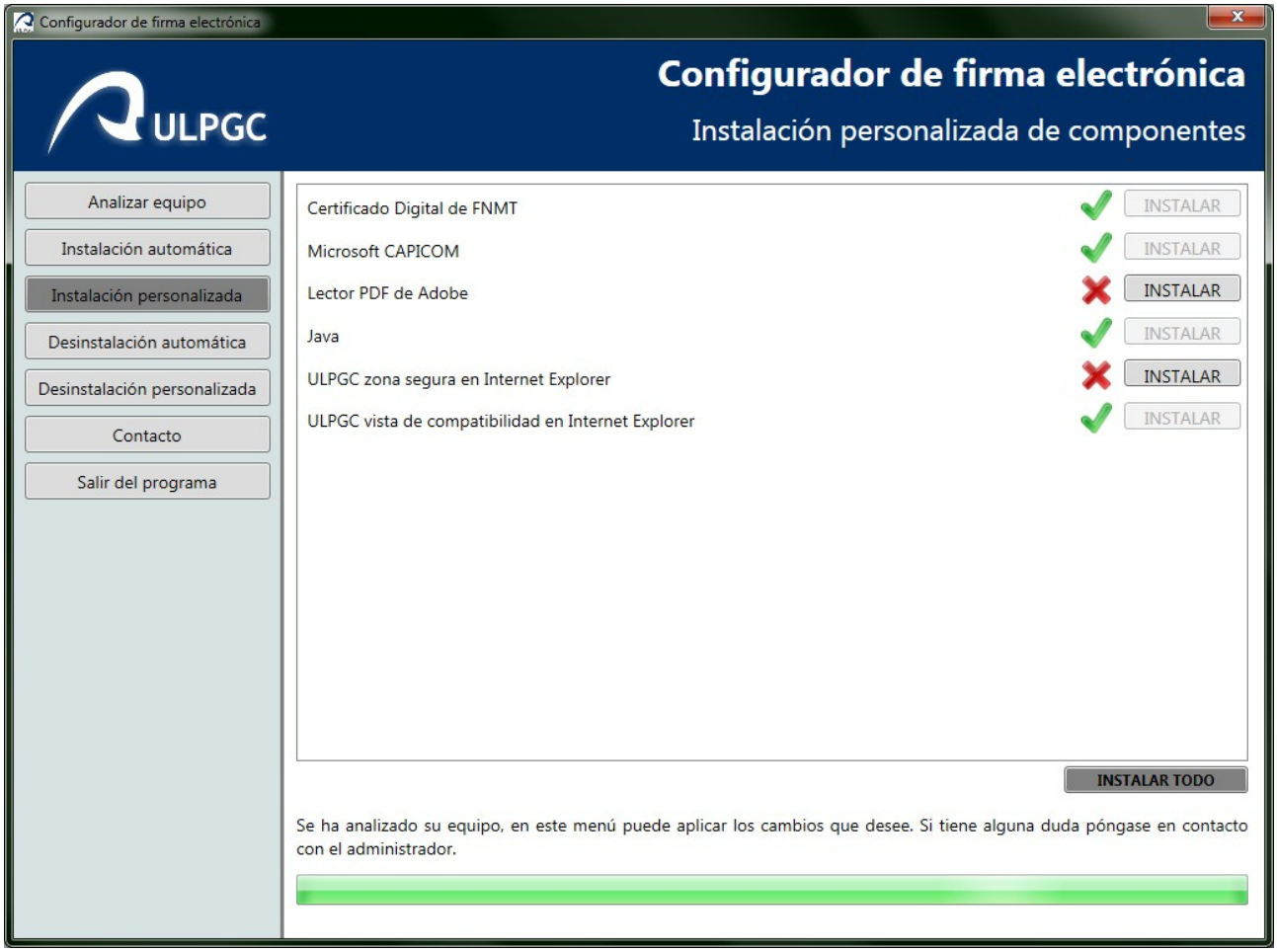


Ilustración 24: Menú de instalación personalizada

Se ha procedido a realizar una instalación automática, vemos una imagen de su realización cuando aún no ha terminado:

