Universidad de las Ciencias Informáticas

Facultad 6

Un nuevo Front-End para la plataforma alasGRATO.

Trabajo de Diploma para Optar por el Título de Ingeniero en Ciencias Informáticas.

Autor:

Julio Antonio Villaverde Martínez

Tutores:

MS.c Aurelio Antelo Collado

Ing. Yunier René Pérez Valdes

Lic. Nelson Rodríguez Proenza

Junio 2009

Pensamiento

La simplicidad llevada al extremo, se convierte en elegancia.

Jon Franklin

Declaración de Autoría

Declaro que soy el único autor de este trabajo y autorizo al polo de Bioinformática de la Universidad de las Ciencias Informáticas; así como a dicho centro para que hagan el uso que estimen pertinente con este trabajo. Para que así conste firmo la presente a los días del mes de del año .

Julio Antonio Villaverde Martínez

Firma del Autor

MSc. Aurelio Antelo Collado

Firma del Tutor

Ing. Yunier René Pérez Valdes

Firma del Tutor

Lic. Nelson Rodríguez Proenza

Firma del Tutor

Datos de Contactos

Tutores:

Aurelio Antelo Collado.

Ingeniero Industrial.

Master en Matemática Aplicada.

Profesor Asistente con 4 años de experiencia.

Email: aantelo@uci.cu

Yunier René Pérez Valdes

Ingeniero en Ciencias Informáticas.

Email: yperezval@uci.cu

Nelson Rodríguez Proenza

Licenciado en Ciencias de la Computación.

Email: nelson.proenza@googlemail.com

Agradecimientos

A mi mama y a mi hermano, que con su apoyo, ejemplo y confianza han logrado que este sueño se convierta en realidad. Gracias por guiarme en la vida, sepan que todo el mérito de este trabajo se los debo a ustedes.

A mi abuela María Victoria, por ser la persona más paciente y cariñosa que he conocido en la vida, gracias a tu ayuda por fin hoy soy Ingeniero.

A mi Tío Ángel Felipe, que ha sido como un padre para mí, gracias por brindarme todo tu apoyo y estar en los momentos en que muchos no estuvieron.

A mi papá, por brindarme un ejemplo a seguir en mi vida profesional. Ojalá y algún día logre la mitad de todos tus méritos en las áreas del conocimiento.

A mis sobrinos, sepan que han sido una fuente inagotable de deseos de seguir superándome, para que un día se sientan orgullosos de mí. Los quiero mucho.

A Diana Mabel (puchi), por brindarme la posibilidad de sentirme hoy el padre más afortunado de este mundo, y saber siempre que puedo contar contigo. Gracias corazón.

A Ariadna, mi cuñada linda, gracias por siempre tener una palabra dulce o un gesto tranquilizador. En gran medida este trabajo también se debe a ti.

A mis primos Mónica y Alejandro, por siempre preocuparse por mí y por los míos. Les voy a estar eternamente agradecidos, gracias.

A Aurelio y a Yanet, gracias de todo corazón. Hoy no tengo suficientes palabras para demostrar todo mi agradecimiento hacia ustedes. Siempre podrán contar conmigo.

Agradecimiento

A Ramón Carrasco, por brindarme todo su conocimiento, gracias tutor. ¿Qué digo tutor? Amigo.

A Yunier y Yanet, gracias por tenerme tanta paciencia, de no ser por ustedes hoy no estaría aquí.

A mi mejor amiga Mayli Medero Cuadrado, por tantos momentos de alegría y angustias que pasamos juntos. Hoy por fin, al termino de este trabajo, celebramos un sueño que nos debíamos hace mucho tiempo. Siempre te querré.

A Jeanlup, Migue, Alberto, Lázaro, Tonysé, Cesar, Daniel, Rigo, Iosev, gracias de todo corazón por estar siempre ahí y brindarme una mano amiga, cuando, muchos de los que decían ser mis amigos me la negaron.

En general a todos mis amigos, discúlpenme que no los mencione uno a uno pero es que si lo hiciera sería infinita la lista. A todos siempre los tendré presente en mi corazón.

A todos mis profesores, por siempre estar presentes cada vez que tuve alguna duda. Mi formación se la debo a ustedes. Son lo mejor de esta universidad.

A las secretarias de la facultad 6 de la Universidad de las Ciencias Informáticas, por toda su paciencia infinita. La verdad es que las voy a extrañar muchísimo. Gracias por ayudarme tanto en esta dura tarea que comienza con este trabajo.

A todos aquellos que hicieron posible este sueño y, ¿Por qué no?, también a todos aquellos que hicieron todo lo posible para que no se hiciera realidad.

Dedicatoria

A mi hijo, espero que algún día te sientas tan orgulloso de mí como hoy lo estoy yo de ti.

Resumen

Como se sabe, una de las tendencias mundiales actuales es la elaboración de Interfaces de Usuarios o Front-End que permitan mayores funcionalidades. En consonancia con tales exigencias, se ha desarrollado un nuevo Front–End en el marco del Proyecto CITMA 0170006: "Plataforma Inteligente para la Predicción de Actividad Biológica de Compuestos Orgánicos", que se ejecuta en la Facultad de Bioinformática de la Universidad de las Ciencias Informáticas. La nueva Interfaz permitirá -al usuario final- incorporar funciona-lidades desarrolladas como plug-ins, entre las que se encuentran la visualización y edición de estructuras químicas en varios formatos, así como el cálculo de descriptores. Otras funcionalidades previstas, pero aún en fase de desarrollo son: hacer búsquedas de fragmentos similares tanto en 2D como en 3D y realizar análisis de datos mediante diferentes técnicas de inteligencia artificial. Por su concepción, esta nueva interfaz constituye la ayuda propia de la plataforma. También, contribuye y permite una rápida familiarización del investigador con el software, al facilitarle un aprovechamiento más óptimo de este recurso informático.

Palabras Clave: Front-End, Bioinformática, Arquitectura de Plug-ins, Arquitectura Informacional.

Tabla de Contenidos

Índice de figuras

Introducción

En la actualidad, los usuarios esperan productos generados por la tecnología informática con interfaces fáciles de utilizar y de aprender. Por eso, uno de los factores claves para el éxito de un software será el desarrollo de una interfaz de usuario (IU) efectiva, útil, segura y agradable. Ella permite la interacción del usuario con cualquier sistema o programa, además de ser el primer elemento con el cual se interactúa. De ahí que el diseño de una IU debe tenerse en consideración, toda vez que nos permite satisfacer expectativas mediante acciones agradables. Cabe señalar que los diseñadores están concientes de esto, pues cada día son mejores las propuestas de interfaz que muestran.

A manera de historia, vale recordar cómo -al principio- predominaban los software de pantallas oscuras, donde la interfaz de usuario era la propia consola del sistema operativo. O sea, el usuario, para poder trabajar, debía conocer un poco más "de lo básico" en computación: tenía que aprenderse todos -o al menos la mayoría- de los comandos del software; era la forma de comunicarse entonces. Esto traía grandes inconvenientes, puesto que- de no saberlos- el trabajo se veía retrasado. Afortunadamente, ya los diseños de interfaces de usuario han transitado por varios conceptos, los cuales van desde las interfaces sencillas hasta los llamados Front-End.

En la Universidad de las Ciencias Informáticas - específicamente en la Facultad no.6 - se trabaja en la creación de una "Plataforma Inteligente para la Predicción de Actividad Biológica de Compuestos Orgánicos". Se desea, también, que su IU facilite el trabajo de los especialistas que se dedican al diseño de fármacos asistido por computadora.

En estos momentos la plataforma cuenta con una interfaz que permite gestionar los diferentes módulos que la componen, pero sucede que esta primera versión ha sido desarrollada para que exprese la funcionalidad del sistema, sin emplear demasiado esfuerzo en la presentación gráfica.

Asimismo, dicha plataforma soporta un conjunto de módulos afines para el trabajo predictivo que, a su vez, son acoplados y manejados en tiempo de ejecución como plug-ins. Ellos se acoplan y toman como centro el Visualizador Molecular (alasGRATOViewer), además de mostrarse a través de ventanas independientes, lo cual imposibilita la interacción usuario-ordenador con la Plataforma

A esto se suman las experiencias y sugerencias obtenidas en las consultas sostenidas con los bioquímicos y químicos que colaboran en la ejecución de este proyecto, quienes hablan a favor de reajustar y reorientar indicadores, en función del usuario final de la plataforma; tales como: facilidad de aprendizaje, eficiencia, efectividad y satisfacción.

Por todo lo anterior, nos planteamos como **problema científico** de este trabajo : ¿Cómo mejorar la interacción de los usuarios con la plataforma alasGRATO?. Y su **objeto de estudio** será entonces la interacción hombre-ordenador, enmarcado en el **campo de acción** diseño e implementación de interfaces de usuario.

Para dar respuesta al problema antes mencionado se plantea, como **objetivo general**, desarrollar un Front-End dinámico y multiplataforma en función de la plataforma alasGRATO, pero con nuevas facilidades para el usuario final.

Con vistas a cumplimentar este objetivo general se trazaron los siguientes **objetivos específicos**:

- Desarrollar una nueva arquitectura informacional para el producto alasGRATO.
- Desarrollar una nueva arquitectura de plug-ins para el producto alasGRATO.
- Desarrollar una nueva interfaz para el producto alasGRATO.
- Validar el Front-End a desarrollar para el producto alasGRATO.

Para alcanzar dichos objetivos se planteó desarrollar las siguientes **tareas**:

- Organización y rotulación de Front-End.
- Estudio del lenguaje de etiquetado XML.
- Estudio de librerías para el trabajo con el leguaje de etiquetado XML.
- Selección de los patrones adecuados para la arquitectura de plug-ins.

- Gestión de plug-ins mediante la descripción de funcionalidades en un formato estándar
- Estudio de librerías para el trabajo visual del producto alasGRATO.
- Reutilización de nuevos componentes visuales para el producto alasGRATO.
- Selección de criterios de evaluación y de validación.
- Selección de los datos necesarios para la realización de las pruebas.

Este trabajo de diploma se ha estructurado como sigue: resumen, introducción, cuatro capítulos, conclusiones generales, recomendaciones, referencias bibliográficas, bibliografía y anexos. A continuación, se reseñan los contenidos de cada capítulo.

Capítulo 1. Fundamentación Teórica. En este capítulo se presenta el marco teórico estudiado y seleccionado para desarrollar nuestro objeto de estudio, orientado a la interacción usuario-ordenador. Se presenta una panorámica sobre los conceptos y herramientas necesarios para alcanzar los objetivos planteados .

Capítulo 2. Características del Sistema. Se realiza una breve descripción de la solución propuesta, sus requisitos funcionales y no funcionales, así como los actores que intervienen en ella. Se presenta el modelo de dominio del sistema, el diagrama de caso de usos del sistema y la descripción de los mismos.

Capítulo 3. Diseño del Sistema. Se describe la representación arquitectónica del sistema. Se hace énfasis en los patrones de arquitectura utilizados, asi como los patrones de diseño que fueron empleados, además, se aborda la arquitectura de plug-ins del Front-End. Se incluye la descripción, a través, de un esquema del fichero XML de un plug-in. Se abordan los temas de arquitectura informacional, usabilidad y se lleva a cabo el diseño del sistema.

Capítulo 4. Describe la implementación del sistema, mostrando los diagramas de componentes y una serie de pasos obligatorios a seguir para implementar un nuevo plug-in. Además se presenta todo lo relacionado con las pruebas realizadas al sistema.

Capítulo 1

Fundamentación Teórica

No en pocas ocasiones son utilizados los términos de *biología computacional o biocomputación* para referirse a disciplinas científicas estrechamente vinculadas entre si, como son: la informática, la matemática aplicada, la estadística, la química y la bioquímica; por sólo mencionar algunas de ellas. En conjunto, bajo el referente de *bioinformática*, ellas han tributado a solucionar problemas científicos en otros campos de estudios, a saber: en la biología de sistemas, la evolución molecular, el diseño de fármacos, entre otros. De ahí que *bioinformática* defina aquella disciplina que gestiona y analiza datos biológicos, mediante la aplicación de técnicas computacionales

Actualmente, para facilitarles el trabajo a los investigadores que centran sus estudios en los campos de la bioinformática, existen compañías en todo el mundo que invierten millones de dólares en el desarrollo de software o aplicaciones informáticas. De esta forma, se garantiza reducir el costo del tiempo invertido en las investigaciones realizadas.

Un aspecto importante en el desarrollo de estas aplicaciones es tener en cuenta una concepción de interfaz de usuario intuitiva, que facilite la comunicación entre persona-ordenador.

1.1. Interacción Persona-Ordenador.

Hoy en día, los ordenadores son utilizados por disímiles personas y para todo tipo de objetivos. Algunos sistemas informáticos funcionan con poca intervención humana, pero la mayor parte son interactivos, en los cuales, las personas o usuarios están involucrados en la resolución de tareas. En estos casos la interacción

persona-ordenador (IPO) - facilitada por la interfaz de usuario- es un factor bien importante para el éxito de todo sistema interactivo.

Así, puede definirse como Interacción persona-ordenador, a la disciplina que estudia el intercambio de información entre las personas y los ordenadores, cuyo objetivo será el garantizar un intercambio eficiente, que minimice errores, e incremente la satisfacción. En resumen, que haga más productiva las tareas que envuelven tanto a personas como a ordenadores [1].

A juzgar por la bibliografía consultada, para el desarrollo de la IPO han influido positivamente algunos factores. Según Rebeca [2], estos son: La creatividad humana, el estado del arte de la tecnología, y el mercado de los ordenadores.

Para Shackel [3], al comienzo los ordenadores eran grandes máquinas que procesaban instrucciones en lenguaje máquina en modo bash. Las operaciones se organizaban en grupos o lotes de trabajo, con largos tiempos de espera; las entradas se realizaban por tarjetas perforadas y la salida por impresoras línea a línea. Por lo que los usuarios eran, básicamente, los propios programadores.

Luego, con el vertiginoso avance de la tecnología y de los sistemas operativos, aparecieron las pantallas con fondo oscuro, letras verdes y los sistemas basados en caracteres, los cuales exigían- entre otras cosas- la memorización de comandos, además de ser ejecutados en la consola del sistema [2] y [4]; modo que dificultaba la interacción del usuario con el ordenador, ya que tenía que dominar la aplicación a utilizar para alcanzar las metas fijadas.

Con la aparición del mercado de computadoras, la tecnología se pone a disposición no sólo de la comunidad científica. Ahora los ordenadores están a disposición de todas las personas del mundo y, por ello, las aplicaciones desarrolladas para ejecutarse en consolas quedaban atrás, dándose paso a las nuevas formas de interacción con los ordenadores. Es así como surge la idea de cambiar la interfaz ya conocida por una nueva idea: desarrollar interfaces gráficas que generen escenas familiares e intuitivas, las cuales deben facilitar la interacción y la comunicación usuario-sistema [5].

El desarrollo de estas interfaces gráficas pasaba a ser una de las principales etapas en el desarrollo del software. Ahora sus productores no sólo se preocuparían por el desarrollo en sí de la aplicación. A partir de este momento es que –también- empezarían a tener en cuenta al usuario final en dichas aplicaciones.

Si tenemos en consideración a Marrero [6] la interfaz gráfica - como artefacto tecnológico- transita por tres períodos fundamentales. El primero, su nacimiento , que tiene lugar en el año 1973 en el centro de investigación Xerox Alto, donde el objetivo básico era encontrar un modelo óptimo de interacción persona-ordenador. El segundo, es cuando pasa por un proceso de eclosión y madurez, definiéndose sus elementos básicos. En el tercero, se convierte en un producto de consumo estético dentro de los sistemas interactivos, donde la interfaz-más allá de un medio de interacción óptimo-, se transforma en un objeto inteligente, abierto a los procesos de personalización por parte del usuario.

Por otra parte, la percepción e impresión que el usuario posee de una aplicación lo determina la interfaz. Sin embargo, no pocas veces existen dificultades con relación al uso de las nuevas metodologías para lograr desarrollarla bien. Debora Hix [7] resume con acierto esta cuestión al afirmar que: "Una buena interfaz es como el teléfono o como la luz eléctrica; cuando funciona nadie la nota. Una buena interfaz parece algo obvio, pero lo que no parece ser tan obvio es cómo desarrollar una interfaz que tenga un alto grado de usabilidad".

1.2. Usabilidad

El concepto de Usabilidad se define como "el grado en que un producto puede ser utilizado por usuarios específicos para conseguir objetivos específicos con efectividad, eficiencia y satisfacción en un determinado contexto de uso" [8].

Como puede apreciarse, la definición de este concepto implica la fusión de 3 factores sumamente importantes: los usuarios, las condiciones y las necesidades específicas. Ello significa que la usabilidad de un sistema no es un atributo independiente de éste, sino que está estrechamente relacionado con el entorno de uso y con los usuarios finales que utilizarán el software.

Por tanto podemos plantear que la usabilidad es un concepto bien abstracto y que su medida estará dada a través de cinco factores fundamentales, conocidos como atributos de usabilidad o características orientadas al usuario. Estos, según Xavier [9], son los siguientes:

- **Facilidad de Aprendizaje**: Trata de medir el tiempo empleado por el usuario en el aprendizaje de las funcionalidades básicas de un sistema determinado, para llevar a cabo una tarea específica. La facilidad estará dada en operar la tarea en el menor tiempo posible. Este atributo es recomendable medirlo en usuarios noveles.

- **Eficiencia**: Mide el número de operaciones que el usuario puede realizar en una unidad de tiempo dado. O sea, cuanto mayor sea la usabilidad del sistema, más rápido será aprehendido por el usuario al poder realizar el trabajo deseado.
- **Recuerdo en el tiempo**: Refleja el recuerdo de los usuarios acerca de cómo funciona el sistema, después de un tiempo, sin ser utilizado. O sea que ,para usuarios intermitentes, el sistema tiene que estar preparado para cuando el usuario -luego de haber pasado un tiempo-, tenga necesidad de volver a utilizarlo , sin tener que empezar desde cero otra vez, para lograr la tarea que se proponga realizar.
- **Tasa de Errores**: Refiere la cantidad de errores que comete un usuario determinado al realizar una tarea específica. Es decir que, mientras menos errores cometa el usuario mayor será la usabilidad que tendrá el sistema.
- **Satisfacción**: Muestra la impresión que el usuario obtiene del sistema y es, de todos los atributos, el más subjetivo.

A menudo, muchas personas identifican la usabilidad de un sistema informático con las características de la interfaz gráfica y no con el nivel de interacción del sistema con el usuario final. También se relaciona con el modo en que se realizan las operaciones con el sistema, lo cual es otra forma errónea de concebir la usabilidad de un sistema [9]. Es importante señalar que la interfaz gráfica de un software es la parte visible de tal interacción, además de ser – sin dudas- una parte importante del sistema. De ahí que un buen diseño de ella puede hacer que un sistema aumente su nivel de usabilidad.

Así, la mayoría de los sistemas informáticos interactúan con las personas a través de la interfaz gráfica (ventanas, iconos, menús, mensajes y la forma en que están organizados) y mediante la información provista en la documentación (manuales, tutoriales y ayuda en línea). Por lo que se puede plantear que para que un producto de software resulte “usable”, todos sus componentes habrán de ser consistentes y trabajar armónicamente juntos [10].Por ello, es necesario tener presente la arquitectura informacional en el momento de concebir y desarrollar un software, con vistas a lograr su usabilidad.

1.3. Arquitectura de Información de un Software

Las interfaces gráficas de las aplicaciones informáticas están orientadas a los usuarios, quienes actualmente han dejado de ser consumidores de dichos productos para convertirse en principales actores durante el desarrollo

de las mismas. A ello se suma el acelerado crecimiento informativo, la evolución del hardware, así como la creciente necesidad que tienen los usuarios de tener una información organizada y precisa. Se hace necesario, entonces, trabajar en función de ayudar a que ellos puedan encontrar y gestionar la información de manera efectiva, utilizando un buen diseño, organización, etiquetado y navegación en los sistemas que se desarrollen.

Por lo anteriormente expuesto es que surge la disciplina Arquitectura de la Información, encargándose ella de estudiar la organización de la información con el objetivo de permitirle al usuario encontrar su vía de cómo navegar hacia el conocimiento y la comprensión. Para el reconocido especialista en el tema, Paul Kahn, ella "es solo la construcción de la organización de la información."[11]

Esta disciplina tiene la capacidad de adaptarse a los nuevos requerimientos que tiene el software en la actualidad, al aplicar técnicas ya conocidas, entre las cuales encontramos las siguientes:

- **Técnicas de interacción con el usuario**: consiste en concertar tanto reuniones, como entrevistas y encuestas con los usuarios finales del sistema en cuestión. Además de la realización de diseños por escenarios, diseño participativo y esquemas de técnicas de interacción con el usuario final.
- **Técnicas de interacción con el contexto**: se basa principalmente en la evaluación de productos similares. Es importante para el análisis de la competencia y la creación de esquemas de interacción con el contexto.
- **Técnicas matemáticas (co-ocurrencia)**: Esta otra técnica se basa en la organización de tarjetas (card sorting) y en el análisis de secuencias. Además de la creación de técnicas matemáticas.
- **Técnicas de representación de Información**: Técnica que se basa en la creación de diagramas, representación de etiquetas, prototipado (creación de maquetas) y esquemas de técnicas de representación de información.

Hasta el momento, la experiencia acumulada indica que al desarrollar una buena arquitectura informacional se puede garantizar el 80 % del éxito total de las aplicaciones. El usuario final quedará satisfecho y la aplicación informática no perecerá, toda vez que ganará en cuanto a su usabilidad.

Una de las principales actividades en esta área es el desarrollo de nuevos sistemas de interfaces para el usuario, lo que implica desarrollar nuevos dispositivos de interfaz, nuevas técnicas de graficación por computadora y técnicas de diálogo [12].

1.4. Interfaces Gráficas de Usuarios

La interfaz gráfica de usuario (GUI) permite aprovechar las capacidades de despliegue gráfico de la computadora, al hacer más sencillo el uso de los programas [13]. Ella se conoce también como un espacio de trabajo bidimensional, donde, como en un escritorio, se tienen los implementos que el usuario utiliza en su trabajo diario. Según Jacobo [14] algunos de los componentes gráficos comunes en las interfaces gráficas son:

- **Apuntador**: símbolo que aparece en la pantalla y que se desplaza para seleccionar objetos y comandos.
- **Dispositivo apuntador**: dispositivo, como el ratón, que permite seleccionar objetos de la pantalla.
- **Iconos**: Pequeñas imágenes que representan comandos, archivos o ventanas.
- **Menús**: Opciones o comandos agrupados en forma de listas.
- **Ventanas**: Son áreas que dividen la pantalla, y en cada una de ellas puede ejecutarse un programa o abrir un archivo diferente.
- **Escritorio**: Es el área de la pantalla donde los iconos son agrupados.

La arquitectura comúnmente usada para el desarrollo de la GUI es a través del patrón modelo-vista-controlador (MVC), que permite separar los componentes gráficos, el control de los dispositivos periféricos y eventos, así como sus efectos en el modelo. (ver figura 1.1)

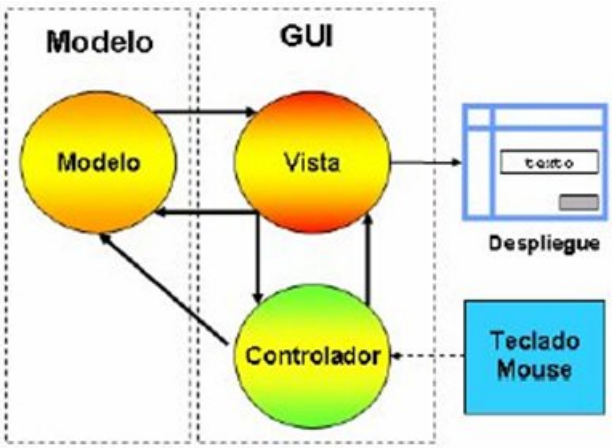


Figura 1.1: Patrón MVC como base de una GUI

Una tendencia adicional es la personalización del diseño gráfico de los elementos de la interfaz gráfica. Para ello se ha introducido el concepto de tema, donde existe una separación entre el trabajo de los diseñadores gráficos y los programadores [15]. De esta manera, puede cambiarse el diseño y la distribución de los controles sin tener que recompilar el código encargado del GUI. (ver figura 1.2)

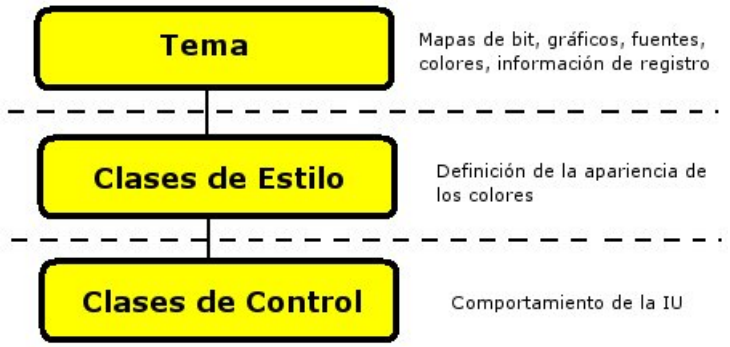


Figura 1.2: Arquitectura de una GUI que integra la metáfora de tema

1.4.1. Características de una Interfaz de Usuario.

Las características que deben presentar las interfaces de usuarios según Toledo [16] son:

- Poseer un monitor gráfico de alta resolución.
- Poseer un dispositivo apuntador (típicamente un ratón).
- Promover la consistencia de la interfaz entre programas.
- Los usuarios podrán ver en la pantalla los gráficos y textos, tal como se verán impresos.
- Siguir el paradigma de la interacción objeto-acción.
- Permitir la transferencia de información entre programas.
- Manipular en la pantalla directamente los objetos y la información.
- Proveer elementos de interfaz estándar como menús y diálogos.
- La existencia de una muestra visual de la información y los objetos (iconos y ventanas).
- Proporcionar respuesta visual a las acciones del usuario.
- Existencia de información visual de las acciones y modos del usuario/sistema (menús, paletas).

- Existencia de controles gráficos (widgets) para la selección e introducción de la información.
- Permitir a los usuarios personalizar la interfaz y las interacciones.
- Proporcionar flexibilidad en el uso de dispositivos de entrada (teclado/ratón).

Una característica importante de la interfaz de usuario es que permite manipular los objetos e informaciones de la pantalla, no sólo presentarla. Además, los usuarios para utilizarlas deben conocer (o aprender) : cómo está organizado el sistema (ficheros, directorios en Win95), los diferentes tipos de íconos y efectos de las acciones sobre ellos, los elementos básicos de una ventana, el uso de los controles y el uso del ratón.

1.4.2. Principios para el Diseño de Interfaces de Usuario.

Existen principios básicos para el diseño e implementación de una IU, los cuales, según Leopoldo Sebastián [17] son los siguientes:

- **Anticipación**: Las aplicaciones deberían intentar anticiparse a las necesidades del usuario y no esperar a que el usuario tenga que buscar la información, recopilarla o invocar las herramientas que va a utilizar.
- **Autonomía**: El entorno de trabajo debe estar a disposición del usuario. Se debe dar un ambiente flexible para que se pueda aprender rápidamente a usar la aplicación. Es importante utilizar mecanismos indicadores del estado del sistema, que mantengan a los usuarios alertas e informados, en ubicaciones fáciles de visualizar.
- **Percepción del Color**: Aunque se utilicen las convenciones de color, deberían usarse otros mecanismos secundarios, que permitan proveer la información a aquellos usuarios con problemas para visualizarlos.
- **Valores por Defecto**: No se debe utilizar la palabra "Defecto" en una aplicación o servicio. Puede ser reemplazada por "Estándar" o "Definida por el Usuario", "Restaurar Valores Iniciales" o algún otro término específico que describa lo que está sucediendo. Los valores por defecto deberían ser opciones inteligentes y sensatas. Además, los mismos tienen que ser fáciles de modificar.
- **Consistencia**: Para lograr una mayor consistencia en la IU se requiere profundizar en diferentes aspectos que están catalogados por niveles. Se realiza un ordenamiento de mayor a menor consistencia:

1. **Interpretación del comportamiento del usuario**: la IU debe comprender el significado que le atribuye un usuario a cada requerimiento. Ejemplo: mantener el significado de los comandos abreviados (shortcut-keys) definidos por el usuario.

   a) **Estructuras invisibles**: se requiere una definición clara de ellas, de lo contrario el usuario nunca podría llegar a descubrir su uso. Ejemplo: la ampliación de ventanas mediante la extensión de sus bordes.

   b) **Pequeñas estructuras visibles**: se puede establecer un conjunto de objetos visibles capaces de ser controlados por el usuario, que permitan ahorrar tiempo en la ejecución de tareas específicas. Ejemplo: ícono y/o botón para impresión.

   c) **Una sola aplicación o servicio**: la IU permite visualizar la aplicación o servicio utilizado como un componente único. Ejemplo: La IU despliega un único menú, pudiendo además acceder al mismo mediante comandos abreviados.

   d) **Un conjunto de aplicaciones o servicios**: la IU visualiza la aplicación o servicio utilizado como un conjunto de componentes. Ejemplo: La IU se presenta como un conjunto de barras de comandos desplegadas en diferentes lugares de la pantalla, las cuales pueden ser desactivadas de forma independiente.

   e) **Consistencia del ambiente**: la IU se mantiene en concordancia con el ambiente de trabajo. Ejemplo: La IU utiliza objetos de control como menús o botones de comandos, de manera análoga a otras IU que se usen en el ambiente de trabajo.

   f) **Consistencia de la plataforma**: La IU concuerda con la plataforma. Ejemplo: La IU tiene un esquema basado en ventanas, el cual es acorde al manejo del sistema operativo Windows.

- **Eficiencia del Usuario**: Se debe priorizar la productividad del usuario antes que la productividad de la máquina. Si el usuario debe esperar la respuesta del sistema por un período prolongado, habrá pérdidas económicas para la organización. Los mensajes de ayuda deben ser sencillos y proveer respuestas a los problemas. Los menús y etiquetas de botones deberían tener las palabras claves del proceso.

- **Ley de Fitt**: El tiempo para alcanzar un objetivo está en función de la distancia y tamaño del objetivo. Conviene por ello usar objetos grandes para las funciones importantes.

- **Interfaces Explorables**: Siempre que sea posible se debe permitir que el usuario pueda salir ágilmente,

dejando una marca que indique el estado de avance de su trabajo, con vistas a continuarlo en otro momento.

- **Objetos de Interfaz Humana**: Los objetos de interfaz humana no son necesariamente los objetos que se encuentran en los sistemas orientados a objetos. Estos pueden ser vistos, escuchados, tocados o percibidos de alguna forma. Además, estos objetos deberían ser entendibles, consistentes y estables.
- **Uso de Metáforas**: Las buenas metáforas crean figuras mentales fáciles de recordar. La interfaz puede contener objetos asociados al modelo conceptual de forma visual, con sonido u otra característica perceptible por el usuario que le ayude a simplificar el uso del sistema.
- **Curva de Aprendizaje**: El aprendizaje de un producto y su usabilidad no son mutuamente excluyentes. Lo ideal es que la curva de aprendizaje sea nula, y que el usuario principiante pueda alcanzar el dominio total de la aplicación sin esfuerzo.
- **Reducción de Latencia**: Siempre que sea posible, el uso de tramas (multi-threading) permite colocar la latencia en segundo plano (background). Las técnicas de trabajo multitarea posibilitan el trabajo ininterrumpido del usuario y se realizan las tareas de transmisión y computación de datos en un segundo plano.
- **Protección del Trabajo**: Se debe asegurar que el usuario nunca pierda su trabajo, ya sea por error de su parte, problemas de transmisión de datos, de energía, o alguna otra razón inevitable.
- **Auditoría del Sistema**: La mayoría de los navegadores de Internet (browsers), no mantienen información acerca de la situación del usuario en el entorno, pero para cualquier aplicación es conveniente conocer un conjunto de características, tales como: hora de acceso al sistema, ubicación del usuario en el sistema y lugares a los que ha accedido, entre otros. Además, el usuario debería poder salir del sistema y -al volver a ingresar- continuar trabajando en el lugar dónde se había quedado.
- **Legibilidad**: Para que la IU favorezca la usabilidad del sistema de software, la información que se exhiba en ella debe ser fácil de ubicar y de leer. Para lograr este resultado deberá tenerse en cuenta que: el texto que aparezca en la IU tenga un alto contraste; se deben utilizar combinaciones de colores, por ejemplo, el texto en negro sobre fondo blanco o amarillo suave. El tamaño de las fuentes tiene que ser lo suficientemente grande como para poder ser leído en monitores estándar. Es importante hacer clara la presentación visual (colocación/agrupación de objetos) y evitar la presentación excesiva de información.

- **Interfaces Visibles**: El uso de Internet ha favorecido la implementación de interfaces invisibles. Esto significa que el usuario siempre ve una página específica, pero nunca puede conocer la totalidad del espacio de páginas de Internet. La navegación en las aplicaciones debe ser reducida a la mínima expresión. El usuario debe sentir que se mantiene en un único lugar y que el que va variando es su trabajo. Esto no solamente elimina la necesidad de mantener mapas u otras ayudas de navegación, sino que además brindan al usuario una sensación de autonomía.

Para poder cumplir con los principios para el diseño de interfaces de usuario, es necesario conocer los componentes visuales que existen en el lenguaje de programación Java.

1.5. Componentes visuales en Java.

En la mayoría de los sistemas operativos actuales, se ofrece una cantidad de código para simplificar la tarea de programación. Este código toma la forma, normalmente, de un conjunto de librerías dinámicas que las aplicaciones pueden llamar cuando lo necesiten. Pero la Plataforma Java está pensada para ser independiente del sistema operativo subyacente, por lo que las aplicaciones no pueden apoyarse en funciones dependientes de cada sistema en concreto. La Plataforma Java ofrece un conjunto de librerías estándares, que contienen funciones reutilizables y disponibles en los sistemas operativos actuales.

Las librerías de Java tienen tres propósitos dentro de la Plataforma Java. Al igual que otras librerías estándares, ofrecen al programador un conjunto bien definido de funciones para realizar tareas comunes, como manejar listas de elementos u operar de forma sofisticada sobre cadenas de caracteres. Además, las librerías proporcionan una interfaz abstracta para tareas que son altamente dependientes del hardware de la plataforma destino y de su sistema operativo. Tareas tales como manejo de las funciones de red o acceso a ficheros, suelen depender fuertemente de la funcionalidad nativa de la plataforma destino. En el caso concreto anterior, las librerías java.net y java.io implementan el código nativo internamente, y ofrecen una interfaz estándar para que aplicaciones Java puedan ejecutar tales funciones. Finalmente, no todas las plataformas soportan todas las funciones que una aplicación Java espera. En estos casos, las librerías bien pueden emular esas funciones usando lo que esté disponible, o bien ofrecer un mecanismo para comprobar si una funcionalidad concreta está presente.

1.5.1. Librería gráfica AWT

Java AWT (Abstract Window Toolkit) es una extensión al lenguaje Java, la cual permite al programador crear interfaces gráficas en las que se pueden manejar imágenes, botones, menús y otros elementos básicos para interactuar con el usuario.

Es una librería portable entre Solaris, Windows 95/NT y Mac System 7.X para crear aplicaciones o applets. Además, se pueden correr en ambientes como IRIX, SunOS, HP/UX, Linux y cualquiera que soporte Netscape 2.0 o superior, y Microsoft Internet Explorer 3.0 o superior.

AWT provee un conjunto de clases para programar. Es la conexión para el programador entre su aplicación y la interfaz gráfica nativa del Sistema Operativo. AWT le esconde los detalles internos del GUI que su aplicación usa y así permite un alto nivel de abstracción. Toma el mínimo común denominador entre las plataformas para mantener portabilidad. No se permiten cosas tales como barras de botones flotantes o Globos de Ayuda.

Es un paquete de Java y puede ser usado en cualquier programa de Java al importar la clase java.awt.* por medio de la instrucción "import".

La estructura del AWT es simple: Los componentes son agregados en contenedores por medio de manejadores de capas. Tiene una variedad de manejo de eventos para menús, tipos de letras y clases gráficas.

Además de usar los elementos tal y como se definen, estos pueden combinarse con el uso de hilos, audio, I/O y clases de red dentro de elementos de AWT. Requiere de programación para crear diálogos y ventanas predefinidas [18].

1.5.2. Librería gráfica SWT

El Standard Widget Toolkit (SWT) ha sido creado por IBM como reemplazo de la Java Abstract Windowing Toolkit (AWT) y Swing. De manera más precisa, el SWT es un conjunto de widgets para desarrolladores en Java, que ofrece una API portable, además de brindar una integración muy ligada con la interfaz gráfica de usuario, nativa al sistema operativo de cada plataforma. Esto puede entrar en contradicción con la filosofía que tiene Java de ser independiente de la plataforma, pero ofrece ventajas para la unicidad del aspecto del GUI diseñado, sea cual sea la plataforma sobre la que se ejecute. Hasta el momento los entornos a los que se

ha portado SWT son Windows, Linux GTK, Linux Motif, Solaris Motif, AIX Motif, HPUX Motif, Photon, QNX, y Mac OS X [19].

Los gráficos en Java han tenido una larga y exitosa evolución: empezaron con el básico AWT, viéndose ampliados por el potente paquete Swing.

SWT ofrece un conjunto de elementos que hacen uso directo de los controles nativos, a través de la Interfaz Nativa de Java (JNI). Si los controles no están ofrecidos por el sistema operativo, SWT crea los suyos propios según las necesidades. Esto significa que se precisa de un código nativo para poder funcionar en cada sistema operativo, pero IBM ha sido capaz de adaptar SWT a un buen número de sistemas. Es destacable la importancia que ha tenido SWT, al permitir el entorno gráfico de desarrollo que viene con Eclipse y que debe ser utilizado en los plug-ins desarrollados para este. Eclipse es una herramienta de desarrollo altamente potente, de libre distribución, código abierto y adaptable a cualquier lenguaje de programación.

SWT se compone esencialmente de tres elementos:

- En el nivel inferior se encuentra una librería nativa que interacciona directamente con el sistema operativo. Es la denominada JNI, la Java Native Interface. Al ser la parte más dependiente de la plataforma, debe ser portada en función de la misma.
- Por encima de la anterior capa está la clase Display, la interfaz por la que el SWT se comunica con la interfaz gráfica.
- El nivel superior lo forma la clase shell, el cual representa el tipo de ventana de más alto nivel y es donde se alojan los widgets, controles o componentes. El shell es la parte de la interfaz que está directamente controlada por el sistema de ventanas del sistema operativo. La clase shell es hija de la clase Display. Ella es la ventana o marco principal a partir de la cual se construye el resto de la interfaz, o bien, la hija de otro shell. Es el ejemplo más común de las ventanas de diálogo.

1.5.3. Librería gráfica SWING

El paquete Swing es parte de la JFC (Java Foundation Classes) en la plataforma Java. La JFC provee facilidades para ayudar a construir GUIs. Swing abarca componentes tales como: botones, tablas, o marcos. Las componentes Swing se identifican porque pertenecen al paquete javax.swing.

Swing existe desde la JDK 1.1 (como un agregado). Antes de su existencia, las interfaces gráficas con el usuario se realizaban a través de AWT (Abstract Window Toolkit), de quien Swing hereda todo el manejo de eventos. Usualmente, para toda componente AWT, existe una componente Swing que la reemplaza, por ejemplo, la clase Button de AWT es reemplazada por la clase JButton de Swing (el nombre de todas las componentes Swing comienza con "J").

Las componentes de Swing utilizan la infraestructura de AWT, incluyendo el modelo de eventos AWT, el cual rige cómo un componente que reacciona a eventos de teclado, del ratón, entre otros. Es por esto que la mayoría de los programas Swing necesitan importar dos paquetes: AWT: java.awt.* y java.awt.event.* [20].

1.5.4. Otras Librerías Implementadas con SWING

En primer lugar Flamingo, es un paquete de componentes para Windows, gratis y de gran utilidad, con una funcionalidad visualmente similar a la del menú superior del Vista Explorer y Microsoft Office 2007. Los componentes incluidos, básicamente librerías JAVA, aportan visualizaciones consistentes en el procesador y permiten nuevas configuraciones al usuario. Además, permite rediseñar el Explorador de Windows, un visor de archivos SVG (archivos que por lo general contienen imágenes) y un redimensionador de íconos operativos con los formatos de imagen IP.ico, IS.ico, JPG.ico y JPEG.ico. Para que Flamingo funcione es necesario tener instalado la máquina virtual de Java 6.0 o superior [21], [22].

En segundo lugar SwingX, que es una extensión del propio Swing, donde se pretenden mejorar algunos de los controles existentes e incluir otros nuevos. Así, por ejemplo, SwingX añade funciones avanzadas de ordenación, filtrado, selección y resaltado a controles básicos de Swing, como tablas (JTable), árboles (JTree) y listas (JList). Añade nuevas funcionalidades a otros controles como cuadros de diálogo (JDialog) o paneles (JPanel), e incluye algunos controles nuevos no presentes en Swing como JXTreeTable, JXLoginPanel o JXDatePicker, para la presentación de tablas jerarquizadas, controles de autenticación o cuadros de selección de fecha.

En tercer lugar Substance, que proporciona uno de los más atractivos look&feel para aplicaciones Swing que existen actualmente, y aunque históricamente ha tenido algunos problemas de rendimiento, parece que ya -en las últimas revisiones- ha sido mejorando en ese sentido. Además, Substance tiene el valor añadido de incluir soporte para las otras dos librerías comentadas anteriormente, Flamingo y SwingX.

Por ultimo MyDoggy, es una fuente abierta de java desarrollada, para la gestión de ventanas secundarias, dentro de la ventana principal. MyDoggy permite mover, cambiar el tamaño o la extracción de las ventanas secundarias. Además, presta apoyo a la gestión de contenidos de la ventana principal. Por ultimo apoya el concepto de "futuros", utilizando grupos mydoggy [23], [24].

1.6. Herramientas y metodologías utilizadas.

El presente trabajo esta enmarcado dentro del proyecto "Plataforma para la predicción de actividad biológica de compuestos orgánicos", el cual tiene su arquitectura definida en trabajos anteriores de Marrero [25]. Es por ello que en este epígrafe sólo se abordarán las herramientas y la metodología definidas para esa arquitectura.

1.6.1. Metodología: OpenUP

OpenUP [20] es un proceso unificado que aplica acercamientos iterativos e incrementales dentro de un ciclo de vida estructurado. Además abraza una filosofía pragmática, ágil, y enfocada en la naturaleza de colaboración del desarrollo del software. Es una herramienta que se puede ampliar a una gran variedad de tipos del proyecto.

El esfuerzo personal en un proyecto de OpenUP se organiza en micro-incrementos, los cuales representan unidades cortas de trabajo que producen un paso constante y mensurable en el progreso del proyecto. El proceso aplica la colaboración intensiva como sistema de desarrollo incremental. Estos micro-incrementos proporcionan un lazo de regeneración extremadamente corto, que conducen a decisiones adaptables dentro de cada iteración. El ciclo de vida de un proyecto en OpenUP está estructurado en cuatro fases: inicio, elaboración, construcción y transición.

1.6.2. Lenguaje de modelado: UML

El Lenguaje de Modelado Unificado (UML - Unified Modeling Language) es un lenguaje que permite modelar, construir y documentar los elementos que forman un producto de software que responde a un enfoque orientado a objetos. Este lenguaje fue creado por un grupo de estudiosos de la Ingeniería de Software formado por: Ivar Jacobson, Grady Booch y James Rumbaugh en el año 1995. Desde entonces, se ha convertido en el estándar internacional para definir, organizar y visualizar los elementos que configuran la arquitectura de una aplicación orientada a objetos.

UML no es un lenguaje de programación, sino un lenguaje de propósito general para el modelado orientado a objetos. También puede considerarse como un lenguaje de modelado visual que permite una abstracción del sistema y sus componentes.

1.6.3. Herramientas CASE: Visual Paradigm

Como herramienta CASE se empleó Visual Paradigm para el trabajo con UML. La herramienta está diseñada para una amplia gama de usuarios que incluye a: ingenieros de software, analistas de sistemas, analistas de negocios y arquitectos de sistemas, interesados en la creación de Grandes Sistemas de Software de manera confiable, a través del paradigma Orientado a Objetos. VP-UML soporta los últimos estándares de anotaciones de JAVA y UML y provee soporte para la generación de código y la ingeniería inversa para Java. Además, VP-UML se integra con Eclipse, Borland® JBuilder®, NetBeans IDE/Sun™ ONE, IntelliJ IDEA™, Oracle JDeveloper y BEA WebLogic Workshop™, para soportar las fases de implementación en el desarrollo de software. Las transiciones del análisis al diseño, y de éste a la implementación, están adecuadamente integradas dentro de la herramienta CASE, de manera que reduce significativamente los esfuerzos de todas las etapas del ciclo de desarrollo de software.

1.6.4. Lenguaje de programación: Java

El lenguaje de programación Java es un lenguaje de propósito general, concurrente, basado en clases y orientado a objetos. Su diseño fue concebido para que los programadores puedan lograr fluidez con el lenguaje. Java ofrece toda la funcionalidad de un lenguaje potente, pero sin las características menos usadas y más confusas de éstos. C++ es un lenguaje que adolece de falta de seguridad, pero C y C++ son lenguajes más difundidos, por ello, Java se diseñó para ser parecido a C++ y así facilitar un rápido y fácil aprendizaje, además, el mismo elimina muchas de las características de otros lenguajes como C++, para mantener reducidas las especificaciones del lenguaje y añadir características muy útiles como el garbage collector (reciclador de memoria dinámica o recolector de basura). No es necesario preocuparse de liberar memoria, el reciclador se encarga de ello y como es un thread (hilo) de baja prioridad, cuando entra en acción, permite liberar bloques de memoria muy grandes, lo que reduce la fragmentación de esta.

Una de las características más importante de Java es que posee una arquitectura neutral, es decir el compilador Java compila su código a un fichero objeto de formato independiente de la arquitectura de la máquina en que se ejecutará. Cualquier máquina que tenga el sistema de ejecución (run-time) puede ejecutar ese código

objeto, sin importar en modo alguno la máquina en que ha sido generado.

Actualmente existen sistemas run-time para Solaris 2.x, SunOs 4.1.x, Windows 95, Windows NT, Linux, Irix, Aix, Mac, Apple y probablemente haya grupos de desarrollo trabajando para la portabilidad a otras plataformas.

Más allá de la portabilidad básica por ser de arquitectura independiente, Java implementa otros estándares de portabilidad para facilitar el desarrollo. Los enteros son siempre enteros y además, enteros de 32 bits en complemento a 2. Java, además, construye sus interfaces de usuario a través de un sistema abstracto de ventanas, de forma que ellas puedan ser implantadas en entornos Unix, Pc o Mac.

1.6.5. Entornos de desarrollo: Eclipse

Eclipse es un poderoso entorno integrado de desarrollo (IDE) que permite la construcción de aplicaciones en Java. Admite la incorporación de otros plug-ins para obtener un mayor número de funcionalidades. Tiene, además, los beneficios siguientes:

- Es una herramienta de código abierto.
- Soporta herramientas que manipulan diferentes tipos de lenguajes, como por ejemplo Java, C, C++,
- Corre en una gran cantidad de sistemas operativos incluyendo Windows y Linux.
- Provee a los desarrolladores, herramientas (ej.- PDE) que facilitan la creación de plug-ins.

A las características señaladas se suman: la capacidad de ser soportado para distintas arquitecturas, control de versiones con cvs o con subversión, resaltado de sintaxis, autocompletado, tabulador de un bloque de código seleccionado, asistentes (wizards): para la creación, exportación e importación de proyectos; para generar esqueletos de códigos (templates), permite la integración con la herramienta CASE (Computer Aided Software Engineering, Ingeniería de Software Asistida por computadoras) Visual Paradigm.

Por todas estas razones fue seleccionado Eclipse como IDE para el desarrollo de la solución a implementar.

1.7. Conclusiones

Con el objetivo de lograr una interfaz de usuario que mejore la interacción persona – ordenador, fueron seleccionados, para el diseño e implementación del Front-End, los siguientes principios:

1. Percepción del Color, Consistencia, Ley de Fitt y Legibilidad.
2. Selección de Swing como librería a utilizar, por ser portable, además de tener el apoyo directo de Sun.
3. Uso de las librerías implementadas bajo Swing, Flamingo y MyDoggy , por presentar mejores componentes, para lograr una mayor interacción persona-ordenador.

Capítulo 2

Características del Sistema

2.1. Introducción

En este capítulo se realiza una breve descripción de la solución propuesta, sus requisitos funcionales y no funcionales, así como los actores que intervienen en ella. Se presenta el diagrama de caso de usos del sistema, la descripción de los mismos y el modelo de dominio.

2.2. Breve Descripción del Sistema

El diseño del nuevo Front-End facilitará a las personas encontrar y utilizar toda la gama de características que proporciona el sistema, con un área de trabajo despejada que mejora la interacción persona-ordenador. Además, permitirá adicionar y eliminar módulos utilizando una arquitectura de plug-ins, la cual, permite aumentar o disminuir funcionalidades al sistema y mejora la manipulación de ventanas**.**

2.3. Modelo de Dominio

El modelo del dominio es una representación de los conceptos u objetos del mundo real, significativos para un problema. Tiene como objetivo fundamental la descripción de las clases más importantes en el sistema y representa conceptos del mundo real, no de los componentes de software.