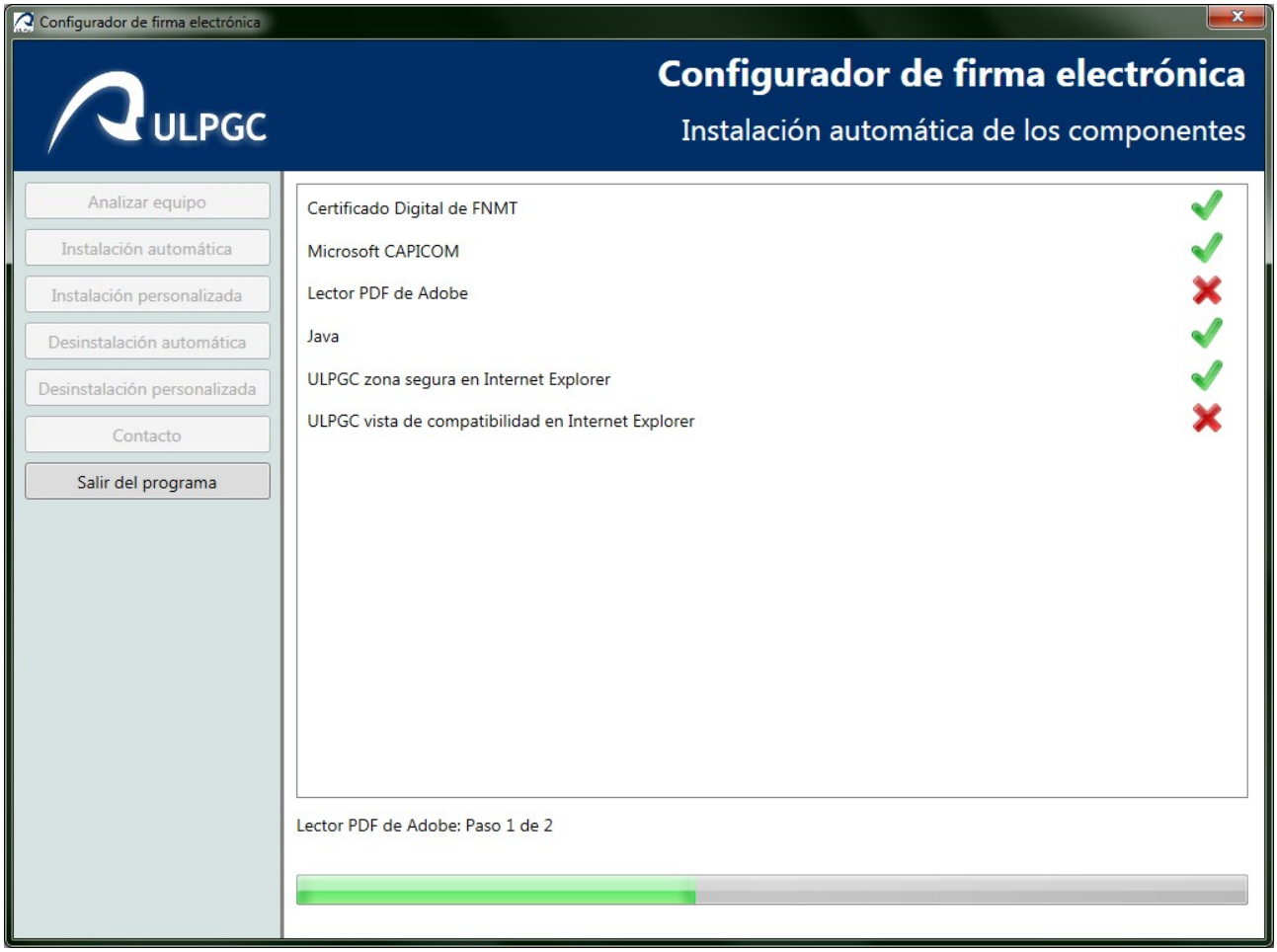


Ilustración 25: Realización de instalación automática

Durante la instalación automática, antes de instalar un requerimiento, se comprueba si está ya instalado, el programa procede a instalarlo sólo en caso negativo.

Ahora que se ha terminado correctamente el proceso de configuración, volvemos al menú de análisis, en esta ocasión nos muestra este mensaje:

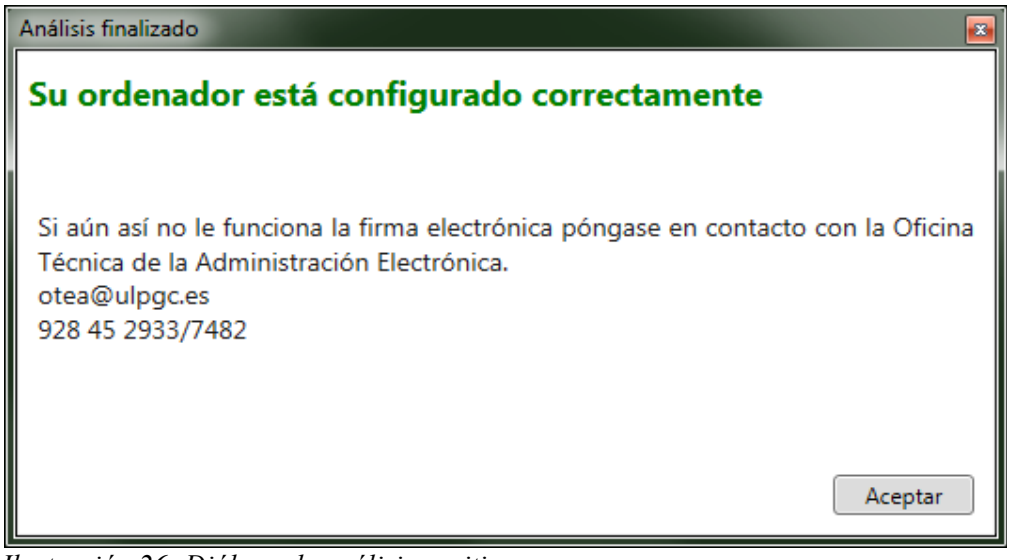


Ilustración 26: Diálogo de análisis positivo

Este es el aspecto de la pantalla principal:

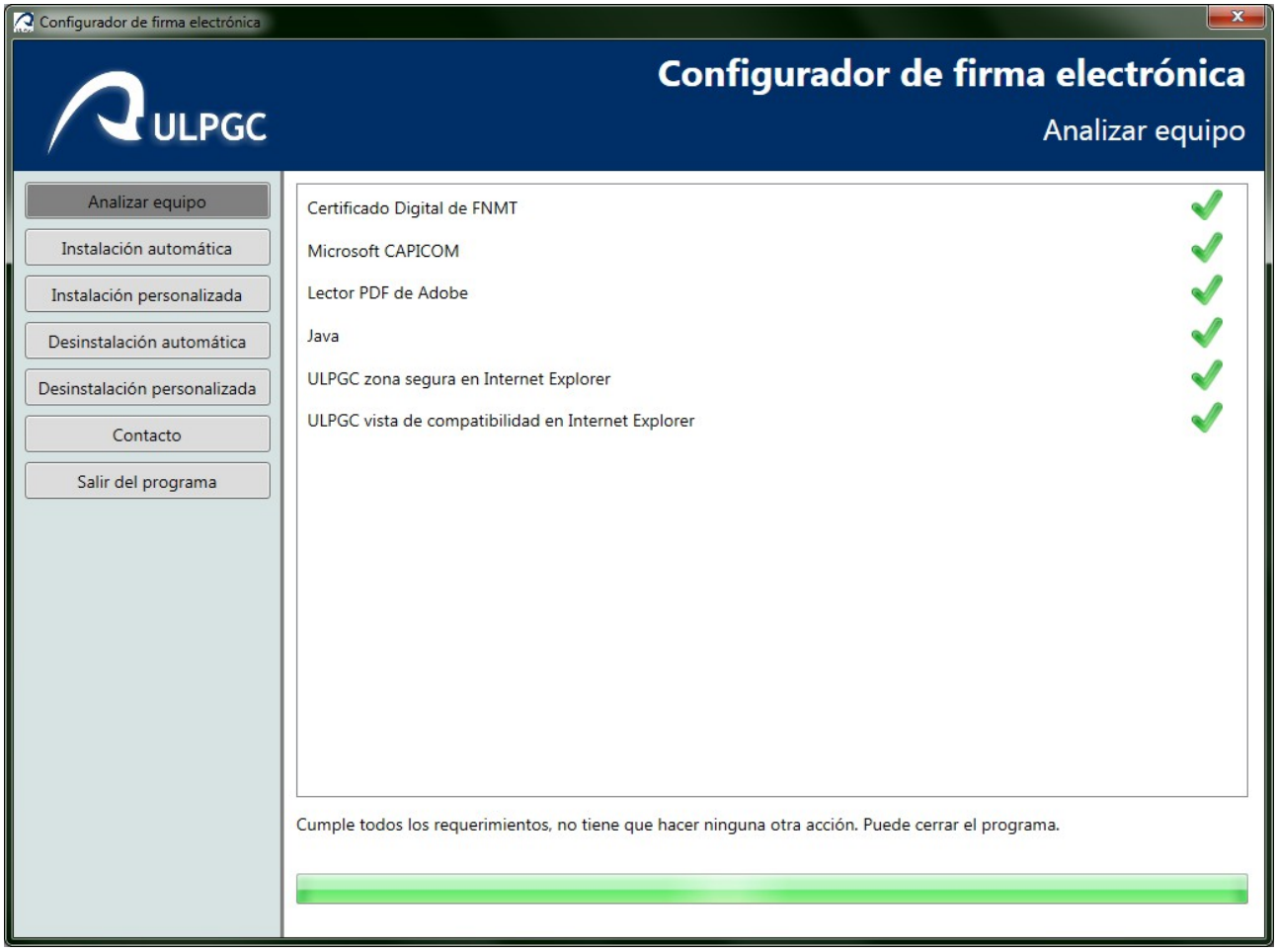


Ilustración 27: Menú de análisis del equipo, análisis positivo

En el menú de desinstalación personalizada vemos que nos da la posibilidad de eliminar componentes instalados por el programa.