El diagrama de clases del modelo de dominio del presente trabajo se muestra en la figura 2.1. , donde se aprecia que el especialista interactuará directamente con el Front-End ,el cual ganará funcionalidades, a través de

plug-ins desarrollados con anterioridad por el equipo de desarrollo de la plataforma en general. El especialista será capaz, además, de configurar preferencias generales de la plataforma como son: la ubicación física del servidor y la presentación de la plataforma. Una vez que el Front-End tenga los plug-ins incorporados, estos interactuarán con los Compuestos Orgánicos descritos en su totalidad por la información estructural, que es conocida de antemano por el usuario final de la plataforma.

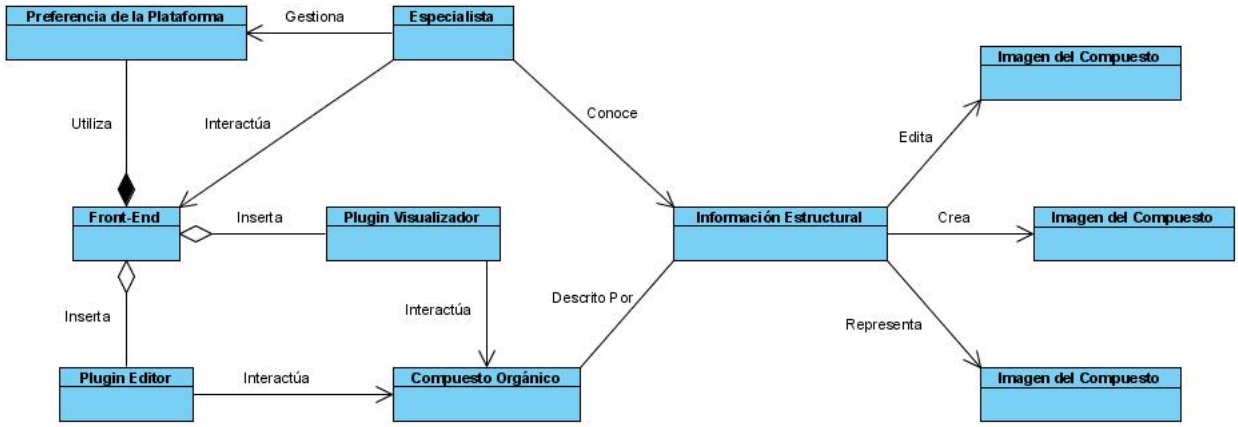


Figura 2.1: Diagrama del dominio del Front-End.

Para una mejor comprensión, a continuación se describen cada una de las clases que intervienen en el diagrama mostrado.

Especialista: Persona capacitada para interactuar con la aplicación.

Compuestos Orgánicos: Sustancias químicas basadas en cadenas de carbono e hidrógenos. En muchos casos contienen oxígenos, nitrógenos, azufres, fósforos y halógenos.

Información Estructural: Dato o descripción que permite conocer cómo está formado el compuesto en términos de átomos y la relación espacial que existe entre ellos.

Imagen del Compuesto: Representación tridimensional del compuesto; se tiene en cuenta la geometría espacial del mismo.

Front-End: Parte visual que adquiere los datos que serán procesados por el Back-End.

Plug-ins: Código enchufable que permite que las aplicaciones aumenten sus funcionalidades.

Preferencias de la Plataforma: Permite configurar a través de ella las preferencias generales de la plataforma.

2.4. Especificación de los Requisitos del Sistema

Los requisitos son condiciones o capacidades que necesita el usuario para resolver un problema o conseguir un objetivo determinado. Esta definición se extiende y se aplica a las condiciones que debe cumplir o poseer un sistema -o uno de sus componentes- ,para satisfacer un contrato, una norma o una especificación.

2.4.1. Requisitos Funcionales

Los requisitos funcionales de un sistema describen su funcionalidad, los servicios que de él se esperan, o los que proveerá, entre ellos: sus entradas, salidas y excepciones. A continuación se muestran los requisitos funcionales del Front-End (alasGRATO) y del plug-in Visualizador Molecular (alasGRATOViewer):

Requisitos del Front-End (alasGRATO)

RF1.- Gestionar Plug-ins

RF1.1.- Adicionar Plug-ins

RF1.2.- Remover Plug-ins

RF1.3.- Activar Plug-ins

RF2.- Gestionar Preferencias

RF2.1.- Cambiar Ambiente

RF2.2.- Cambiar ubicación de los servicios

RF3.- Eliminar Fichero

RF4.- Manipular Ventanas

RF4.1.- Mover hacia arriba

RF4.2.- Mover hacia a bajo

RF4.3.- Mover hacia la derecha

RF4.4.- Mover hacia la izquierda

RF4.5.- Embeber ventanas en la plataforma

RF4.6.- Extraer ventana de la plataforma

Requisitos del Plug-in Visualizador Molecular (alasGRATOViewer)

RV1 Abrir Fichero.

RV1.1 Abrir Molécula.

RV1.2 Abrir Proteína.

RV1.3 Abrir Reciente.

RV1.4 Abrir Fichero de Preferencias de Visualización.

RV2 Gestionar Fichero.

RV2.1 Guardar Fichero.

RV2.1.1 Guardar Molécula.

RV2.1.2 Guardar Proteína.

RV2.1.3 Guardar Fichero de Preferencias de Visualización.

RV2.2 Cerrar Fichero.

RV2.2.1 Cerrar el Fichero Activo.

RV2.2.2 Cerrar Todo los Ficheros.

RV2.3 Exportar Fichero.

RV2.3.1 Exportar Fichero a PDF.

RV2.3.2 Exportar Fichero a POVRAY.

RV3 Seleccionar Átomos

RV3.1: Seleccionar Átomos.

RV4 Seleccionar Proteína.

RV4.1: Seleccionar Toda la Proteína.

RV4.2: Seleccionar BackBone.

RV4.3: Seleccionar Cadenas Secundarias.

RV4.4: Seleccionar Aminoácidos.

RV4.4.1: Neutros.

RV4.4.2: Negativos.

RV4.4.3: Positivos.

RV4.4.4: Polares.

RV4.4.5: No Polares.

RV5 Seleccionar Ácido Nucleicos.

RV5.1: Seleccionar Todas las Bases.

RV5.2: Seleccionar los Pares.

RV5.2.1: G-C.

RV5.2.2: A-U.

RV5.2.3: A-T.

RV5.3: Seleccionar las Bases.

RV5.3.1: A.

RV5.3.2: T.

RV5.3.3: G.

RV5.3.4: C.

RV5.3.5: U.

RV 5.4: Seleccionar Cadenas Secundarias.

RV6 Seleccionar Heteros.

RV6.1: Seleccionar Todos.

RV6.2: Seleccionar Agua.

RV6.3: Seleccionar Disolvente.

RV6.4: Seleccionar Disolvente Distinto de Agua.

RV6.5: Seleccionar Todo Excepto Agua.

RV6.6: Seleccionar Ligandos.

RV6.7: Otros.

RV7 Visualizar Moléculas.

RV7.1 Visualizar Bolas y cilindros.

RV7.2 Visualizar Alambres.

RV7.3 Visualizar Radios de Van Der Waals.

RV8 Selección de Superficies.

RV8.1 Seleccionar Superficie De Puntos.

RV8.2 Seleccionar Superficie Van Der Waals.

RV8.3 Seleccionar Superficie Molecular.

RV8.4 Seleccionar Superficie Disolvente.

RV8.5 Seleccionar Superficie Accesible al Disolvente.

RV8.6 Seleccionar Superficie Carga.

RV8.7 Seleccionar Superficies Opacas.

RV8.8 Seleccionar Superficies Traslucidas.

RV8.9 Seleccionar Superficie Ocultar.

RV9 Selección de Vistas.

RV9.1 Seleccionar Vista Frontal.

RV9.2 Seleccionar Vista Superior.

RV9.3 Seleccionar Vista Inferior.

RV9.4 Seleccionar Vista Izquierda.

RV9.5 Seleccionar Vista Derecha.

RV9.6 Seleccionar Vista Estereográficas.

RV9.6.1 Restaurar.

RV9.6.2 Gafas Rojo + Cian.

RV9.6.3 Gafas Rojo + Azul.

RV9.6.4 Gafas Rojo + Verde.

RV9.6.5 Seleccionar Visión Cruzada.

RV9.6.6 Seleccionar Visión Paralela.

RV10 Selección de Representaciones.

RV10.1 Seleccionar representación en BackBone.

RV10.2 Seleccionar representación en Cartoon.

RV10.3 Seleccionar representación en Cohetes Cartoon.

RV10.4 Seleccionar representación en Cintas.

RV10.5 Seleccionar representación en Cohetes.

RV10.6 Seleccionar representación en Hilos.

RV10.7 Seleccionar representación en Trazas.

RV11 Cambiar Átomos.

RV11.1 Ocultar.

RV11.2 15 % de Van Der Waals.

RV11.3 20 % de Van Der Waals.

RV11.4 25 % de Van Der Waals.

RV11.5 Radios de Van Der Waals.

RV12 Cambiar Enlaces.

RV12.1 Ocultar.

RV12.2 Modelo de Alambre.

RV12.3 Radios.

RV12.3.1 0,10 Angstroms.

RV12.3.2 0,15 Angstroms.

RV12.3.3 0,20 Angstroms.

RV13 Cambiar Etiquetas.

RV13.1 Ocultar.

RV13.2 Nombre.

RV13.3 Símbolos.

RV13.4 Números.

RV14 Alinear Etiquetas.

RV14.1 Centradas.

RV14.2 Superior Izquierdo.

RV14.3 Superior Derecho

RV14.4 Inferior Izquierdo

RV14.5 Inferior Derecho

RV15 Cambiar Preferencias de Visualización.

RV15.1 Mostrar Hidrógenos.

RV15.2 Mostrar Mediciones.

RV15.3 Mostrar Caja.

RV15.4 Mostrar Ejes.

RV15.5 Mostrar Perspectiva.

RV15.6 Rotar Molécula.

RV15.7 Rotar Proteína.

RV15.8 Mostrar Halos de Selección.

RV15.9 Mostrar solo Selección.

RV16 Cambiar Unidad de Medida.

RV16.1 Nanómetros.

RV16.2 Angstroms.

RV16.3 Picómetros.

RV17 Convertir Molécula a 3D.

2.4.2. Requisitos no Funcionales

Los requisitos no funcionales de un sistema se refieren a las propiedades emergentes del sistema como son: la fiabilidad, el tiempo de respuesta, la capacidad de almacenamiento, la capacidad de los dispositivos de entrada/salida, y la representación de datos que se utiliza en las interfaces del sistema.

Los requisitos no funcionales definidos para la creación del Front-End son los que se muestran a continuación:

2.4.2.1. Usabilidad

- El Front-End de la plataforma será diseñado para aquellos usuarios que posean conocimientos básicos de química y para investigadores en el campo del diseño de fármacos.
- El Front-End de la plataforma deberá ofrecer la posibilidad de visualizar tanto moléculas como proteínas, editar moléculas, calcular descriptores topológicos, topográficos e híbridos, tanto atómicos como moleculares; Además, permitirá predecir actividad biológica utilizando técnicas de inteligencia artificial.
- El Front-End de la plataforma dará la posibilidad de almacenar los resultados obtenidos de los cálculos de descriptores, así como los modelos creados y los resultados de la predicción. También permitirá guardar la molécula creada ,así como las preferencias de visualización del usuario. Todo esto mediante interfaces que permitan el intercambio de datos con el usuario.
- Se debe lograr un diseño flexible, con capacidad de poder adicionar nuevas funcionalidades, sin impactar el resto de los requerimientos contemplados en el sistema.

2.4.3. Fiabilidad

- Cada módulo debe ser adicionado y eliminado de la plataforma en tiempo de ejecución. Asimismo, deberá garantizarse la funcionalidad independiente de estos, aunque exista dependencia entre ellos.

- Confidencialidad: Se requiere que los módulos que se le adicionen a la plataforma sean reconocidos por el servidor encargado de ofrecer los servicios, para que los mismos puedan funcionar. De ocurrir lo contrario el usuario podrá utilizar el módulo, pero no los servicios que brinda.
- Disponibilidad: Aún si el módulo que se adicione a la plataforma es reconocido, por el servidor encargado de ofrecer los servicios, se le podrá acceder en todo momento.

2.4.3.1. Soporte

- Todos los módulos y servicios que presenta la plataforma estarán bien documentados, para que el equipo de desarrollo pueda darle mantenimiento en el menor tiempo posible.
- Cuando se le de soporte a un servicio de la plataforma, sólo se verá afectada la funcionalidad de los módulos que utilicen dicho servicio, es decir, los módulos siguieran funcionando sin usar el servicio afectado.
- Los módulos de la plataforma deben funcionar en cualquier sistema operativo sobre el cual se haya instalado la máquina virtual de Java 1.6 o superior.
- **Instalación**: La instalación del sistema debe caracterizarse por su facilidad, claridad y sencillez. Debe garantizar que el usuario instale el módulo que desee.
- **Portabilidad**: El sistema debe ser multiplataforma.

2.4.3.2. Interfaz

Interfaces del sistema

- La interfaz externa debe ser fácil de usar, profesional, clara y sencilla. Debe garantizar una buena interacción persona-ordenador.
- La interfaz externa debe tener la funcionalidad de cambiar la apariencia (ya sea color o idioma) ante la necesidad del usuario.

Interfaces de usuario

- Navegación y distribución: la navegación y distribución de las interfaces de cada módulo deben estar diseñadas- de forma tal- que garanticen la interacción persona-ordenador.

- Consistencia: las interfaces de los módulos de la plataforma deben tener consistencia. Para garantizar que los datos compartidos entre los módulos tengan consistencia, se deberá mantener una correcta relación entre ellos.
- Personalización y adaptación al cliente: el cliente podrá ajustar toda la interfaces de los módulos a su gusto en cuanto a idioma y color.

Interfaces Software

- El Front-End: debe permitir mantener el control del sistema y la gestión de los restantes módulos de la plataforma. Representa la cara principal del sistema. Es él quien permite la adición o eliminación de módulos a la plataforma, debido a que contiene las clases pertinentes para adicionar y eliminar plug-ins a ella.
- Los módulos de la plataforma son archivos .jar, ellos son adicionados y eliminados a la plataforma en forma de plug-ins en tiempo de ejecución. La comunicación entre ellos se establece a través del Front-End, que es la interfaz principal de la plataforma

2.4.3.3. Requisitos de Licencia

- El sistema se utilizará primeramente en la Universidad de Camagüey. Posteriormente, en aquellos centros que -en el país-, se dediquen al diseño de fármacos. Deberá contarse con una licencia en el caso que se decida comercializarse en el extranjero.

2.4.3.4. Requisitos Legales, de Derecho de Autor y otros.

- Los derechos de autor serán registrado por la UCI.

2.5. Definición de los Casos de Uso del Sistema

Los casos de uso se utilizan para obtener información de cómo debe trabajar el sistema, son descripciones de las funcionalidades del sistema. Independientemente de la implementación, describen- bajo la forma de acciones y reacciones- el comportamiento de un sistema desde el punto de vista del usuario.

2.5.1. Actores del Sistema

Se le llama Actor a toda entidad externa al sistema que guarda una relación con este y que le demanda una funcionalidad. Esto incluye a los operadores humanos. Pero también incluye a todos los sistemas externos, así como a entidades abstractas como el tiempo.

Autor	Descripción
Especialista	Representa al usuario que va a hacer uso del sistema, y quien tiene la posibilidad de interactuar con todas las funcionalidades de este.

2.5.2. Diagrama de Casos de Uso del Sistema

El diagrama de casos de uso del sistema sirve para especificar la comunicación y el comportamiento de un sistema, mediante su interacción con los usuarios y/u otros sistemas. A continuación se muestra, en la figura 2.2, el diagrama de casos de uso del sistema del presente trabajo.

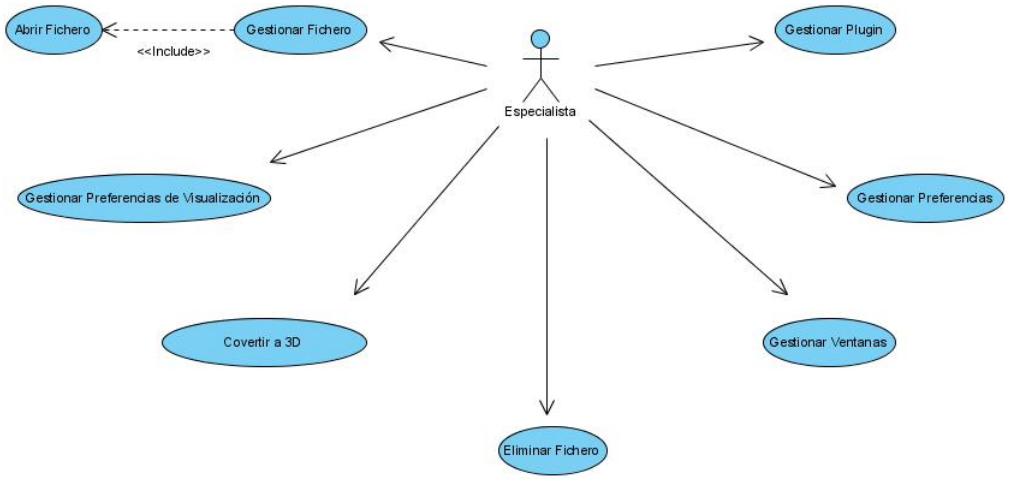


Figura 2.2: Diagrama de casos de uso del sistema

2.5.3. Descripción de los Casos de Uso del Sistema

Un caso de uso es una técnica para la captura de requisitos potenciales de un nuevo sistema o una actualización de software. Cada caso de uso proporciona uno o más escenarios que indican cómo debería interactuar el sistema con el usuario, o con otro sistema, para conseguir un objetivo específico. A continuación, se describen los casos de uso del Front-End (alasGRATO) y los casos de uso más significativos del plug-in Visualizador

Molecular (alasGRATOViewer), ya que el mismo fue desarrollado para validar las funcionalidades del Front-End. Por ello, con la descripción de los casos de uso significativo, se realiza esa validación.

2.5.3.1. Descripción de los Casos de Uso del Front-End (alasGRATO)

Gestionar Plug-ins.

Caso de Uso	**Gestionar Plug-ins**
Actores	**Especialista**
Propósito	Adicionar o eliminar plug-ins a la plataforma por el especialista para su posterior uso en la aplicación.
Resumen	El caso de uso inicia cuando el especialista selecciona una de las siguientes opciones: Adicionar, Eliminar, Activar o Desactivar plug-ins del Menú Principal.
Referencia	RF1.1, RF1.2, RF1.3, RF1.4
Precondiciones	Que existan el plug-in creado y no esté siendo utilizado por el especialista.
Prioridad	Crítico.
Flujo Normal de Eventos	
Actores	**Especialista**
1.- El especialista selecciona la opción Administración de Plug-ins del Menú "Principal".	1.1.- El sistema muestra un panel lateral con las diferentes opciones que puede realizar el especialista.

2.- El especialista puede seleccionar una de las siguientes opciones del panel lateral: • Adicionar Plug-ins. • Eliminar Plug-ins. • Activar Plug-ins.	2.1.- El sistema ejecuta una de las siguientes secciones: • Si seleccionó Adicionar Plug-ins ir a la Sección "Adicionar Plug-ins". • Si seleccionó Eliminar Plug-ins ir a la Sección "Eliminar Plug-ins". • Si seleccionó Activar Plug-ins ir a la Sección "Activar Plug-ins".
Sección "Adicionar Plug-ins" Flujo Normal de Eventos	
Actores	**Especialista**
1.- El especialista selecciona la opción: Adicionar Plug-ins.	1.1.- El sistema abrirá una ventana de diálogo para que el especialista escoja la ubicación de los plug-ins.
2.- El especialista indica la dirección donde se encuentran ubicado los plug-ins, a través del botón localización de las extensiones.	2.1.- El sistema abrirá una ventana de diálogo para que el especialista escoja la ubicación de la carpeta contenedora de los plug-ins.
3.- El especialista selecciona la carpeta que contiene los plug-ins y presiona Abrir.	3.1.- El sistema añade los plug-ins ,que se encuentran en la carpeta seleccionada, a la lista de posibles plug-ins a importar.
4.- El especialista selecciona el Plug-in que desea importar y presiona el botón Importar.	4.1.- Si el especialista tiene un plug-in seleccionado: • El sistema adiciona el plug-in a la carpeta de extensiones, muestra el mensaje "Plug-in importado satisfactoriamente" y elimina de la lista de plug-ins disponibles el plug-in importado.

5.- El especialista selecciona cerrar la ventana.	5.1.- El sistema cierra la ventana correspondiente.
Flujo Alternos	
Actores	**Especialista**
3.- El especialista selecciona el botón Cancelar.	3.1.- Se retorna al flujo normal de eventos número 2.
	4.1.- Si el especialista no ha seleccionado ningún plug-in el sistema muestra el mensaje "Debe seleccionar un plug-in".
Sección "Eliminar Plug-ins" Flujo Normal de Eventos	
Actores	**Especialista**
1.- El especialista selecciona la opción: Eliminar Plug-ins.	1.1.- El sistema abrirá una ventana de diálogo con los plug-ins existentes.
2.- El especialista selecciona el plug-in que desea eliminar y luego presionará el botón Eliminar.	2.1.- El sistema eliminará totalmente de la plataforma el plug-in indicado.
Sección "Eliminar Plug-ins" Flujo Alternos	
Actores	**Especialista**
2.- El especialista selecciona cerrar la ventana.	2.1.- El sistema cierra la ventana correspondiente.
Sección "Activar Plug-ins" Flujo Normal de Eventos	
Actores	**Especialista**

1.- El especialista selecciona la opción: Activar Plug-ins.	3.1.- Si el plug-in no ha sido activado el sistema lo activa creando el TaskRibbon correspondiente al mismo y crea la ventana que contiene el área de trabajo del plug-in, ubicándola en el lado oeste del sistema.
Flujo Alternos	
Actores	**Especialista**
	3.1- Si el plug-in ya fue activado anteriormente el sistema muestra al especialista este plug-in que se deseaba activar.
Poscondiciones:	Un nuevo plug-in es añadido o eliminado de la carpeta "ext" del Front-End.

Gestionar Preferencias.

Caso de Uso	**Gestionar Preferencias**
Actores	**Especialista**
Propósito	Cambia la ubicación de los servicios o el ambiente de la plataforma.
Resumen	El caso de uso se inicia cuando el especialista selecciona la opción "Servidor" o la opción "Tema de Colores", presentes en el Menú Principal.
Referencia	RF2.1, RF2.2
Precondiciones	Que la dirección donde esté ubicado el servicio sea válida.
Prioridad	Secundario.
Flujo Normal de Eventos	
Actores	**Especialista**

1.- El especialista puede seleccionar una de las siguientes opciones del Menú "Principal": • Preferencias del Servidor. • Tema de Colores.	1.1- El sistema ejecuta una de las siguientes secciones: • Si seleccionó Preferencias del Servidor ir a la Sección "Preferencias del Servidor". • Si seleccionó Tema de Colores ir a la Sección "Tema de Colores".
Sección "Preferencias del Servidor" Flujo Normal de Eventos	
Actores	**Especialista**
1.- El especialista selecciona la opción : Preferencias del Servidor	1.1.- El sistema muestra una ventana de diálogo donde el especialista debe especificar el ip del servidor y el puerto.
2.- El especialista introduce los datos y presiona el botón Aceptar.	2.1.- El sistema recoge los datos introducidos por el especialista, los valida y configura con ellos, dos variables globales para el uso de todos los plug-ins de la aplicación.
Flujo Alternos	
Actores	**Especialista**
2.- El especialista presiona el botón Cancelar.	2.1.- El sistema cierra la ventana correspondiente.
Sección "Tema de Colores" Flujo Normal de Eventos	
Actores	**Especialista**
1.- El especialista selecciona la opción Tema de Colores.	1.1.- El sistema muestra en el panel lateral los temas disponibles que el especialista puede escoger para que sean aplicados a la plataforma.

2.- El especialista selecciona el tema que desea aplicar a la plataforma.	2.1.- El sistema realiza el cambio de color de acuerdo a la selección del especialista.
Poscondiciones:	El ambiente de la plataforma o la dirección física del servidor cambiará de acuerdo a la opción seleccionada por el especialista.

Eliminar Fichero.

Caso de Uso	**Eliminar Fichero**
Actores	**Especialista**
Propósito	Eliminar un fichero de la plataforma.
Resumen	El caso de uso se inicia cuando el especialista presiona clic derecho sobre un fichero que se encuentra en el árbol de ficheros de la plataforma.
Referencia	RF3
Precondiciones	Que exista al menos un fichero cargado en el árbol de la plataforma.
Prioridad	Secundario.
Flujo Normal de Eventos	
Actores	**Especialista**
1.- El especialista selecciona el fichero que desea eliminar y presiona clic derecho.	1.1.- El sistema muestra un Popup que contiene la opción “Eliminar fichero”.

2.- El especialista presiona la opción Eliminar fichero.	2.1.- El sistema busca qué plug-ing se encuentra haciendo uso del fichero a eliminar. 2.2.- El sistema elimina el fichero del plug-in en que se encuentre. 2.3.- El sistema elimina el fichero seleccionado del árbol de la plataforma.
Poscondiciones:	Queda eliminado totalmente de la plataforma el fichero deseado.

Manipular Ventanas.

Caso de Uso	**Manipular Ventanas**
Actores	**Especialista**
Propósito	Que el usuario pueda trabajar con varios plug-ins a la vez y configurar su entorno de trabajo para su conveniencia.
Resumen	El caso de uso se inicia cuando el especialista decide trabajar con más de un plug-in a la vez o desee cambiar su entorno de trabajo.

Referencia	RF4.1, RF4.2, RF4.3, RF4.4, RF4.5, RF4.6.
Precondiciones	Debe existir al menos un plug-in cargado en la plataforma.
Prioridad	Secundario.
Flujo Normal de Eventos	
Actores	**Especialista**
1.- El especialista tiene la posibilidad de realizar una de las siguientes operaciones: • Mover ventana hacia arriba. • Mover ventana hacia abajo. • Mover ventana hacia la derecha. • Mover ventana hacia la izquierda. • Embeber ventana. • Extraer ventana.	1.1.- El sistema ejecuta una de las siguientes secciones: • Si seleccionó Mover ventana hacia arriba ir a la Sección "Mover ventana hacia arriba". • Si seleccionó Mover ventana hacia abajo ir a la Sección "Mover ventana hacia abajo". • Si seleccionó Mover ventana hacia la derecha ir a la Sección "Mover ventana hacia la derecha". • Si seleccionó Mover ventana hacia la izquierda ir a la Sección "Mover ventana hacia la izquierda". • Si seleccionó Embeber Ventana ir a la Sección "Embeber Ventana". • Si seleccionó Extraer Ventana ir a la Sección "Extraer Ventana".
Sección "Mover ventana hacia arriba" Flujo Normal de Eventos	
Actores	**Especialista**
1.- El especialista selecciona la ventana con la que quiere trabajar y la mueve hacia la parte superior.	1.1.- El sistema moverá la ventana hacia la parte superior del área de trabajo.
Sección "Mover ventana hacia abajo" Flujo Normal de Eventos	
Actores	**Especialista**

1.- El especialista selecciona la ventana con la que quiere trabajar y la mueve hacia la parte inferior.	1.1.- El sistema moverá la ventana hacia la parte inferior del área de trabajo.
Sección "Mover ventana hacia la derecha" Flujo Alternos	
Actores	**Especialista**
1.- El especialista selecciona la ventana con la que quiere trabajar y la mueve hacia la parte derecha.	1.1.- El sistema moverá la ventana hacia la parte derecha del área de trabajo.
Sección "Mover ventana hacia la izquierdas" Flujo Normal de Eventos	
Actores	**Especialista**
1.- El especialista selecciona la ventana con la que quiere trabajar y la mueve hacia la parte izquierda.	1.1.- El sistema moverá la ventana hacia la parte izquierda del área de trabajo.
Sección "Embeber Ventana" Flujo Normal de Eventos	
Actores	**Especialista**
1.- El especialista selecciona la opción de "Fijar Ventana" de la ventana que quiere fijar a la plataforma.	1.1.- El sistema fija la ventana flotante a la plataforma.
Sección "Extraer Ventana" Flujo Normal de Eventos	

Actores	**Especialista**
1.- El especialista selecciona la opción de "Ventana Flotante" de la ventana que quiere extraer de la plataforma.	1.1.- El sistema convierte la ventana a una ventana flotante y la extrae de la plataforma.
Poscondiciones:	Se configura una nueva disposición de las ventanas de trabajo en la plataforma.

2.5.3.2. Descripción de los Casos de Uso del Visualizador Molecular (alasGRATOViewer)

Abrir Fichero

Caso de Uso	**Abrir Fichero**
Actores	**Especialista**
Propósito	Cargar fichero para su posterior uso en la aplicación.
Resumen	El especialista iniciará el caso de uso al seleccionar una de las opciones : Abrir, Abrir Reciente o Abrir Preferencias de Visualización del menú Archivo, y podrá cargar uno o varios ficheros a la vez.
Referencia	RV1.1, RV1.2, RV1.3, RV1.4
Precondiciones	Que el o los ficheros existan.
Prioridad	Crítico.
Flujo Normal de Eventos	
Actores	**Especialista**

1.- El especialista selecciona una de las siguiente opciones del menú “Archivo”: • Abrir. • Abrir Reciente. • Abrir Preferencias de Visualización.	1.1.- El sistema ejecuta una de las siguientes secciones: • Si escoge Abrir ir a la sección ”Abrir”. • Si escoge Abrir Reciente ir a la sección ”Abrir Reciente”. • Si escoge Abrir Preferencias de Visualización ir a la sección ”Abrir Preferencias de Visualización”.
Sección ”Abrir” Flujo Normal de Eventos.	
Actores	**Especialista**
1.- El especialista selecciona la opción Abrir del menú “Archivo”	1.1.- El sistema muestra una ventana de diálogo donde permite seleccionar la extensión y el o los ficheros, ya sea Molécula o Proteína, que desee abrir.
2.- El especialista selecciona la extensión y la(s) Molécula(s) o Proteína(s) que desea abrir y luego presiona el botón Abrir.	2.1.- El sistema verifica si la extensión es distinta de mol o pdf, de ser así, le solicita al servicio de conversión de formato que convierta el o los ficheros a mol. 2.2.- De ser uno, el sistema copia el mismo en la carpeta tmp. 2.3.- El sistema adiciona el ficheros al final del árbol de ficheros cargados. 2.4.- Adiciona el fichero en la lista de ficheros recientes. 2.5.- Visualiza el fichero en el área de visualización.
Flujos Alternos	
Actores	**Especialista**

	2.1.1.- Si el servicio de conversión no está disponible muestra el siguiente error “No está ejecutándose el servicio de conversión”. 2.2.- De ser mas de uno, el sistema copia los ficheros en la carpeta tmp. 2.3.- El sistema adiciona los ficheros al final del árbol de ficheros cargados. 2.4.- Adiciona los 10 últimos fichero en la lista de ficheros recientes.
Sección ”Abrir Reciente” Flujo Normal de Eventos.	
Actores	**Especialista**
1.- El especialista selecciona la opción Abrir Reciente del menú “Archivo”.	1.1.- El sistema muestra una serie de ficheros que fueron cargados recientemente.
2.- El especialista selecciona el fichero que desea abrir.	2.1.- El sistema carga el fichero seleccionado y lo visualiza en el área de visualización.
Sección ”Abrir Preferencias de Visualización” Flujo Normal de Eventos.	
Actores	**Especialista**
1.- El especialista selecciona la opción Abrir Preferencias de Visualización del menú “Archivo”.	1.1.- El sistema muestra una ventana de diálogo donde se podrá seleccionar la Preferencia de Visualización que se desee abrir.
2.- El especialista selecciona la Preferencia de Visualización que desea abrir.	2.1.- El sistema verifica si es un fichero de Preferencia de Visualización. 2.2.- El sistema carga la Preferencia de Visualización seleccionada y visualiza las preferencias en el área de visualización.
Flujos Alternos	

Actores	**Especialista**
	2.2.1.- El sistema muestra un mensaje de error "Error en selección de tipo de fichero."
Poscondiciones:	El o los fichero(s) queda(n) cargado(s).

Gestionar Fichero

Caso de Uso	**Gestionar Fichero**
Actores	**Especialista**
Propósito	Cerrar, guardar y exportar ficheros moleculares.
Resumen	El caso de uso se inicia cuando el especialista va a realizar alguna de las siguientes opciones: Cerrar Fichero: cuando el especialista cierra la pestaña del fichero actual o selecciona la opción de Cerrar del menú Archivo. Guardar Fichero: cuando el especialista selecciona la opción Guardar o Guardar Preferencias de Visualización del menú "Archivo". Exportar Fichero: cuando el especialista exporta ficheros a formato PDF.
Referencia	RV2.1.1, RV2.1.2, RV2.1.3, RV2.2.1, RV2.2.2, RV2.3.1, RV2.3.2
Precondiciones	Que el o los ficheros estén cargados y en ocasiones seleccionados.
Prioridad	Crítico.
Flujo Normal de Eventos	
Actores	**Especialista**

1.- El especialista selecciona de las siguientes opciones cual desea realizar : •Cerrar Fichero. •Guardar Fichero. •Exportar Fichero.	1.1.- El sistema ejecuta una de las siguientes secciones: •Si escoge Cerrar Fichero ir a la sección “Cerrar Fichero”. •Si escoge Guardar Fichero ir a la sección “Guardar Fichero ”. •Si escoge Exportar Fichero ir a la sección “Exportar Fichero”.
Sección ”Cerrar Fichero” Flujo Normal de Eventos.	
Actores	**Especialista**
1.- El especialista selecciona en el menú ”Archivo” una de las siguientes opciones: • Cerrar. • Cerrar Todo.	1.1.- Si el especialista selecciona ”Cerrar” el sistema cierra el fichero que esté visualizado sin quitarlo de la lista de ficheros abiertos. 2.1.- Si el especialista selecciona ”Cerrar Todo” el sistema cierra todos los ficheros visualizados sin quitarlos de la lista de ficheros abiertos.
Sección ”Guardar Fichero” Flujo Normal de Eventos.	
Actores	**Especialista**
1.- El especialista selecciona en el menú ”Archivo” la opción Guardar.	1.1.- El sistema muestra una ventana de diálogo donde permite buscar el lugar y la extensión con que se quiere guardar el fichero.
2.- El especialista selecciona dónde desea guardar el fichero, escribe el nombre, selecciona la extensión con la que quedará guardado y presiona el botón Guardar.	2.1.- El sistema verifica si la extensión es distinta de mol y utiliza el servicio de conversión de formato para convertir a la extensión especificada. 2.2.- El sistema guarda el fichero en la dirección especificada por el usuario.

Flujos Alternos	
Actores	**Especialista**
	2.1.1.- El sistema muestra el mensaje "No esta corriendo el servicio".
Sección "Guardar Fichero de Visualización" Flujo Normal de Eventos.	
1.- El especialista selecciona la opción Guardar Preferencias de Visualización del menú "Archivo"	1.1.- El sistema muestra una ventana de diálogo que permite seleccionar el lugar donde se quiere guardar las Preferencias de Visualización.
2.- El especialista selecciona dónde desea guardar las Preferencias de Visualización, escribe el nombre del fichero y presiona el botón Guardar.	2.1.- El sistema guarda el fichero de Preferencias de Visualización en el destino indicado, con el nombre especificado.
Sección "Exportar Fichero" Flujo Normal de Eventos.	
Actores	**Especialista**
1.- El especialista selecciona en el menú "Archivo" la opción de Exportar a PDF.	1.1.- El sistema muestra una ventana de diálogo que permite buscar el lugar donde se quiere exportar el fichero.
2.- El especialista selecciona dónde desea guardar el fichero, escribe el nombre y presiona el botón Exportar.	2.1.- El sistema guarda el fichero en el destino indicado, con el nombre y extensión especificada.

Poscondiciones:	El o los fichero(s) queda(n) Cerrado(s), Guardado(s) o Exportado.

Convertir a 3D

Caso de Uso	**Convertir a 3D**
Actores	**Especialista**
Propósito	Convertir a 3D la molécula activa.
Resumen	El especialista inicia el caso de uso cuando selecciona del menú Molécula la opción "Convertir a 3D".
Referencia	RV17
Precondiciones	Molécula cargada y activa en el área de trabajo.
Prioridad	Crítico.
Flujo Normal de Eventos	
Actores	**Especialista**
1.- El especialista selecciona en el menú "Molécula" la opción Convertir a 3D.	1.1.- El sistema envía la molécula seleccionada al servicio de conversión en 3D. 1.2.- El sistema recibe la molécula convertida del servicio y la visualiza en el área de visualización.
Flujos Alternos	
Actores	**Especialista**
	1.1.1.- El sistema muestra el mensaje "No esta corriendo el servicio".
Poscondiciones:	Se ve la conversión de la molécula en 3D en el área de trabajo.

2.6. Conclusiones

- Los conceptos definidos fueron relacionados mediante un diagrama de Modelo de dominio y se brindó de forma general cómo se desarrolla este proceso.
- Se brinda una clara definición de los requisitos que debe cumplir el sistema, seleccionando los actores, así como los casos de uso de dicho sistema.

- Se ganó claridad en cuanto a la concesión del sistema a construir. También se sentaron las bases para las restantes fases del proceso de diseño e implementación, a través de los modelos de casos de uso.

Capítulo 3

Diseño del Sistema

3.1. Introducción

En este capítulo se describe la representación arquitectónica del sistema, haciendo énfasis en los patrones de arquitectura utilizados. Además, se desarrolla el diseño del sistema.

3.2. Representación Arquitectónica

Un avance importante dentro de la construcción del software ha sido el desarrollo de la arquitectura de software, toda vez que permite representar la estructura de un sistema, a un nivel mayor que el dado por la programación, o incluso por el diseño [28]. Como ya se ha dicho, el presente trabajo responde a un proyecto cuya arquitectura está bien definida ya en trabajos anteriores de Marrero [25]. Por ello, aquí sólo se abordará la interfaz visual de la plataforma; es decir, el Front-End y el trabajo con los plug-ins.

Un patrón es un par problema/solución con nombre que se puede aplicar en nuevos contextos, con consejos acerca de cómo aplicarlos en nuevas situaciones y discusiones sobre sus puntos fuertes y débiles. Existen varios tipos de patrones ,ya conocidos, entre los que se encuentran los patrones de requisitos, de arquitectura, de diseño y de programación, entre otros. A continuación se presentan los patrones que fueron utilizados para la implemantación del sistema.

3.2.1. Patrón Arquitectónico Empleado

Los patrones arquitectónicos especifican un conjunto predefinido de subsistemas con sus responsabilidades y una serie de recomendaciones, para organizar los distintos componentes.

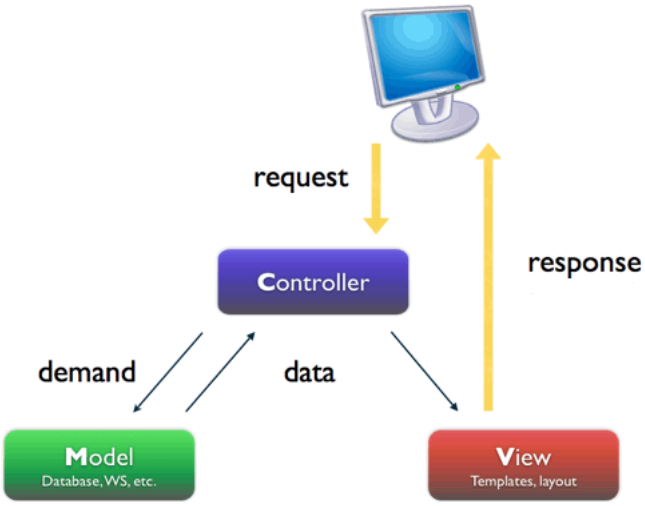


Figura 3.1: Esquema del patrón modelo-vista-controlador

El patrón Modelo – Vista – Controlador mostrado en la figura 3.1 está catalogado como un patrón de arquitectura de software donde:

- **Modelo**: Representa específicamente el dominio de la información sobre la cual funciona la aplicación. El modelo es otra forma de llamar a la capa de dominio. La lógica de dominio añade significado a los datos. El modelo encapsula los datos y las funcionalidades. El modelo es independiente de cualquier representación de salida y/o comportamiento de entrada.
- **Vista**: Presenta el modelo en un formato adecuado para interactuar, usualmente con un elemento de interfaz de usuario. Muestra la información al usuario. Pueden existir múltiples vistas del modelo. Cada vista tiene asociado un componente controlador.
- **Controlador**: Responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista. Los eventos son traducidos a solicitudes de servicio ("services request") para el modelo o la vista.

A continuación se presenta en la figura 3.2 un ejemplo de cómo se aplica este patrón en el Front-End desarrollado.

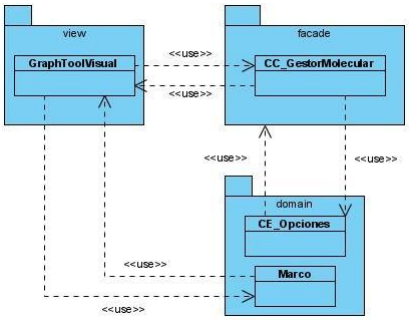


Figura 3.2: Ejemplo de aplicación de patrón MVC

3.2.2. Patrones de Diseño Empleados

A continuación se abordarán los patrones de diseños utilizados en el sistema desarrollado para asegurar una solución mucho más confiable.

Patrón Factory: Proporciona una manera flexible de instanciar objetos cuando la clase puede cambiar por alteraciones en el diseño o bien por cambios en la ejecución. El código debe estar abierto a la extensión y cerrado a la modificación.

A continuación en la figura 3.3 se muestra un pequeño diagrama de clases que evidencia el empleo del patrón en la aplicación desarrollada.

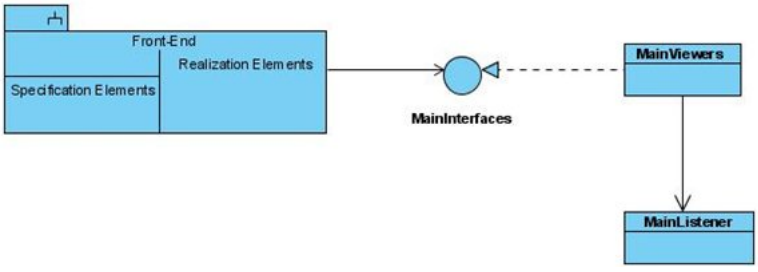


Figura 3.3: Aplicación del patrón Factory en la implementación del Front-End.

Patrón Facade: Provee una interfaz unificada para un conjunto de interfaces en un subsistema. Define una interfaz de alto nivel que permite usar fácilmente un subsistema.

A continuación en la figura 3.4 se muestra un diagrama de paquetes donde se evidencia el uso del patrón Facade, en la aplicación desarrollada.

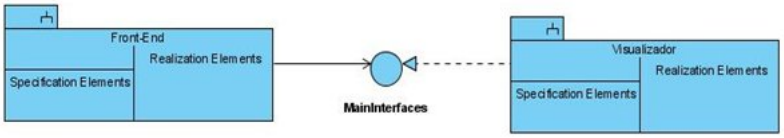


Figura 3.4: Aplicación del patrón Facade en la implementación del Front-End

Patrón Creador: Este patrón es el encargado de que una clase B cree una instancia de una clase A, siempre que:

- La clase B contenga a la clase A
- B sea una agregación (o composición) de A
- B almacene a A
- B tenga los datos de inicialización de A(datos que requiere su constructor)
- B use a A

Este patrón se utilizo en la implementación de las clases del Front-End, ejemplo de algunas son: GraphToolVisual, AddPluginVisual y RemovePluginVisual.

Patrón Bajo Acoplamiento: Este patrón asigna la responsabilidad de controlar el flujo de eventos del sistema a clases específicas. Esto facilita la centralización de actividades. El controlador no las realiza, las delega en otras clases con las que mantiene un modelo de alta cohesión. En la figura 3.5, se brinda un ejemplo de cómo se ve evidenciado dicho patrón.

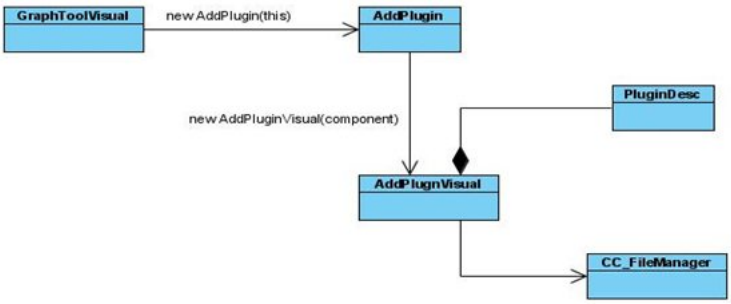


Figura 3.5: Aplicación del patrón Bajo Acoplamiento en la implementación del Front-End

Patrón Alta Cohesión: la cohesión es una medida de fuerza con las que se relacionan las clases y el grado de focalización de las responsabilidades de un elemento. Cada elemento de nuestro diseño debe realizar una labor única y auto - identificable dentro del sistema; no desempeñada por otro de ellos. Una clase con baja cohesión hace muchas operaciones no relacionadas, o trabaja en demasía.

Este patrón fue utilizado en el sistema desarrollado cuando -por ejemplo- se implementó la clase controladora CC_GestorMolecular, quien es la encargada de gestionar los datos de los compuestos a cargar en el sistema, auxiliándose de la clase CE_Opciones.

3.2.3. Arquitectura de Plug-ins

La arquitectura de plug-ins se desarrolló bajo ciertos patrones de diseño, para asegurar una robusta y rápida arquitectura. Patrones tales como el 3 capaz, y Factory, son utilizados a la hora de implementar la arquitectura propuesta. Donde las clases visuales, entre ellas, GraphToolVisual, AddPluginVisual y RemovePluginVisual, se conectarán a una clase controladora llamada CC_GestorPlugin. Ella accederá a los datos de los plug-ins a través de la interfaz CE_Plugin -que brinda el Front-End-, la cual servirá de comunicación con los restantes módulos que se deseen incorporar a la plataforma. En la figura 3.6 se muestra una vista de esta arquitectura.

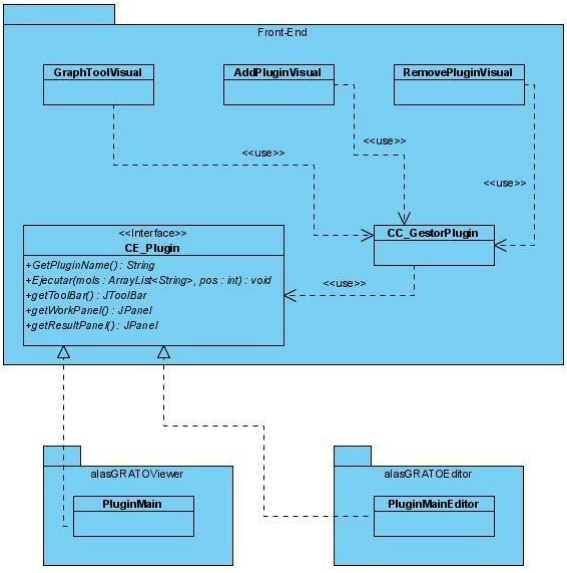


Figura 3.6: Diagrama de la arquitectura de plug-in implementada en el Front-End

3.2.3.1. Fichero XML

En la arquitectura de plug-ins, explicada anteriormente, se utiliza un fichero XML para la construcción de los toolBars de los plug-ins que se desean incorporar al Front-End. El esquema del mismo se muestra en la figura 3.7.

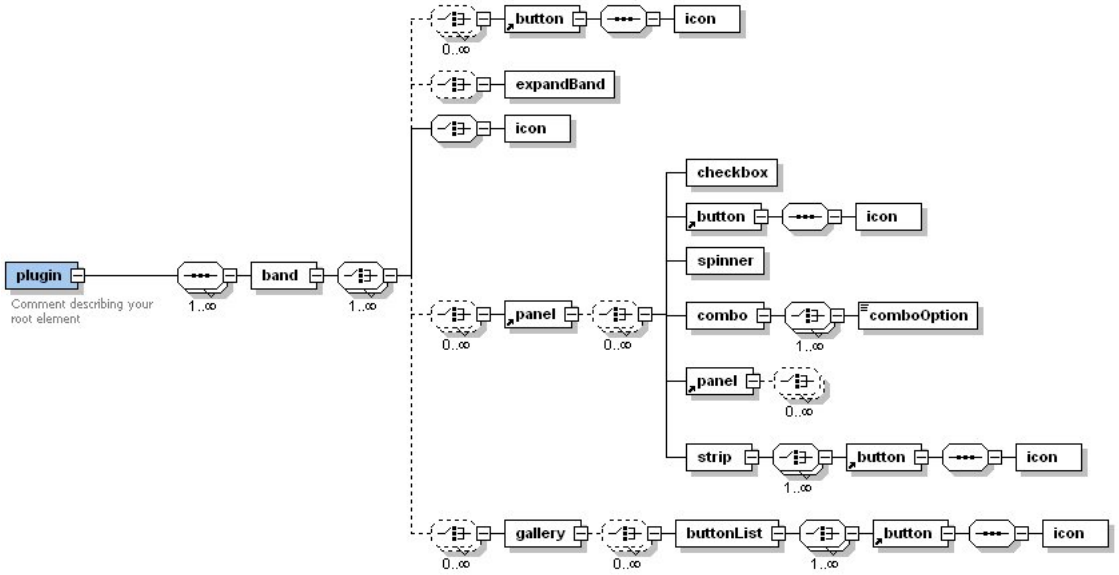


Figura 3.7: Esquema del XML implementado

3.2.3.2. Arquitectura informacional del Front-End

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

1.- Organización de la información.