Desinstalación de componentes

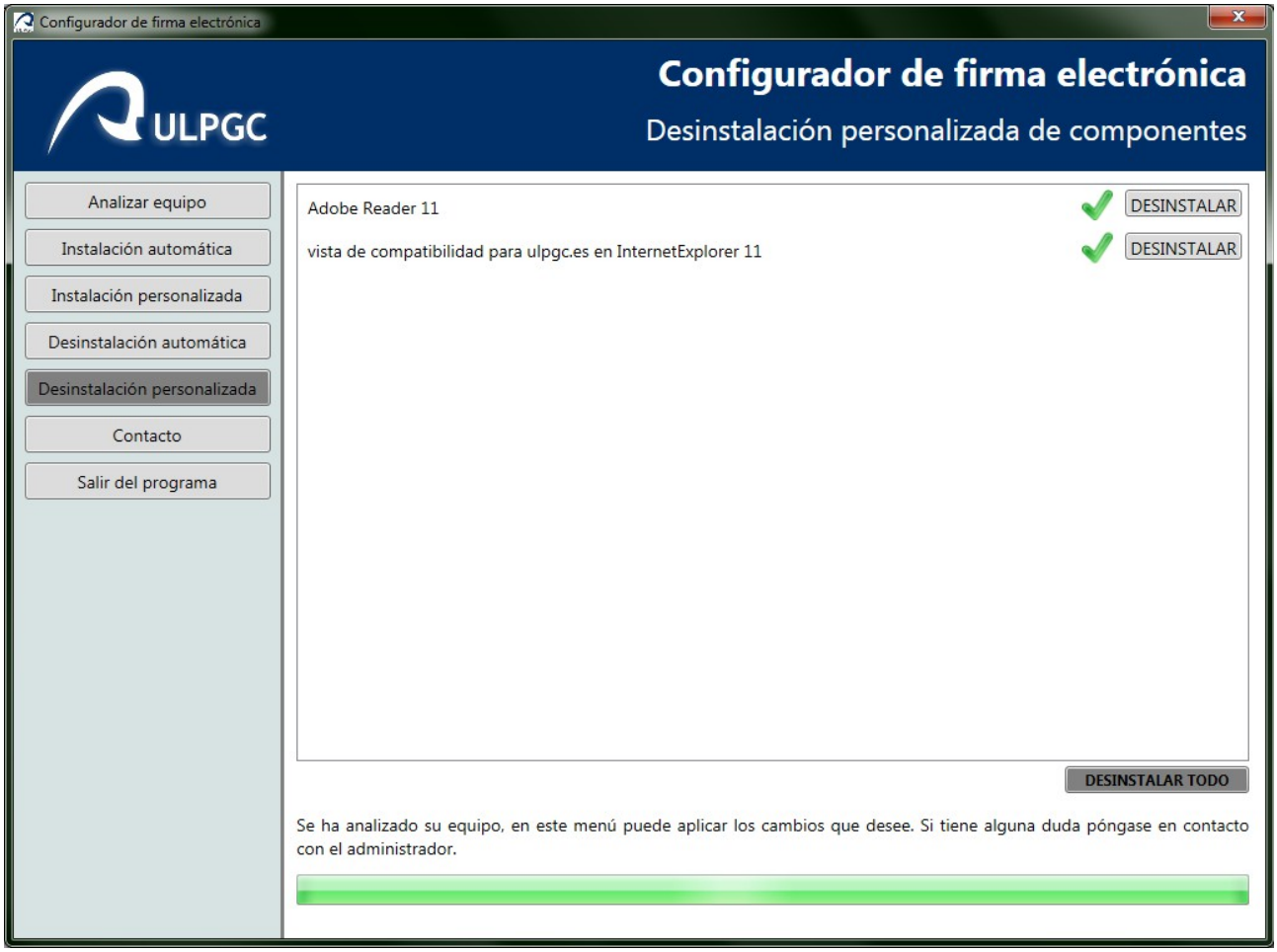


Ilustración 28: Menú de desinstalación personalizada

También se pueden eliminar todos los componentes instalados uno tras otro automáticamente, desde la opción 'Desinstalación automática'. Nos confirmará la decisión:

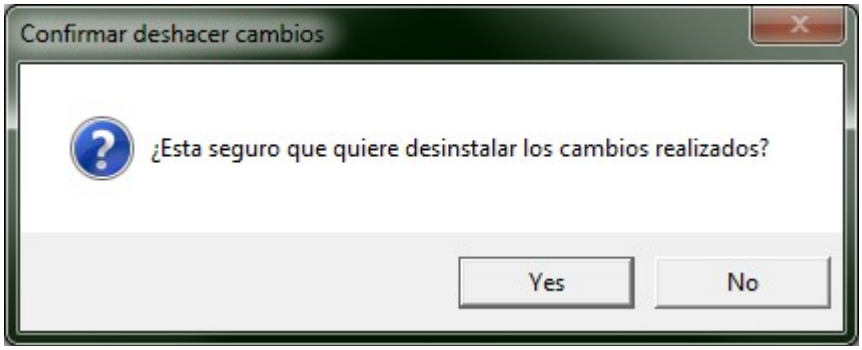


Ilustración 29: Confirmación de desinstalación automática

Si se acepta, el programa procede a desinstalar los componentes instalados, tras haber terminado, mostrará la siguiente pantalla:

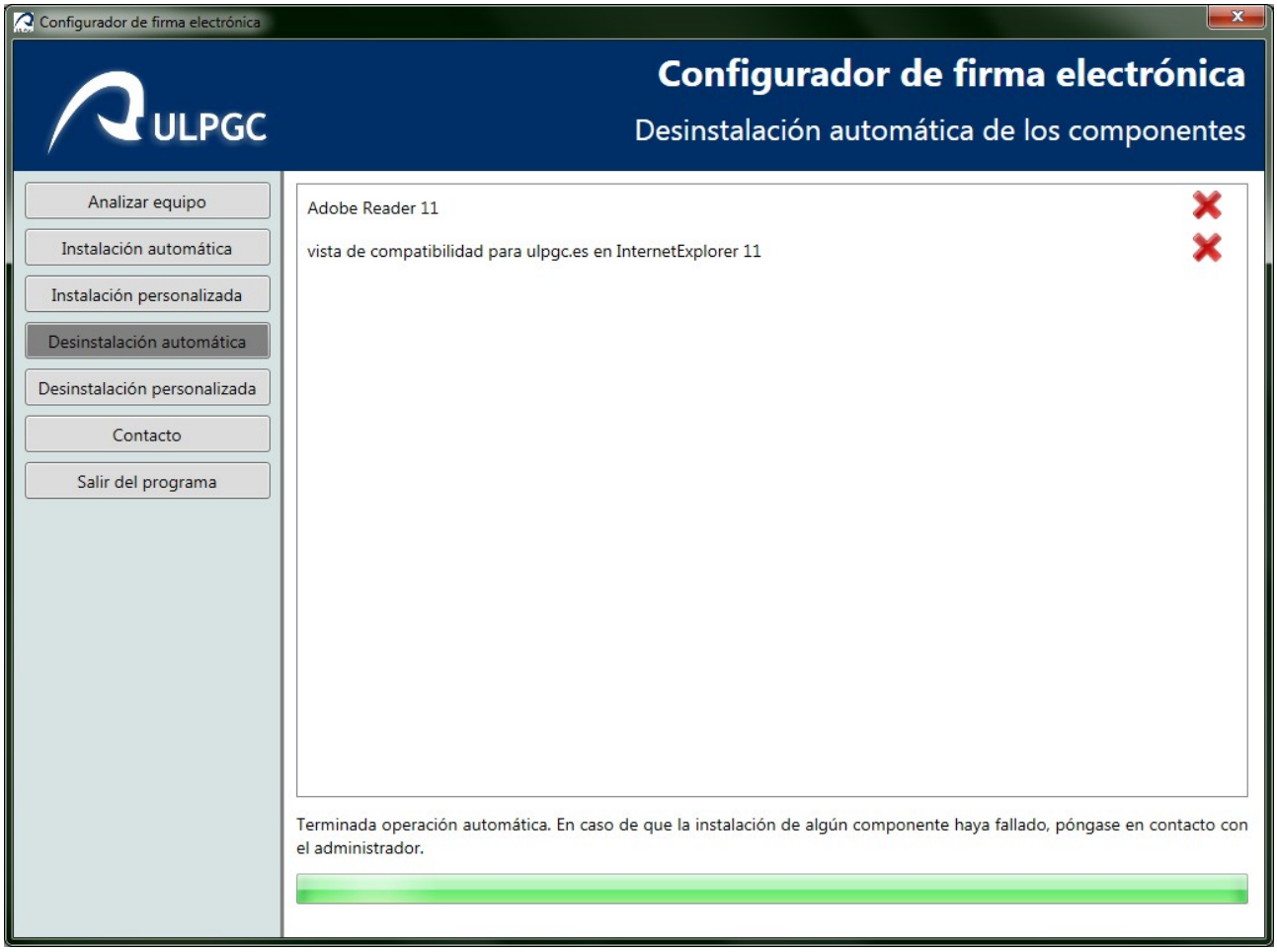


Ilustración 30: Menú de desinstalación automática

Ayuda e información de contacto

Si el usuario tiene cualquier duda, puede ponerse en contacto con los técnicos del departamento de informática de la ULPGC, puede consultar las formas de contactar en la opción 'Contacto' del menú de opciones

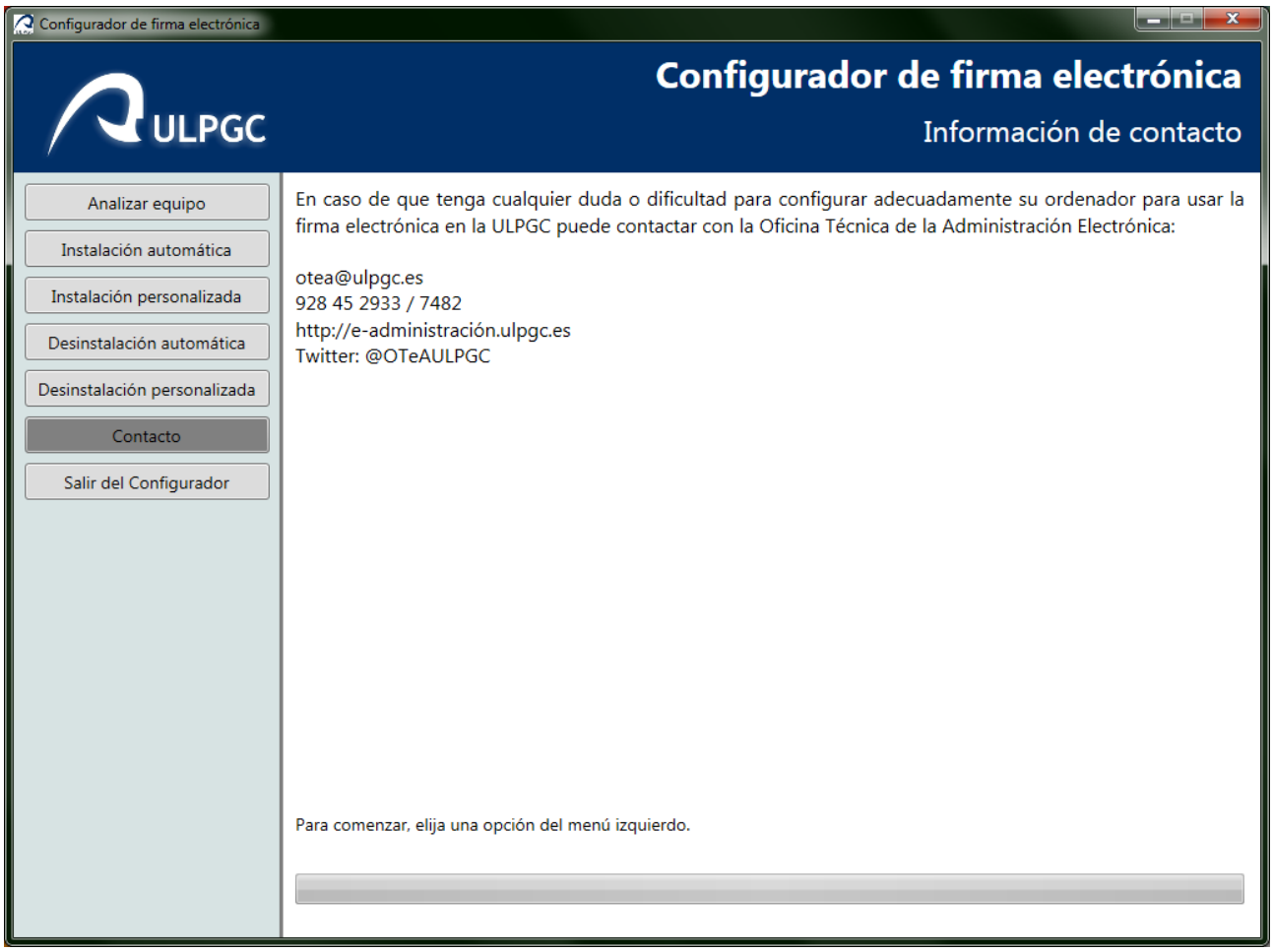


Ilustración 31: Menú de Contacto

Por último, si no se desea realizar ninguna otra acción, se puede cerrar el programa mediante la opción 'Salir del Configurador' o pulsando sobre el botón 'X'.