La información se organiza como se describe a continuación: se mostrarán los plug-ins que se le adicionaran al Front-End en forma de tabs, donde el título de este será el nombre del plug-in. El tab estará compuesto por un toolbar conformado por paneles que a su vez estarán compuestos por botones, galerías, popups, spiner, combox, checkBox entre otros componentes que garantizarán los requisitos funcionales identificados por el equipo de desarrollo. En la figura 3.8 se muestra un ejemplo de cómo debe diseñarse el tab.

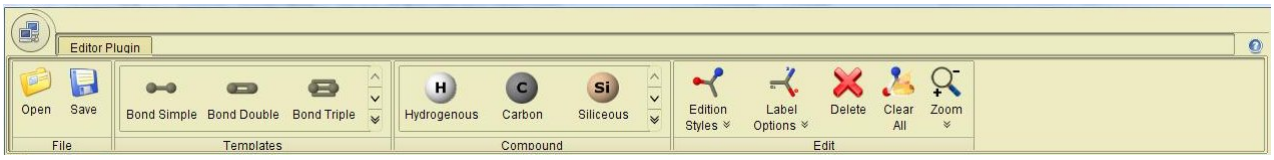


Figura 3.8: Ejemplo de cómo debe ser diseñado un tab.

2.- Ubicar las opciones para facilitar la localización y focalización de la información relevante.

Las funcionalidades serán ubicadas según las semejanzas entre ellas, además de tenerse en cuenta una jerarquía entre los componentes, donde se ubicaron dichas funcionalidades. Todo esto se hace con el objetivo de focalizar al cliente en lo que quiere y para que los componentes queden localizados según las semejanzas entre las acciones a realizar. Dicha jerarquía es fundamental para la concepción de una buena arquitectura informacional por eso a continuación pasamos a explicarla.

Los requisitos con mayor importancia serán representados mediante botones con prioridad máxima, con un mayor tamaño e identificados con sus nombres. Los requisitos secundarios tendrán una prioridad media y serán representados como botones más pequeños con los nombres de los mismos. Los requisitos menos importantes para el plug-in desarrollado tendrán prioridad mínima y serán representados por botones pequeños y estos no tendrán nombres. Las galerías tendrán prioridad máxima y serán ubicadas por delante de los botones con prioridad máxima. Los demás componentes serán representados por la prioridad que tengan: entre media y baja.

A continuación mostramos en la figura 3.9 una imagen que ejemplifica lo antes planteado.

Figura 3.9: Prioridad de los iconos en el Front-End.

3.- Las categorías creadas deben estar de acuerdo con las necesidades de los usuarios.

Las categorías que se creen tendrán mucho que ver con los paneles en donde estén agrupadas las funcionalidades. Cada nombre de ellos deberá reflejar la necesidad del usuario final de la plataforma en su forma más general.

4.- Seguir los estándares aceptados y no confundir a los visitantes con colores.

El sistema está desarrollado según las pautas de diseño de la marca “alas” definidas por la comercializadora de software Albet. Por lo que es necesario que- al incluírsele un plug-in- se tengan en consideración estas pautas.

5.- La estructura soportará los cambios del crecimiento del mismo.

En el toolbar del plug-in existen estructuras que soportan el crecimiento de estas, como por ejemplo, la galería y los popus. Estas estructuras están diseñadas e implementadas de forma tal que si los desarrolladores quisieran agregar una nueva funcionalidad, contenida dentro de estas estructuras, pues solamente tienen que agregarlas al XML, y será visualizado el componente sin ningún tipo de problema, con la nueva funcionalidad en este.

A continuación en la figura 3.10 se muestra un ejemplo de cada uno de estos componentes.

Figura 3.10: Ejemplo de componentes desplegables

6.- Trabajará el sistema aun cuando se adicionen o supriman funcionalidades.

La navegación es asegurada a través del uso de un fichero XML donde se especifican las conformaciones de los toolbars de los plug-ins ,de forma que si el grupo de desarrollo deseara suprimir algún componente de algun toolbar ,para retirar una funcionalidad especifica, sólo debe suprimir ese componente del XML y el sistema cargaría el resto de los componentes sin afectaciones.. Lo mismo pasaría dado el caso en que el grupo deseara incorporar alguna nueva funcionalidad, lo único que en vez de suprimir el componente tendrían que añadirlo bajo el esquema en que se definió el XML.

3.3. Modelo de diseño

Es una abstracción del Modelo de Implementación y su código fuente, el cual se emplea – fundamentalmente- para representar y documentar su diseño. Es usado como entrada esencial en las actividades relacionadas con la implementación. Representa a los casos de uso en el dominio de la solución. A continuación abordaremos los temas relacionados con el diseño del Front-End.

3.3.1. Diagramas de Clases

Un diagrama de clases es un tipo de diagrama estático que describe la estructura de un sistema; también muestra sus clases, atributos y las relaciones entre ellos. Los diagramas de clases son utilizados durante el proceso de análisis y diseño de los sistemas, cuando se crea el diseño conceptual de la información que se manejará en el sistema, conjuntamente con los componentes que se encargarán del funcionamiento y la relaciones entre unos y otros. En los epígrafes siguientes se muestran los diagramas de clases y los diagramas de secuencias tanto del Front-End como del Visualizador Molecular.

3.3.1.1. Diagrama de Clases del Front-End (alasGRATO)

Dada la complejidad del diagrama de clase del Front-End, y para su mejor entendimiento, el diagrama se agrupó por paquetes. El que se muestra a continuación en la figura 3.11

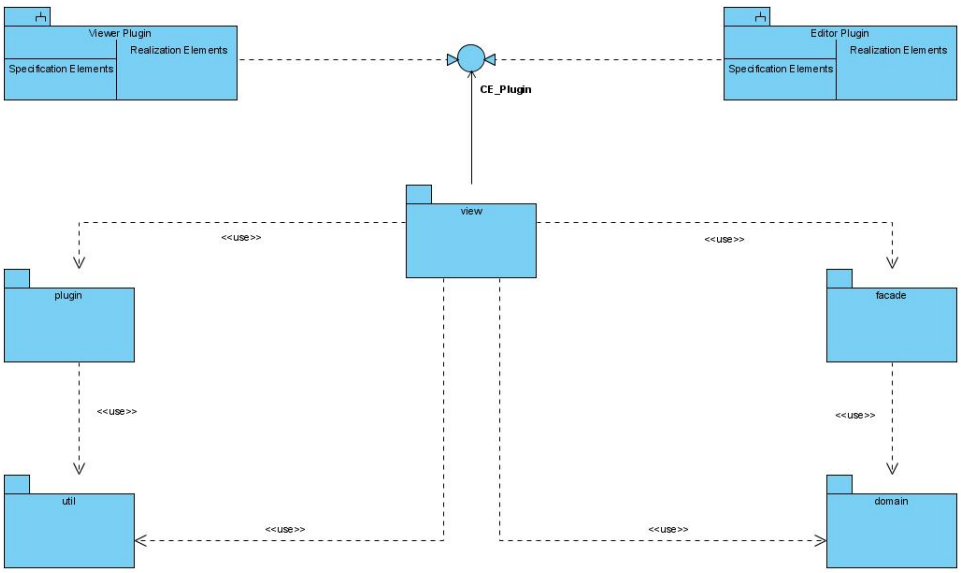


Figura 3.11: Diagramas de Paquete del Front-End (alasGRATO)

A continuación daremos una breve explicación del mismo: A través de la clase CE_Plugin las clases del paquete view recogen la información necesaria para la conformación del plug-in que se desea añadir en el Front-End, como son: el toolbar con sus funcionalidades, el área de trabajo y el área de resultado. Con esta información las clases del paquete view utilizan a las clases contenidas en el paquete plug-ins y ,auxiliándose del las clases del paquete útil construyen el plug-in mencionado con anterioridad. Y se accede a las funcionalidades que él mismo brinda a través de las clases que se encuentran en el paquete facade, el cual gestiona la información de las clases que se encuentran en el paquete domain.

A continuación se muestra el diagrama de clase realizado para cada caso de uso del Front-End

Caso de uso: Gestionar plug-ins

Este caso de uso fue dividido por escenarios, los cuales se muestran a continuación con sus diagramas de clases incluidos.

Escenario Adicionar Plug-ins

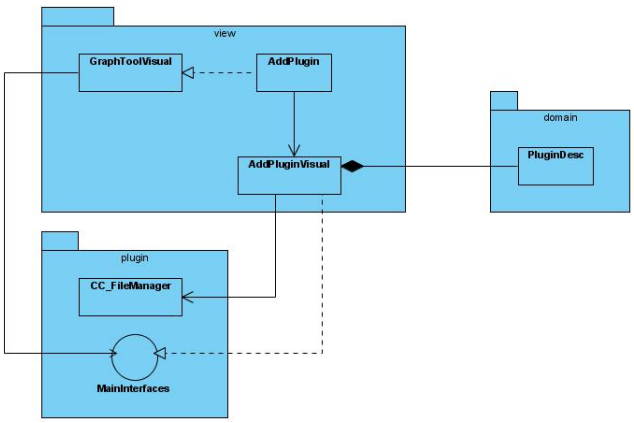


Figura 3.12: Diagrama de clases del escenario adicionar plug-ins

Escenario Remover Plug-in

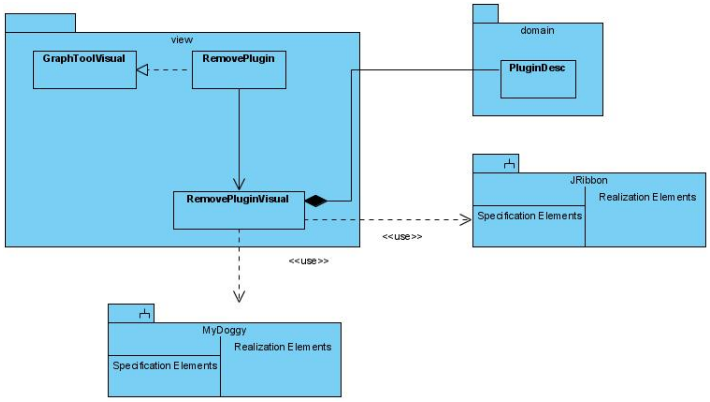


Figura 3.13: Diagrama de clases del escenario remover plug-ins

Escenario Activar Plug-in

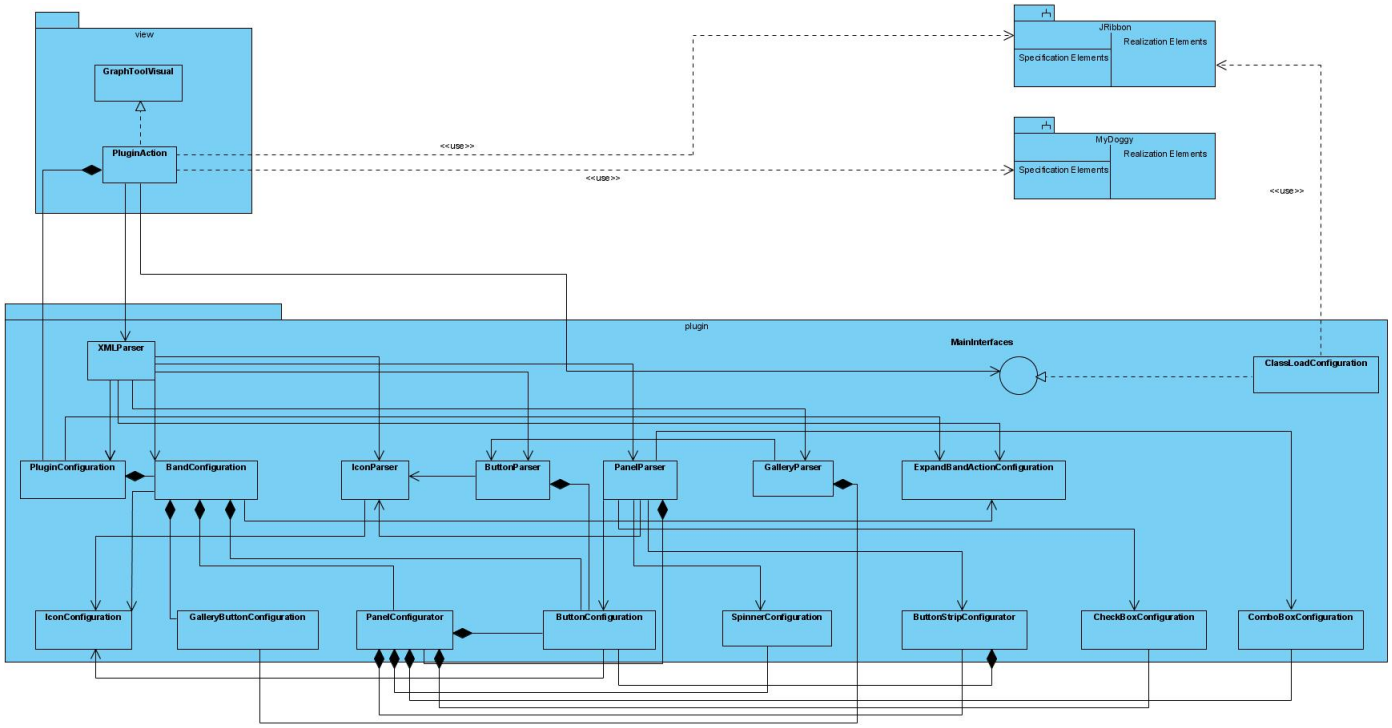


Figura 3.14: Diagrama de clases del escenario activar plug-ins

Caso de uso: Gestionar Preferencia

Este caso de uso fue dividido por escenarios, los cuales se muestran a continuación con sus diagramas de clases incluidos.

Escenario Preferencias de Temas

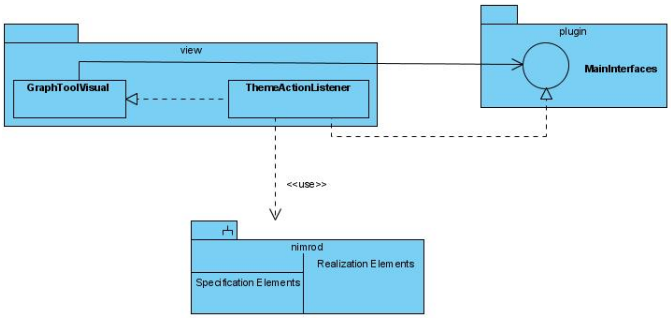


Figura 3.15: Diagrama de clases del escenario preferencias de temas

Escenario Preferencias del Servidor

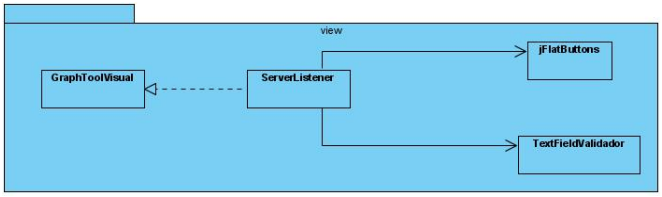


Figura 3.16: Diagrama de clases del escenario preferencias del servidor

Caso de uso: Eliminar Fichero

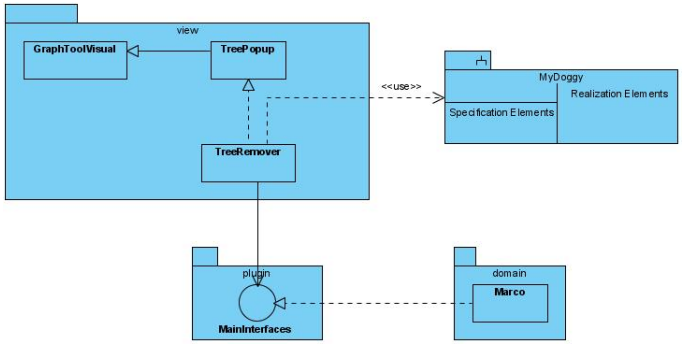


Figura 3.17: Diagrama de clases del caso de uso eliminar fichero

Caso de uso: Manipular Ventanas

En este caso de uso no se presenta ningún diagrama de clase porque todas las funcionalidades que incluye se encuentran desarrolladas en el componente MyDoggy-1.4.2, la cual es utilizada en el desarrollo del Front-End

3.3.1.2. Diagrama de Clases del Visualizador Molecular (alasGRATOViewer)

Por la complejidad del diagrama de clase del plug-in alasGRATOViewer, y para su mejor entendimiento, el diagrama se agrupó por paquetes. El que se muestra a continuación, en la figura 3.18

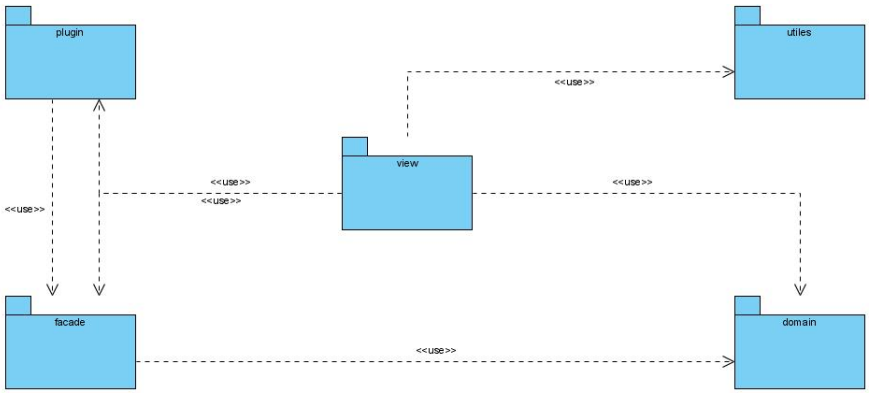


Figura 3.18: Diagramas de Paquete del Visualizador Molecular (alasGRATOViewer)

A continuación daremos una breve explicación del mismo: Las clases del paquete view utilizan a las clases del paquete facade para, a través de estas, obtener información de las clases del paquete domain, que será manipulada en el sistema, quien, además utiliza a las clases del paquete plug-in, para obtener la información necesaria para que, cobren vida, las funcionalidades contenidas en los plug-ins desarrollados. Por último las clases del paquete view se apoyan en el paquete útil donde existen clases de propósitos generales, las cuales permitirán -a las funcionalidades ya antes mencionadas- la creación de componentes necesarios para su funcionamiento.

Caso de uso: Abrir Fichero

Este caso de uso fue dividido por escenarios, los cuales se muestran a continuación con sus diagramas de clases incluidos.

Escenario Abrir

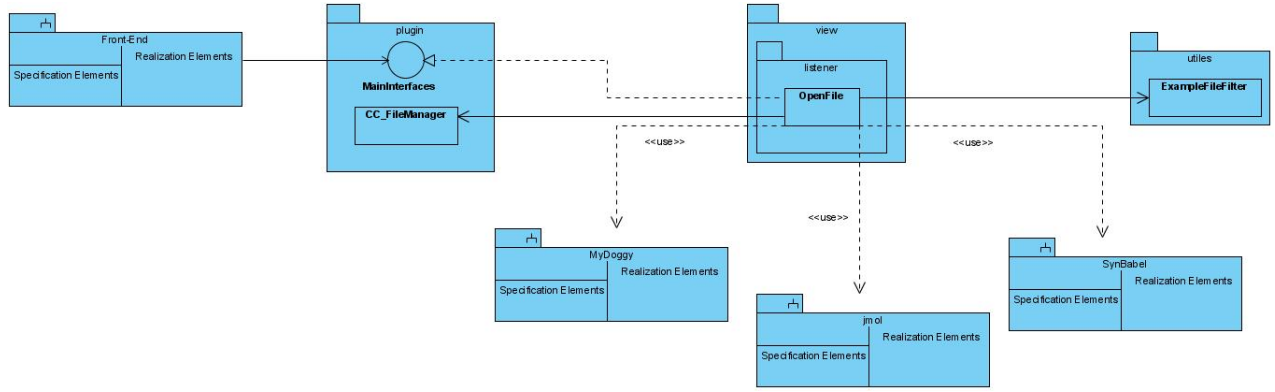


Figura 3.19: Diagrama de clases del escenario abrir

Escenario Abrir Reciente

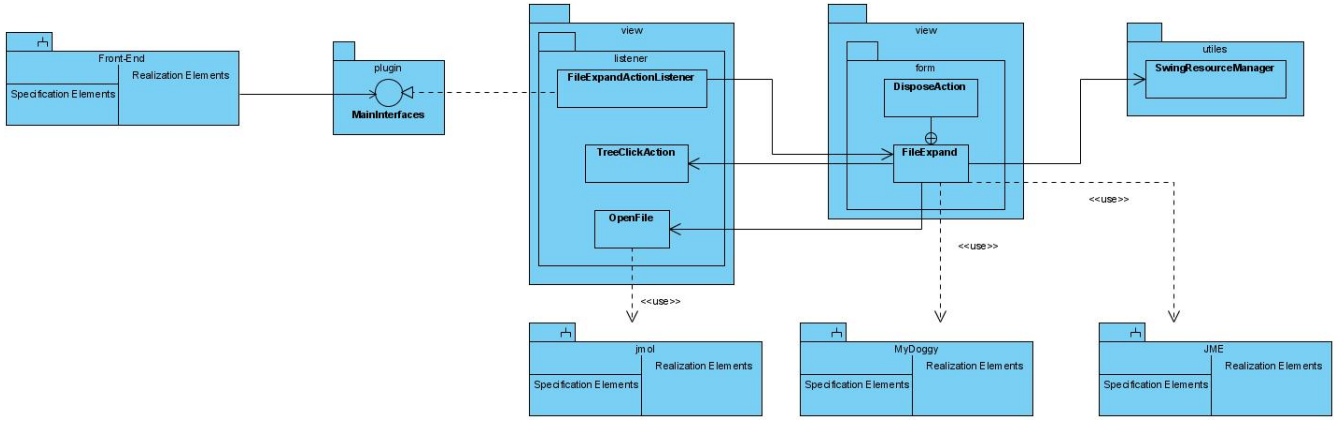


Figura 3.20: Diagrama de clases del escenario abrir recientes

Escenario Abrir Preferencias de Visualización

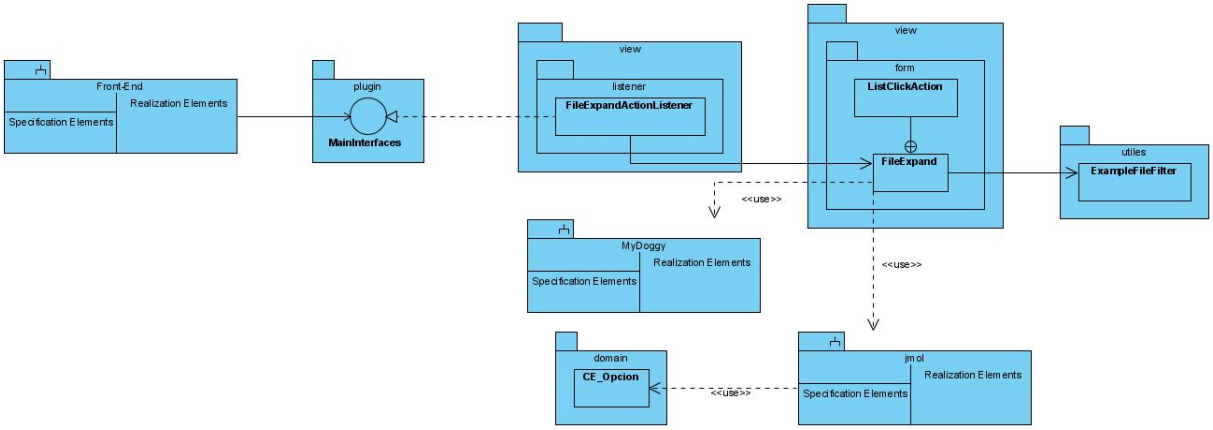


Figura 3.21: Diagrama de clases del escenario abrir preferencias de visualización

Caso de uso: Gestionar Fichero

Este caso de uso fue dividido por escenarios, los cuales se muestran a continuación con sus diagramas de clases incluidos.

Escenario Cerrar Fichero

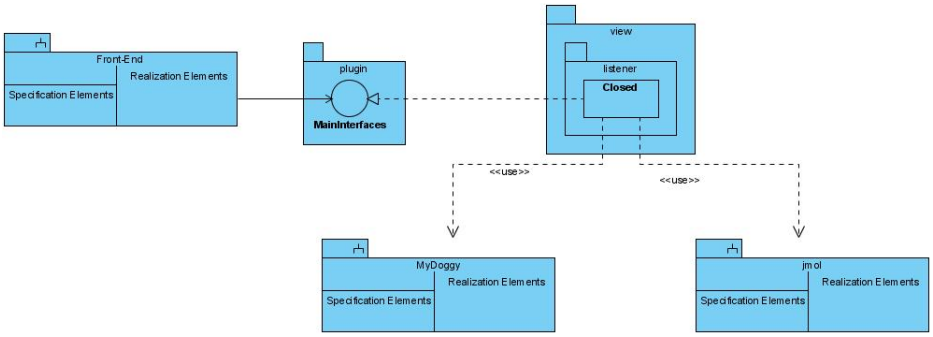


Figura 3.22: Diagrama de clases del escenario cerrar fichero

Escenario Guardar Fichero

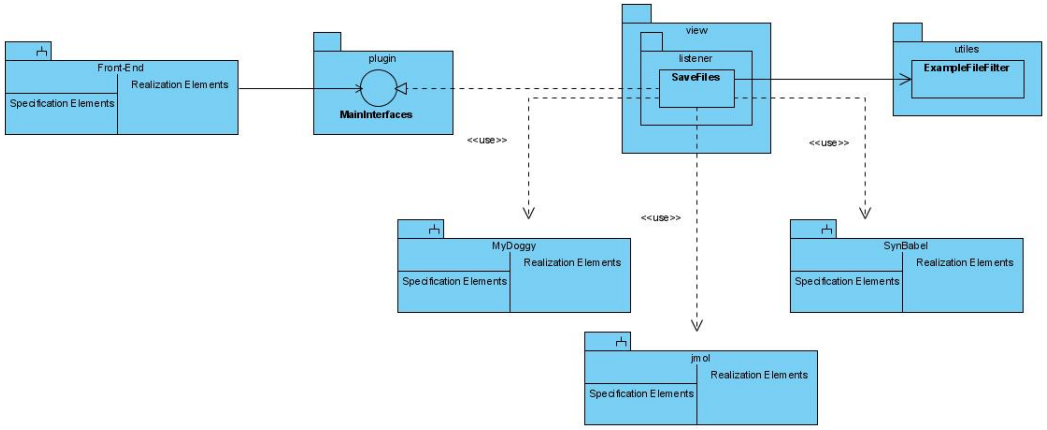


Figura 3.23: Diagrama de clases del escenario guardar fichero

Escenario Exportar Fichero

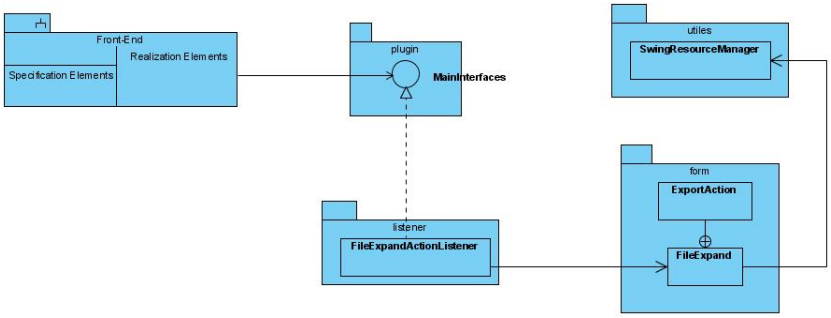


Figura 3.24: Diagrama de clases del escenario exportar fichero

3.3.2. Descripción de las Clases

La descripción de las clases del Front-End (alasGRATO) se encuentra en el anexo 1.

3.3.3. Diagramas de Interacción

Los diagramas de interacción se utilizan para modelar los aspectos dinámicos de un sistema, lo que conlleva modelar instancias concretas o prototípicas de clases interfaces, componentes y nodos, junto con los mensajes enviados entre ellos. Todo en el contexto de un escenario que ilustra un comportamiento. En el contexto de las clases se describe la forma en que grupos de objetos colaboran para proveer un comportamiento. Mientras que un diagrama de casos de uso presenta una visión externa del sistema, las funcionalidades de dichos casos de uso se recoge como un flujo de eventos, y se utilizan para ello, interacciones entre sociedades de objetos.

Para documentar el diseño desde el punto de vista de los casos de uso, se muestran a continuación los diagramas de secuencia por caso de uso, tantos del Front-End como del Visualizador molecular.

3.3.3.1. Diagramas de Secuencias del Front-End (alasGRATO)

Caso de uso: Gestionar plug-ins

Sección Adicionar Plug-ins

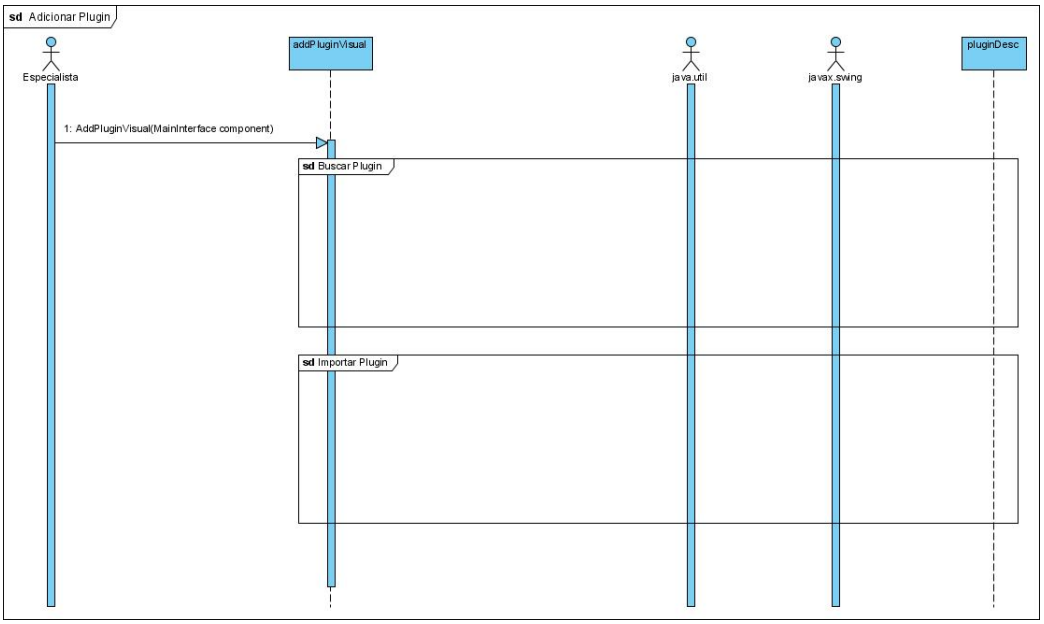


Figura 3.25: Diagrama de secuencia de la sección adicionar plug-ins

Por lo extenso que es el diagrama y para una mejor comprensión del mismo lo separamos en los diagramas de: buscar plug-ins e importar plug-ins. Ver anexo 2.

Sección Remover Plug-ins

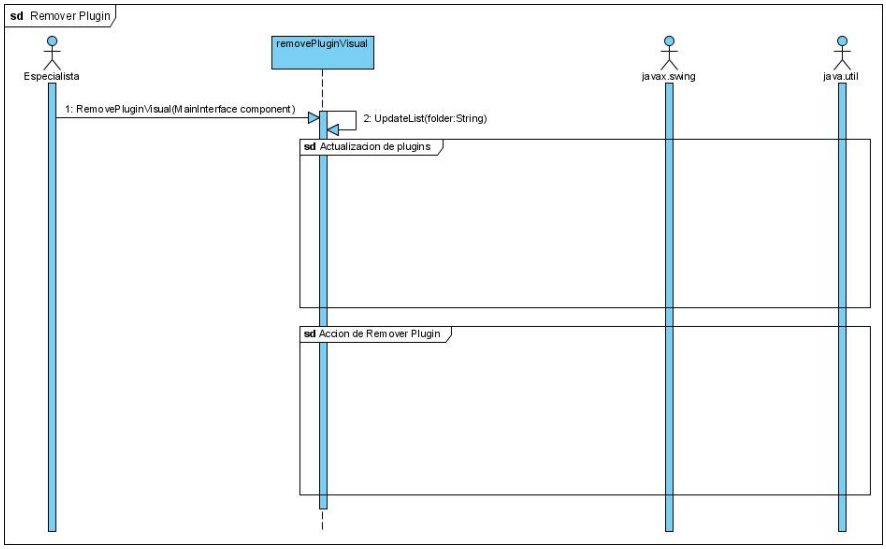


Figura 3.26: Diagrama de secuencia de la sección remover plug-ins

Por lo extenso que resulta el diagrama y para su mejor comprensión, lo separamos en los diagramas de: actualización de plug-ins y acción de remover plug-ins .Ver anexo 3.

Sección Activar Plug-ins

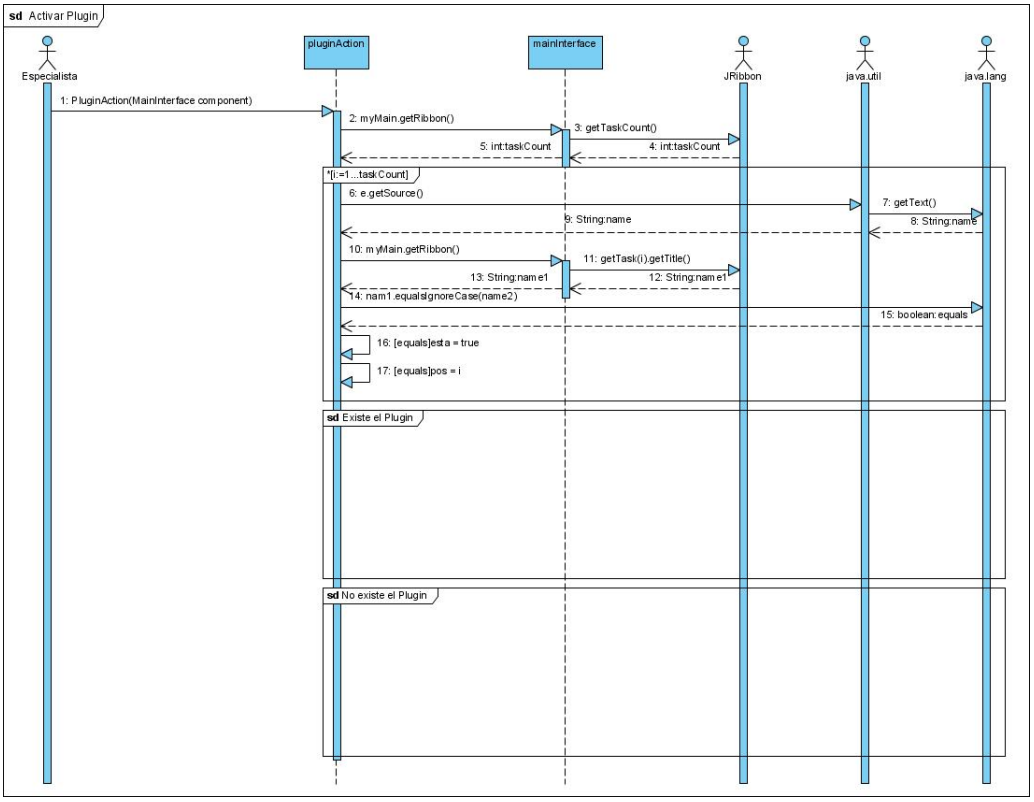


Figura 3.27: Diagrama de secuencia de la sección activar plug-ins

Por lo extenso que es el diagrama y para una mejor comprensión del mismo lo separamos en los diagramas de: existe el plug-in y no existe el plug-in .Ver anexo 4.

Caso de uso: Gestionar Preferencias

Sección Preferencias de Temas

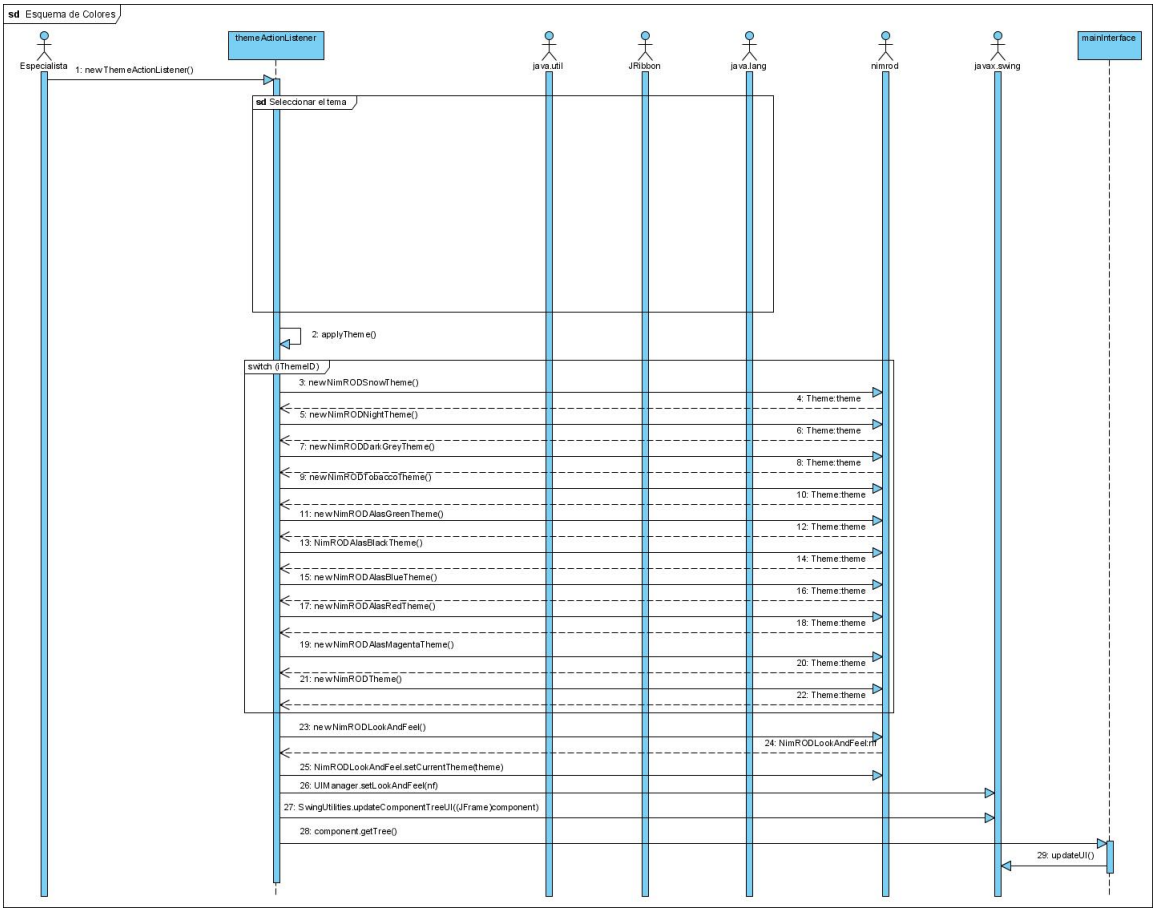


Figura 3.28: Diagrama de secuencia de la sección preferencias de temas

Por lo extenso que es el diagrama y para su mejor comprensión, lo separamos en el diagrama de seleccionar el tema. Ver anexo 5.

Sección Preferencias del Servidor

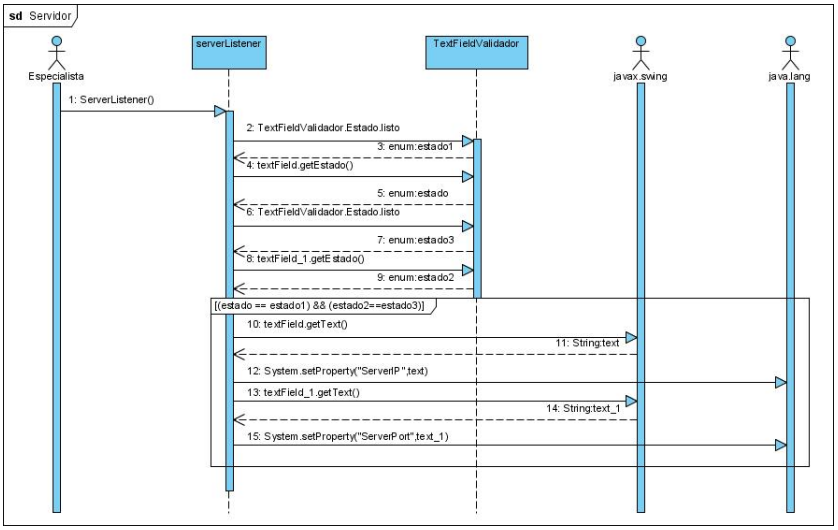


Figura 3.29: Diagrama de secuencia de la sección preferencias del servidor

Caso de uso: Eliminar Fichero

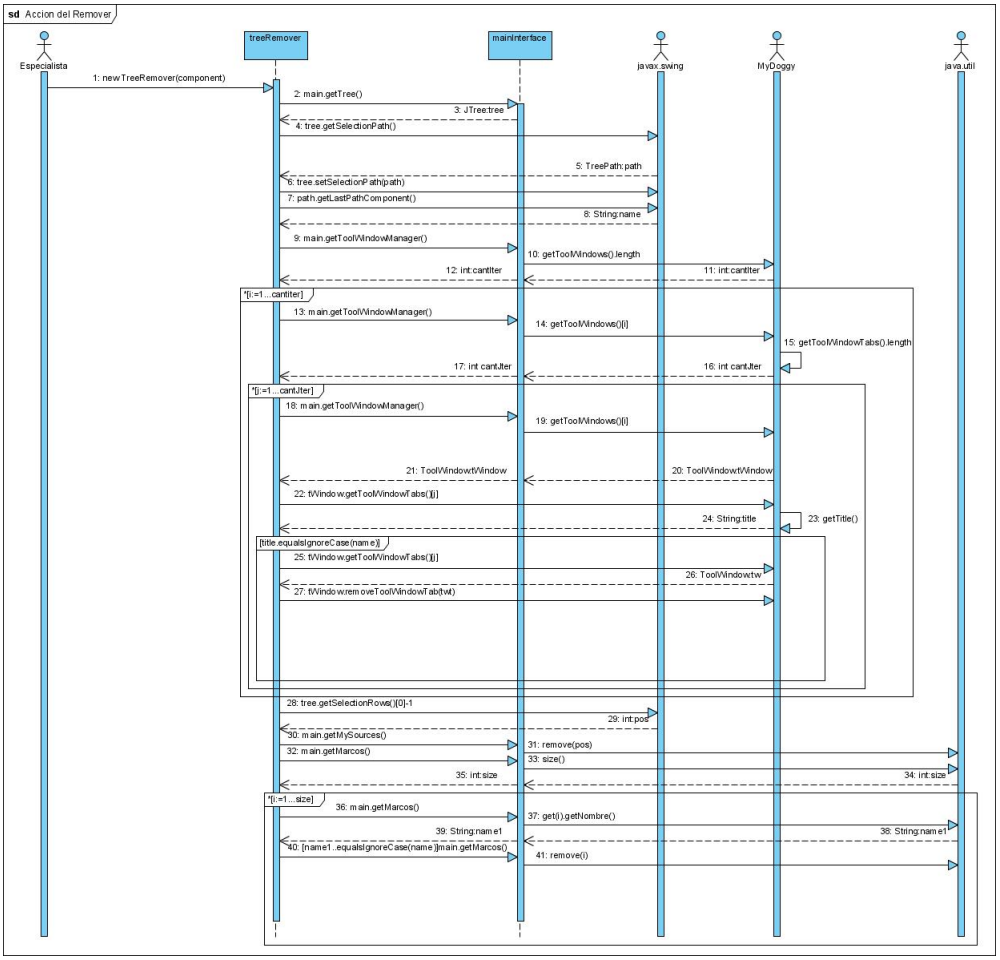


Figura 3.30: Diagrama de secuencia del caso de uso eliminar fichero

3.3.3.2. Diagramas de Secuencias del Visualizador Molecular (alasGRATOViewer)

Caso de uso: Abrir Fichero

Sección Abrir

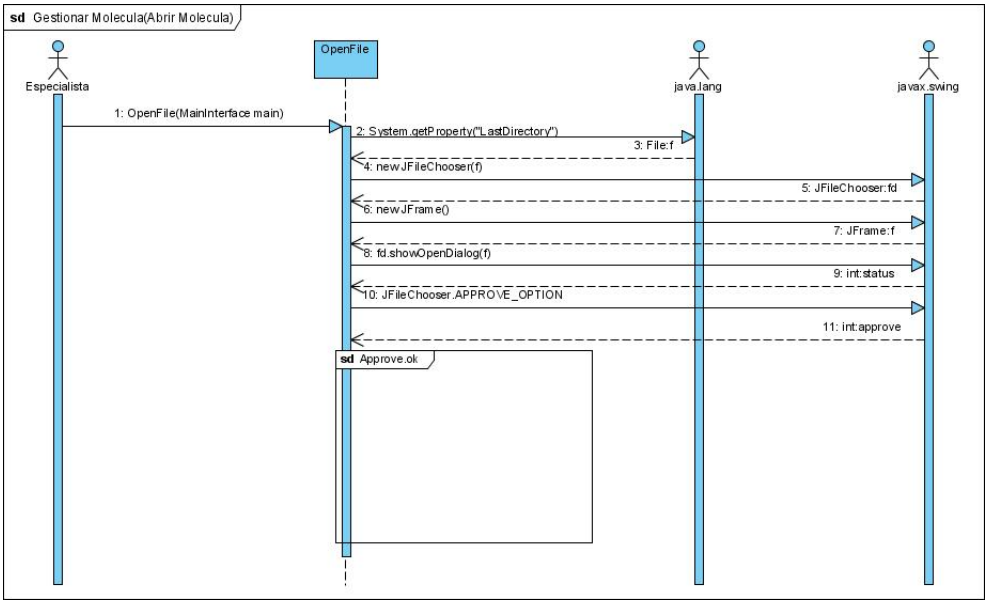


Figura 3.31: Diagrama de secuencia de la sección abrir

Por lo extenso que es el diagrama y para su mejor comprensión , lo separamos en el diagrama Approve ok .Ver anexo 6.

Sección Abrir Reciente

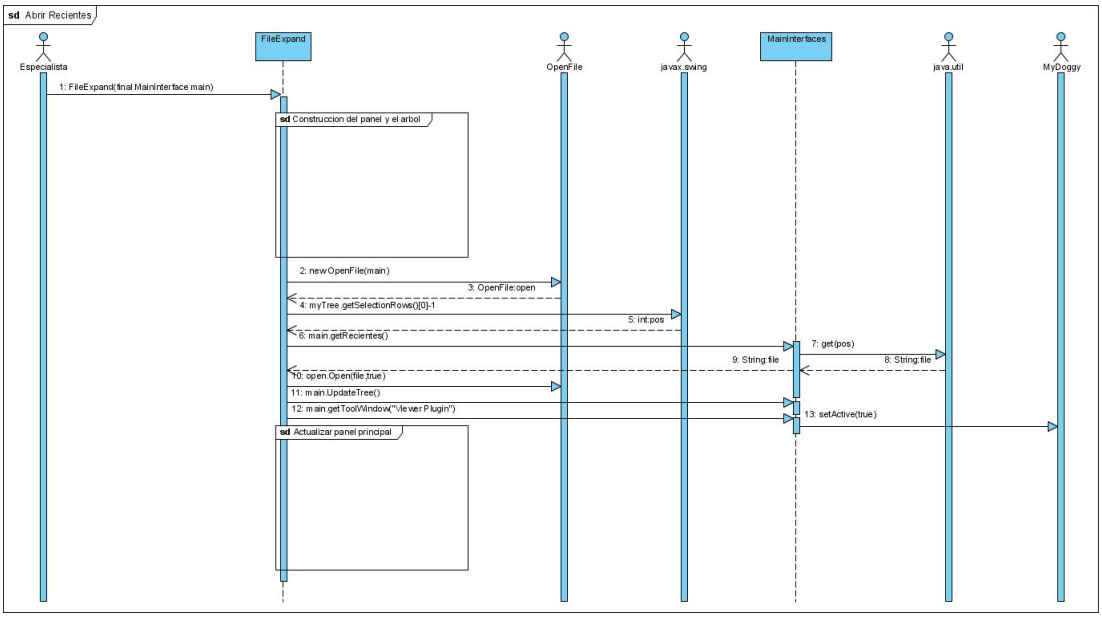


Figura 3.32: Diagrama de secuencia de la sección abrir reciente

Dada la extensión del diagrama y para su mejor comprensión , lo separamos en los diagramas: construcción del panel y el árbol que se encuentran y actualizar panel principal.Ver anexo 7.

Sección Abrir Preferencias de Visualización

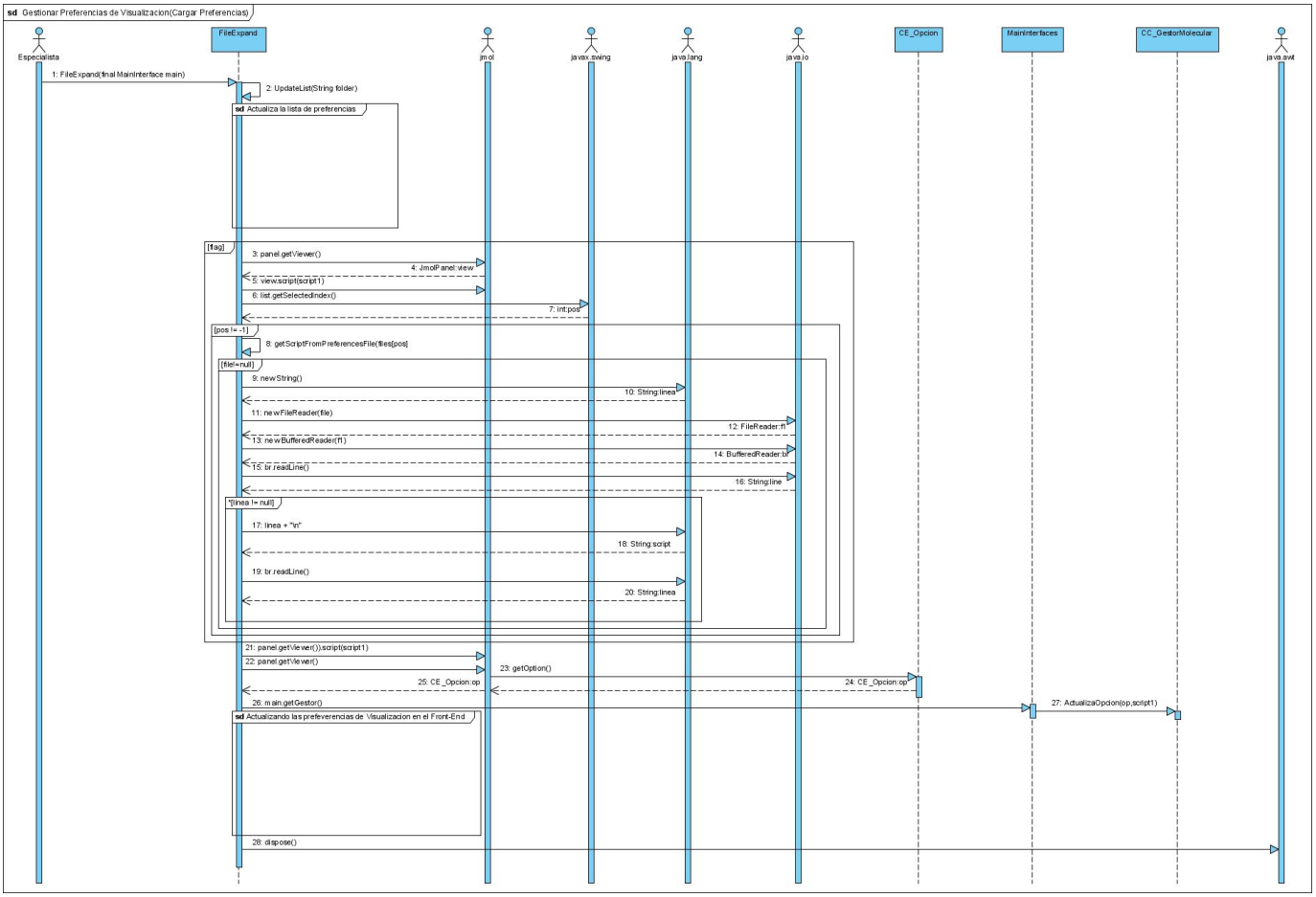


Figura 3.33: Diagrama de secuencia de la sección abrir preferencias de visualización

Por lo extenso que es el diagrama y para su mejor comprensión , lo separamos en los diagramas: actualizar la lista de preferencias y actualizando las preferencias de visualización en el Front-End .Ver anexo 8.

Caso de uso: Gestionar Fichero

Sección Cerrar Fichero

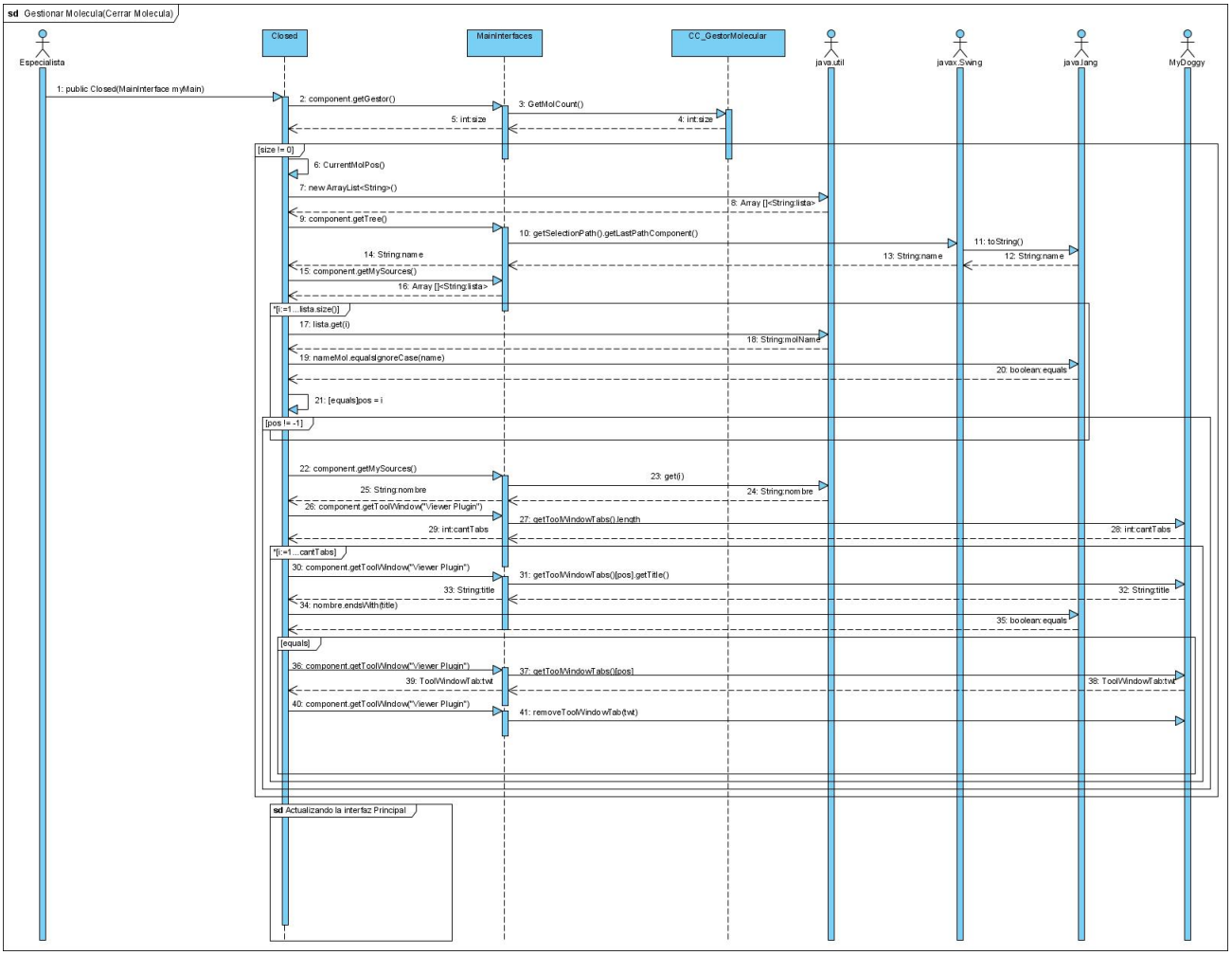


Figura 3.34: Diagrama de secuencia de la sección cerrar fichero

Por lo extenso que es el diagrama y para una mejor comprensión, lo separamos en el diagrama actualizando la interfaz principal. Ver anexo 9.

Sección Guardar Fichero

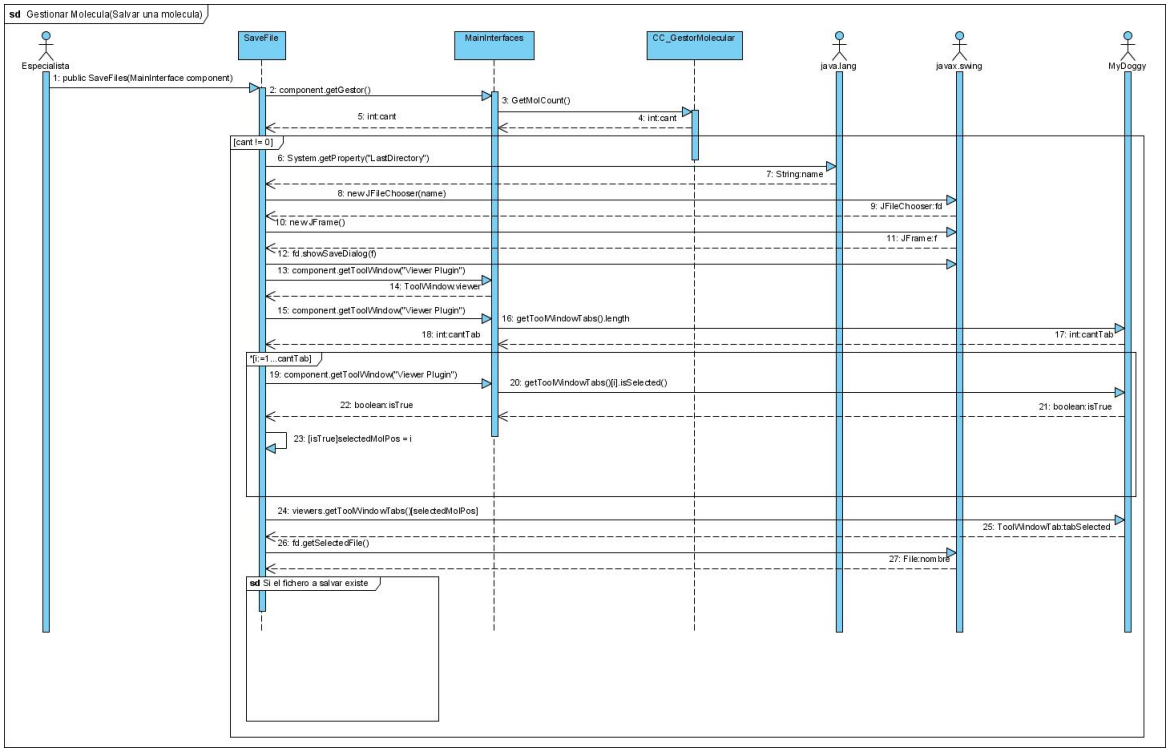


Figura 3.35: Diagrama de secuencia de la sección guardar fichero

Por lo extenso que es el diagrama y para su mejor comprensión, lo separamos en el diagrama si el fichero a cargar existe .Ver anexo 10.

Sección Exportar Fichero

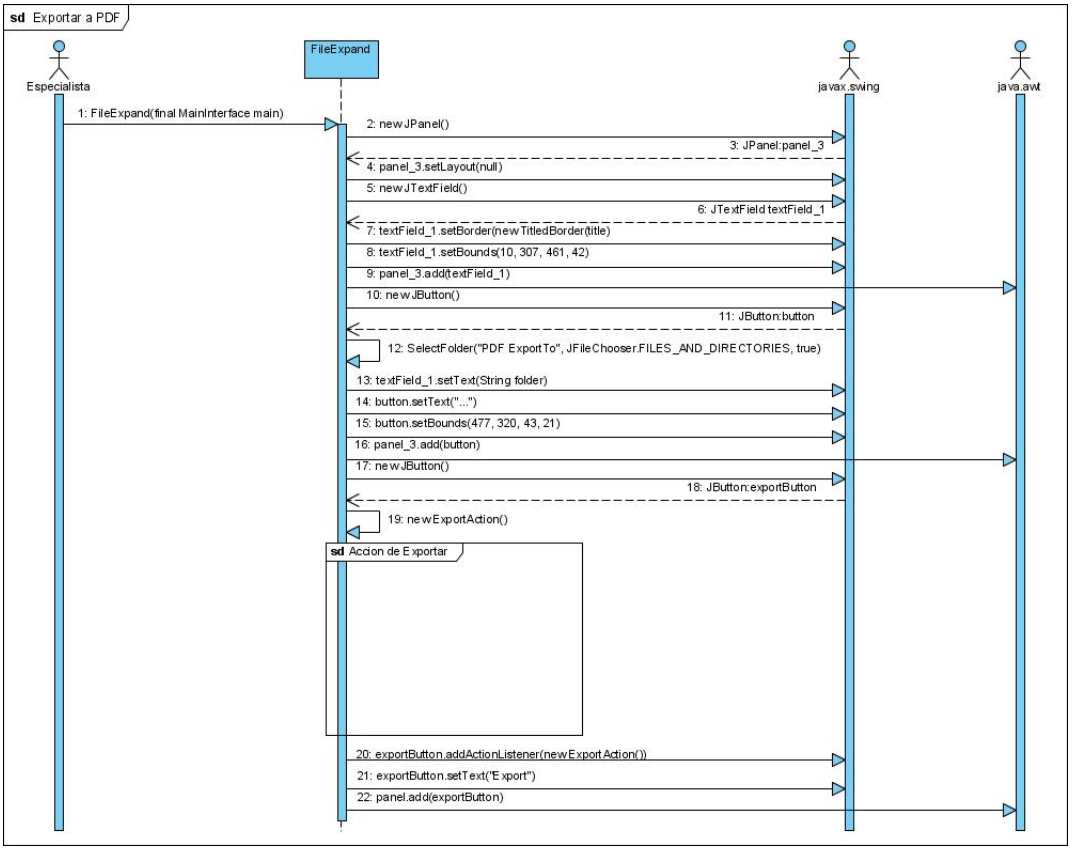


Figura 3.36: Diagrama de secuencia de la sección exportar fichero

Por lo extenso que es el diagrama y para su mejor comprensión, lo separamos en el diagrama acción de exportar. Ver anexo 11.

Caso de uso: Convertir a 3D

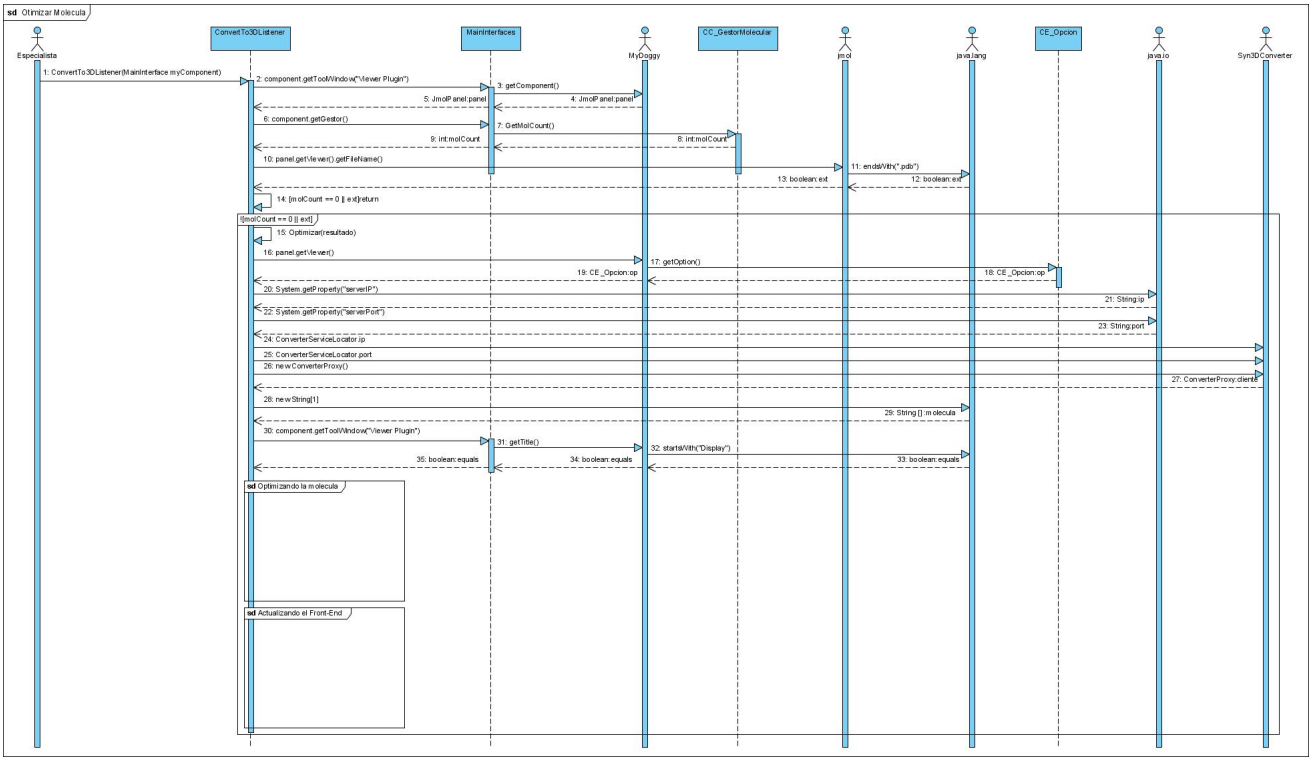


Figura 3.37: Diagrama de secuencia del caso de uso convertir a 3D

Por lo extenso que es el diagrama y para su mejor comprensión, lo separamos en los diagramas: optimizando la molécula y actualizando el Front-End. Ver anexo 12.

3.3.4. Principios para el Diseño del Front-End (alasGRATO)

Para el diseño e implementación del Front-End se utilizaron los mismos principios de Leopoldo Sebastián[17]:

1.- **Percepción del Color**: Aunque se utiliza una convención de color por defecto en el Front-End desarrollado, el mismo presenta un mecanismo secundario llamado “cambiar temas de colores”, el cual permite que el usuario -en tiempo de ejecución- pueda seleccionar la combinación de colores que desee dentro de las existentes.

En la figura 3.38, se muestra la cantidad de temas de colores que presenta el Front-End desarrollado hasta estos momentos.

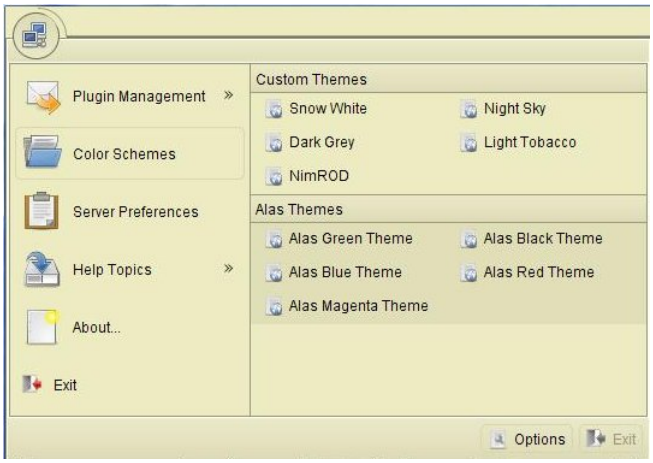


Figura 3.38: Cantidad de temas de colores que presenta el Front-End.

2.- **Consistencia**: Para lograr una mayor consistencia en la IU se requiere profundizar en diferentes aspectos que están catalogados por niveles, los cuales se muestran a continuación.

2.1.- **Estructuras invisibles**: Para las estructuras invisibles se requiere una definición clara de las mismas. Para lograr esto en el Front-End desarrollado se utiliza la ampliación de ventanas, mediante la extensión de sus bordes. En la figura 3.39, se muestra con flechas de color rojo los símbolos utilizados para la extensión de los bordes.

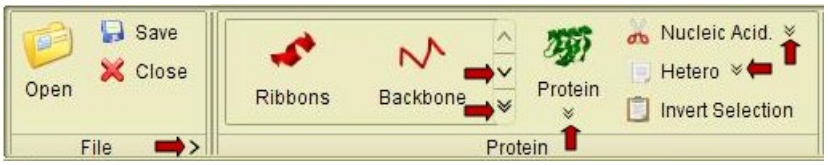


Figura 3.39: Símbolos utilizados para la extensión de los bordes

2.2.- **Una sola aplicación**: El Front-End se visualiza como un componente único, es decir, despliega un único menú. En la figura 3.40, se muestra el menú principal del Front-End.

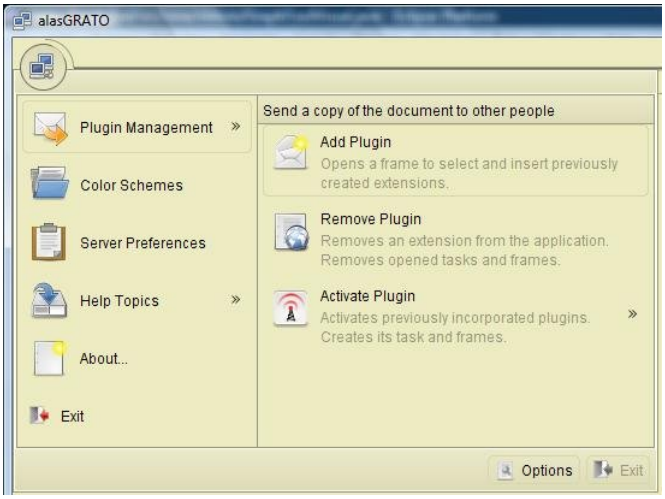


Figura 3.40: Menú Principal del Front-End

2.3.- **Un conjunto de aplicaciones o servicios**: El Front-End permite visualizar un conjunto de módulos o servicios. Se utiliza un componente que genera una barra de comandos distribuida en un tab, la cual puede ser activada o removida de forma independiente. En la figura 3.41, se muestra un ejemplo de un tab creado por el Front-End.

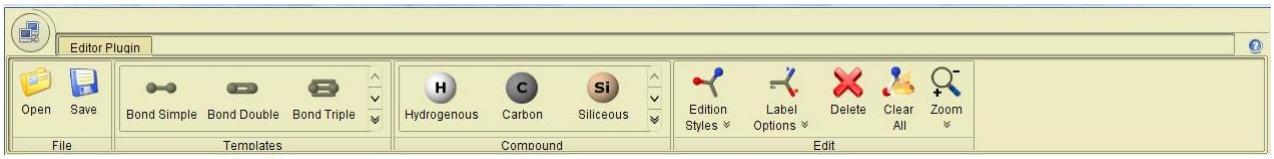


Figura 3.41: Ejemplo de barra de tareas del Front-End.

2.4.- **Consistencia del ambiente**: Todos los módulos incorporados al Front-End presentan el mismo ambiente de trabajo, debido a que se usan objetos de control de comandos de manera análoga.

2.5.- **Consistencia de la plataforma**: El Front-End presenta un esquema basado en ventanas, para ello, utiliza el componente MyDoggy, el cual es compatible al manejo de ventana en los sistemas operativos.

3.- **Ley de Fitt**: El tiempo para alcanzar un objetivo es una función de la distancia y tamaño del objetivo. Es por ello, que existe en el Front-End una política de prioridad de los componentes, en la medida de su importancia y su uso. Esta política establece que existen 3 prioridades: alta, media y baja.

La Alta se representa con un icono más grande y con su nombre; la media se representa con un icono mediano y con su nombre; y la baja se representa con un icono pequeño y sin nombre. A continuación, en la figura 3.42, se muestra un ejemplo de esta política de prioridad.

Figura 3.42: Política de prioridad de iconos en el Front-End.

4.- **Legibilidad**: El Front-End fue desarrollado bajo las pautas de la marca "alas", definidas por la comercializadora Albet. Se garantiza de esta forma que el texto presente un alto contraste, que las combinaciones de colores utilizadas sean correctas y que el tamaño de las fuentes sean las adecuadas para poder ser leído en monitores estándares.

3.3.5. Diagrama de Despliegue

En la figura 3.43 se muestra el diagrama de despliegue que presenta la "Plataforma Inteligente para la Predicción de Actividad Biológica en Compuestos Orgánicos".En ella se resalta -en el rectángulo de color azul- donde será desplegado el Front-End desarrollado. Este se desplegará en las PC clientes, debido a que será la interfaz de usuario que utilizaran los clientes para interactuar con la plataforma.

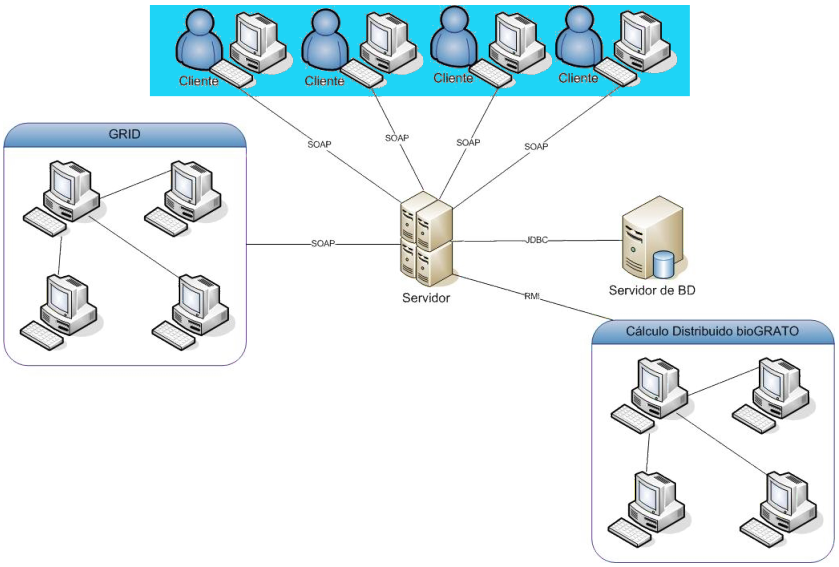


Figura 3.43: Diagrama de despliegue

3.4. Conclusiones

- Para la implementación del sistema, se utilizó el patrón arquitectónico modelo-vista controlador
- Para realizar el diseño del Front-End, se utilizaron los patrones de diseño: Factory, Facade, Creador, Bajo Acoplamiento y Alta Cohesión.
- Para llevar a cabo el diseño del Front-End se aplicaron algunos de los principios para el diseño que se comentaron en este capítulo como son: Percepción del Color, Legibilidad, Ley de Fitt, entre otros.

Capítulo 4

Implementación y Prueba del Sistema

4.1. Introducción

En este capítulo se describe la implementación del sistema, mostrando algunos de los diagramas de componentes y una serie de pasos obligatorios a seguir para implementar un nuevo plug-in. Además se presenta todo lo relacionado con las pruebas realizadas al sistema.

4.2. Implementación

4.2.1. Diagrama de Componentes

Los diagramas de componentes muestran las organizaciones y dependencias lógicas entre componentes de software, sean estos de código fuente, binarios, archivos, bibliotecas cargadas dinámicamente o ejecutables.

Por la complejidad del sistema a elaborar, se consideró que no era factible una representación del diagrama de componentes de forma explícita. Así, se tomó la determinación de que este fuera agrupado por paquetes, en concordancia con lo ya planteado anteriormente al representar el diagrama de clases del diseño del sistema. En la figura 4.1 se muestra el diagrama de componentes del Front-End agrupado por paquetes.

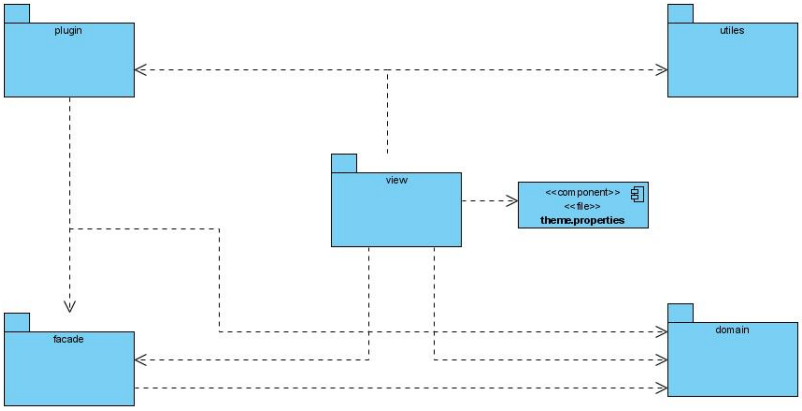


Figura 4.1: Diagrama de componentes del Front-End.

En el anexo 13, se muestra el diagrama de componentes de cado uno de los paquetes presentes en la figura 4.1.

Para la compresión del mismo explicaremos algunos componentes que consideramos fundamentales para la implementación del sistema desarrollado. Se irán mostrando las partes del diagrama donde se evidencia el uso de dichos componentes.

Flamingo: Componente que le permite al sistema la construcción de toolBars con la filosofía de Office, los mismos son especificados por los desarrolladores a través de un fichero XML. La utilización se puede apreciar en la figura 4.2, donde se muestra el diagrama de componentes del paquete plug-in.

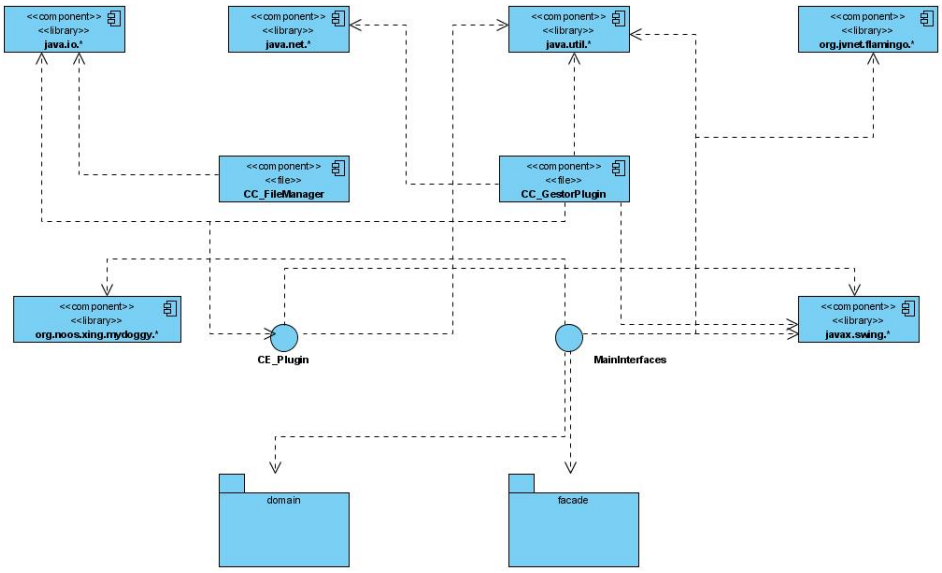


Figura 4.2: Diagrama de componentes del paquete plug-in.

MyDoggy: Componente que permite al sistema la manipulación de ventanas. Gestiona el caso de uso manipular ventanas descrito en el capítulo 2. La utilización del mismo se puede apreciar en la figura 4.3, donde se muestra el diagrama de componentes del paquete plug-in en el visualizador molecular.

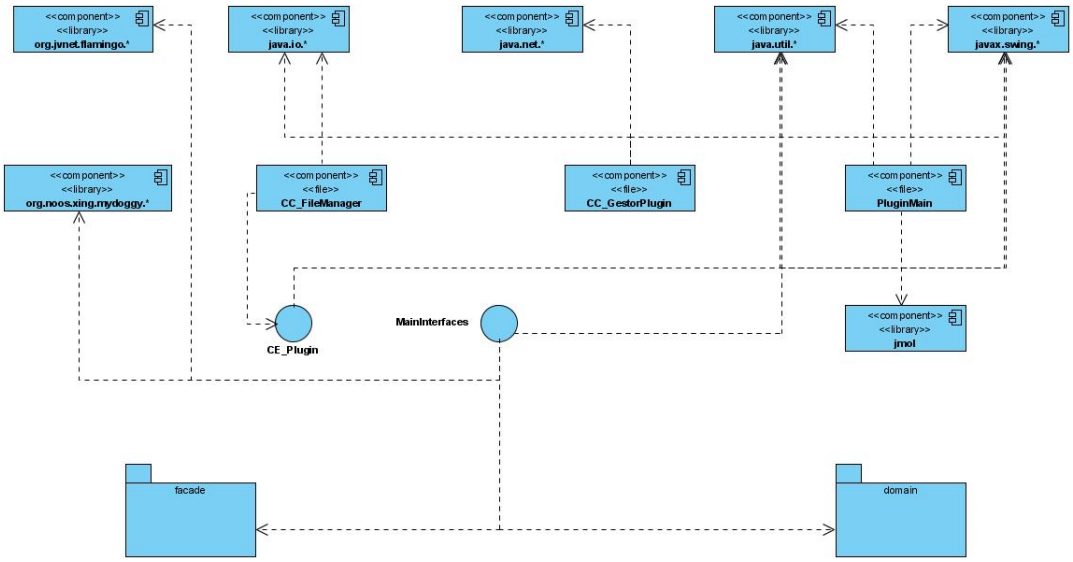


Figura 4.3: Diagrama de componentes del paquete plug-in en el visualizador molecular.

Jaxen: Componente que permite al Front-End ser capaz de obtener las descripciones de los toolBars de los plug-ins a partir del fichero XML para su posterior construcción. La utilización del mismo se puede apreciar

en la figura 4.4, donde se muestra el diagrama de paquetes plugin_parser.

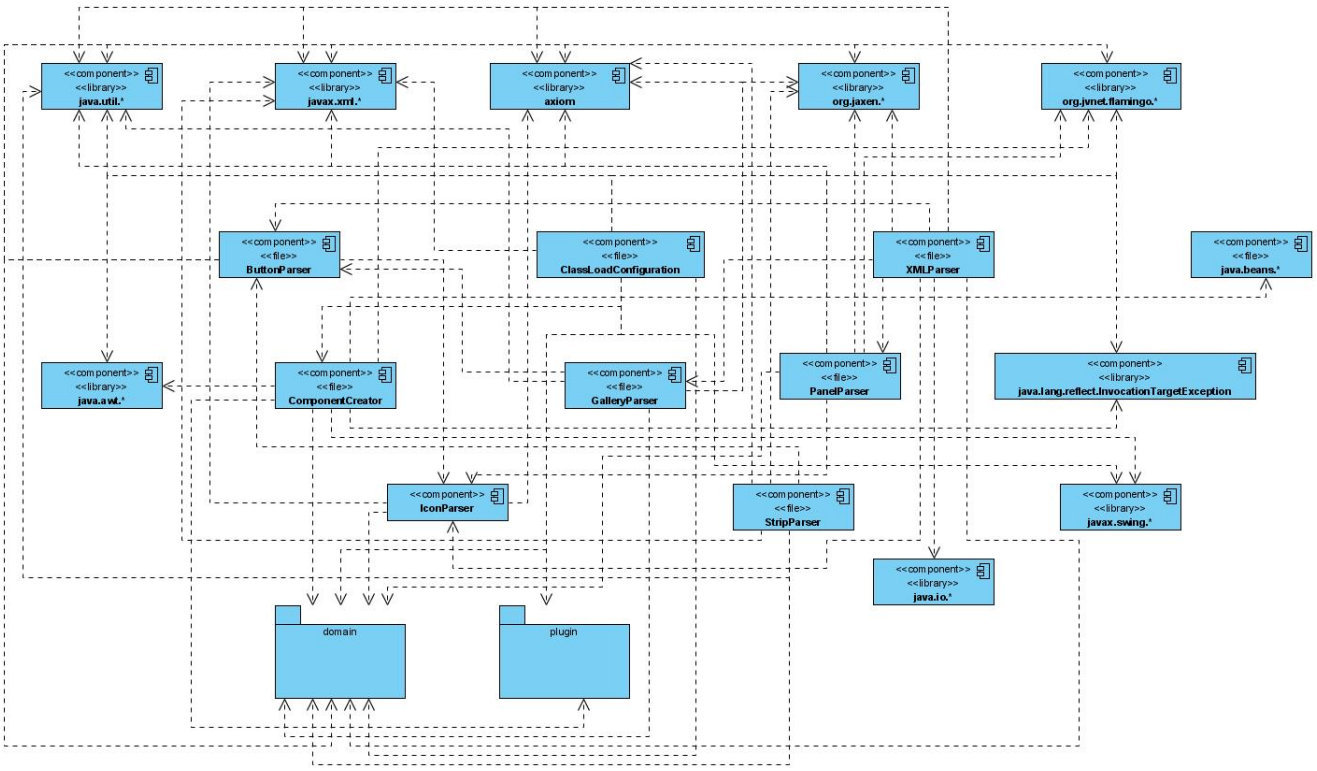


Figura 4.4: Diagrama de componentes del paquete plugin_parser

Theme.propierte: Fichero que se utiliza para guardar en memoria el estado del trabajo del cliente en un momento determinado, siendo luego utilizado por la plataforma para su retomo posterior.

Además de un fichero .XML, que se encuentra contenido en el compilado o .jar, donde el sistema leerá para obtener las especificaciones de los aspectos visuales de los toolBars y donde además, estarán recogidos los eventos a realizar por los diferentes componentes visuales que se le incorporen a los toolBars.

4.2.2. Pasos para Implementar un Plug-in para el Front-End.

Para la implementación de un nuevo plug-in para la plataforma, es necesario cumplir con los siguientes pasos:

1.- Conformar un XML con el mismo nombre del plug-in que cumpla con el XSD desarrollado por el grupo de arquitectura del Front-End, donde se reflejan datos como: nombre del plug-in y acciones que realiza el mismo, o sea, la dirección de las clases de las acciones a desarrollar, además de las extensiones con las que

trabajará el plug-in. Por último, para adicionar los componentes que se desean al toolbar, se deberá respetar la jerarquía que se presenta en el XSD.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v5 U (http://www.xmlspy.com) by Julio (UCI) -->
<plugin xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="D:\Trabajo\Work\JRibbon2\plugins.xsd"
name="Viewer Plugin" accion="view.listener.ViewMolAction" exts="mol pdb"
description="View Molecule">
    <band name="File">
        <icon height="32" width="32" iconsource="view/resource/svg/
        preferences-desktop-theme.svg"/>
        <button name="Open" action="view.listener.OpenFile"
        classType="" mnemonic="O" popupClassType=""
        tooltipText="Open file" priority="TOP">
            <icon height="32" width="32" iconsource="view/resource/
            svg/open.svg"/>
        </button>
        <button name="Save" action="view.listener.SaveFiles"
        classType="" mnemonic="S" popupClassType=" "
        tooltipText="Save Current Molecule As..."
        priority="MEDIUM">
            <icon height="32" width="32" iconsource="view/resource/
            svg/save.svg"/>
        </button>
        <button name="Close" action="view.listener.Closed"
        classType="" mnemonic="C" popupClassType=" "
        tooltipText="Close Current Viewer"
        priority="MEDIUM">
            <icon height="32" width="32" iconsource="view/resource/
            svg/close.svg"/>
```

```
        </button>
        <expandBand expandActionListener="view.listener.
      FileExpandActionListener"/>
  </band>
```

2.- Implementar el plug-in cumpliendo estrictamente con la estructura de paquete definido en la arquitectura, donde las acciones estarán separadas de las vistas, debido a que se utiliza programación orientada a evento.

En la figura 4.5, se observa la estructura de paquete que muestra cómo deben agruparse las clases para implementar un plug-in.

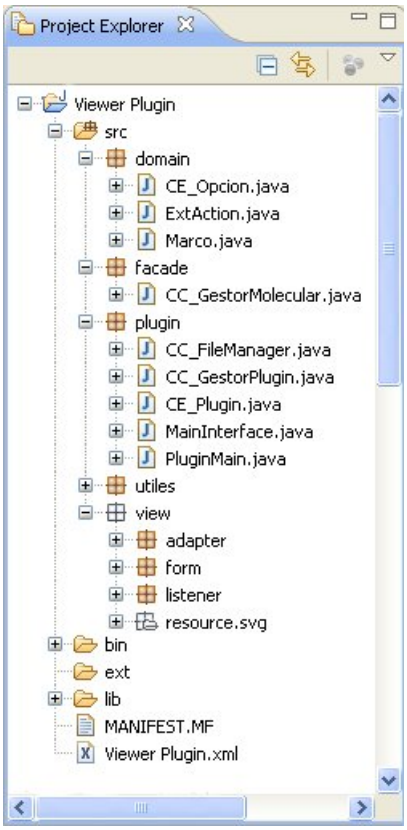


Figura 4.5: Estructura de paquete para implementar un plug-in

3.- La clase principal del plug-in a desarrollar se implementará como la clase CE_Plugin, por lo tanto, es obligatorio redefinir los métodos abstractos presente en la misma.

4.- En el fichero Manifest del plug-in se especificará la dirección de la clase principal del plug-in en el campo PluginBaseClass y las librerías utilizadas.

```
Manifest-Version: 2.0

Name: Grato
Created-By: Julio Antonio Villaverde Martinez
PluginName: alasGRATOViewer
PluginBaseClass: plugin.PluginMain
Description: Molecular Viewer for alasGRATO Platform
Class-Path: lib/Syn3DConverter.jar lib/SynBabel.jar lib/jmol.jar
 lib/jme.jar
Version: 1.0
```

5.- La carpeta que contiene el ejecutable (.jar) del plug-in, es de obligatoria inclusión, en caso que utilice librerías de una carpeta llamada lib y que contenga las librerías antes mencionadas.

4.2.3. Prototipos funcionales del Front-End

A continuación se muestra en la figura 4.6 un prototipo funcional del Front-End (alasGRATO).

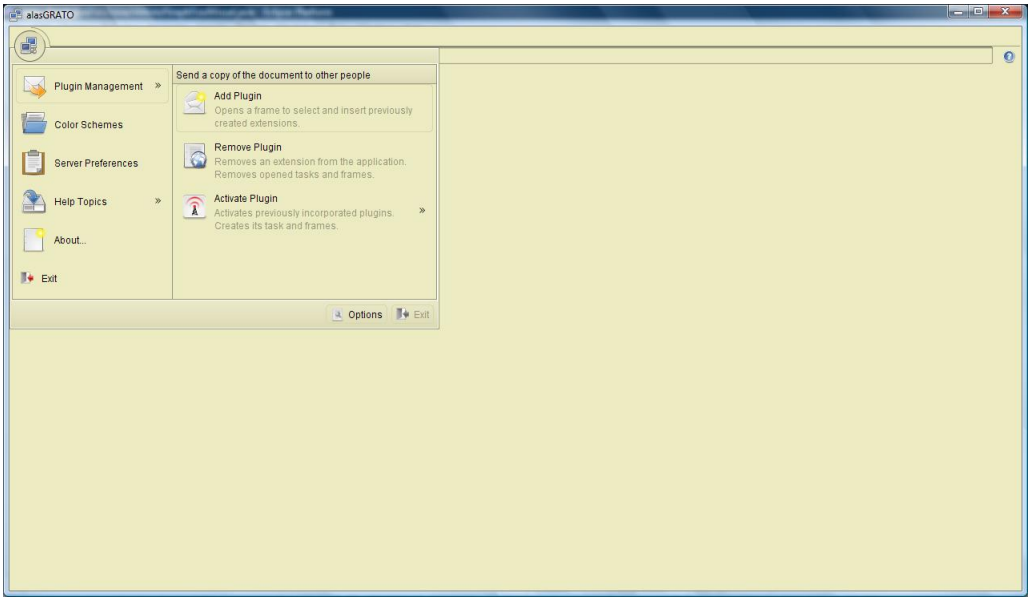


Figura 4.6: Prototipo Funcional del Font-End

Además se muestra en la figura 4.7 un prototipo funcional del plug-in visualizador molecular (alasGRATOViewer).

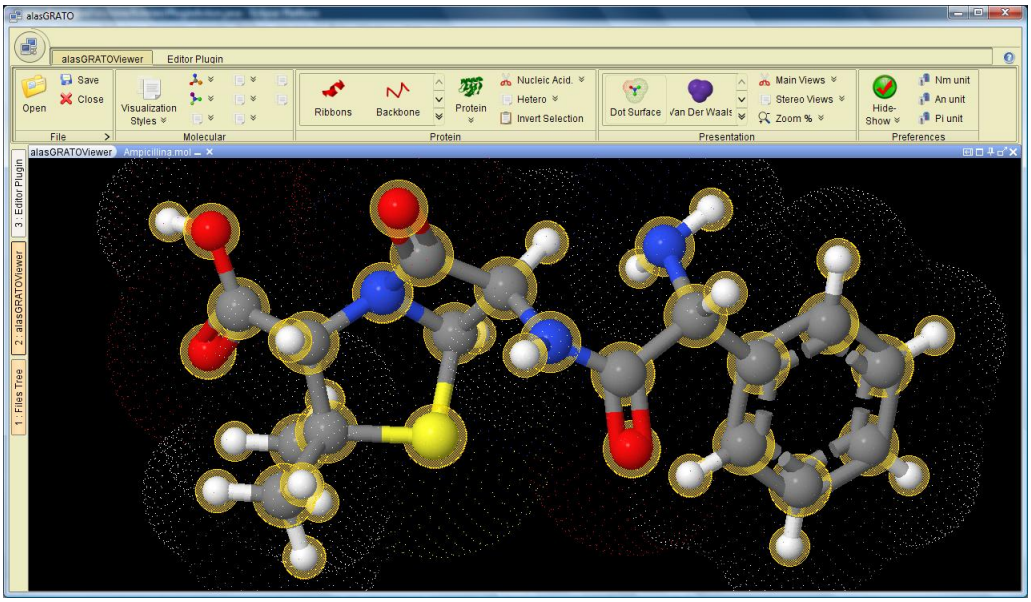


Figura 4.7: Prototipo funcional del plug-in Visualizador Molecular

4.3. Pruebas del Sistema

La prueba de software es un proceso que corre en paralelo al proceso de desarrollo del software, usado para identificar posibles fallos de implementación, calidad, o usabilidad de un software. Se realiza por convencimiento de que: todo sistema debe ser revisado con el objetivo de establecer el nivel de calidad requerido. Básicamente, es una fase en el desarrollo de software que consiste en probar las aplicaciones construidas. Ellas se integran a las diferentes fases del ciclo del desarrollo del software y dentro de la propia Ingeniería.

4.3.1. Plan de Prueba

Un plan de pruebas está constituido por un conjunto de pruebas. Cada prueba debe dejar claro: qué tipo de propiedades se quieren probar (corrección, robustez, fiabilidad, amigabilidad); cómo se mide el resultado; especificar en qué consiste la prueba (hasta el último detalle de cómo se ejecuta) y definir cuál es el resultado que se espera. En lo adelante se abordara todo lo relacionado con las pruebas realizadas al sistema.

4.3.1.1. Configuración del entorno de Prueba

La configuración del entorno donde se vayan a ejecutar las diferentes pruebas que se realizan a un software es un aspecto muy importante dentro del proceso de pruebas, pues si no se analizan bien los recursos de software y hardware que necesita el producto que se está construyendo, a la hora de probarlo se prescindirá de los elementos necesarios para la ejecución de un proceso de pruebas exitoso.

Por ello se tuvo en cuenta, a la hora de llevar a cabo todo el proceso de pruebas, algunos requerimientos de hardware y de software que hicieron posible un mejor desarrollo de las mismas, en aras de que se lograran minimizar los errores de la aplicación en desarrollo. Algunos de los requerimientos que se consideraron necesarios para las pruebas son los referenciados a continuación:

Requerimientos de software:

- Una PC con Windows XP o superior.
- Una PC con Linux (Ubuntu).
- Máquina Virtual de Java 1.6.

Requerimientos de hardware:

- PC con Microprocesador Pentium IV a 2.41 GHz o superior.
- Con 160 GB de Disco Duro.
- 512 MB de memoria RAM o superior.

Fecha de Inicio	**Fecha de Fin**	**Actividades**	**Personal Implicado**
14/04/2009	14/04/2009	Aceptación y firma del Plan de Prueba.	Julio Antonio Villaverde Martínez. Ramón Carrasco Velar.
18/04/2009	20/04/2009	Verificación de las condiciones previas para el inicio de las pruebas.	Julio Antonio Villaverde Martínez. Ramón Carrasco Velar.
Primera Iteración del Front-End(alasGRATO).			
23/04/2009	25/04/2009	Ejecución de las pruebas de Caja Negra al caso de uso Gestionar Plug-ins.	Julio Antonio Villaverde Martínez.
27/04/2009	28/04/2009	Ejecución de las pruebas de Caja Negra al caso de uso Eliminar Fichero.	Julio Antonio Villaverde Martínez.
29/04/2009	29/04/2009	Ejecución de las pruebas de Caja Negra al caso de uso Gestionar Preferencias de la Plataforma.	Julio Antonio Villaverde Martínez.
30/05/2009	30/05/2009	Análisis de las no conformidades y las solicitudes de cambio.	Julio Antonio Villaverde Martínez. Ramón Carrasco Velar.
Primera Iteración del plug-in Visualizador Molecular(alasGRATOViewer).			

Fecha de Inicio	Fecha de Fin	Actividades	Personal Implicado
04/05/2009	05/05/2009	Ejecución de las pruebas de Caja Negra al caso de uso Abrir Fichero.	Julio Antonio Villaverde Martínez.
06/05/2009	08/05/2009	Ejecución de las pruebas de Caja Negra al caso de uso Gestionar Fichero.	Julio Antonio Villaverde Martínez.
09/05/2009	09/05/2009	Ejecución de las pruebas de Caja Negra al caso de uso Convertir a 3D.	Julio Antonio Villaverde Martínez.
11/05/2009	12/05/2009	Ejecución de las pruebas de Caja Negra al caso de uso Gestionar Preferencias de Visualización.	Julio Antonio Villaverde Martínez.
14/05/2009	14/05/2009	Análisis de las no conformidades y las solicitudes de cambio.	Julio Antonio Villaverde Martínez. Ramón Carrasco Velar.

En la segunda iteración se realizaron las mismas pruebas planificadas que en la primera iteración.

4.3.2. Diseño de las Pruebas de Unidad (Caja Negra)

4.3.2.1. Diseño de las pruebas del Front-End (alasGRATO)

Caso de uso: Gestionar Plug-ins

Secciones a probar en el caso de uso.

Nombre de la sección	**Escenarios de la sección**	**Descripción de la funcionalidad**	**Flujo Central**
SC1: Adicionar Plug-ins.	EC1.1:Adicionar un plug-in.	El usuario selecciona en el menú Administrar Plug-ins la opción Adicionar Plug-ins y le aparece un diálogo donde se escoge la ruta donde se encuentra(n) el(los) plug-in(s). Selecciona el plug-in y presiona el botón importar se muestra el mensaje "Plug-in importado satisfactoriamente".	Escoge la opción Adicionar Plug-ins presente en el menú Administrar Plug-ins.

Nombre de la sección	**Escenarios de la sección**	**Descripción de la funcionalidad**	**Flujo Central**
SC1: Adicionar Plug-ins.	EC1.2: Adicionar un plug-in que no exista.	El usuario selecciona en el menú Administrar Plug-ins la opción Adicionar Plug-ins y le aparece un diálogo en el cual se escoge la ruta donde se encuentra(n) el(los) plug-in(s). Si no selecciona el plug-in y presiona el botón importar, se muestra el mensaje "Debe seleccionar un plug-in".	Escoge la opción Adicionar Plug-ins presente en el menú Administrar Plug-ins.
SC2: Eliminar Plug-ins.	EC2.1: Eliminar un plug-in.	El usuario selecciona en el menú Administrar Plug-ins la opción Eliminar Plug-ins. Selecciona el plug-in que desea eliminar y luego presionará el botón Eliminar.	Escoge la opción Eliminar Plug-ins presente en el menú Administrar Plug-ins.
SC3: Activar Plug-ins.	EC3.1: Activar plug-in ya activado.	El usuario selecciona en el menú Administrar Plug-ins la opción Activar Plug-ins. Si el plug-in ya fue activado anteriormente el sistema muestra al especialista dicho plug-in.	Escoge la opción Activar Plug-ins presente en el menú Administrar Plug-in.

Nombre de la sección	**Escenarios de la sección**	**Descripción de la funcionalidad**	**Flujo Central**
SC3: Activar Plug-ins.	EC3.2: Activar plug-in no activado	El usuario selecciona en el menú Administrar Plug-ins la opción Activar Plug-ins. Si el plug-in no ha sido activado el sistema lo activa creando el TaskRibbon correspondiente al mismo y crea la ventana que contiene el área de trabajo del plug-in ubicándola en el lado oeste del sistema.	Escoge la opción Activar Plug-ins presente en el menú Administrar Plug-ins.

El diseño de las pruebas por cada sección definida anteriormente se encuentra en el anexo 14.

Caso de uso: Gestionar Preferencias

Secciones a probar en el caso de uso.

Nombre de la sección	**Escenarios de la sección**	**Descripción de la funcionalidad**	**Flujo Central**
SC1: Servidor.	EC1.1: Cambiar preferencias del servidor.	El especialista escoge la opción Servidor el sistema muestra una ventana de diálogo donde el especialista debe especificar el ip del servidor y el puerto y presiona el botón Aceptar. El sistema recoge los datos introducidos por el especialista, los valida y configura con estos, dos variables globales para el uso de todos los plug-ins de la aplicación.	Escoge la opción Servidor presente en el menú Principal del Front-End.
SC1: Servidor.	EC1.2: Cancelar preferencias del servidor.	El especialista escoge la opción Servidor el sistema muestra una ventana de diálogo donde el especialista debe especificar el ip del servidor y el puerto y presiona el botón Cancelar. El sistema cierra la ventana correspondiente.	Escoge la opción Servidor presente en el menú Principal del Front-End.
SC1: Temas de Colores.	EC1.1: Cambiar temas de colores.	El especialista selecciona la opción Temas de Colores .El sistema muestra en el panel lateral los temas disponibles que el especialista puede escoger para la configuración de su aplicación. El sistema realiza el cambio de color de acuerdo a la selección del especialista.	Escoge la opción Tema de Colores presente en el menú Principal del Front-End.

El diseño de las pruebas por cada sección definida anteriormente se encuentra en el anexo 15.

Caso de uso: Eliminar Fichero

Secciones a probar en el caso de uso.

Nombre de la sección	**Escenarios de la sección**	**Descripción de la funcionalidad**	**Flujo Central**
SC1: Eliminar Fichero.	EC1.1: Eliminar fichero existente.	El especialista selecciona el fichero que desea eliminar y presiona clic derecho y selecciona la opción Eliminar Fichero. El sistema elimina el fichero del plug-in en que se encuentre y del árbol de la plataforma.	Escoge la opción Eliminar Fichero presente en el menú Principal del Front-End.

El diseño de las pruebas por cada sección definida anteriormente se encuentra en el anexo 16.

Caso de uso: Manipular Ventanas

A este caso de uso no fue necesario realizarle pruebas de funcionalidad, debido a que se utilizó la librería MyDoggy la cual fue probada por sus desarrolladores.

4.3.2.2. Diseño de las pruebas del Visualizador Molecular (alasGRATOViewer)

Caso de uso: Abrir Fichero

Secciones a probar en el caso de uso.

Nombre de la sección	**Escenarios de la sección**	**Descripción de la funcionalidad**	**Flujo Central**
SC1: Abrir.	EC 1.1: Abrir una o varias moléculas o Proteínas.	El usuario selecciona en la menú Archivo la opción Abrir aparece un diálogo donde se escoge nombre y ruta de la(s) molécula(s) o la(s) proteína(s).	Escoge la opción Abrir presente en el menú Archivo.
SC 2: Abrir Reciente.	EC 2.1: Abrir una fichero reciente.	El usuario selecciona un archivo de la lista de ficheros recientes.	Escoge la opción Abrir Reciente presente en el menú Archivo.
SC 3:Abrir Preferencias de Visualización.	EC 3.1: Abrir un fichero de Preferencia de Visualización.	El usuario selecciona en la menú Archivo la opción Abrir Preferencias de Visualización aparece un diálogo donde se escoge nombre y ruta del fichero.	Escoge la opción Abrir Preferencia de Visualización presente en el menú Archivo.
SC 3:Abrir Preferencias de Visualización.	EC 3.2: Abrir un fichero de Preferencias de Visualización que no exista.	El sistema muestra un mensaje de error “Error en selección de tipo de fichero.”	Escoge la opción Abrir Preferencia de Visualización presente en el menú Archivo.

El diseño de las pruebas por cada sección definida anteriormente se encuentra en el anexo 17.

Caso de uso: Gestionar Fichero

Secciones a probar en el caso de uso.

Nombre de la sección	**Escenarios de la sección**	**Descripción de la funcionalidad**	**Flujo Central**
SC1: Cerrar Fichero.	EC 1.1: Cerrar una moléculas o proteínas.	El usuario da click sobre la cruz que tiene cada tab en donde se visualizan los compuestos, y este queda cerrado.	Escoge la opción Cerrar presente en el menú Archivo.
SC2: Guardar Fichero.	EC 2.1: Guardar.	El usuario selecciona en el menú Archivo la opción Guardar como, y aparece un diálogo donde se entra el nombre, la ruta y la extensión del fichero a guardar.	Escoge la opción Guardar como, presente en el menú Archivo.
SC3: Exportar Fichero	EC 3.1: Exportar a PDF	El usuario selecciona en el menú Archivo la opción Exportar a PDF aparece un diálogo que permite buscar el lugar donde se quiere exportar el fichero.	Escoge la opción Exportar a PDF presente en el menú Archivo.

El diseño de las pruebas por cada sección definida anteriormente se encuentra en el anexo 18.

Caso de uso: Convertir a 3D

Secciones a probar en el caso de uso.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo Central
SC1: Convertir a 3D.	EC1.1: Convertir a 3D.	El usuario selecciona en el menú Molécula la opción Convertir a 3D y se mostrará en el área de trabajo la molécula convertida a 3D.	Escoge la opción Convertir a 3D presente en el menú Molécula.
SC1: Convertir a 3D.	EC1.2: Convertir a 3D cuando no se esta ejecutando el servicio.	El usuario selecciona en el menú Molécula la opción Convertir a 3D y se mostrará el siguiente mensaje "El servicio no esta corriendo".	Escoge la opción Convertir a 3D presente en el menú Molécula.

El diseño de las pruebas por cada sección definida anteriormente se encuentra en el anexo 19.

4.3.2.3. No conformidades Detectadas

Luego de concluir las pruebas realizadas se detectaron las siguientes no conformidades.

Elemento	No	No conformidad	Aspecto correspondiente	Etapa de detección	Signif.	No Signif.
Interfaz.	1	Al activar un plug-in los componentes visuales no eran capaces de cargar las acciones que se le definieron.	Cuando se activa el plug-in alasGRATOViewer en la opción Activar Plug-ins.	Etapa de Diseño y Aplicación de Pruebas al Front-End.	X	

Elemento	**No**	**No conformidad**	**Aspecto correspondiente**	**Etapa de detección**	**Signif.**	**No Signif.**
Interfaz.	2	Los servicios no estaban disponibles.	Cuando se definía la dirección del servidor donde se encontraban los servicios en la opción Servidor.	Etapa de Diseño y Aplicación de Pruebas al Front-End.	X	
Interfaz.	3	No se eliminaba del árbol el fichero.	Cuando se intenta eliminar un fichero del árbol de ficheros cargados.	Etapa de Diseño y Aplicación de Pruebas al Front-End.	X	
Interfaz.	4	No se activa el plug-in alasGRATOViewer.	Cuando se desea activar el plug-in alasGRATOViewer en la opción Activar Plug-ins.	Etapa de Diseño y Aplicación de Pruebas al Front-End.	X	
Interfaz.	5	Existe conflicto en el plug-in alasGRATOViewer cuando se trata de carga una molécula cargada con anterioridad.	Cuando se desea abrir con el plug-in alasGRATOViewer un fichero ya cargado con anterioridad.	Etapa de Diseño y Aplicación de Pruebas al Front-End.	X	

4.4. Conclusiones

- Se logró implementar con éxito el Front-End(alasGRATO) propuesto, el cual servirá como interfaz de la "Plataforma Inteligente para la Predicción de Actividad Biológica en Compuestos Orgánicos" perteneciente al polo de bioinformática de la UCI.
- Se logró implementar el plug-in Visualizador molecular(alasGRATOViewer), el cual fue utilizado para probar las funcionalidades propuestas del Front-End.
- Se definieron y elaboraron los pasos necesarios para implementar un nuevo plug-in al Front-End desarrollado.

Conclusiones

1. Se desarrolló un Front-End, dinámico y multiplataforma, encargado de la portabilidad de plug-ins que garantiza:

   - Unificar la carga de los plug-ins a través de descriptores XML.
   - Eliminar impedacias en la interoperabilidad de los plug-ins.
   - Ganar en soportes de esacalabilidad para el ingreso de futuros plug-ins.
   - Ganar en el manejo de memoria en ejecución controlando todos los componentes visuales desde un mismo marco.
   - Garantiza el principio de conservación, pues en la evolución del Front-End, los plug-ins desarrollados siempre serán compatibles.

2. Se desarrolló una nueva arquitectura informacional para el producto alasGRATO, utilizando los principios de diseños: Percepción del Color, Consistencia, Ley de Fitt y Legibilidad. Donde los usuarios podrán encontrar y utilizar todas las opciones que brinda la plataforma, manteniendo el área de trabajo despejada y con una mejor manipulación de sus ventanas. La cual podrá servir de punto de partida en el desarrollo de la arquitectura informacional para software de escritorio del polo de bioinformática.

3. Se desarrolló una nueva arquitectura de plug-ins para el producto alasGRATO, permitiendo de esta forma adicionar y eliminar módulos en tiempo de ejecución, logrando aumentar o disminuir funcionalidades al sistema. La nueva arquitectura se desarrolló utilizando las librerías Flamingo y MyDoggy.

4. Se validaron las arquitecturas desarrolladas con la implementación del plug-ins Visualizador Molecular (alasGRATOViewer) y a través de las pruebas unitarias de caja negra. Se detectaron solamente 5 no conformidades en la primera iteración, hecho que habla a favor de su validación.

Recomendaciones

1. Continuar la incorporación de nuevas funcionalidades al Front-End, con el objetivo de seguir mejorando la interacción usuario-ordenador

2. Continuar con la implementación de nuevos plug-ins en el proyecto alasGRATO, con el objetivo de ampliar la potencialidad de la plataforma.

3. Contribuir a mejorar el rendimiento del Front-End desarrollado, mediante una mayor exigencia de los componentes JRibbon, logrando de esta forma perfeccionar el diseño visual del Front-End desarrollado.

4. Realizar un test de usabilidad a los usuarios potenciales de la plataforma alasGRATO, con el objetivo de obtener nuevas sugerencias para mejorar -aún más- la interacción usuario-ordenador.

Referencias Bibliográficas

1. Manchón, Eduardo. ¿Qué es la Interacción Persona-Ordenador?, 2003. [2009] Disponible en: http://www.ingenieriasimple.com/usabilidad/QueEsHCI.pdf

2. Ribera Turró, Mireia. Evolución y tendencias en la interacción persona–ordenador. El profesional de la información, 2005, noviembre–diciembre, vol. 15, no. 6, p.414-422.

3. Shackel, Brian. Human–computer interaction: whence and whither. Journal of the American Society for Information Science, 1997, vol. 48, no. 11, p. 970–986.

4. Lorés, Jesús. La interacción persona-ordenador [en línea]. Univertat de Lleida, Mayo 2002 [citado 10 abril 2009]. Disponible en: http://griho.udl.es/ipo/ipo/pdf/01Introd.pdf

5. Argollo Jr., Miguel y M. Olguín, Carlos José. Considerando el Usuario en el Proceso de Desarrollo de Aplicaciones con Interfaces Gráficas. En IntEmpres' 2008. Ponencias del Congreso Internacional de Información - Info 2008, La Habana, 21-25 abril 2008. Al cuidado de IDICT, Cuba: La Habana, 2008. p: 1-11.

6. Marrero Expósito, Carlos. Diseño Gráfico y Comunicación Visual. Tesis doctoral, Universidad de la Laguna, 2006.

7. Hix Deborah. y Rex Hartson, H. Developing User Interfaces: Ensuring Usability Through Product & Process. 1993, John Wiley & Sons Inc.

8. ISO 9241-11. Ergonomic requirements for office work with visual display terminals. ISO, 1998.

9. Ferré Grau, Xavier. Principios Básicos de Usabilidad para Ingenieros Software [en línea]. Universidad Politécnica de Madrid, [citado 12 abril 2009] Disponible en: http://is.ls.fi.upm.es/xavier/papers/ usabilidad.pdf

10. Albano, José Luis. Entendiendo qué es la usabilidad de un producto software [en línea]. Facultad Regional de Rosario, UTN, [citado 12 abril 2009] Disponible en: http://www.utn.edu.ar/download.aspx?idFile=3523

11. Kahn, Paul. WebSite Information Architecture. 1998 .Indianapolis, IN: New Riders.

12. T. Hewett, Thomas. ACM SIGCHI Curricula for Human-Computer Interaction. SIGCHI. Ohio State University, USA., Abril 2008 [citado 15 abril 2009]. Disponible en: http://sigchi.org/cdg/index.html

13. GUI – Webopedia. Graphical User Interface GUI, 2004. [2009]. Disponible en: http://www. webopedia.com/TERM/G/Graphical_User_Interface_GUI.html

14. H. Valdelamar, Eugenio J. Consideraciones para el diseño de interfaces de usuario en sistemas operativos [en línea]. Fundación Arturo Rosenblueth, México, [citado 15 abril 2009]. Disponible en: http://www.scribd.com/doc/2522263/Consideraciones para el diseno de interfaces de usuario en sistemas operativos

15. WCEfA. Using Automotive Graphical User Interface. [2009]. Disponible en: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/apcguide/htm/acc_guide_1.asp

16. Aimacaña Toledo, Carlos. Interfaz de Usuario [en línea]. [2009] Disponible en: http://www. wikiciencia.org/informatica/programacion/iusuario/index.php

17. M. Gómez, Leopoldo Sebastián. Diseño de Interfaces de Usuario Principios, Prototipos y Heurísticas para Evaluación [en línea]. Poder Judicial del Neuquén, Argentina, [citado 18 abril 2009]. Disponible en: http://200.14.84.223/apuntesudp/showDoc.php?id=1209&ramo=ICI2423

18. Tutorial de Java. Introducción al AWT. [2009]. Disponible en: http://www.ac.uma.es/localres/manuals/JavaTut-Froufe/Cap4/awt.html#Intro

19. Pérez Váldes, Y. Réne y C. Bravo, Yanet. Interfaz para la Visualización y Edición de Estructuras Químicas. Tesis de grado, Universidad de las Ciencias Informáticas, 2006

20. Calderón, Marcela y Davis, Emilio. Swing, la solución actual de Java para crear GUIs [en línea]. [citada 18 abril 2009]. Disponible en: http://www.dcc.uchile.cl/~lmateu/CC60H/Trabajos/edavis/swing.html

21. Flamingo. Nuevas versiones de SwingX, Flamingo y Substance. [2009]. Disponible en: http://www.sgoliver.net/blog/?p=451

22. Flamingo. Flamingo 4.0 RC. [2009]. Disponible en:http://www.javahispano.org/contenidos/es/ flamingo_4_0_rc/

23. MyDoggy. Introduction, 2008. [2009]. Disponible en:http://mydoggy.sourceforge.net/

24. MyDoggy. MyDoggy 1.3.1 - My Java Docking Framework, 2007. [2009]. Disponible en: http://www.javalobby.org/java/forums/t102112.html

25. Marrero López, Yadira y Rosales García, Adonis R. Propuesta del diseño arquitectónico de la Plataforma bioGRATO, Universidad de las Ciencias Informáticas, 2007.

26. Open UP. [2009]. Disponible en: http://www.openup.es/

27. Piattini, Mario G. Análisis y diseño detallado de aplicaciones informáticas de gestión. 1ª ed. RA-MA Editorial, Madrid, 1996.

28. Shaw, Mary y Garlan, David. Software Architecture Perspectives on an Emerging Discipline. Prentice Hall, 1996.

Bibliografía

1. Argollo Jr., M., et al., Tecnologia para a Produção de Interfaces Gráficas Portáteis de Alta Qualidade, Memorias del Workshop de Qualidade de Software - IX Simpósio Brasileiro de Engenharia de Software, Recife - PE, 1995.

2. Argollo Jr., M., et al., Un Enfoque para la Obtención y Transferencia de Tecnología para Producción de Interfaces Gráficas, Memorias de INFOCOM'96 - II Congreso Internacional de Informática y Telecomunicaciones, Buenos Aires, Argentina, 1996.

3. Ascui, I.C. Java y la Programación Orientada a Objetos. 2006 [cited 2007 diciembre]; Available from: http://iverclaros.blog.galeon.com/1141775160/.

4. B. Shneiderman. Designing the user interface. Addison Wesley, Reading, Massachusetts, 1998 (Revisado 2009).

5. Browne D., STUDIO: Structured User-Interface Design for Interaction Optimisation. Prentice Hall, 1994.

6. CAMACHO, E., F. CARDESO, and G. NUÑEZ. Arquitecturas de Software. 2004 [cited 2007 noviembre]; Available from: http://prof.usb.ve/lmendoza/Documentos/PS-6116/Guia%20Arquitectura%20v.2.pdf.

7. Carlos Reynoso, N.K. Estilos y Patrones en la Estrategia de Arquitectura de Microsoft. 2004 [cited 2007 noviembre]; Available from: http://www.willydev.net/descargas/prev/Estiloypatron.pdf.

8. CARRILLO, L. V. L. Caracterización de la Prueba de Software: Clasicación y Técnicas, 2005.

9. Clements, P.C. Coming Attractions in Software Architecture. 1996 [cited 2007 noviembre]; Available from: http://www.sei.cmu.edu/pub/documents/96.reports/pdf/tr008.96.pdf.

10. C. H. Lewis y J. Rieman. Task-centered user interface design: A pratical introduction.. 1993.

11. Diseño y Modelación de un Proyecto de Software Utilizando el lenguaje UML. [Consultado el 9/12/2007]. Disponible en: http://www.monografias.com/trabajos28/proyecto-software/proyecto-software.shtml.

12. Heckel P., Software Amigável: Técnicas de Projeto para uma melhor Interface como Usuario. Editora Campus, Rio de Janeiro, 1991.

13. Hix D. y Hartson H., Developing User Interfaces: Ensuring Usability Through Product & Process. John Wiley & Sons, Inc., 1993.

14. Josep Casanovas. De la Acción-Objeto al Objeto-Acción. Revisado 2009. URL http://www.alzado.org/

15. Kay A., User Interface: A Personal View. en The Art of Human-Computer Interface Design. Addison-Wesley Publishing Company, 1990, pp. 191-207.

16. KYBELE., G. D. I. Implementación y Pruebas, 2007. [2007]. Disponible en: http://kybele.escet.urjc.es/Documentos/ISI/Implem_pruebas.pdf.

17. Laurel B., Introduction. en The Art of Human-Computer Interface Design. Addison-Wesley Publishing Company, 1990, pp.xi-xvi.

18. LIMITED, V. P. I. Visual Paradigm for the Unified Modeling Language:VP-UML 6.0 User's Guide, 2007. [Disponible en: http://content.europe.visual-paradigm.com/media/documents/vpuml60ug1/html/ VP-UML_Users_Guide_Part_1_Cover/VP-UML_Users_Guide_Part_1_Cover.html#toc-0

19. MEMBER, A. A. Programación Extrema, 2002. [2007]. Disponible en: http://www.programacionextrema.org/

20. Metodología, Open UP, http://epf.eclipse.org/wikis/openup/index.htm.

21. Molich R. y Nielsen J., Improving Human-Computer Dialogue. Communications of the ACM - Vol.33 No.3, 1990, pp. 338-348.

22. Nielsen J., Usability Engineering. AP Profesional, 1993.

23. Nielsen J., Usability Engineering at a Discount. Em Designing and Using Human- Computer Interfaces and Knowledge Based Systems. Elsevier Science Publishers, 1989, pp. 394-401.

24. Nielsen J., Paybacks from 'Discount' Usability Engineering. IEEE Software - Vol.7 No.3, 1990, pp. 107-108.

25. Potosnak K., Getting the Most Out of Guidelines. IEEE Software - Vol.5 No.1, 1988, pp. 85-86.

26. Pressman, R.S., Ingienería del Software. Un enfoque práctico. V ed. Vol. I. 2005.

27. Reynoso, C.B. Introducción a la Arquitectura de Software. 2004 [cited 2007 noviembre]; Available from: http://www.willydev.net/descargas/prev/IntroArq.pdf.

28. Tsukumo, A.N., et al., Avaliação e Melhoria da Qualidade do Produto de Software Educacional "Escritor", Memorias del Workshop de Qualidade de Software, X Simpósio Brasileiro de Engenharia de Software, São Carlos - SP, 1996.

29. Tsukumo, A.N., et al., The Second Experiment of Application of ISO/IEC 9126 Standards on Quality Evaluation of Brazilian Software Products. Memorias del 6th International Conference on Software Quality (ICSQ), IEEE Computer Society, Ottawa, Canadá, 1996.

30. Wikipedia,Caso de prueba.[En Linea] Enero 2007 [Disponible en: http://es.wikipedia.org/wiki/Caso_de _prueba.

Anexo 1: Descripción de las Clases del Diseño.

Nombre de la Clase: GraphtoolVisual

Atributos	**Tipo**
-MySources	ArrayList<String>
-Recientes	ArrayList<String>
-misExtensiones	ArrayList<String>
-toolWindowManager	MyDoggyToolWindowManager
-treeView	JScrollPane
-tree	JTree
pluginManager	CC_GestorPlugin
-Gestor	CC_GestorMolecular
-Marcos	ArrayList<Marco>
-customClosedIcon	Icon
-customLeafIcon	Icon
-customOpenIcon	Icon
-updater	Timer
-root	DefaultMutableTreeNode
-Extensions	ArrayList<ExtAction>
-pc	PluginConfiguration
-rt	RibbonTask
-pref	Properties
-szCustomThemeFile	String
-THEME_PROP	String
-ServerIp	String
-ServerPort	String
-tm	ThemeActionListener
-nf	NimRODLookAndFeel
-icon1	document_new
-icon2	document_open

Atributos	**Tipo**
-icon3	document_properties
-icon4	document_save_as
-icon5	help_browser
-icon6	mail_forward
-icon7	mail_message_new
-icon8	network_wireless
-icon9	system_log_out
-icon10	text_html
-icon11	x_office_document
-addPlugin	AddPlugin
-removePlugin	RemovePlugin
-pa	PluginAction
-treeUpdater	TreeUpdaterTimer
-t	TreeCreator
-serialVersionUID	Long

Operaciones	**Descripción**
+savePreferences () : void	Este método se encarga de salvar las preferencias generales de la plataforma a un fichero llamado Theme.propierte.
+loadPreferences () : void	Este método se encarga de cargar desde un fichero llamado Theme.propierte las preferencias generales de la plataforma.
+GraphtoolVisual ()	Este método es el constructor del Front-End.
+configureRibbon () : void	Este método se encarga de configurar la ayuda del Front-End.

Operaciones	**Descripción**
+LoadPlugin () : void	Este método se encarga de cargar los plugin que se encuentran en la carpeta ext del Front-End cuando se levanta el mismo.
+SetupTool (name: String, Detached : bolean): Void	Este método se encarga de la configuración del Tool-Window que contiene al árbol.
#configureApplicationMenu() : void	Este método se encarga de la configuración del menú principal del Front-End.
+main (args : String []) : void	Este método se encarga de que el Front-End corra.
+LoadExtensions () : void	Este método inicializa las variables pluginsManager y MisExtensiones para su posterior uso en el método CargarJar().
+LoadJars () : void	Este método le pasa a CargarJar() la dirección donde se encuentran los .jar que se desean cargar.
+CargarJars (string : String) : void	Este método se encarga de cargar los plug-ins que se desean cargar en el Front-End.
+getGestor () : CC_GestorMolecular	Este método se encarga de devolver la clase CC_GestorMolecular para que pueda ser utilizada por los plug-ins que se encuentran en el Front-End.
+getMisExtensions () : ArrayList<String>	Este método se encarga de devolver una lista con los nombres de la las extensiones o plug-ins cargados en el Front-End.
+getMySources () : ArrayList<String>	Este método se encarga de devolver una lista con los nombres de las moléculas que están siendo utilizadas por algún plug-in o para que otro las utilice.
+getPluginManager () : CC_GestorPlugin	Este método se encarga de devolver la clase CC_GestorPlugin para que el Front-End pueda utilizarla.
+getRecientes () : ArrayList<String>	Este método se encarga de devolver una lista de las últimas moléculas cargadas en la plataforma.

Operaciones	**Descripción**
+getToolWindow (key : Object) : ToolWindow	Este método se encarga de devolver un ToolWindow pasándole el nombre del mismo.
+getToolWindowManager () : MyDoggyToolWindowManager	Este método se encarga de devolver el MyDoggyToolWindowsManager que es el encargado de controlar los ToolWindows que se encuentran activos en el Front-End.
+getTree () : JTree	Este método se encarga de devolver el tree para que otros plug-ins puedan utilizarlo en algún momento.
+getMarcos () : ArrayList<Marco>	Este método se encarga de devolver una lista que contiene los marcos creados para el uso de ellos en el Front-End.
+UpdateTree () : void	Este método se encarga de actualizar el árbol del Front-End cada vez que ocurra un cambio en el mismo.
+getExtensions () : ArrayList<ExtAction>	Este método se encarga de devolver una lista de las extensiones de los ficheros que utilizan los plug-ins en su trabajo.
#DeleteMols () : void	Este método se encarga de borrar todas aquellas moléculas que no se encuentran en el arreglo Recientes.
+applyTheme () : void	Este método se encarga de aplicar los temas de colores para el Front-End.

Nombre de la Clase: PluginAction

Atributos	**Tipo**
-myMain	MainInterface
-pc	PluginConfiguration
-rt	RibbonTask
-jar	JarResources

Atributos	**Tipo**
-xmlParser	XMLParser
-clc	ClassLoadConfiguration

Operaciones	**Descripción**
+PluginAction (component : MainInterface)	Este método es el constructor de la clase.
+actionPerformed (e : ActionEvent) ; void	Este método es el encargado de parsear el fichero XML y de construir a partir de este el plug-in que se desea incorporar a la plataforma, además de incluir el ToolWindow o área de trabajo del mismo, además del área de resultados del plug-in si este la tiene.
#setupTool (name : String) : void	Este método es el encargado de configurar el ToolWindow de el plug-in que se piensa incorporar al FrontEnd.

Nombre de la Clase: PluginConfiguration

Atributos	**Tipo**
-bandsConf	List<BandConfiguration>
-expandBandConf	ExpandBandActionConfiguration
-name	String
-action	String
-ext	String
-descr	String

Operaciones	**Descripción**
+getDescr () : string	Este método es el encargado de devolver la descripción del plug-in.
+setDescr (descr : String) : void	Este método es el encargado de cambiar la descripción del plug-in.
+PluginConfiguration ()	Este método es el constructor por defecto de la clase.
+PluginConfiguration (bandsConf : List<BandConfiguration>)	Este método es el constructor de la clase pasándole la configuración de una lista de bandas.
+PluginConfiguration (bandsConf : BandConfiguration, name : String)	Este método es el constructor de la clase pasándole la configuración de una banda y un nombre.
+addBandConf (bandConf : BandConfiguration) : void	Este método es el encargado de agregar una nueva configuración de una banda.
+getBandsConf () : List<BandConfiguration>	Este método es el encargado de devolver una lista de bandas.
+getExpandBandConf () : ExpandBandActionConfiguration	Este método es el encargado de devolver una expansión de una banda.
+setExpandBandConf () : ExpandBandActionConfiguration : void	Este método es el encargado de cambiar la expansión de una banda.
+getName () : string	Este método es el encargado de devolver el nombre del plug-in.
+setName (name : String) : void	Este método es el encargado de cambiar el nombre del plug-in.
+setPluginsConf (pluginsConf : List<BandConfiguration>) : viod	Este método es el encargado de cambiar la configuración del plug-in.
+setBandsConf (bandsConf : List<BandConfiguration>) : void	Este método es el encargado de cambiar la lista de configuración de las bandas.
+getAction () :string	Este método es el encargado de devolver la acción que realiza el clic derecho del árbol que está en el Front-End.

Operaciones	**Descripción**
+setAction (action : String) : void	Este método es el encargado de cambiar la acción que realiza el clic derecho del árbol.
+getExt () :string	Este método es el encargado de devolver la extensión con que trabaja un plug-in determinado.
+setExt (ext : string) : void	Este método es el encargado de cambiar la extensión con que trabaja un plug-in.

Nombre de la Clase: XMLParser

Atributos	**Tipo**
pluginConf	PluginConfiguration
button	ButtonParser
panel	PanelParser
gallery	GalleryParser
icon	IconParser
bandConf	BandConfiguration
expandAction	ExpandBandActionConfiguration

Operaciones	**Descripción**
+parser (fileName : String) : PluginConfiguration	Este método es el encargado de parsear la configuración de un plug-in a partir de una dirección.
-bandsParser (plugin : OMElement, pluginConf : PluginConfiguration) : void	Este método es el encargado de parsear una banda a partir de un configuración de un plug-in.
-expandActionListener (band : OMElement, bandConf: BandConfiguration) : void	Este método es el encargado de parsear una expansión de una banda.

Nombre de la Clase: MainInterfaces

Operaciones	**Descripción**
+getMySources():ArrayList<String>	Este método se encarga de devolver una lista con los nombres de las moléculas que están siendo utilizadas por algún plug-in o para que otro las utilice.
+getRecientes():ArrayList<String>	Este método se encarga de devolver una lista de las últimas moléculas cargadas en la plataforma.
+getRibbon():JRibbon	Este método es el encargado de devolver el JRibbon del Front-End para que los plug-ins puedan utilizarlo.
+getToolWindow(Object key): ToolWindow	Este método es el encargado de devolver un ToolWindow según el nombre que se le pase para su uso por un plug-in determinado.
+getToolWindowManager(): MyDoggyToolWindowManager	Este método se encarga de devolver el MyDoggyToolWindowsManager que es el encargado de controlar los ToolWindows que se encuentran activos en el Front-End.
+getTree():JTree	Este método se encarga de devolver el tree del Front-End.
+getPluginManager(): CC_GestorPlugin	Este método se encarga de devolver la clase CC_GestorPlugin para su uso posterior por algún plug-in.
+getMisExtensiones(): ArrayList<String>	Este método se encarga de devolver una lista con los nombres de la las extensiones o plug-ins cargados en el Front-End.
+getGestor(): CC_GestorMolecular	Este método se encarga de devolver la clase CC_GestorMolecular para su uso posterior por algún plug-in.

Operaciones	**Descripción**
+getMarcos(): ArrayList<Marco>	Este método se encarga de devolver una lista que contiene los marcos creados para el uso de ellos en el Front-End.
+UpdateTree():void	Este método se encarga de actualizar el árbol del Front-End cada vez que ocurra un cambio en el mismo.
+LoadExtensions():void	Este método es el encargado de inicializar las variables pluginsManager y MisExtensiones para su posterior uso en CargarJar().
+getExtensions(): ArrayList<ExtAction>	Este método es el encargado de devolver una lista de las extensiones de los ficheros que utilizan los plug-ins en su trabajo.

Nombre de la Clase: ClassLoadConfiguration

Atributos	**Tipo**
-parent	MainInterface
-CompoCreate	ComponentCreator
-buttonPanel	JPanel
-sizeStrip	JCommandButtonStrip
-band	JRibbonBand
-button	JCommandButton
-myBandPanel	JPanel

Operaciones	**Descripción**
+ClassLoadConfiguration (main : MainInterface)	Este método es el constructor por defecto de la clase.

Operaciones	**Descripción**
-addButtons (bandPanel : JPanel, buttonsList : List<ButtonConfiguration>) : void	Este método es el encargado de añadir botones a una lista de botones que se encuentran en un panel de una banda.
-addChecks (bandPanel : JPanel, listaCheckBoxes : List<CheckBoxConfiguration>) : void	Este método es el encargado de añadir checkBox a una lista de chechBox que se encuentran en un panel de una banda.
-addCombos (bandPanel : JPanel, comboList : List<ComboBoxConfiguration>) : void	Este método es el encargado de añadir comboBox a una lista de comboBox que se encuentran en un panel de una banda.
-addSpinners (bandPanel : JPanel, spinnersList : List<SpinnerConfiguration>) : void	Este método es el encargado de añadir spinners a una lista de comboBox que se encuentran en un panel de una banda.
-addStrips (bandPanel : JPanel, stripsList : List<ButtonStripConfiguration>) : void	Este método es el encargado de añadir buttonStrip a una lista de buttonStrip que se encuentran en un panel de una banda.
-createBand(bandConf : BandConfiguration) : JRibbonBand	Este método es el encargado de crear una banda.
-createButtons (bandConf : BandConfiguration, band : JRibbonBand) : void	Este método es el encargado de crear un botón en una banda.
-createGalleries (bandConf : BandConfiguration, band : JRibbonBand) : void	Este método es el encargado de crear una galería en una banda.
-createPanels (bandConf : BandConfiguration, band : JRibbonBand) : void	Este método es el encargado de crear paneles en una banda.
-createPanels (list : List<PanelConfiguration>, band : JRibbon, bandPanel : JPanel) : void	Este método es el encargado de crear un conjunto de paneles en una banda.

Operaciones	**Descripción**
+createPlugin (pluginConf : PluginConfiguration) : RibbonTask	Este método es el encargado de crear un plugin como RibbonTask.

Nombre de la Clase: BandConfiguration

Atributos	**Tipo**
-buttonsConf	List<ButtonConfiguration>
-expandBandConf	ExpandBandActionConfiguration
-galleriesConf	List<GallerybuttonConfiguration>
-iconConf	IconConfiguration
-name	String
panelConf	List<PanelConfiguration>

Operaciones	**Descripción**
+BandConfiguration()	Este método es el constructor por defecto de la clase.
+BandConfiguration(name : String)	Este método es el constructor de la clase pasándole un nombre a la banda.
+BandConfiguration(name : String, buttonsConf : List< ButtonConfiguration>)	Este método es el constructor de la clase pasándole un nombre a la banda y una lista de botones.
+BandConfiguration(name : String, buttonsConf : List< ButtonConfiguration>, expandBandCond : ExpandBandActionConfiguration)	Este método es el constructor de la clase pasándole un nombre a la banda, una lista de botones y una expansión de la banda.
+addButtonConf(buttonConf : ButtonConfiguration) : void	Este método es el encargado de añadir la configuración de un botón.

Operaciones	**Descripción**
+addPanelConf(panelConf:PanelConfigurator) : void	Este método es el encargado de añadir la configuración de un panel.
+addGalleriesConf(galleryConf : GalleryButtonConfiguration) : void	Este método es el encargado de añadir la configuración de una galería.
+getButtonsConf() : List< ButtonConfiguration>	Este método es el encargado de devolver una lista de botones.
+getExpandBandCond() : ExpandBandActionConfiguration	Este método es el encargado de devolver la expansión de la banda.
+getGalleriesConf() : List<GalleryButtonConfiguration>	Este método es el encargado de devolver una lista de galerías.
+getIconConf():IconConfiguration	Este método es el encargado de de devolver la configuración de un icono.
+getName() : String	Este método es el encargado de devolver el nombre de la banda.
+getPanelConf() : List<PanelConfigurator>	Este método es el encargado de devolver una lista de configuraciones de paneles.
+setButtonsConf(buttonsConf : List<ButtonConfiguration>) : void	Este método es el encargado de cambiar la lista de configuraciones de botones.
+setGalleryConf(galleryConf : List<GalleryButtonConfiguration>) : void	Este método es el encargado de cambiar la lista de configuraciones de galerías.
+setExpandBandCond(expandBandCond : ExpandBandActionConfiguration) : void	Este método es el encargado de cambiar la expansión de la banda.
+setIconConf(iconConf : IconConfiguration) : void	Este método es el encargado de cambiar la configuración de un icono.
+setName(name : String) : void	Este método es el encargado de cambiar el nombre de una banda.
+setPanelConf(panelConf : List<PanelConfigurator>) : void	Este método es el encargado de cambiar la configuración de una lista de paneles.

Nombre de la Clase: IconParser

Atributos	**Tipo**
-iconConf	IconConfiguration

Operaciones	**Descripción**
~iconParser(component : OMElement) : IconConfiguration	Este método es el encargado de crear una configuración de un icono.

Nombre de la Clase: ButtonParser

Atributos	**Tipo**
-icon	IconParser
-buttonConf	ButtonConfiguration
-list	LinkedList<ButtonConfiguration>

Operaciones	**Descripción**
~buttonParser(band OMElement, bandConf : BandConfiguration, rootPath : String) : List< ButtonConfiguration>	Este método es el encargado de crear un botón en una banda determinada a partir de una dirección que se le pasa al método.

Nombre de la Clase: PanelParser

Atributos	**Tipo**
-myPanels	LinkedList<PanelConfigurator>
-panelConf	PanelConfigurator

Atributos	**Tipo**
-buttonC	ButtonConfiguration
-icon	IconParser
-bp	ButtonParser
-spinnB	SpinnerConfiguration
-comboConfig	ComboBoxConfiguration
-buttonSConf	ButtonStripConfigurator
-checkB	CheckBoxConfiguration

Operaciones	**Descripción**
~panelParser(band OMElement, bandConf : BandConfiguration, parentPath : String, parentPanel : PanelConfigurator) : List< PanelConfigurator>	Este método es el encargado de de crear un panel con todos los componentes que tienes un panel en el, como son botones, spinners, comboBox y otros.
+panelParser(band OMElement, bandConf : BandConfiguration, string : String) : void	Este método vuelve a llamar al método anterior cuando existe un panel dentro de otro.

Nombre de la Clase: GalleryParser

Atributos	**Tipo**
-gbc	GalleryButtonConfiguration
-gl	GalleryList
-bp	ButtonParser

Operaciones	**Descripción**
+galleryParser(band OMElement, bandConf : BandConfiguration, parentPath : String) : void	Este método es el encargado de parsear una galería en una banda.
-parseGalleryLists(band OMElement, bandConf : BandConfiguration, gbc : GalleryButtonConfiguration, parentPath : String) : void	Este método es el encargado de parsear los componentes internos de una galería como sería por ejemplo una lista de botones, con todas sus funcionalidades.

Nombre de la Clase: ExpandBandActionConfiguration

Atributos	**Tipo**
-expandBand	String

Operaciones	**Descripción**
+getExpandBand() : String	Este método es el encargado de devolver la expansión de una banda.
+setExpandBand(expandBand : String) : void	Este método es el encargado de cambiar la expansión de una banda.

Nombre de la Clase: IconConfiguration

Atributos	**Tipo**
-height	Integer
-iconSource	String

-width	Integer

Operaciones	**Descripción**
+IconConfiguration()	Este método es el constructor de la clase por defecto.
+IconConfiguration(height : Integer, width : Integer, iconSource : String)	Este método es el constructor de la clase pasándole el altura, ancho y la fuente del icono.
+getHeight() : Integer	Este método es el encargado de devolver la altura del icono.
+getIconSource() : String	Este método es el encargado de devolver la dirección donde se encuentra el icono.
+getWidth() : Integer	Este método es el encargado de devolver el ancho del icono.
+setHeight(height : Integer) : void	Este método es el encargado de cambiar la altura del icono.
+setIconSource(iconSource : String) : void	Este método es el encargado de cambiar la dirección de un icono.
+setWidth(width : Integer) : void	Este método es el encargado de cambiar el ancho de un icono.

Nombre de la Clase: GalleryButtonConfiguration

Atributos	**Tipo**
~buttonLists	List< GalleryList>
~name	String
-columns	int
-rows	int

Operaciones	**Descripción**
+GalleryButtonConfiguration()	Este método es el constructor de la clase.
+add(arg0 : GalleryList) : boolean	Este método es el encargado de añadir un lista de galería.
+get(arg0 : int) : GalleryList	Este método es el encargado de devolver un botón de una galería.
+getButtonLists() : List< GalleryList>	Este método es el encargado de devolver una lista de botones que están en una galería.
+getName() : String	Este método es el encargado de devolver el nombre de una galería.
+isEmpty() : boolean	Este método es el encargado de devolver si una galería está vacía o no.
+setButtonLists(buttonLists : List< GalleryList>) : void	Este método es el encargado de cambiar una lista de botones en una galería.
+setName(name : String) : void	Este método es el encargado de cambiar el nombre de una galería.
+size() : int	Este método es el encargado de devolver el tamaño de una galería.
+getColumnCounts() : int	Este método es el encargado de devolver cuantas columnas va a tener una galería.
+setColumns(columns : int) : void	Este método es el encargado de cambiar el tamaño de las columnas de una galería.
+getRowCounts() : int	Este método es el encargado de devolver cuantas filas tiene una galería.
+setRows(rows : int) : void	Este método es el encargado de cambiar cuantas filas tiene una galería.

Nombre de la Clase: PanelConfigurator

Atributos	**Tipo**
-buttonsList	ButtonConfiguration>
-checkBoxeslist	CheckBoxConfiguration>
-comboList	ComboBoxConfiguration>
-panelList	PanelConfigurator>
-name	String
-orientation	String
-spinnersList	SpinnerConfiguration>
-stripList	ButtonStripConfigurator>

Operaciones	**Descripción**
+PanelConfigurator()	Este método es el constructor por defecto de la clase.
+PanelConfigurator(buttonsList List<ButtonConfiguration>, checkBoxeslist : List<CheckBoxConfiguration>, spinnersList : List<SpinnerConfiguration>)	Este método es un constructor de la clase pasándole una lista de botones, una lista de checkbox, y una lista de spinners.
+PanelConfigurator(buttonsList List<ButtonConfiguration>, spinnersList : List<SpinnerConfiguration>)	Este método es un constructor de la clase pasándole una lista de botones y una lista de spinner.
+PanelConfigurator(mylistaCheckBoxes : List< CheckBoxConfiguration>)	Este método es un constructor de la clase pasándole una lista de checkBox.
+PanelConfigurator(myname : String)	Este método es un constructor de la clase pasándole un nombre.
+getButtonsList() : List<ButtonConfiguration>	Este método es el encargado de devolver una lista de botones.

Operaciones	**Descripción**
+getComboList() : List< ComboBoxConfiguration>	Este método es el encargado de devolver una lista de comboBox.
+getName() : String	Este método es el encargado de devolver el nombre del panel.
+getOrientation() : String	Este método es el encargado de devolver la orientación del panel.
+getSpinnersList() : List<SpinnerConfiguration>	Este método es el encargado de devolver una lista de spinner.
+getStripList() : List<ButtonStripConfigurator>	Este método es el encargado de devolver una lista de buttonStrip.
+ListaCheckBoxes() : List<CheckBoxConfiguration>	Este método es el encargado de devolver una lista de checkbox.
+setButtonsList(buttonsList : List<ButtonConfiguration>) : void	Este método es el encargado de cambiar una lista de botones en un panel.
+setComboList(comboList : List<ComboBoxConfiguration>) : void	Este método es el encargado de cambiar una lista de comboBox en un panel.
+setListaCheckBoxes(mylistaCheckBoxes : List<CheckBoxConfiguration>) : void	Este método es el encargado de cambiar una lista de checkbox en un panel.
+setName(name : String) : void	Este método es el encargado de cambiar el nombre de un panel.
+setOrientation(orient : String) : void	Este método es el encargado de cambiar la orientación de un panel.
+setSpinnersList(spinnersList : List<SpinnerConfiguration>) : void	Este método es el encargado de cambiar una lista de spinner en una panel.
+setStripList(stripList : List< ButtonStripConfigurator>) : void	Este método es el encargado de cambiar una lista de ButtonStrip en un panel.
+getPanelList() : List<PanelConfigurator>	Este método es el encargado de devolver la lista de paneles.

Operaciones	Descripción
+setPanelList() : List<PanelConfigurator>	Este método es el encargado de cambiar la lista de paneles.

Nombre de la Clase: ButtonConfiguration

Atributos	Tipo
-action	String
-classType	String
-iconConf	IconConfiguration
-mnemonic	String
-name	String
-popupClassType	String
-priority	RibbonElementPriority
-tooltipText	String

Operaciones	Descripción
+getAction() : String	Este método es el encargado de devolver la acción que realiza el botón.
+getClassType() : String	Este método es el encargado de devolver que tipo de botón es.
+getIconConf() : IconConfiguration	Este método es el encargado de devolver la configuración del icono.
+getMnemonic() : String	Este método es el encargado de devolver la tecla de acceso rápido al botón.
+getName() : String	Este método es el encargado de devolver el nombre del botón.

Operaciones	**Descripción**
+getPopupClassType() : String	Este método es el encargado de devolver en que clase se encuentra la acción de un popup.
+getPriority() : RibbonElementPriority	Este método es el encargado de devolver la prioridad que tiene el botón a la hora de ponerlo en una banda.
+getTooltipText() : String	Este método es el encargado de devolver el Tooltip de un botón.
+setAction(action : String) : void	Este método es el encargado de cambiar la acción de un botón.
+setClassType(classType : String) : void	Este método es el encargado de cambiar el tipo de un botón.
+setIconConf(iconConf : IconConfiguration) : void	Este método es el encargado de devolver el icono de un botón.
+setMnemonic(mnemonic : String) : void	Este método es el encargado de devolver la tecla de acceso rápido a un botón.
+setName(name : String) : void	Este método es el encargado de cambiar el nombre de un botón.
+setPopupClassType(popupClassType : String) : void	Este método es el encargado de cambiar el popup de un botón.
+setPriority(priority RibbonElementPriority) : void	Este metodo es el encargado de cambiar la prioridad de un botón.
+setTooltipText(tooltipText : String) : void	Este método es el encargado de cambiar el Tooltip de un botón.

Nombre de la Clase: SpinnerConfiguration

Atributos	**Tipo**
~changeListener	String
~spinnerModel	String

Atributos	**Tipo**
˜tooltipText	String

Operaciones	**Descripción**
+getChangeListener() : String	Este método es el encargado de devolver la acción que realiza el spinner.
+getSpinnerModel() : String	Este método es el encargado de devolver el modelo del spinner.
+getTooltipText() : String	Este método es el encargado de devolver el Tooltip del spinner.
+setChangeListener(changeListener : String) : void	Este método es el encargado de cambiar la acción del spinner.
+setSpinnerModel(spinnerModel : String) : void	Este método es el encargado de cambiar el modelo del spinner.
+setTooltipText(tooltipText : String) : void	Este método es el encargado de cambiar el ToolTip del spinner.

Nombre de la Clase: ComboBoxConfiguration

Atributos	**Tipo**
-changeListener	String
-height	int
-name	String
-Objects	List<String>
-tooltipText	String
-width	int

Operaciones	**Descripción**
+ComboBoxConfiguration()	Este método es el constructor por defecto de la clase.
+add(arg0 : String) : boolean	Este método es el encargado de añadir un objeto al combox.
+get(arg0 : int) : String	Este método es el encargado de devolver un objeto del combox.
+getChangeListener() : String	Este método es el encargado de devolver la acción de un objeto en el combox.
+getHeight() : int	Este método es el encargado de devolver el alto de un combox.
+getName() : String	Este método es el encargado de devolver el nombre de un combox.
+getObjects():List<String>	Este método es el encargado de devolver un objeto que esta en un combox.
+getTooltipText() : String	Este método es el encargado de devolver el Tooltip de un combox.
+getWidth() : int	Este método es el encargado de devolver el ancho de un combox.
+isEmpty() : boolean	Este método es el encargado de devolver si un combox esta vacio o no.
+setChangeListener(changeListener : String) : void	Este método es el encargado de cambiar la acción que realiza un objeto en un combox.
+setHeight(height : int) : void	Este método es el encargado de cambiar el alto de un combox.
+setName(nameComponent : String) : void	Este método es el encargado de cambiar el nombre de un combox.
+setObjects(objects:List<String>) : void	Este método es el encargado de cambiar un objeto de combox.
+setTooltipText(tooltipText : String) : void	Este método es el encargado de cambiar el Tooltip de un combox.

Operaciones	**Descripción**
+setWidth(width : int) : void	Este método es el encargado de cambiar el ancho de un combox.
+size() : int	Este método es el encargado de devolver el tamaño de un combox en cuanto a objetos se habla.

Nombre de la Clase: ButtonStripConfigurator

Atributos	**Tipo**
-name	String
-buttonList	ButtonConfiguration>

Operaciones	**Descripción**
+ButtonStripConfigurator()	Este es el constructor por defecto de la clase.
+ButtonStripConfigurator(name : String, buttonList : List<ButtonConfiguration>)	Este método es un constructor de la clase pasandole un nombre, y una lista de botones.
+getName() : String	Este método es el encargado de devolver el nombre del buttonStrip.
+setName(name : String) : void	Este método es el encargado de cambiar el nombre del buttonStrip.
+getButtonList() : List<ButtonConfiguration>	Este método es el encargado de devolver la lista de botones que componen al buttonStrip.
+setButtonList(buttonList : List<ButtonConfiguration>) : void	Este método es el encargado de cambiar la lista de botones que componen al buttonStrip.

Nombre de la Clase: CheckBoxConfiguration

Atributos	**Tipo**
~action	String
~selected	boolean
~text	String
~tooltipText	String

Operaciones	**Descripción**
+getAction() : String	Este método es el encargado de devolver la acción que realiza el checkBox.
+getText() : String	Este método es el encargado de devolver el texto del checkBox.
+getTooltipText() : String	Este método es el encargado de devolver el Tooltip del checkBox.
+isSelected() : boolean	Este método es el encargado de devolver si un checkBox esta seleccionado o no.
+setAction(action : String) : void	Este método es el encargado de cambiar la acción del checkBox.
+setSelected(selected : boolean) : void	Este método es el encargado de cambiar si un checkBox esta seleccionado o no.
+setText(text : String) : void	Este método es el encargado de cambiar el texto del checkBox.
+setTooltipText(tooltipText : String) : void	Este método es el encargado de cambiar el Tooltip del checkBox.

Nombre de la Clase: RemovePlugin

Atributos	**Tipo**
-component	MainInterface

Atributos	**Tipo**
-p	RemovePluginVisual

Operaciones	**Descripción**
+RemovePlugin(component JFrame)	Este método es el constructor de la clase por defecto
+actionPerformed(e:ActionEvent) : void	Este método es el encargado de levantar a RemovePluginVisual.

Nombre de la Clase: AddPlugin

Atributos	**Tipo**
-component	plugin.MainInterface
-p	AddPluginVisual

Operaciones	**Descripción**
+AddPlugin(component :JFrame)	Este método es el constructor por defecto de la clase.
+actionPerformed(e : ActionEvent) : void	Este método es el encargado de llamar a la clase AddPluginVisual.

Nombre de la Clase: CC_FileManager

Operaciones	**Descripción**
+copyDirectory(srcDir : File, dstDir : File) : void	Este método es el encargado de copiar un de un directorio fuente a un directorio destino.

Operaciones	**Descripción**
+copy(src File, dst : File) : void	Este método es el encargado de copiar de un fichero fuente a un fichero destino.

Nombre de la Clase: RemovePluginVisual

Atributos	**Tipo**
-textField_6	JTextField
-textField_5	jJTextField
-textField_4	JTextField
-textField_3	JTextField
-textField_2	jJTextField
-textField_1	JTextField
-importarButton	JButton
-list	JList
-PLUGINS_KEY	String
-PLUGIN_NAME	String
-PLUGIN_AUTHOR	String
-PLUGIN_DESCRIPTION	String
-PLUGIN_CLASSPATH	String
-actualPlugs	ArrayList<PluginDesc>
-Component	MainInterface
-p	PluginDesc
-currentFile	JarFile
-attributes	Attributes
-MisExtensiones	ArrayList<String>
-serialVersionUID	long

Operaciones	**Descripción**
+RemovePluginVisual(component : MainInterface)	Este método es el constructor por defecto de la clase.
~Remove() : void	Este método es el encargado de remover un plug-in visualmente del Front-End.
-checkPluginDependencies(library : String, selectedRow : int) : boolean	Este método es el encargado de devolver si un plug-in existe o no.
~DelPlugin() : void	Este método es el encargado de remover físicamente el plug-in del Front-End.
~ActualizaDatos(p : PluginDesc) : void	Este método es el encargado de actualizar la lista de plug-in que existen para eliminar.
~UpdateLists(folder : String) : void	Este método es el encargado de actualizar la lista cuando un plug-in es removido.
-ChequeaExistencia(string : String) : boolean	Este método es el encargado de devolver si un plugin existe en una lista de plug-in.
-InitTable() : void	Este método es el encargado de actualizar los campos del visual con los datos obtenidos del plug-in.
~SelectFolder() : String	Este método se encarga de devolver el folder donde se encuentran los plug-in a eliminar.

Nombre de la Clase: PluginDesc

Atributos	**Tipo**
~Name	String
~Author	String
~Location	String
~Description	String
~Version	String
~ClassPath	String

Operaciones	**Descripción**
+PluginDesc()	Este método es el constructor por defecto de la clase.
+PluginDesc(name : String, author : String, location : String, description : String, version : String, classPath : String)	Este método es el constructor de la clase pasándole un autor, una locación, una descripción, una versión, y un camino donde se encuentran las librerías.
+getName() : String	Este método es el encargado de devolver el nombre del plug-in.
+setName(name : String) : void	Este método es el encargado de cambiar el nombre del plug-in.
+getAuthor() : String	Este método es el encargado de devolver el autor del plug-in.
+setAuthor(author : String) : void	Este método es el encargado de cambiar el autor del plug-in.
+getLocation() : String	Este método es el encargado de devolver la locación del plug-in.
+setLocation(location : String) : void	Este método es el encargado de cambiar la locación del plug-in.
+getDescription() : String	Este método es el encargado de devolver la descripción del plug-in.
+setDescription(description : String) : void	Este método es el encargado de cambiar la descripción del plug-in.
+getVersion() : String	Este método es el encargado de devolver la versión del plug-in.
+setVersion(version : String) : void	Este método es el encargado de cambiar la versión del plug-in.
+getClassPath() : String	Este método es el encargado de devolver el camino donde están las librerías del plug-in.
+setClassPath(classPath : String) : void	Este método es el encargado de cambiar el camino de las librerías del plug-in.

Operaciones	**Descripción**
+equals(obj : Object) : boolean	Este método es el encargado de instanciar una nueva descripción de un plug-in.

Nombre de la Clase: AddPluginVisual

Atributos	**Tipo**
-textField_6	JTextField
-textField_5	JTextField
-textField_4	JTextField
-textField_3	JTextField
-textField_2	JTextField
-textField_1	JTextField
-importarButton	JButton
-list	JList
-PLUGINS_KEY	String
-PLUGIN_NAME	String
-PLUGIN_AUTHOR	String
-PLUGIN_DESCRIPTION	String
-PLUGIN_CLASSPATH	String
-actualPlugs	ArrayList< PluginDesc>
-Component	MainInterface
-p	PluginDesc
-currentFile	jJarFile
-attributes	Attributes
-MisExtensiones	ArrayList<String>
-serialVersionUID	long

Operaciones	**Descripción**
+main(args : String []) : void	Este método es el encargado de que la clase AddPlugInVisual corra.
+AddPluginVisual(component : MainInterface)	Este método es el constructor por defecto de la clase.
~Importar() : void	Este método es el encargado de importar el plugin deseado hacia el Front-End.
~ActualizaDatos(p : PluginDesc) : void	Este método es el encargado de actualizar los datos a partir de la descripción del plug-in.
~UpdateLists(folder : String) : void	Este método es el encargado de actualizar la lista de plug-in a importar.
-ChequeaExistencia(string : String) : boolean	Este método es el encargado de chequear la existencia de un plug-in dado.
-InitTable() : void	Este método es el encargado de actualizar la tabla del visual.
~SelectFolder() : String	Este método es el encargado de seleccionar el folder donde se encuentran los plug-ins.

Nombre de la Clase: ServerListener

Atributos	**Tipo**
-serialVersionUID	long
-textField	TextFieldValidador
-textField_1	TextFieldValidador
-cancelButton	jFlatButtons
-okButton	jFlatButtons

Operaciones	**Descripción**
+actionPerformed(arg0:ActionEvent) : void	Este método es el encargado de recoger los datos que introducirá el usuario para la ubicación del servidor.

Nombre de la Clase: TreePopup

Atributos	**Tipo**
~component	MainInterface
~remove	TreeRemover

Operaciones	**Descripción**
+TreePopup(mycomp : MainInterface)	Este método es el constructor por defecto de la clase.
-myPopupEvent(e : MouseEvent) : void	Este método es el encargado de construir el popup cuando se da clic derecho sobre el árbol.
+mousePressed(e : MouseEvent) : void	Este método es el encargado de garantizar que el tree no haga nada cuando se presione sobre él.
+mouseReleased(e : MouseEvent) : void	Este método es el encargado de realizar la acción que se definió en el XML para el tree con respecto a un plug-in.

Nombre de la Clase: TreeRemover

Atributos	**Tipo**
~main	MainInterface

Operaciones	**Descripción**
+TreeRemover(main2 : plugin.MainInterface)	Este método es el constructor de la clase por defecto.
+actionPerformed(e:ActionEvent) : void	Este método es el encargado de realizar la acción de eliminar del árbol del Front-End.

Nombre de la Clase: Marco

Atributos	**Tipo**
-marco	JPanel
-nombre	String

Operaciones	**Descripción**
+Marco(marco : JPanel, nombre : String)	Este método es el constructor de la clase pasándole un JPanel y un nombre.
+Marco()	Este método es el constructor por defecto de la clase.
+getMarco():JPanel	Este método es el encargado de devolver un marco que se encuentre activo en la plataforma.
+setMarco(marco : JPanel) : void	Este método es el encargado de cambiar un marco en el Front-End.
+getNombre() : String	Este método es el encargado de devolver el nombre de un marco.
+setNombre(nombre : String) : void	Este método es el encargado de cambiar el nombre de un marco.

Nombre de la Clase: ThemeActionListener

Atributos	**Tipo**
-iThemeID	int
-component	MainInterface

Operaciones	**Descripción**
+ThemeActionListener(component : MainInterface)	Este método es el constructor por defecto de la plataforma.
+actionPerformed(arg0 : ActionEvent) : void	Este método es el que realiza la acción de cambiar el ambiente del Front-End.
-applyTheme() : void	Este método es el que aplica el cambio de ambiente del Front-End.

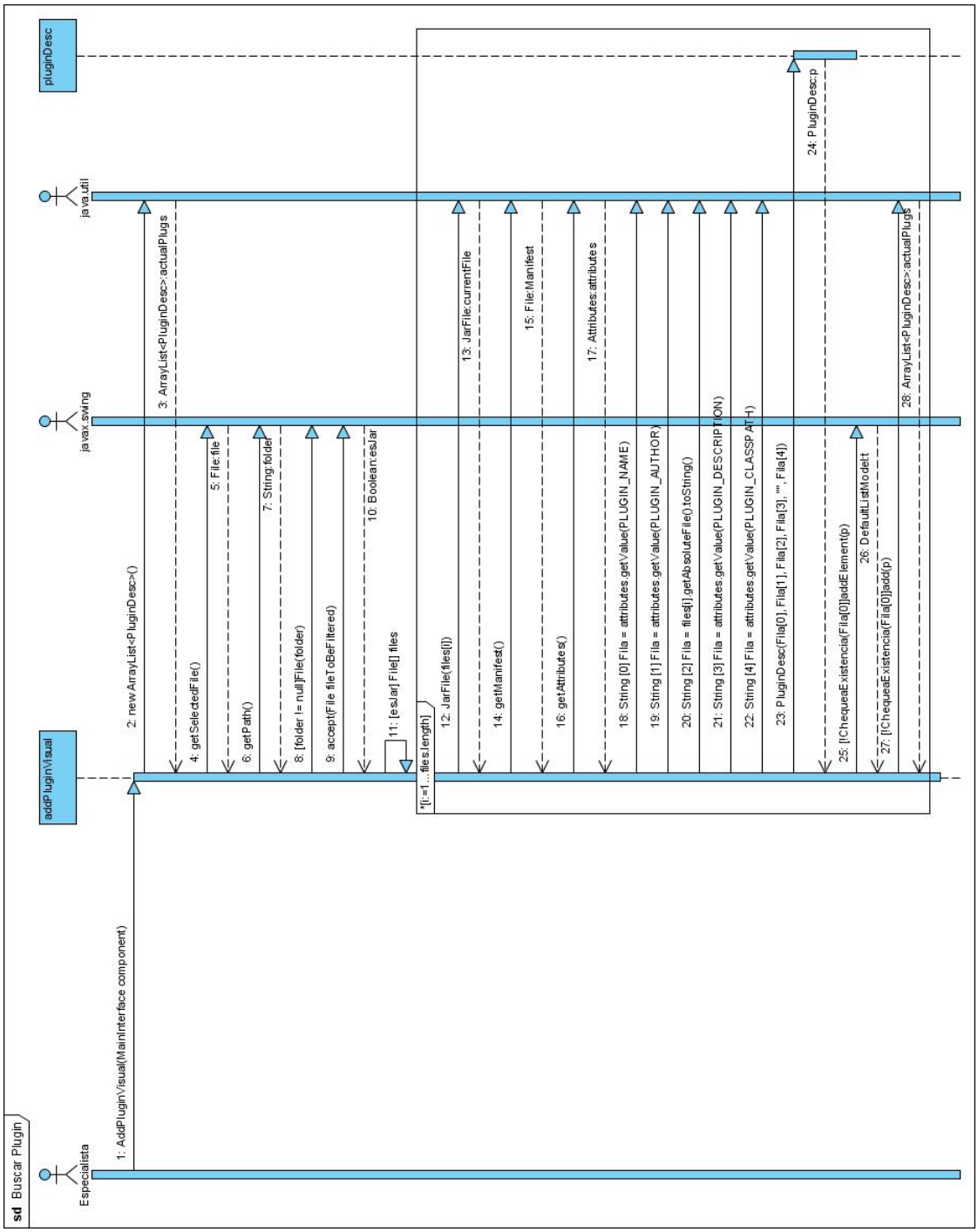
Nombre de la Clase: ComponentCreator

Atributos	**Tipo**
-parent	MainInterface

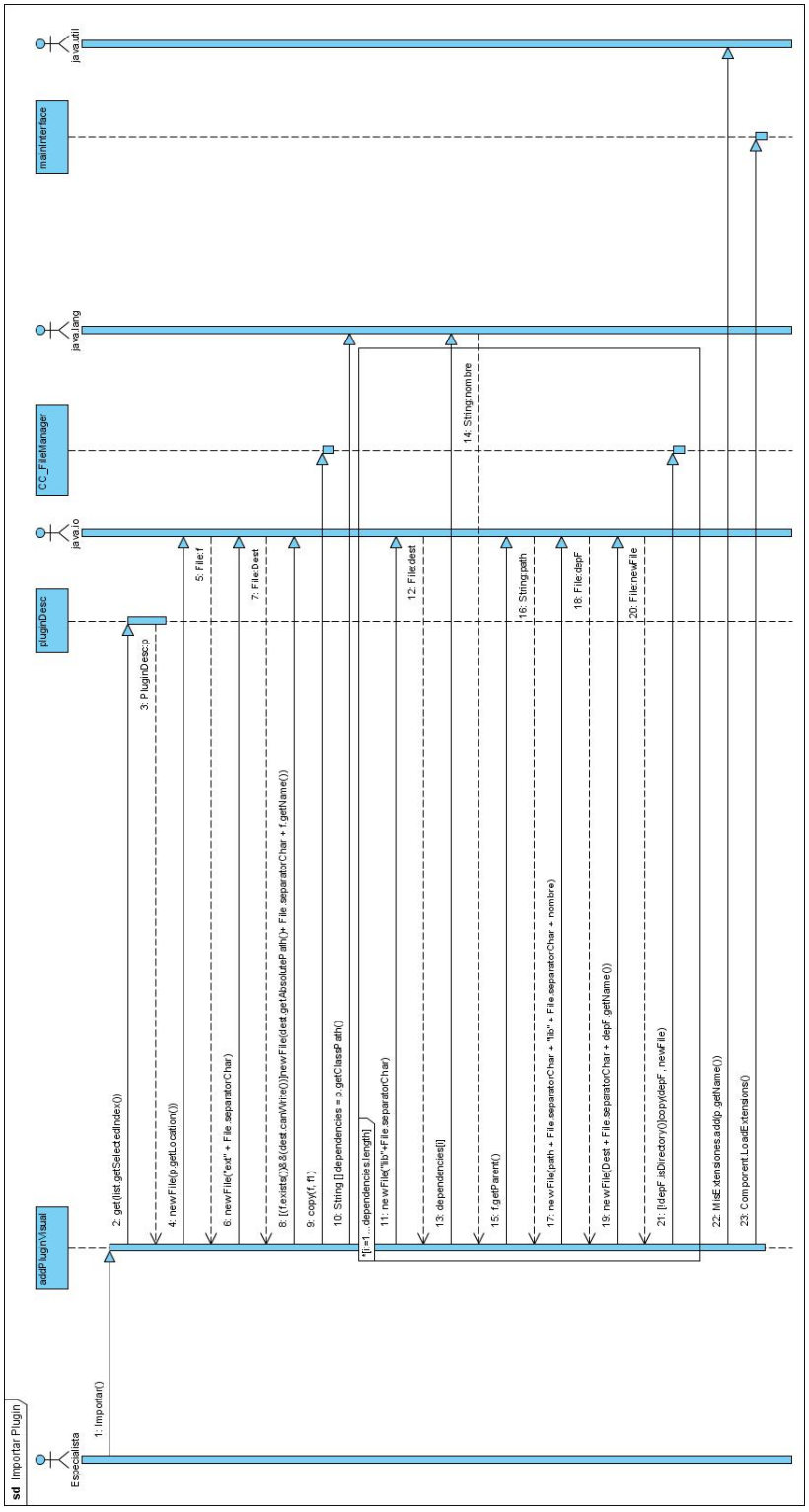
Operaciones	**Descripción**
+ ComponentCreator(MainInterface parent)	Este método es el constructor de la clase por defecto
+ createButton(final ButtonConfiguration buttonConf, int Type): AbstractCommandButton	Este método es el encargado de la creación de un botón para su uso en el JRibbon.
+ createCheck(CheckBoxConfiguration checkConf): JCheckBox	Este método es el encargado de la creación de un checkBox para su uso en el JRibbon.
+createCombo(ComboBoxConfiguration comboConf): JComboBox	Este método es el encargado de la creación de un comboBox para su uso en el JRibbon.

Operaciones	**Descripción**
+ createSpinner(SpinnerConfiguration spinnerConf): JSpinner	Este método es el encargado de la creación de un spinner para su uso en el JRibbon.

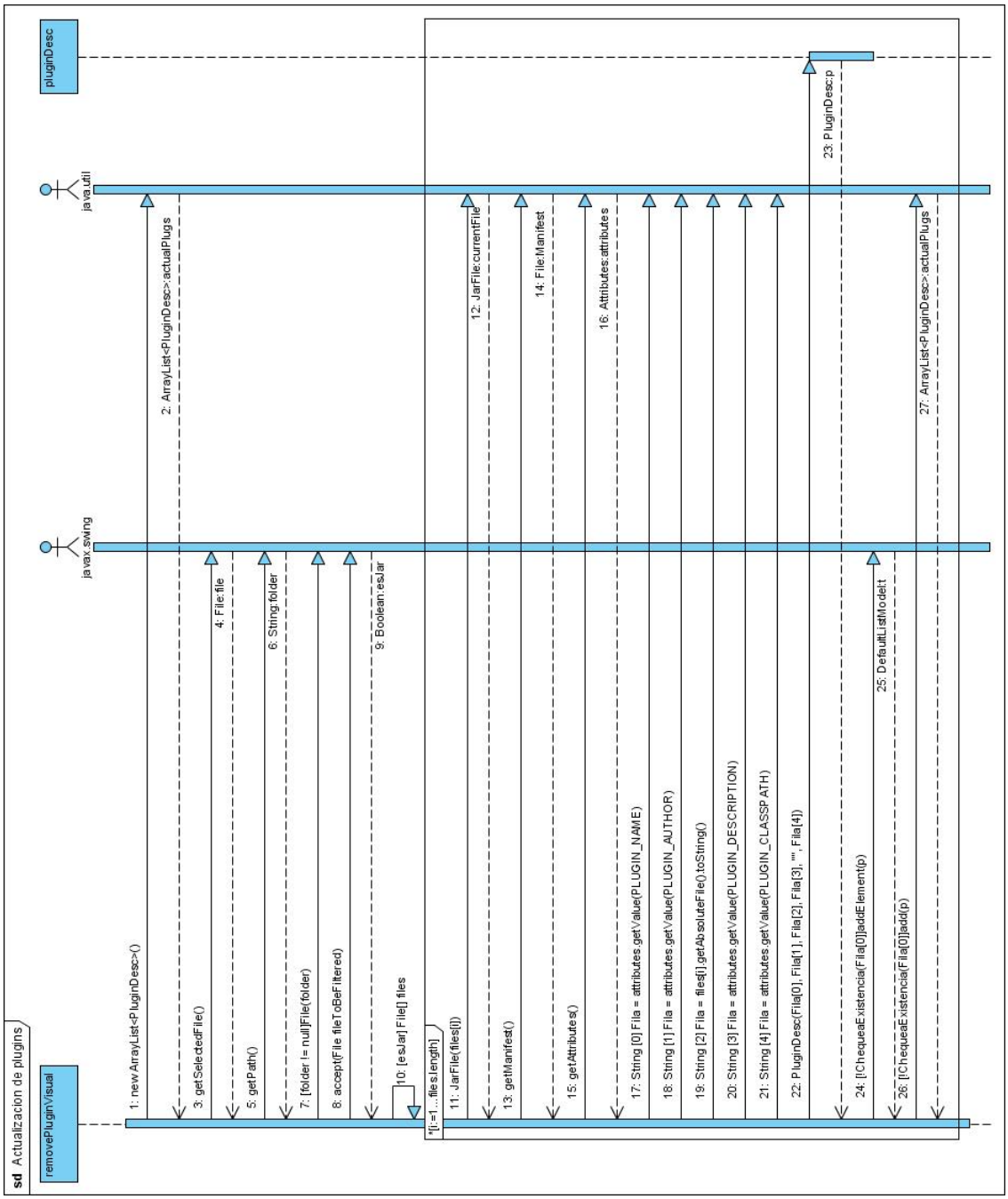
Anexo 2: Diagramas de Secuencias de la Sección Adicionar Plug-ins del caso de uso Gestionar plug-ins.



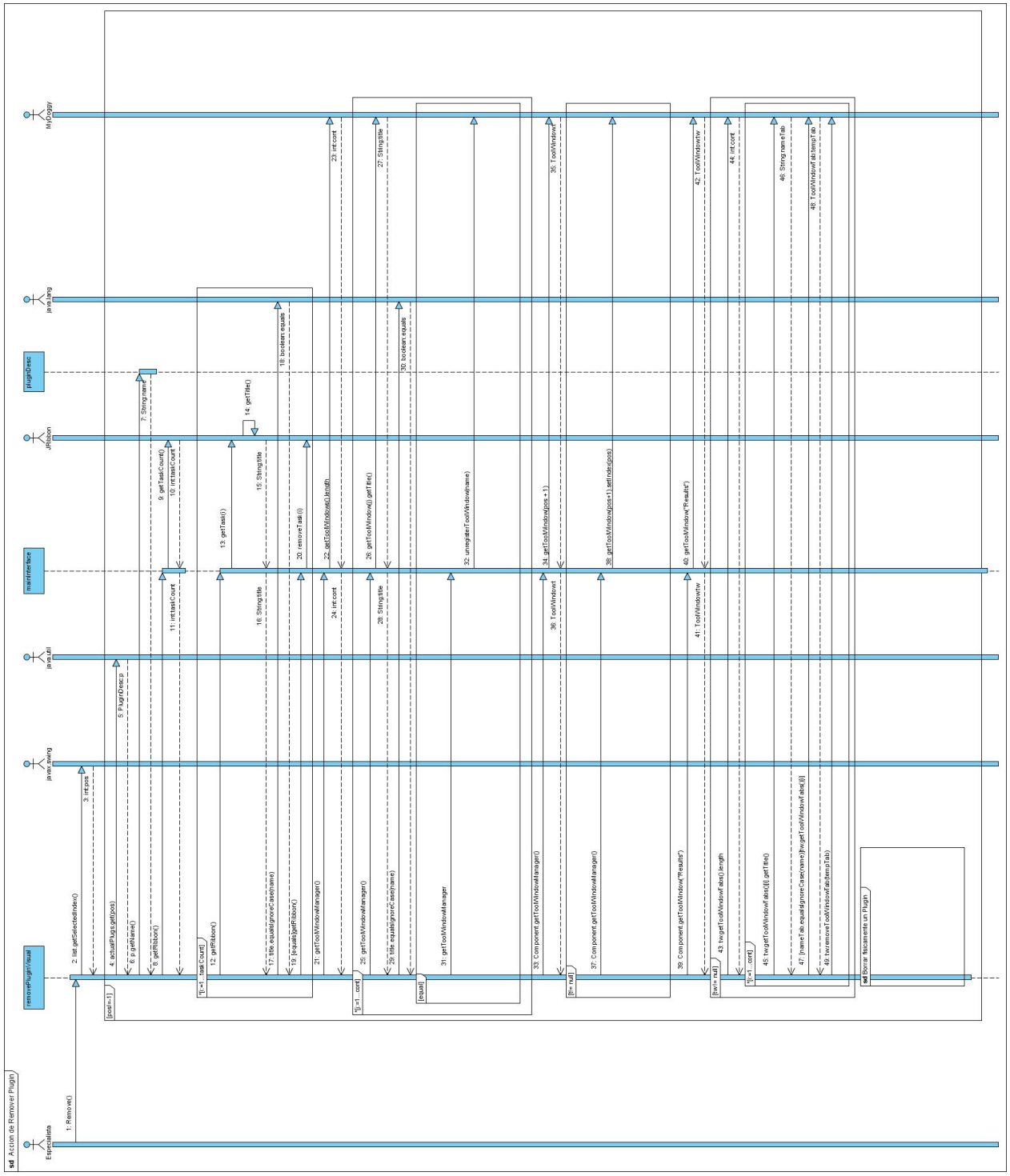
Anexo 2: Diagramas de Secuencias de la Sección Adicionar Plug-ins del caso de uso Gestionar plug-ins (Continuación).



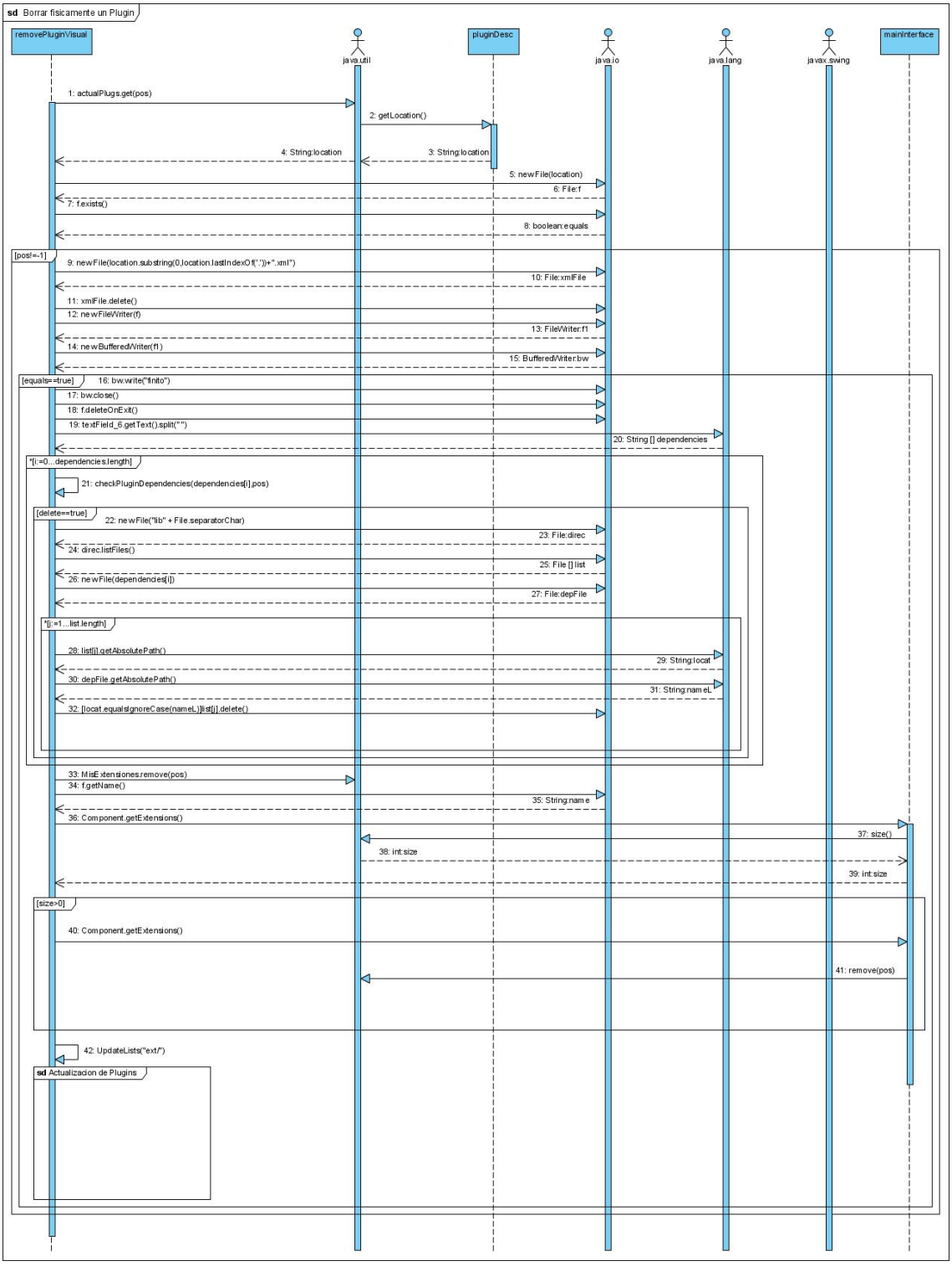
Anexo 3: Diagramas de Secuencias de la Sección Remover Plug-ins del caso de uso Gestionar plug-ins.



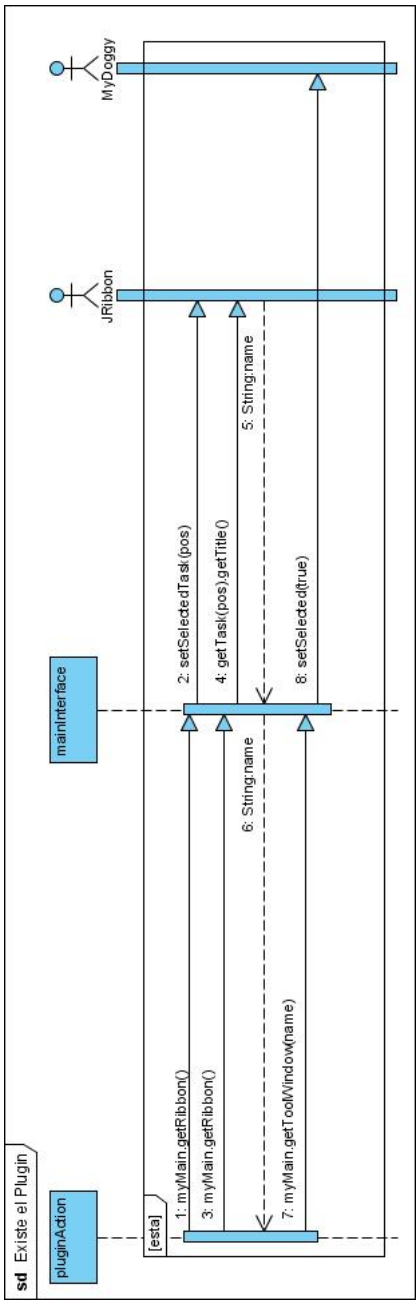
Anexo 3: Diagramas de Secuencias de la Sección Remover Plug-ins del caso de uso Gestionar plug-ins (Continuación).



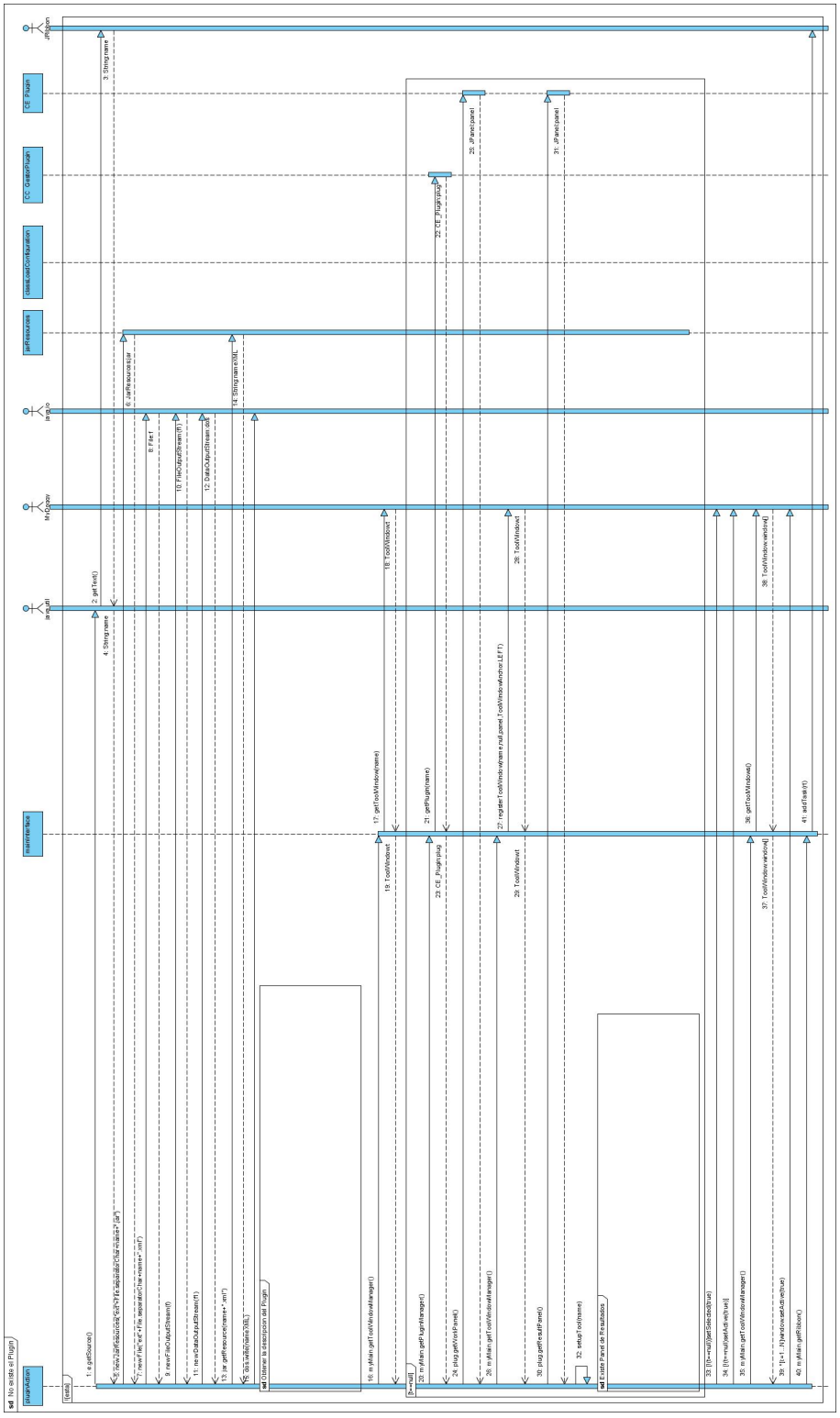
Anexo 3: Diagramas de Secuencias de la Sección Remover Plug-ins del caso de uso Gestionar plug-ins (Continuación).



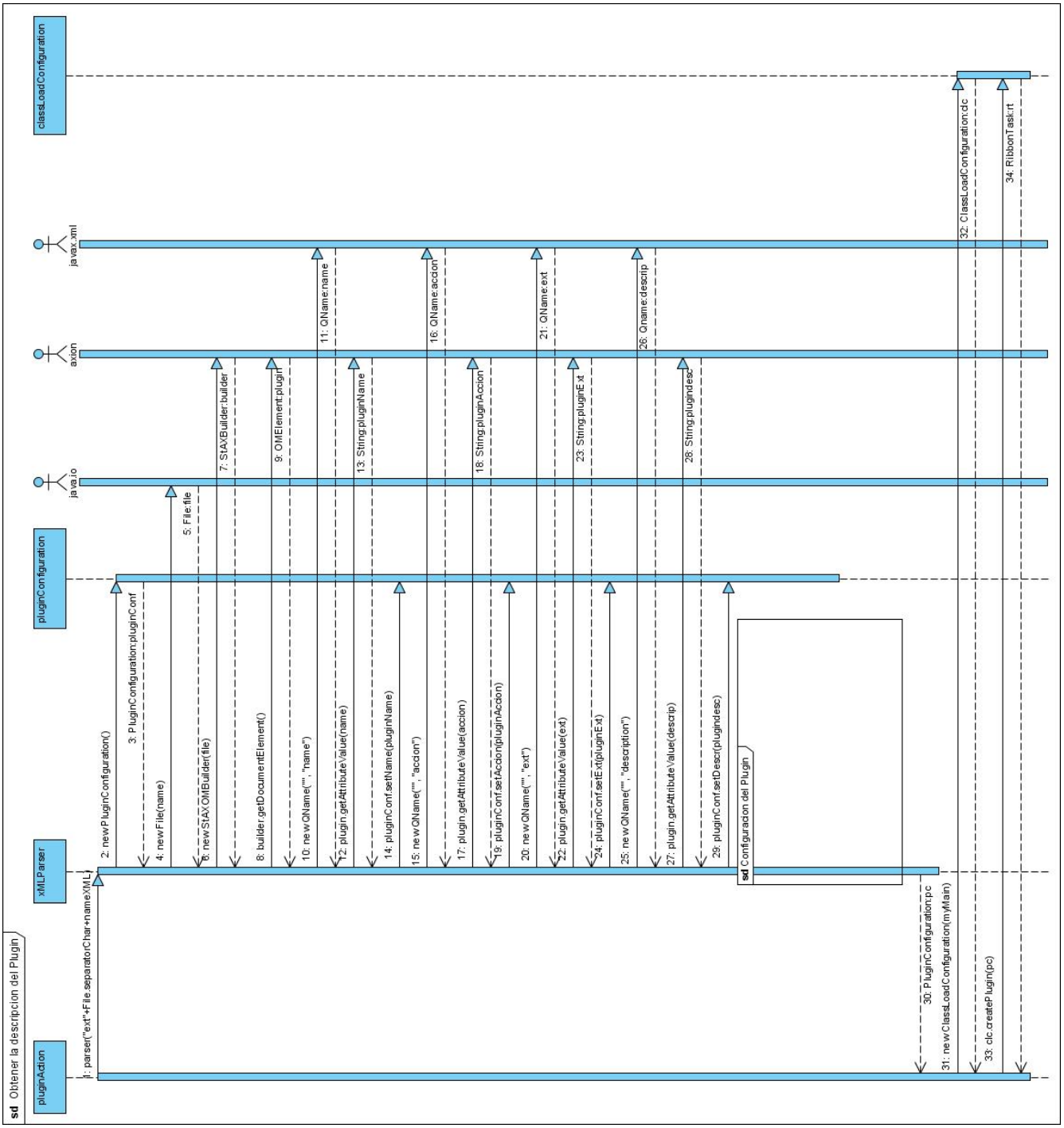
Anexo 4: Diagramas de Secuencias de la Sección Activar Plug-ins del caso de uso Gestionar plug-ins.



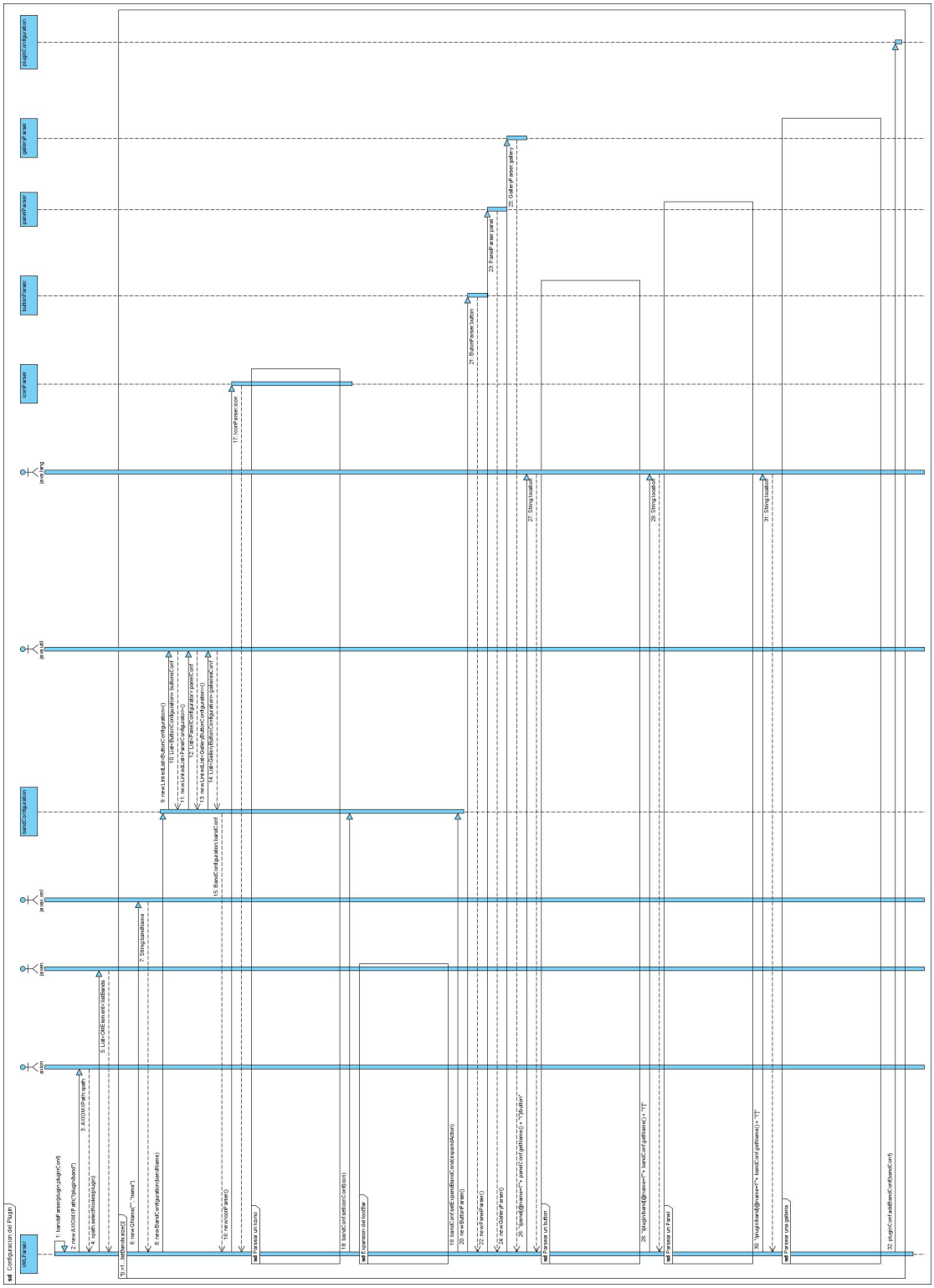
Anexo 4: Diagramas de Secuencias de la Sección Activar Plug-ins del caso de uso Gestionar plug-ins (Continuación).



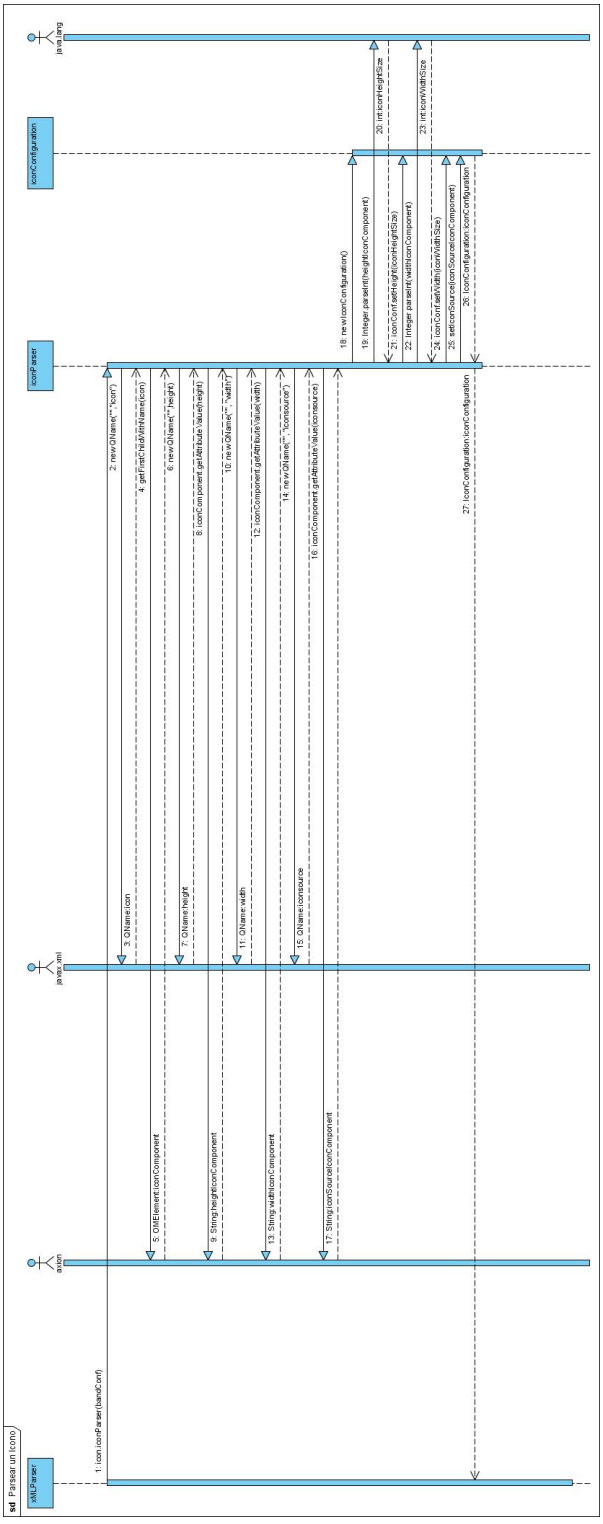
Anexo 4: Diagramas de Secuencias de la Sección Activar Plug-ins del caso de uso Gestionar plug-ins (Continuación).



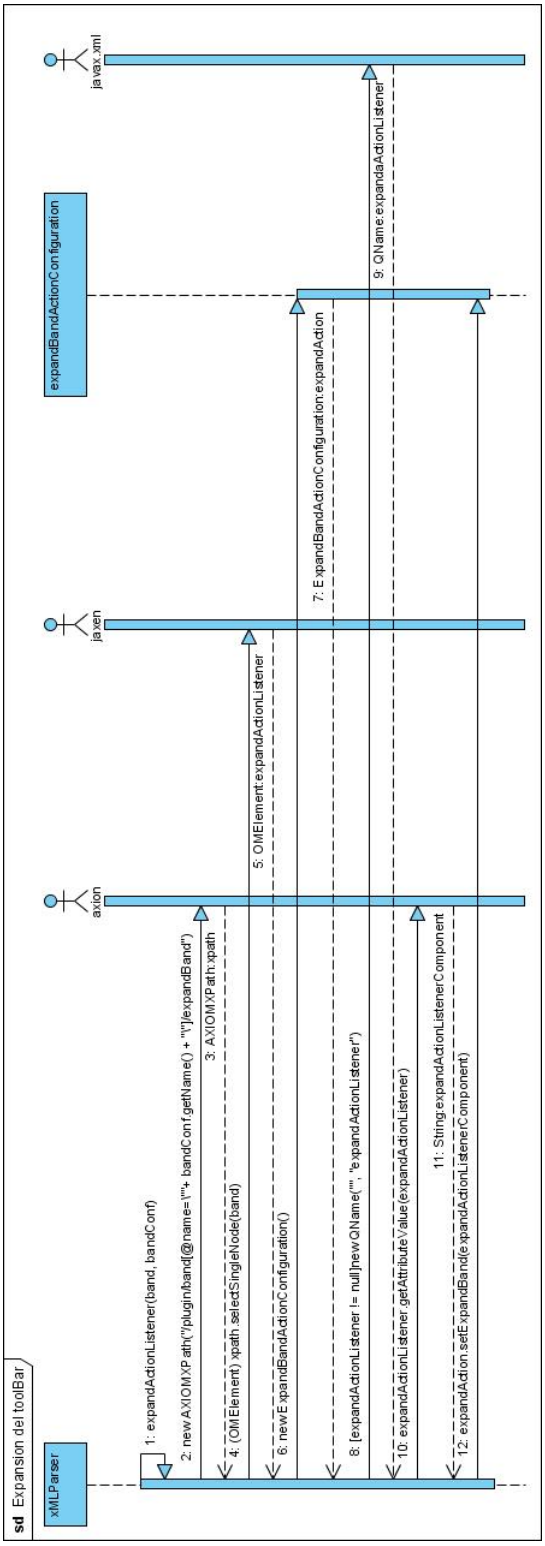
Anexo 4: Diagramas de Secuencias de la Sección Activar Plug-ins del caso de uso Gestionar plug-ins (Continuación).



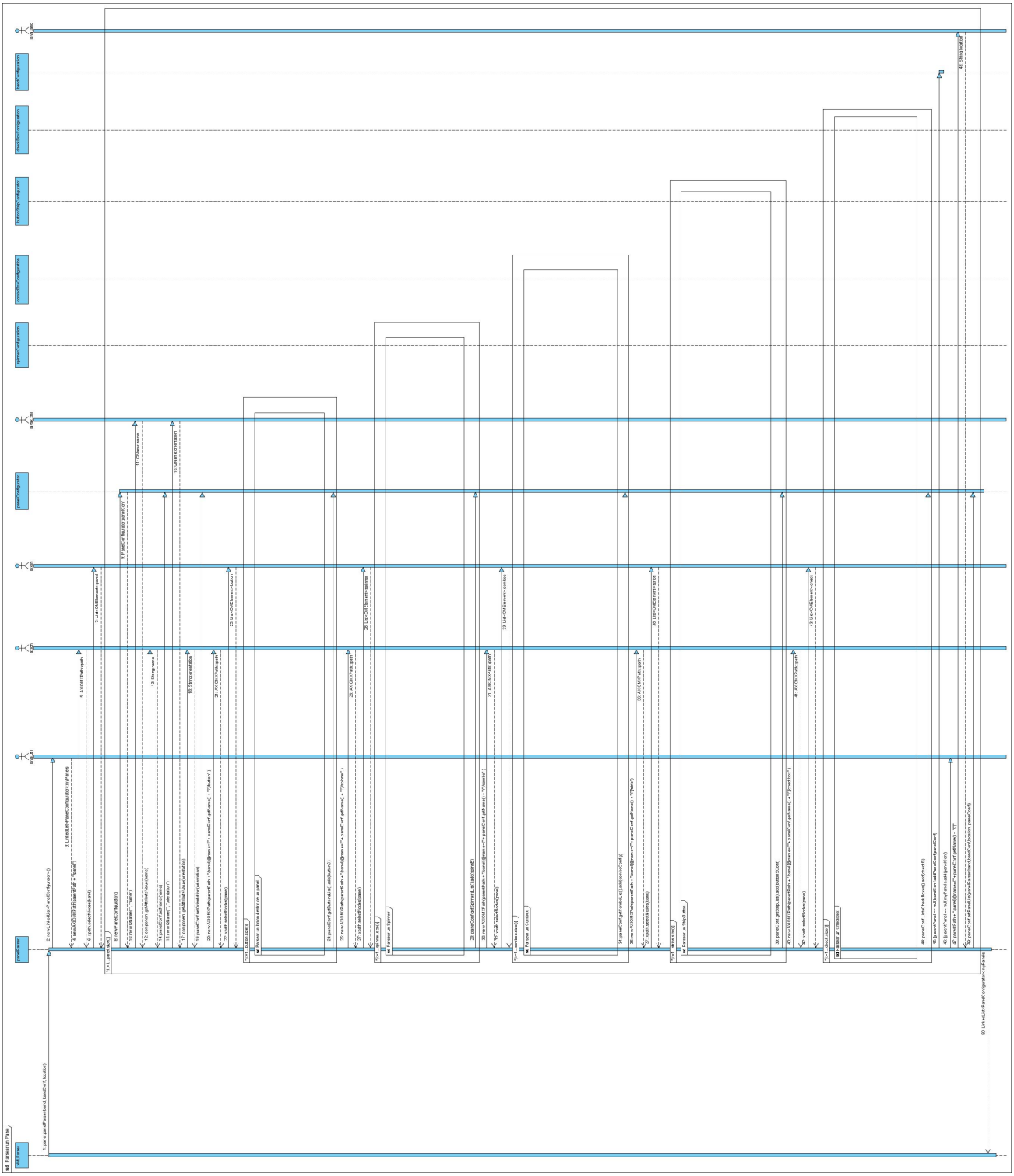
Anexo 4: Diagramas de Secuencias de la Sección Activar Plug-ins del caso de uso Gestionar plug-ins (Continuación).



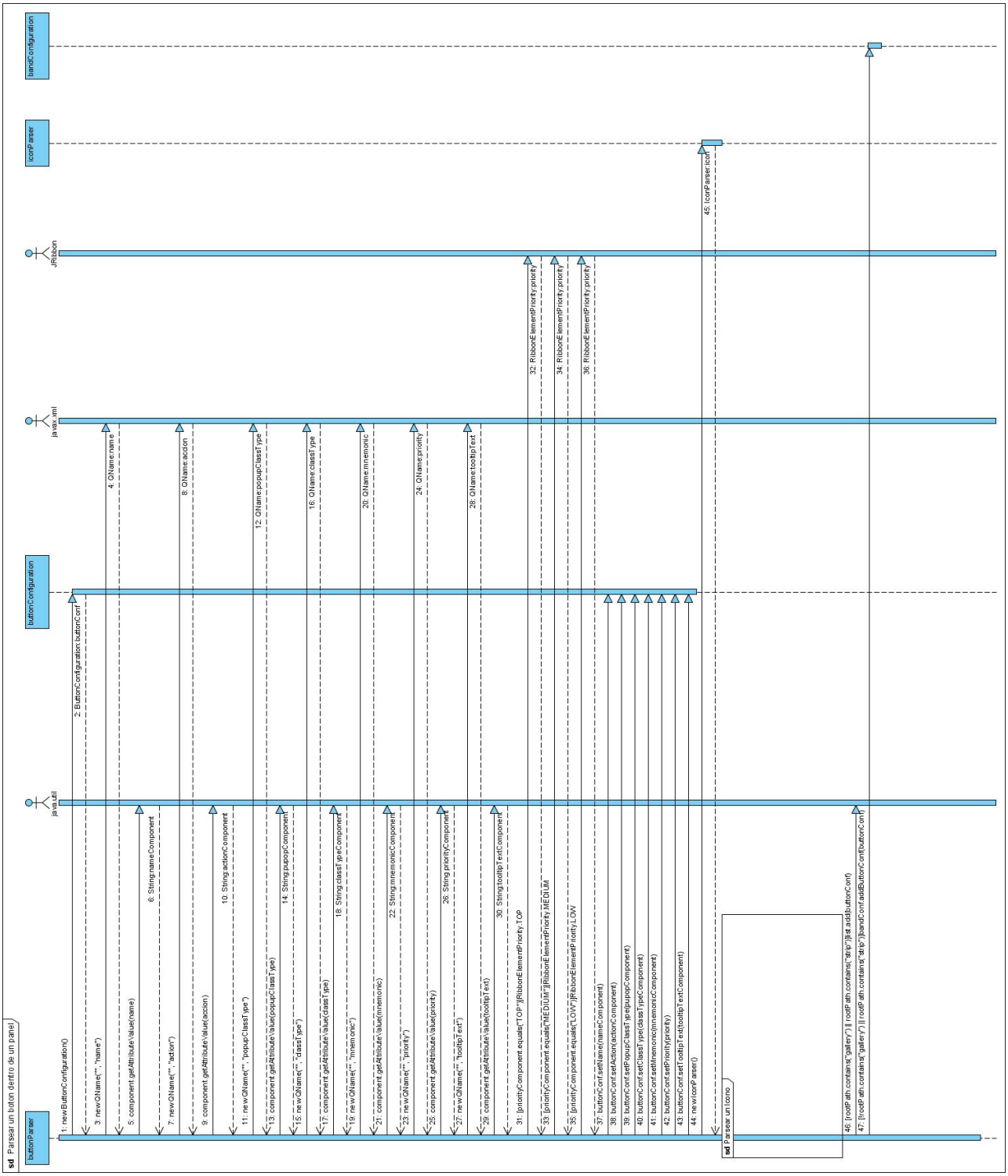
Anexo 4: Diagramas de Secuencias de la Sección Activar Plug-ins del caso de uso Gestionar plug-ins (Continuación).



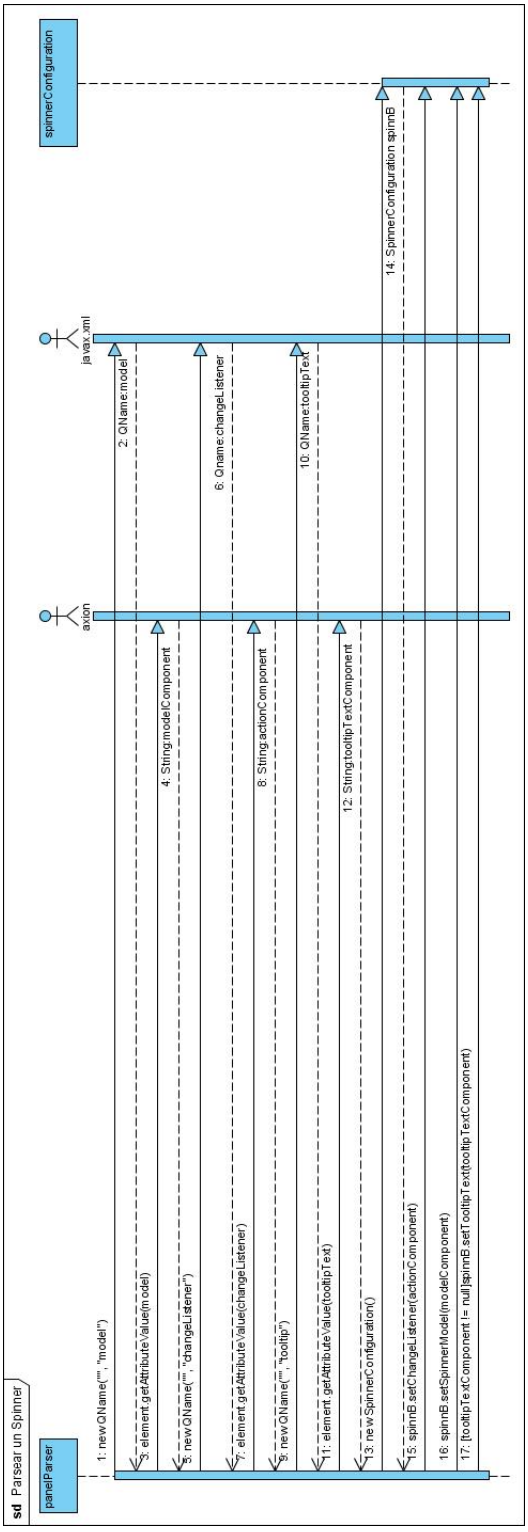
Anexo 4: Diagramas de Secuencias de la Sección Activar Plug-ins del caso de uso Gestionar plug-ins (Continuación).



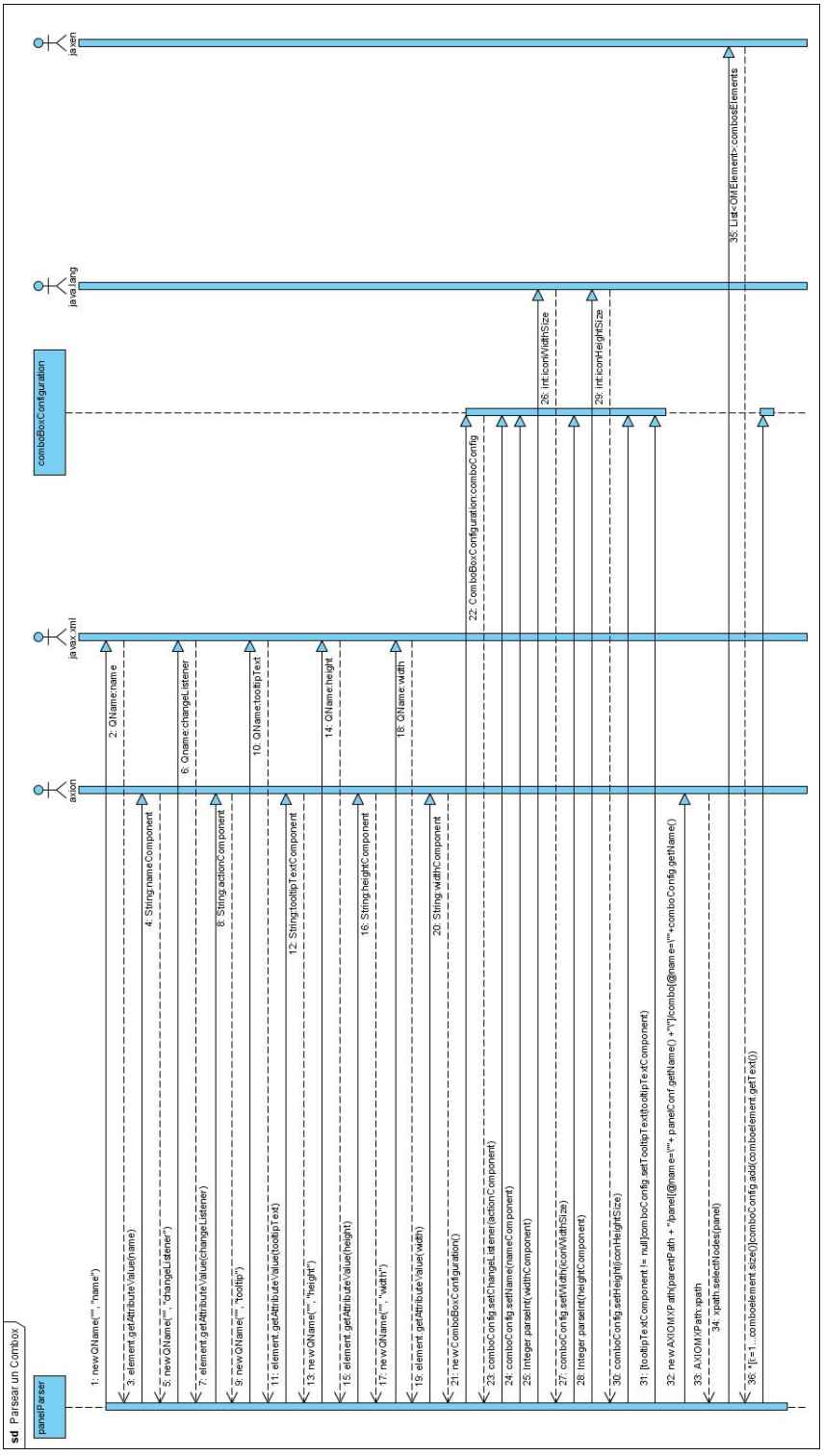
Anexo 4: Diagramas de Secuencias de la Sección Activar Plug-ins del caso de uso Gestionar plug-ins (Continuación).



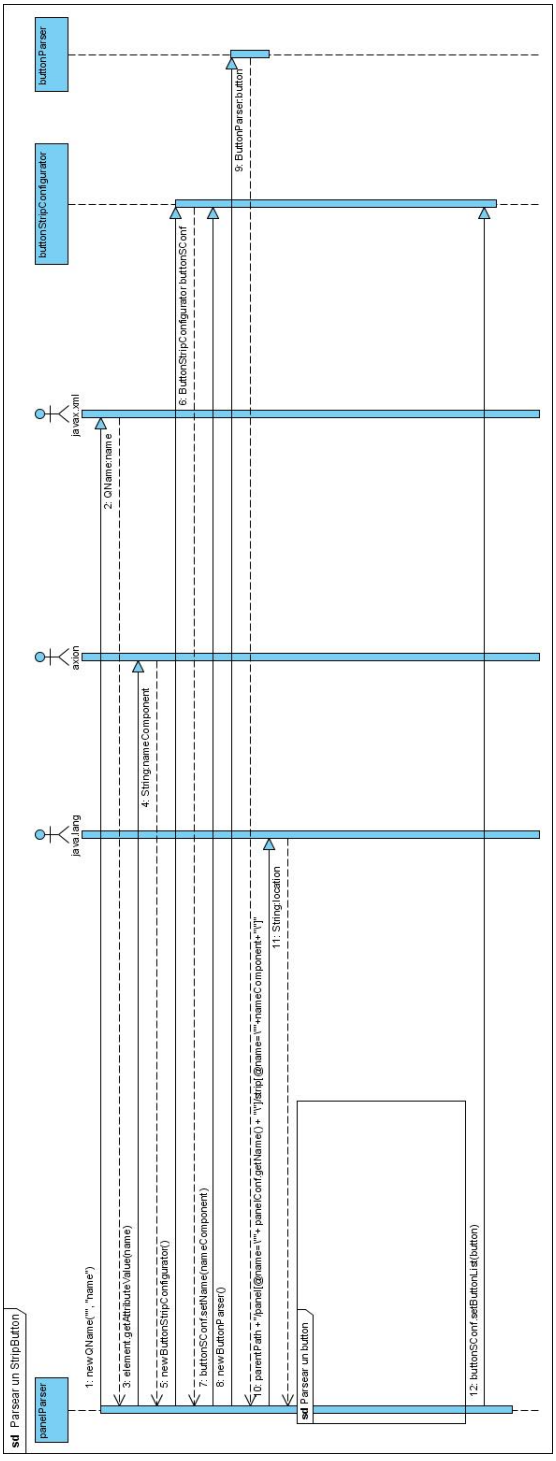
Anexo 4: Diagramas de Secuencias de la Sección Activar Plug-ins del caso de uso Gestionar plug-ins (Continuación).



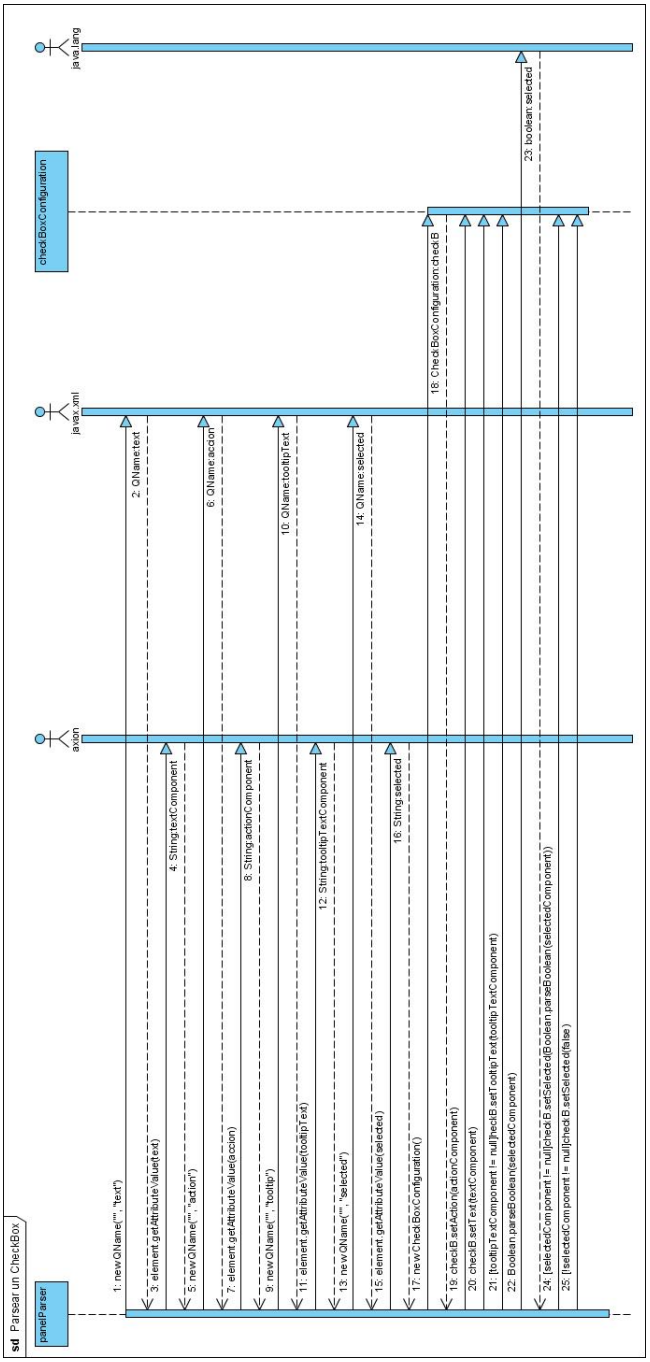
Anexo 4: Diagramas de Secuencias de la Sección Activar Plug-ins del caso de uso Gestionar plug-ins (Continuación).



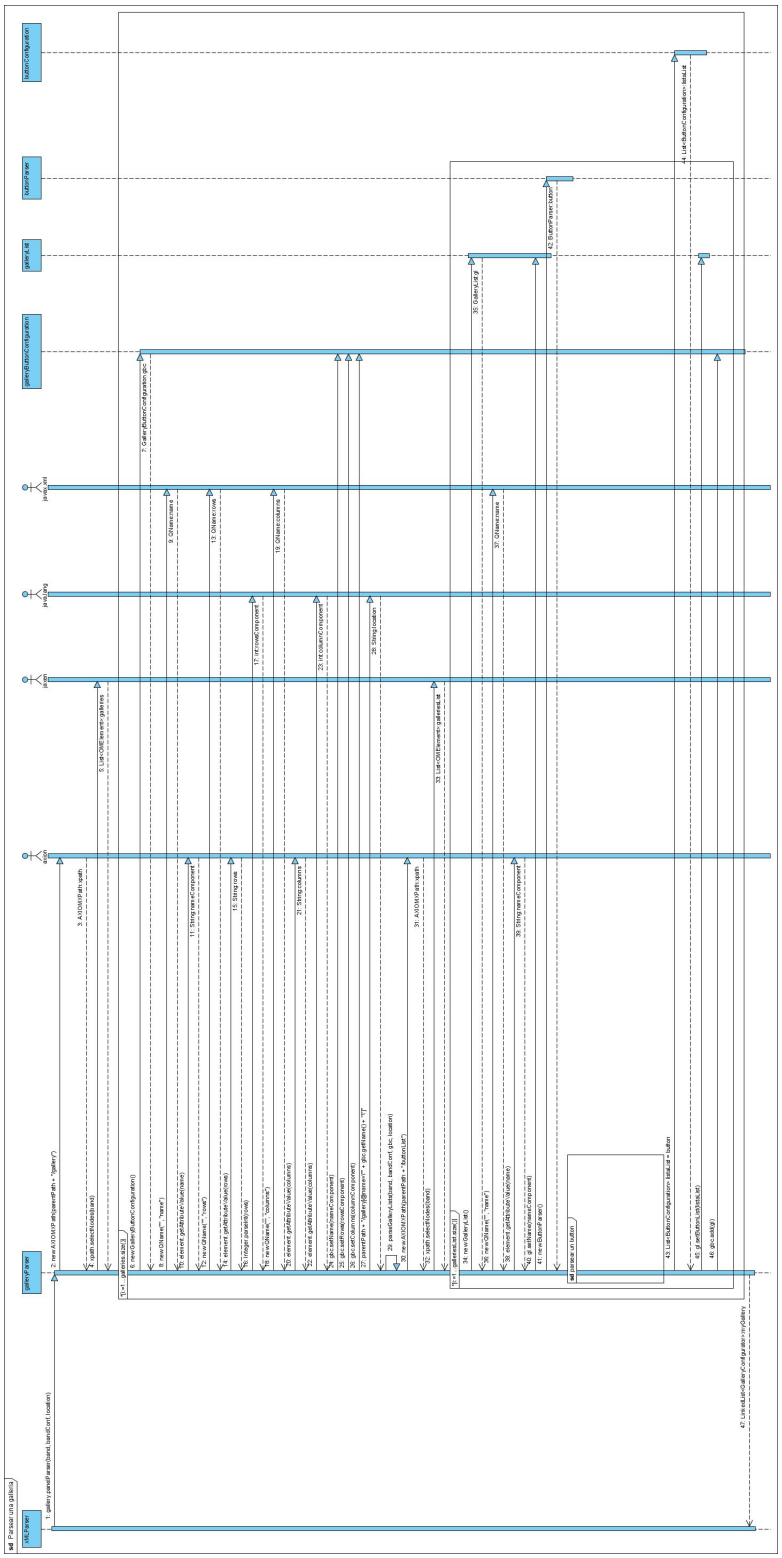
Anexo 4: Diagramas de Secuencias de la Sección Activar Plug-ins del caso de uso Gestionar plug-ins (Continuación).



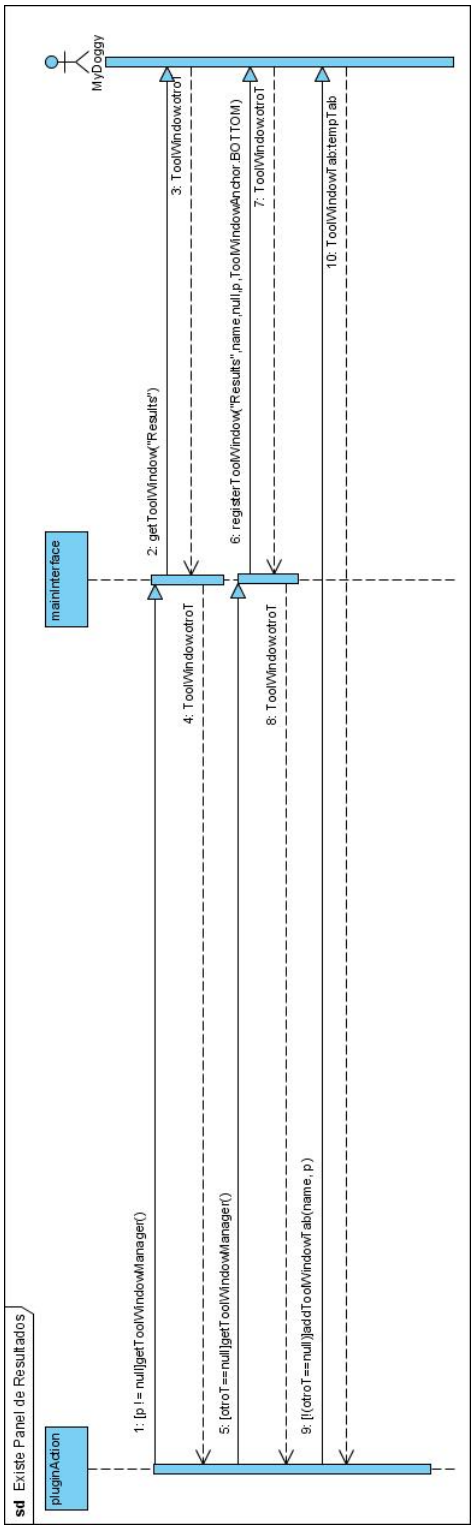
Anexo 4: Diagramas de Secuencias de la Sección Activar Plug-ins del caso de uso Gestionar plug-ins (Continuación).



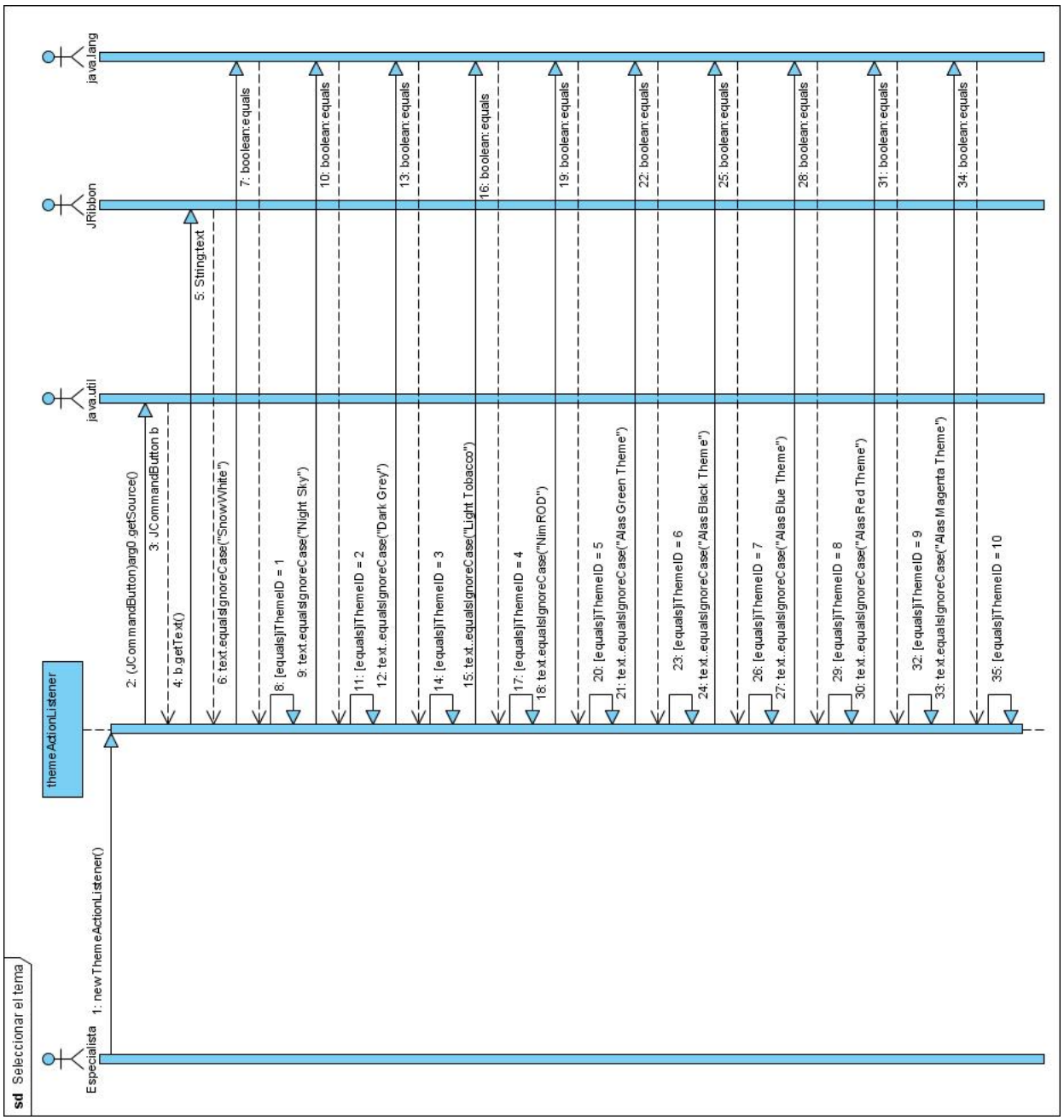
Anexo 4: Diagramas de Secuencias de la Sección Activar Plug-ins del caso de uso Gestionar plug-ins (Continuación).



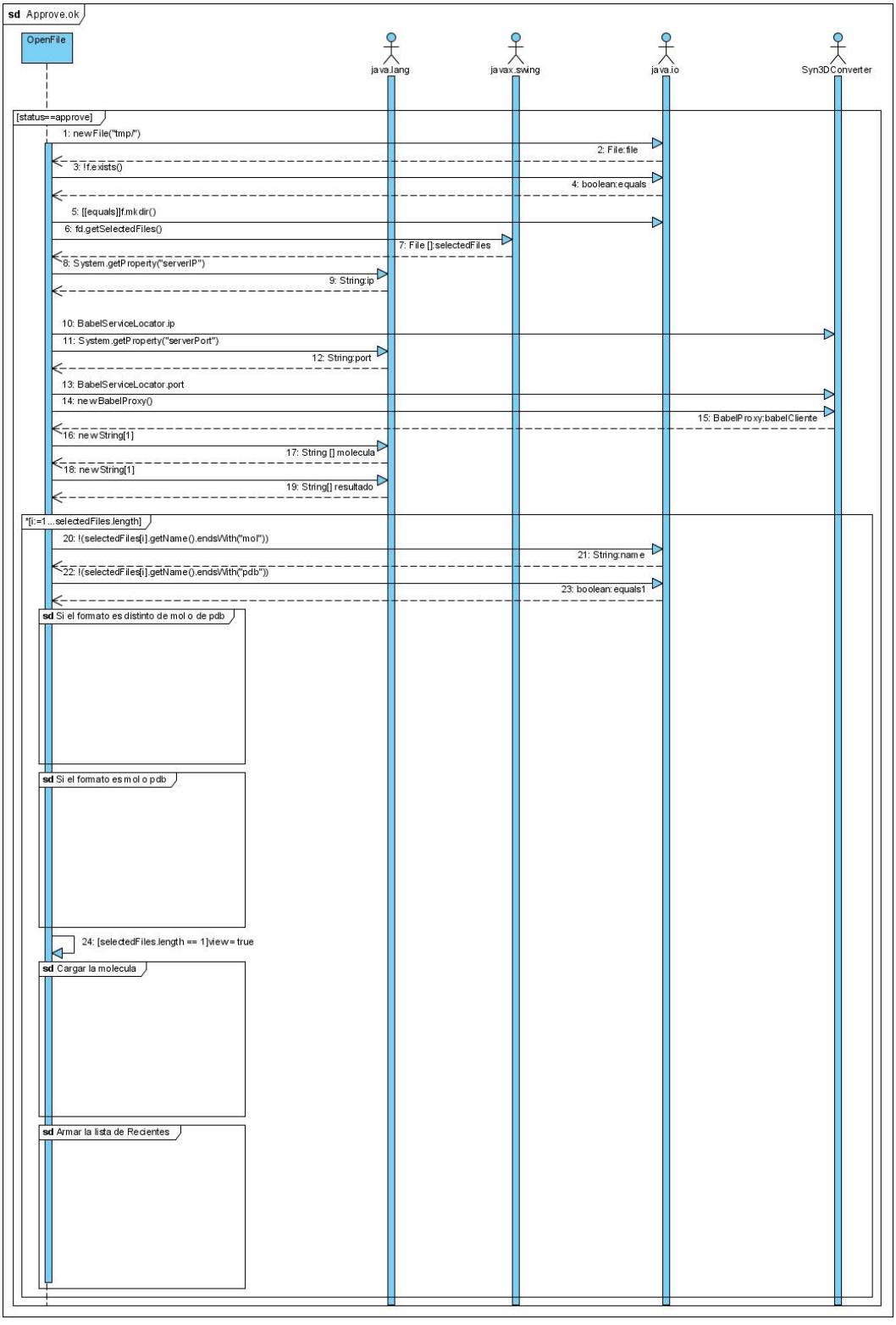
Anexo 4: Diagramas de Secuencias de la Sección Activar Plug-ins del caso de uso Gestionar plug-ins (Continuación).



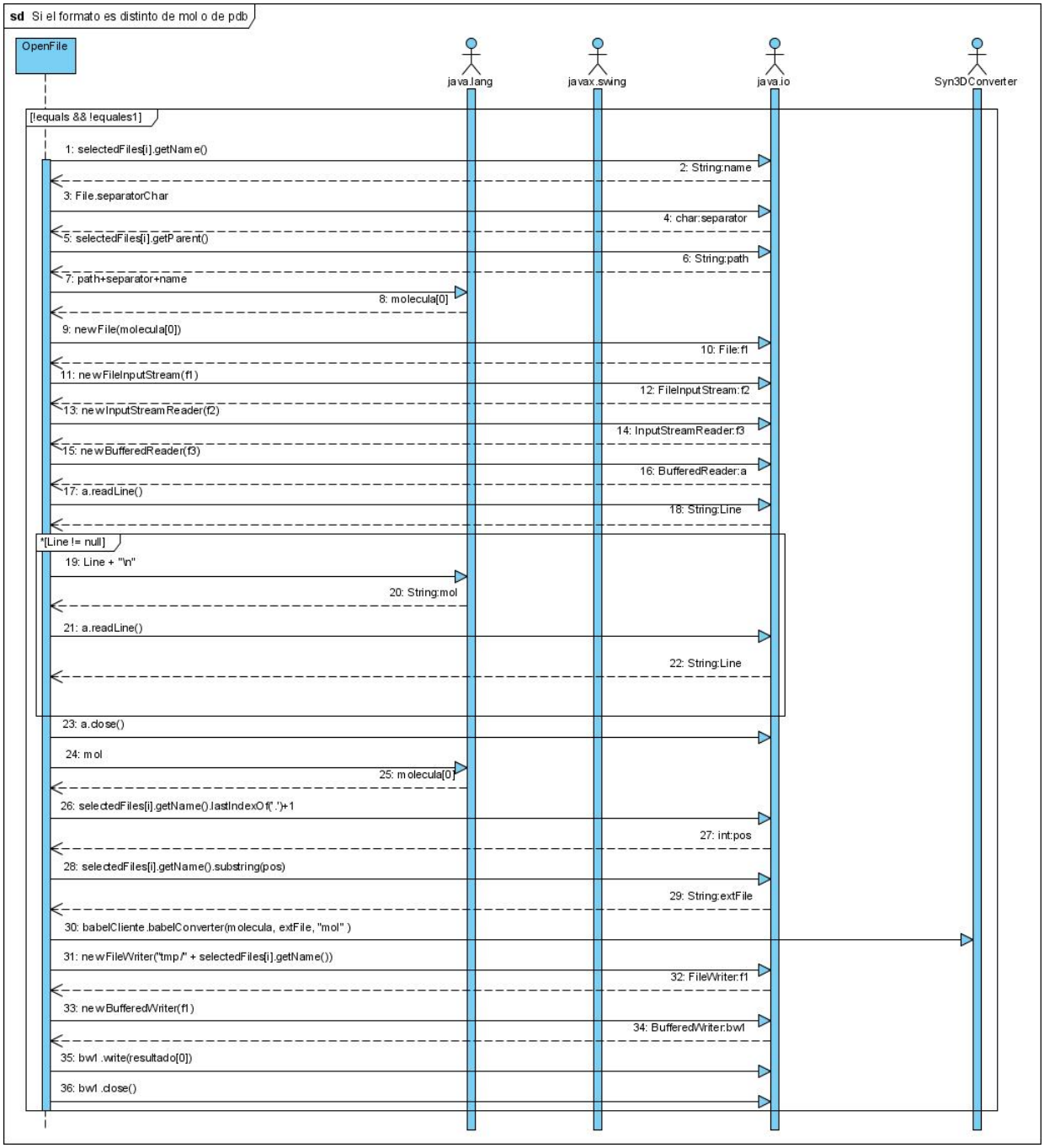
Anexo 5: Diagramas de Secuencias de la Sección Preferencias de Temas del caso de uso Gestionar Preferencia.



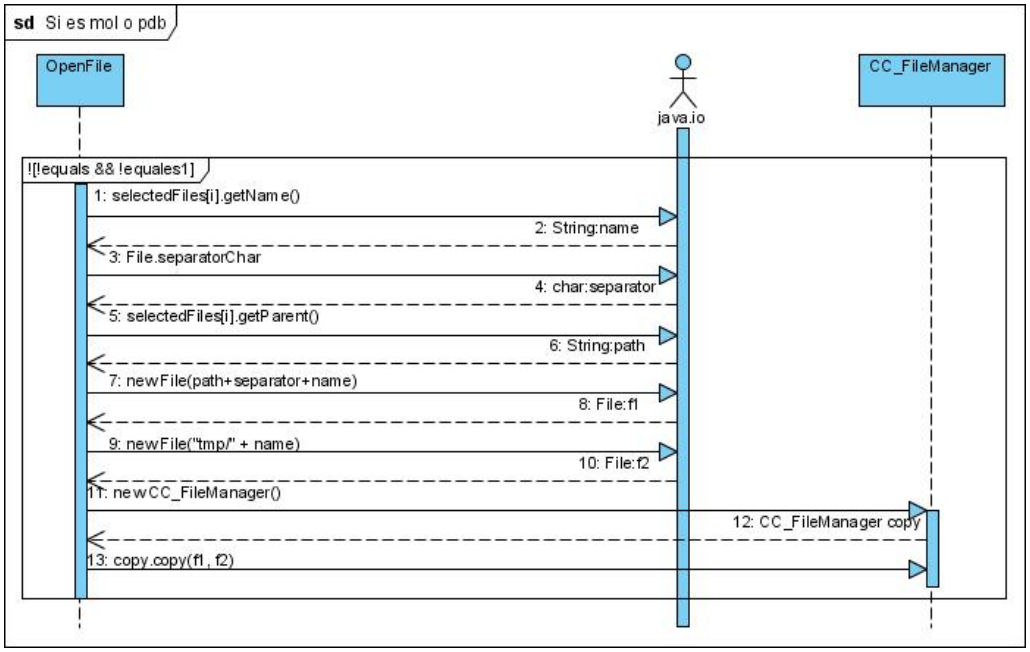
Anexo 6: Diagramas de Secuencias de la Sección Abrir del caso de uso Abrir Fichero.



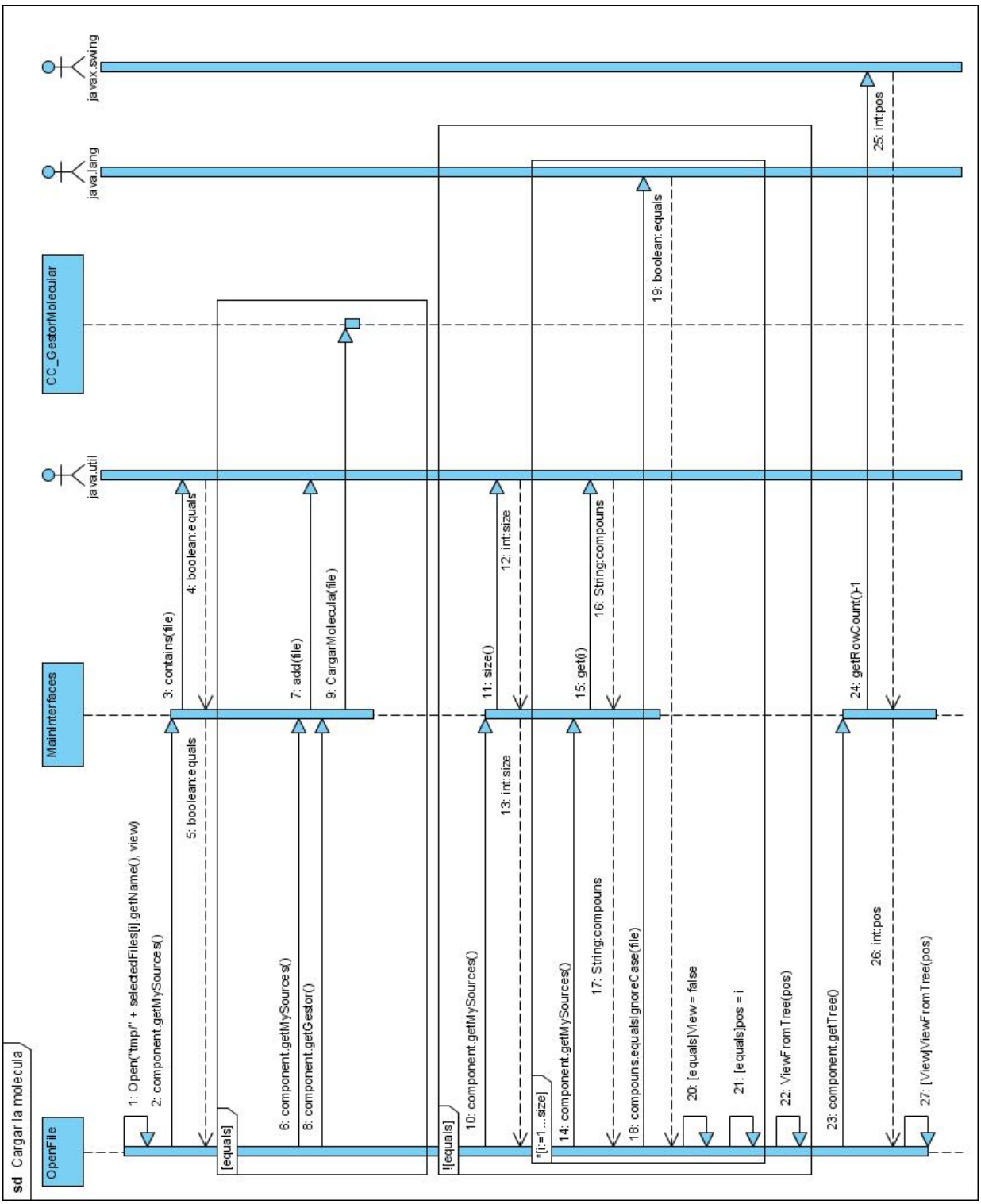
Anexo 6: Diagramas de Secuencias de la Sección Abrir del caso de uso Abrir Fichero (Continuación).



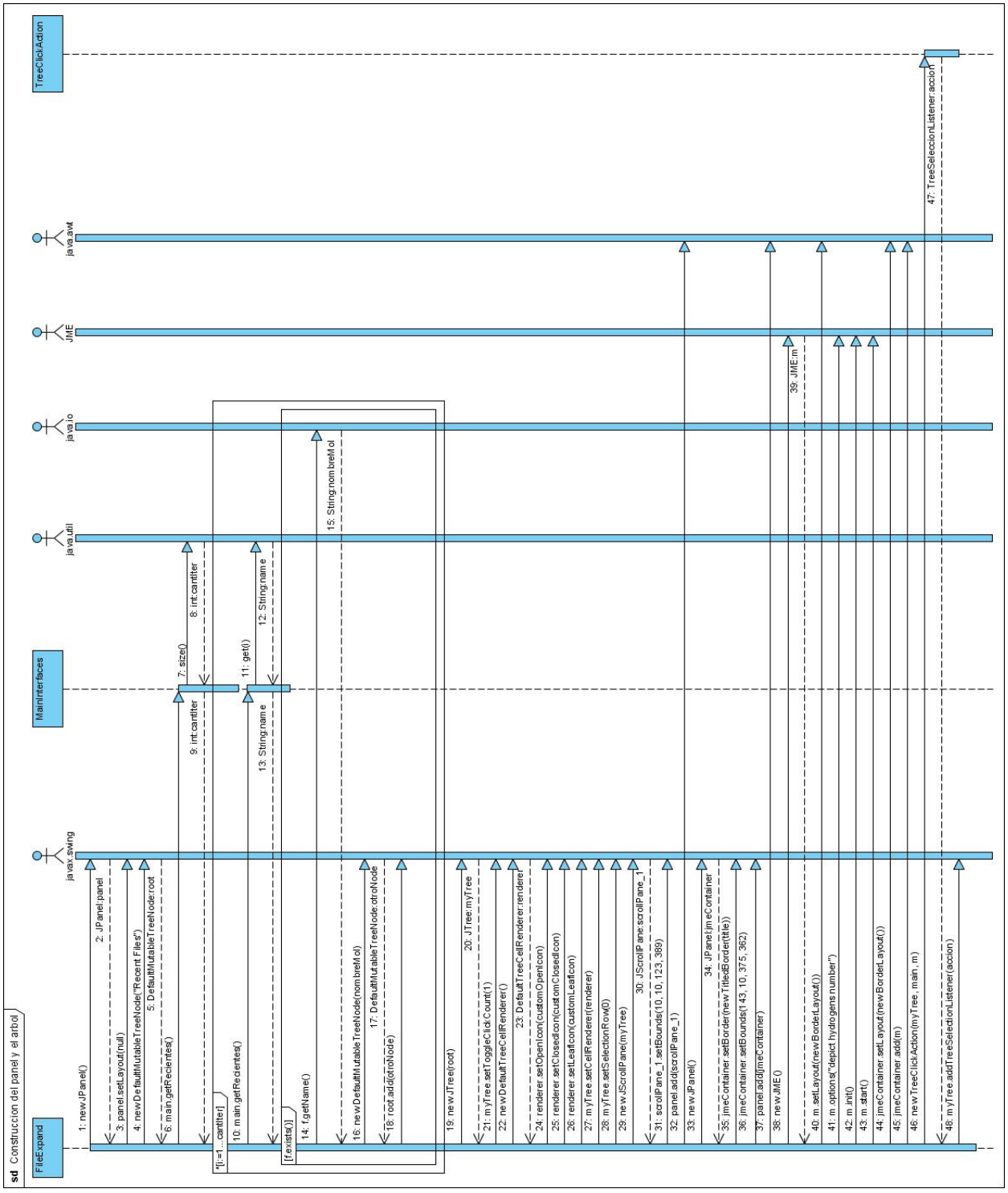
Anexo 6: Diagramas de Secuencias de la Sección Abrir del caso de uso Abrir Fichero (Continuación).



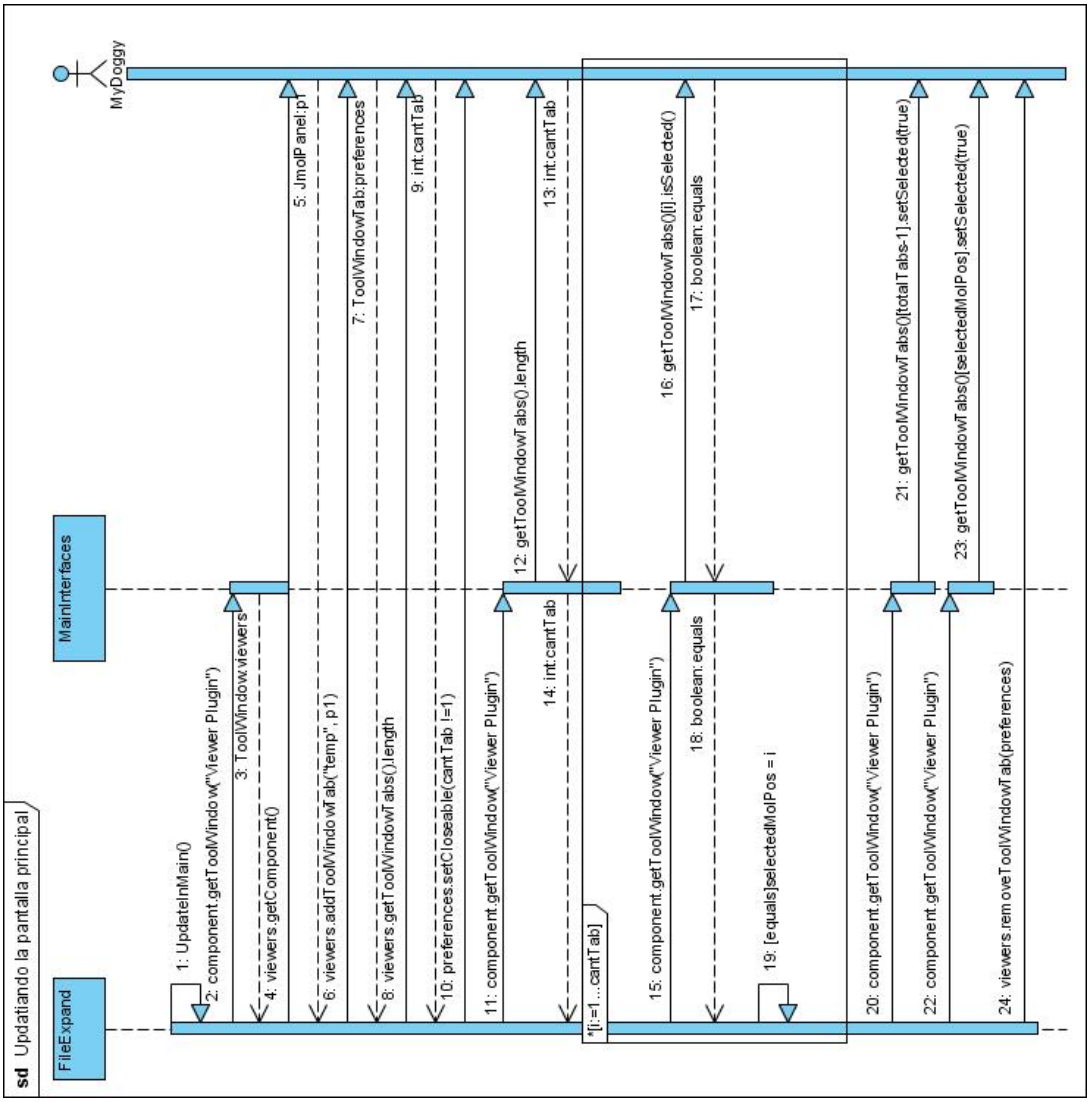
Anexo 6: Diagramas de Secuencias de la Sección Abrir del caso de uso Abrir Fichero (Continuación).



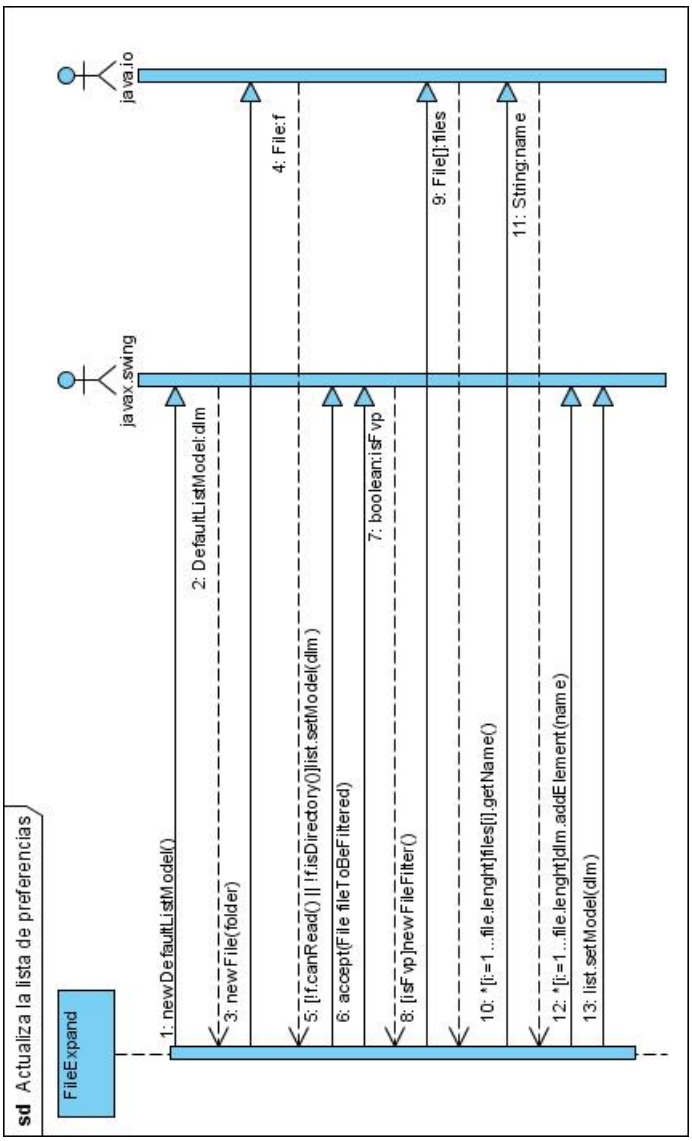
Anexo 7: Diagramas de Secuencias de la Sección Abrir Reciente del caso de uso Abrir Fichero.



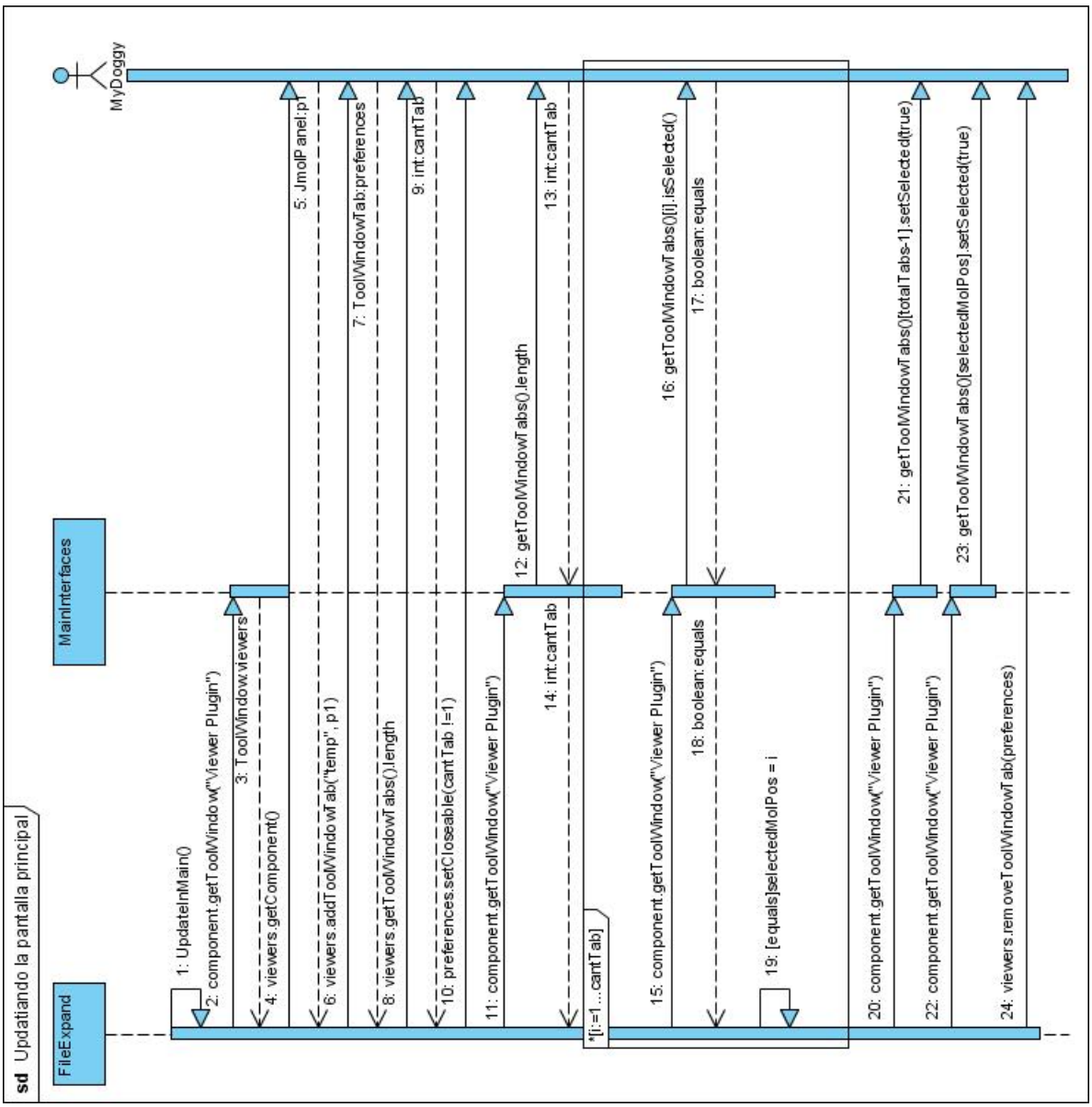
Anexo 7: Diagramas de Secuencias de la Sección Abrir Reciente del caso de uso Abrir Fichero (Continuación).



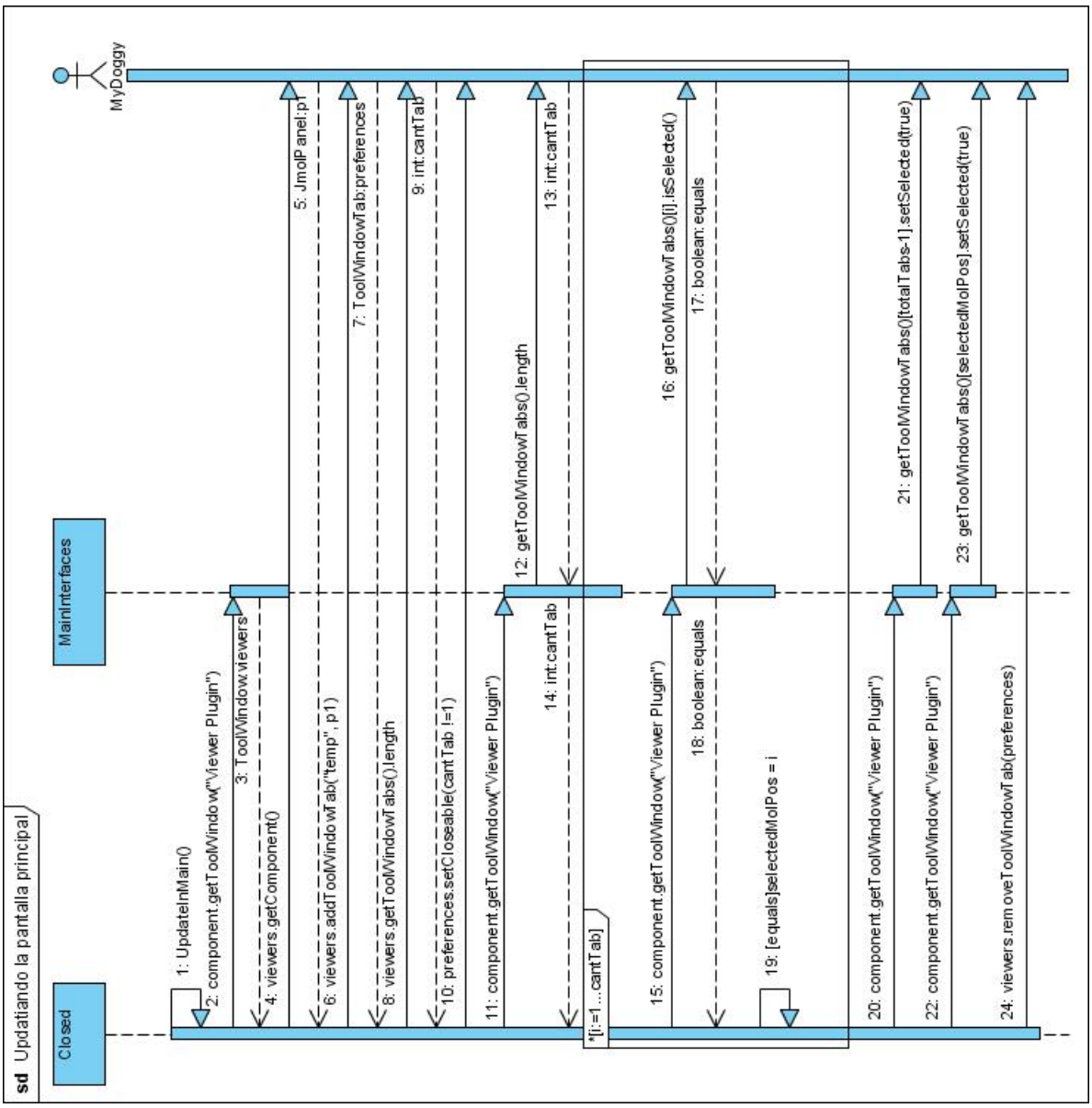
Anexo 8: Diagramas de Secuencias de la Sección Abrir Preferencias de Visualización del caso de uso Abrir Fichero.



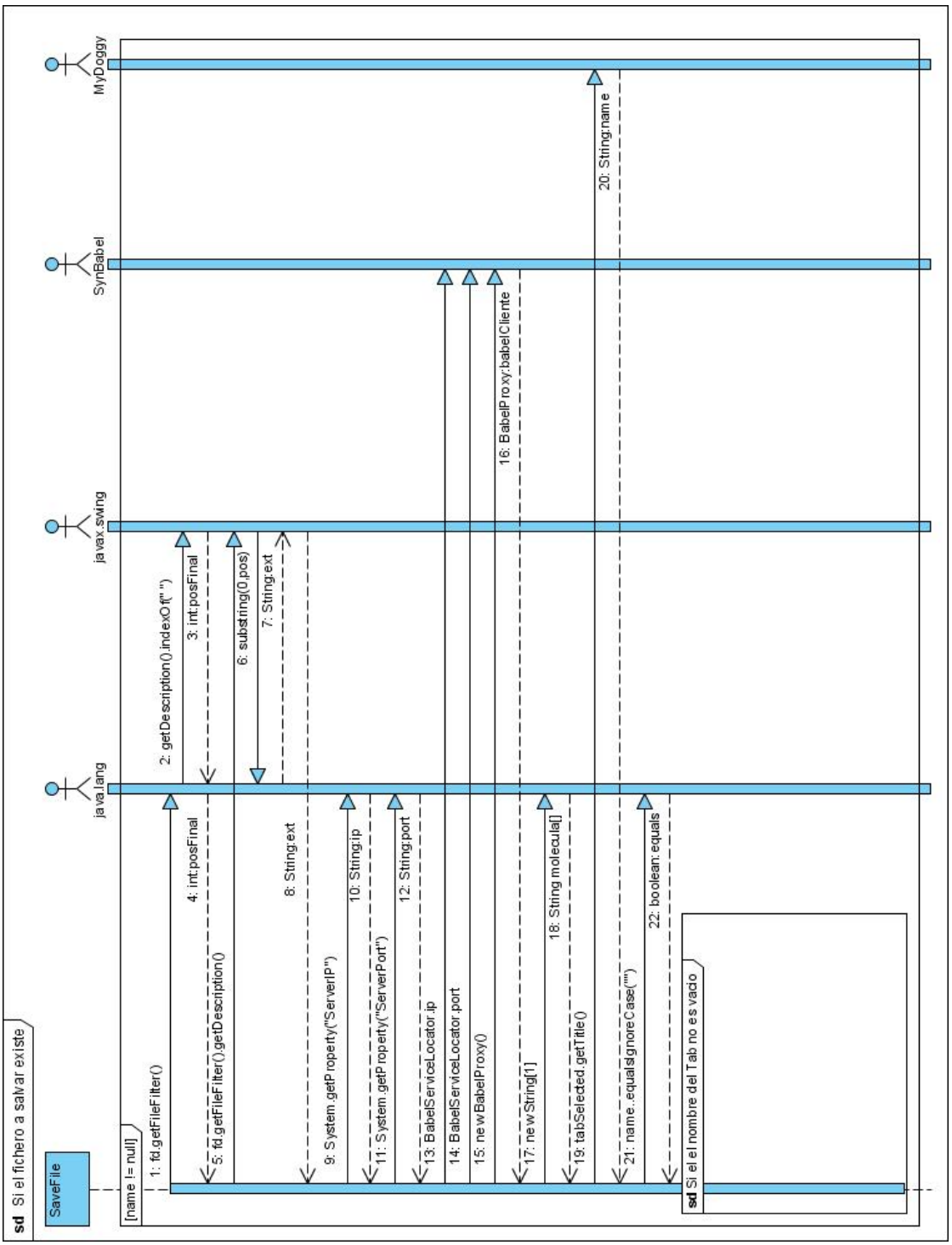
Anexo 8: Diagramas de Secuencias de la Sección Abrir Preferencias de Visualización del caso de uso Abrir Fichero (Continuación).



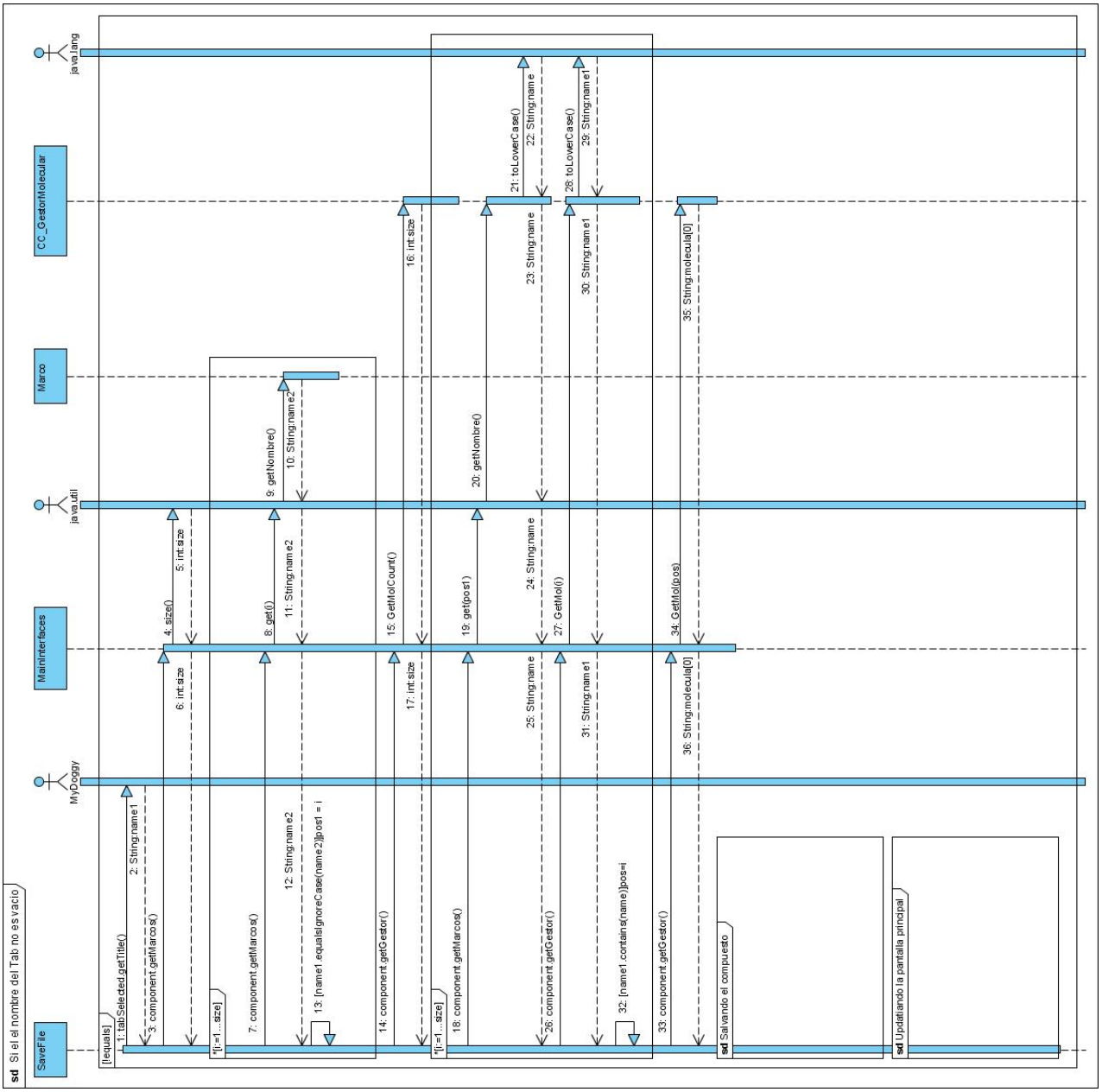
Anexo 9: Diagramas de Secuencias de la Sección Cerrar Fichero del caso de uso Gestionar Fichero.



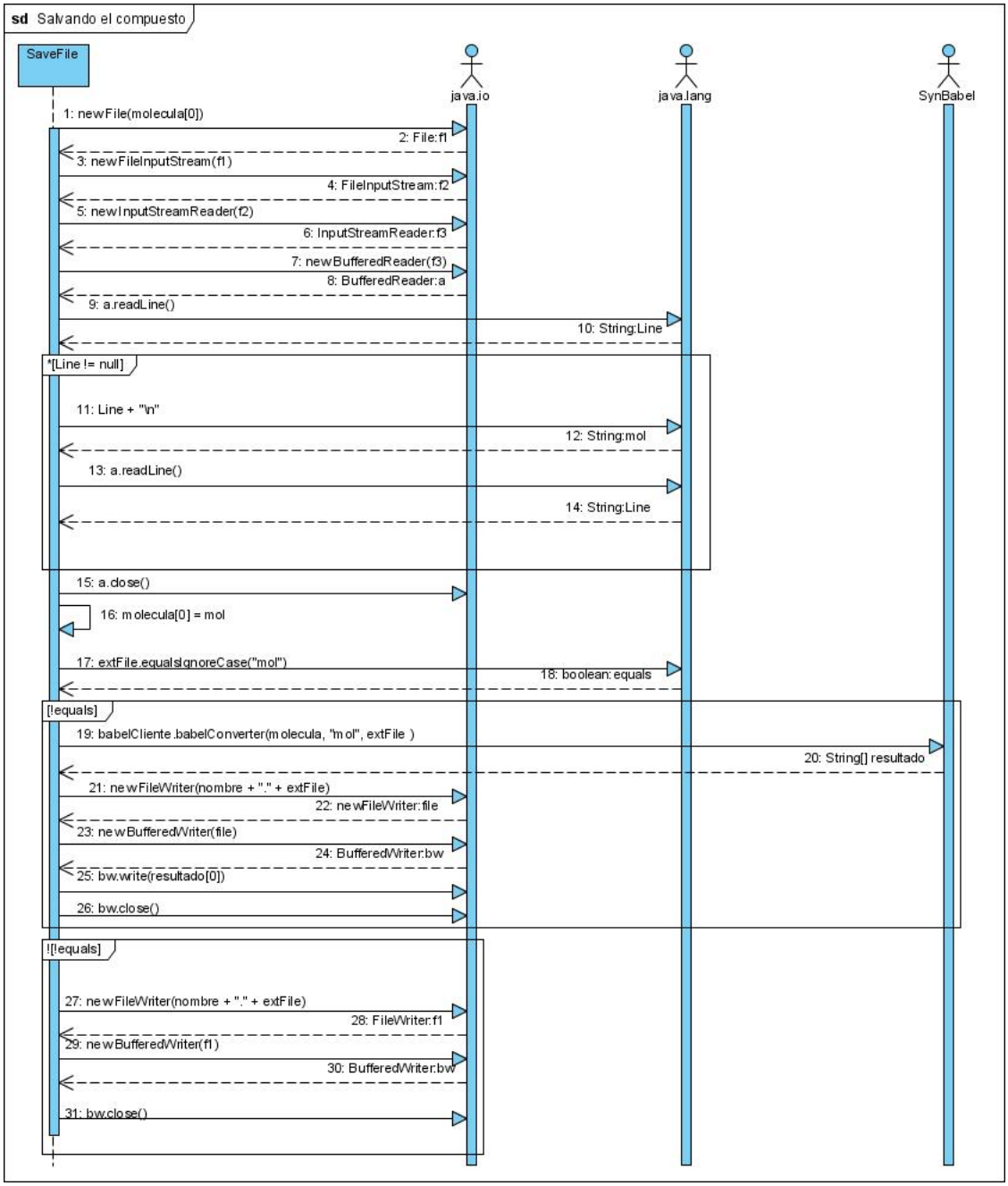
Anexo 10: Diagramas de Secuencias de la Sección Guardar Fichero del caso de uso Gestionar Fichero.



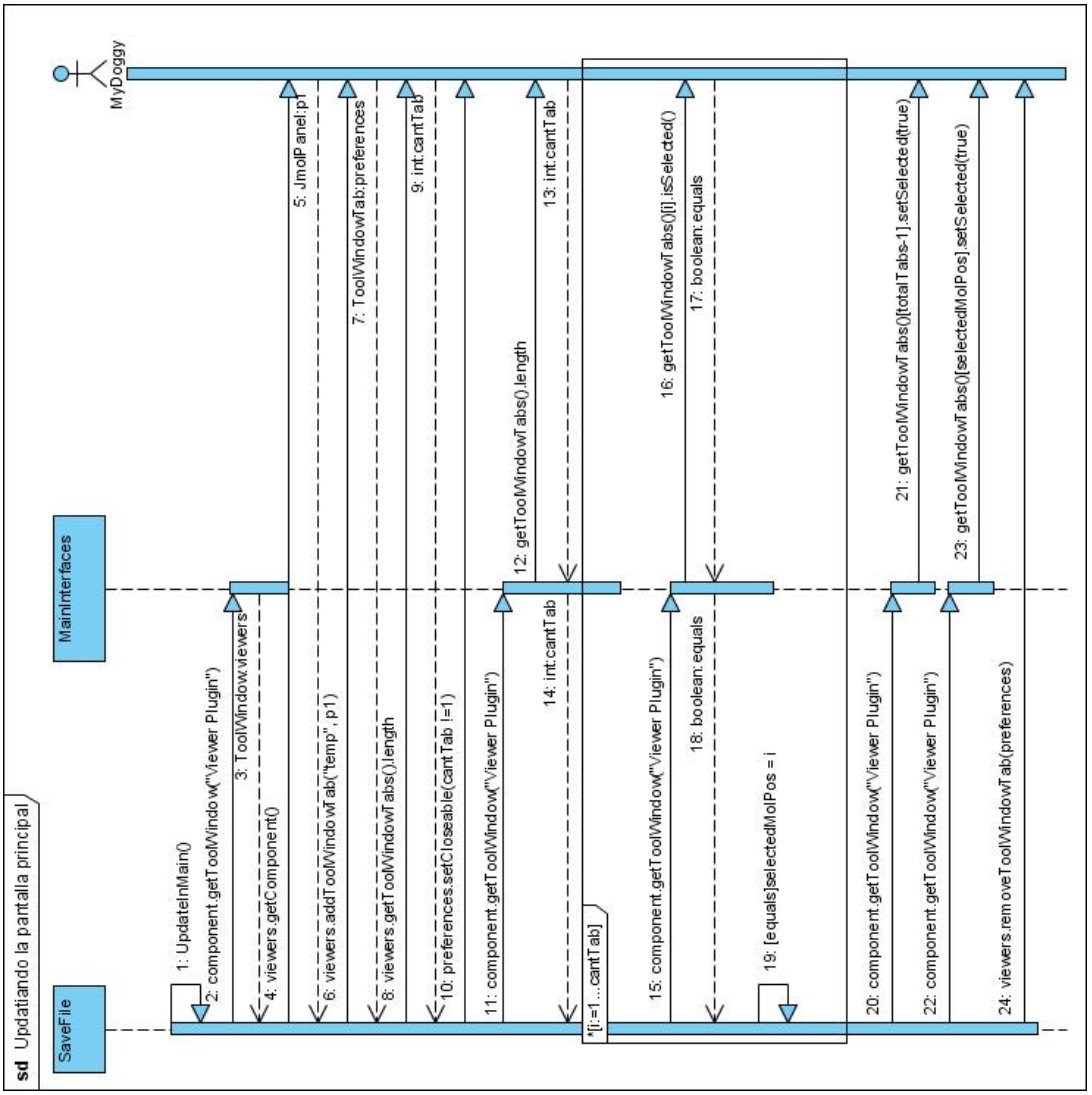
Anexo 10: Diagramas de Secuencias de la Sección Guardar Fichero del caso de uso Gestionar Fichero (Continuación).



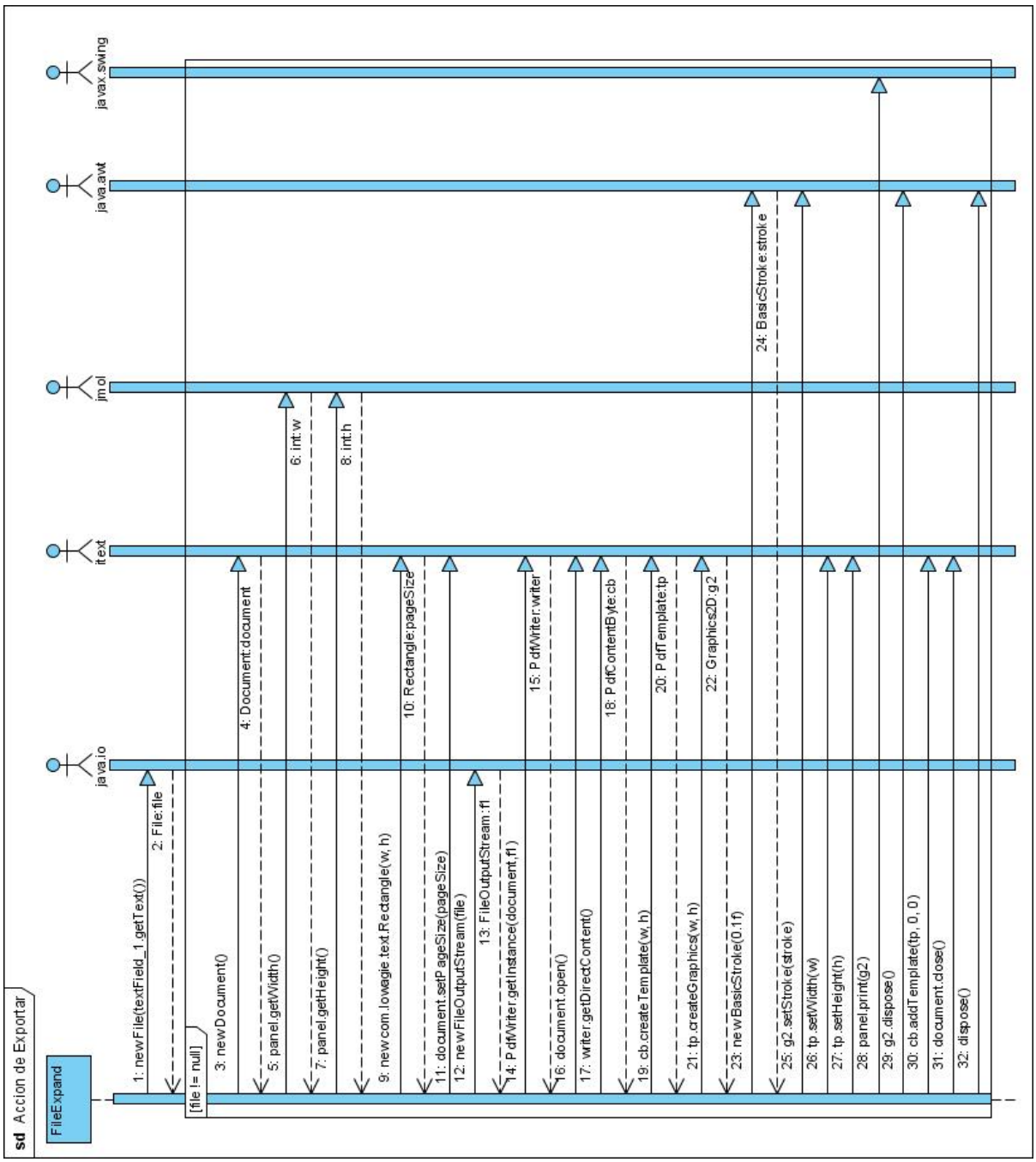
Anexo 10: Diagramas de Secuencias de la Sección Guardar Fichero del caso de uso Gestionar Fichero (Continuación).



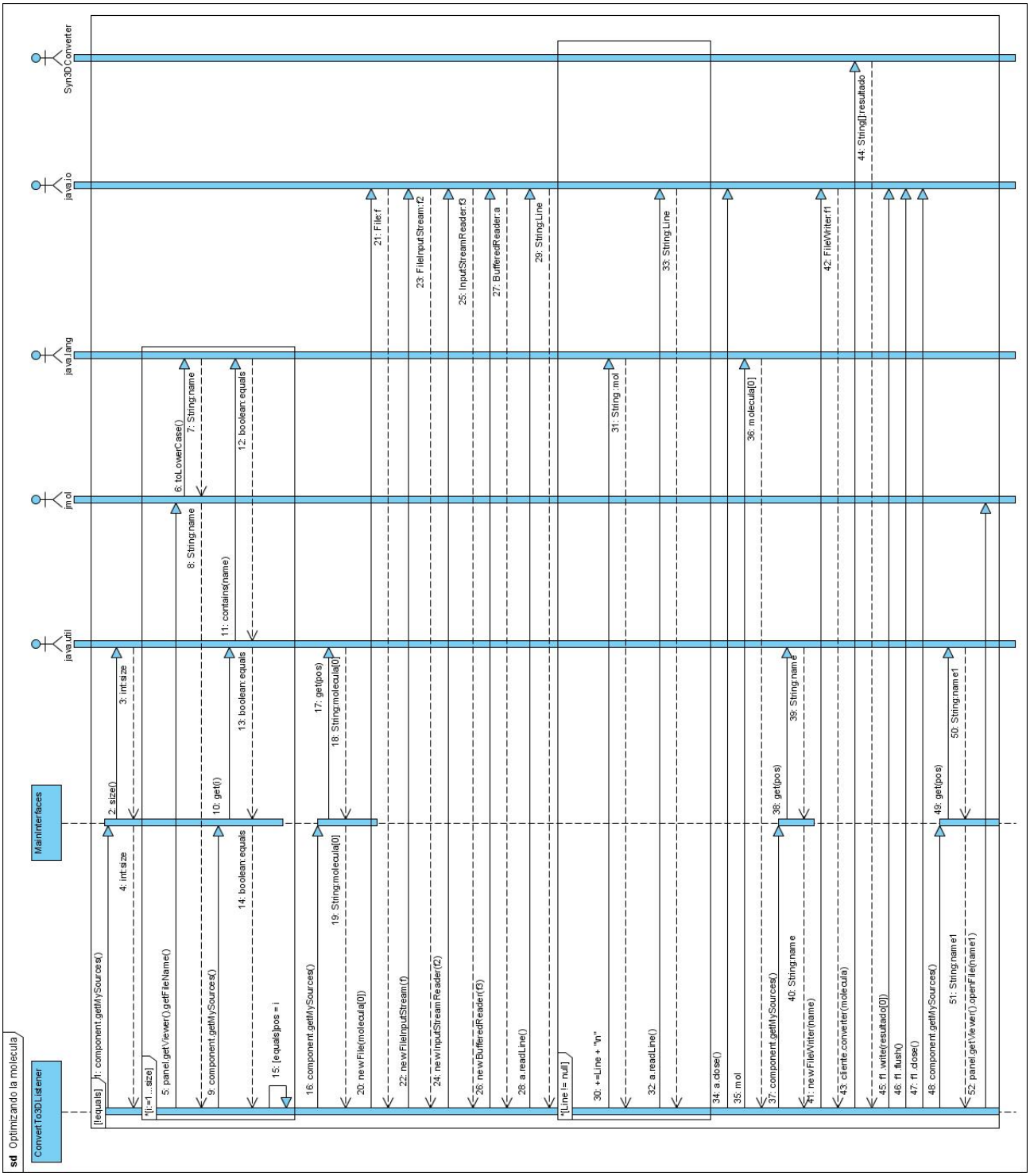
Anexo 10: Diagramas de Secuencias de la Sección Guardar Fichero del caso de uso Gestionar Fichero (Continuación).



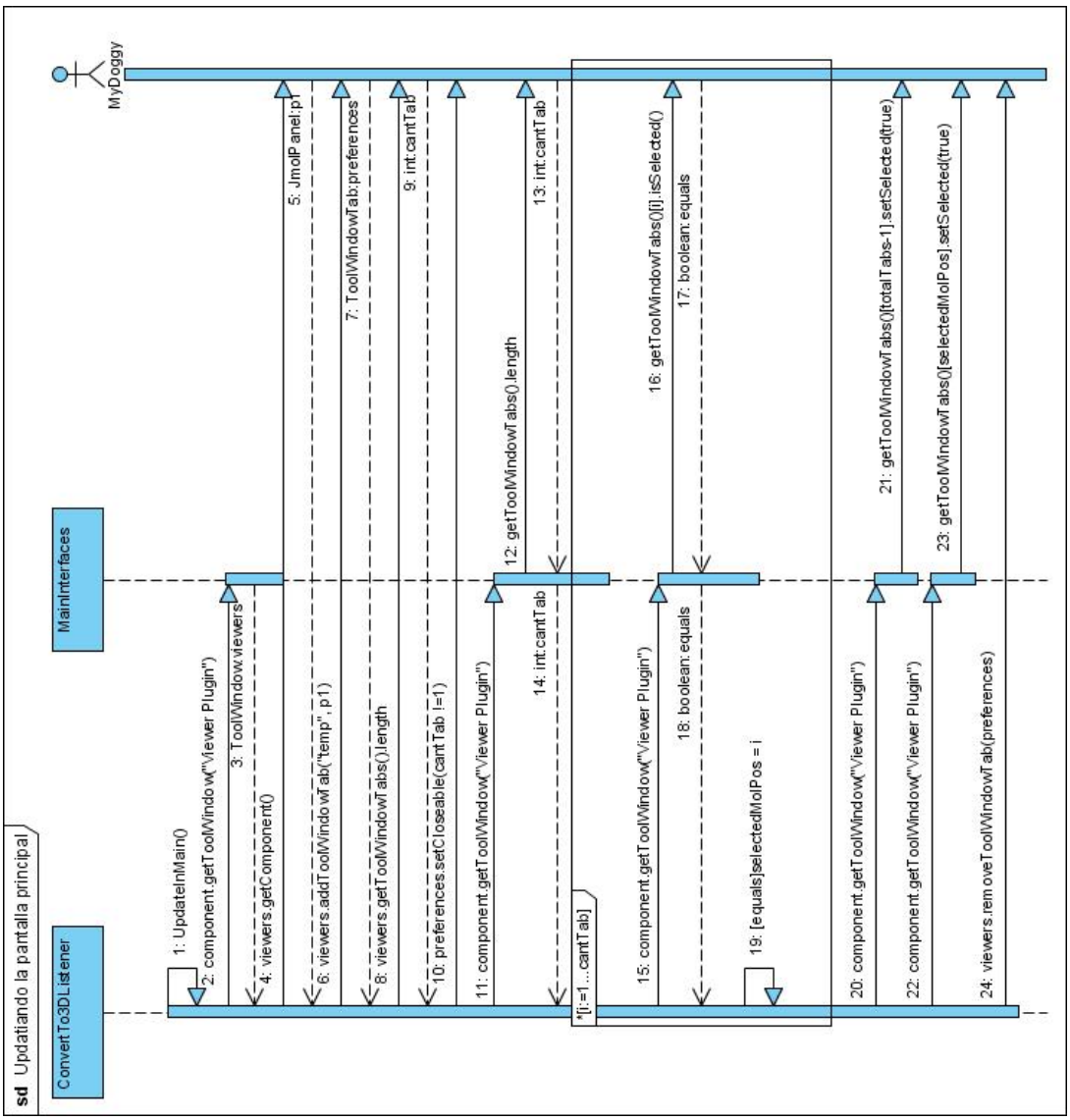
Anexo 11: Diagramas de Secuencias de la Sección Exportar Fichero del caso de uso Gestionar Fichero.



Anexo 12: Diagramas de Secuencias del caso de uso Convertir a 3D.

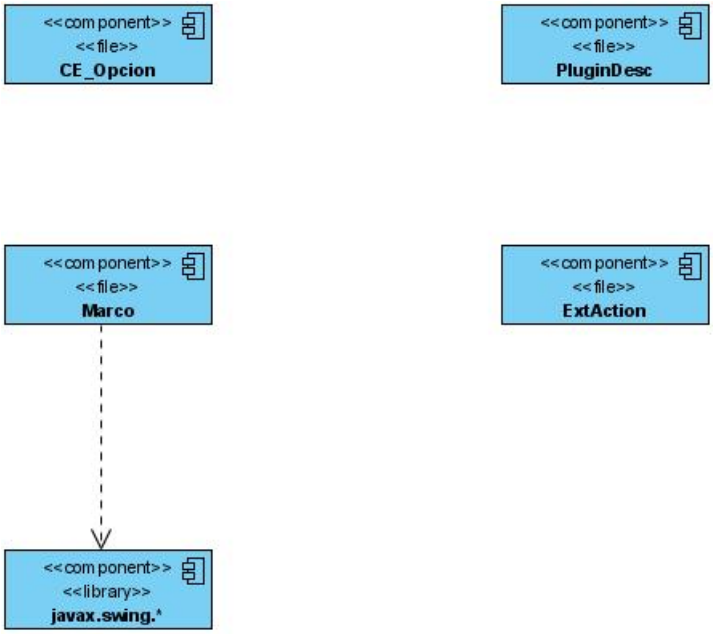


Anexo 12: Diagramas de Secuencias del caso de uso Convertir a 3D (Continuación).

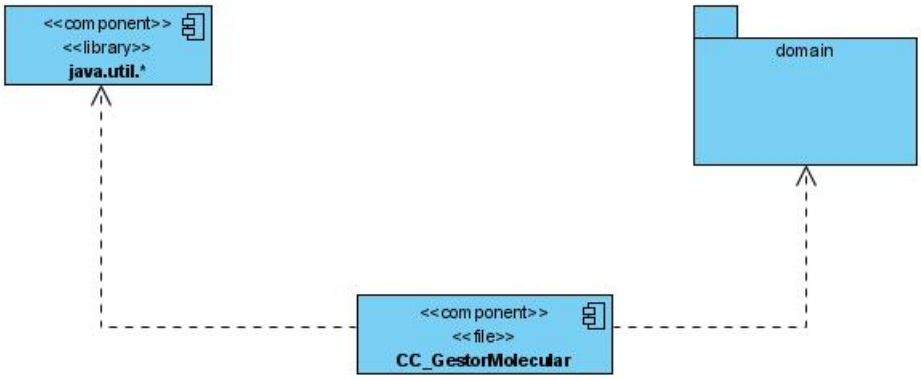


Anexo 13: Diagrama de Componentes del Front-End.

Paquete Domain

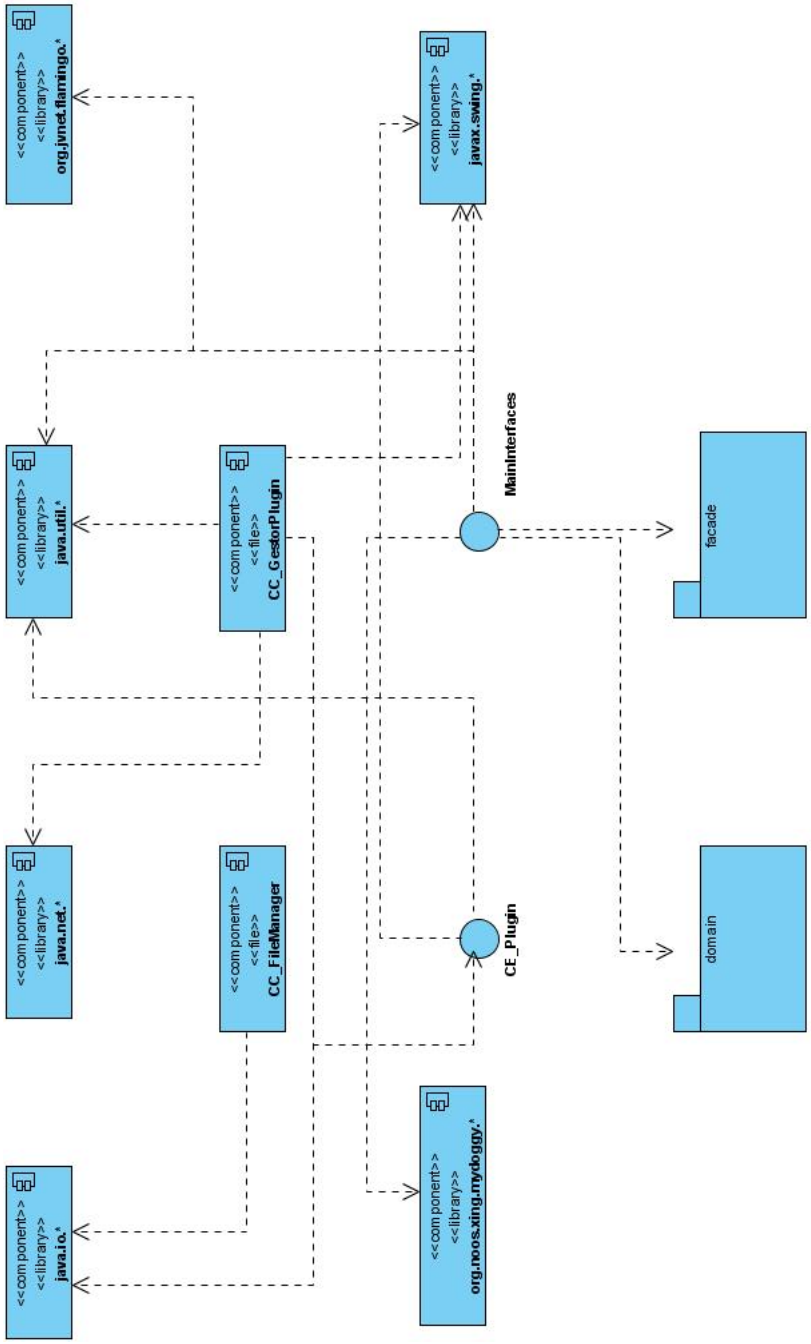


Paquete Facade



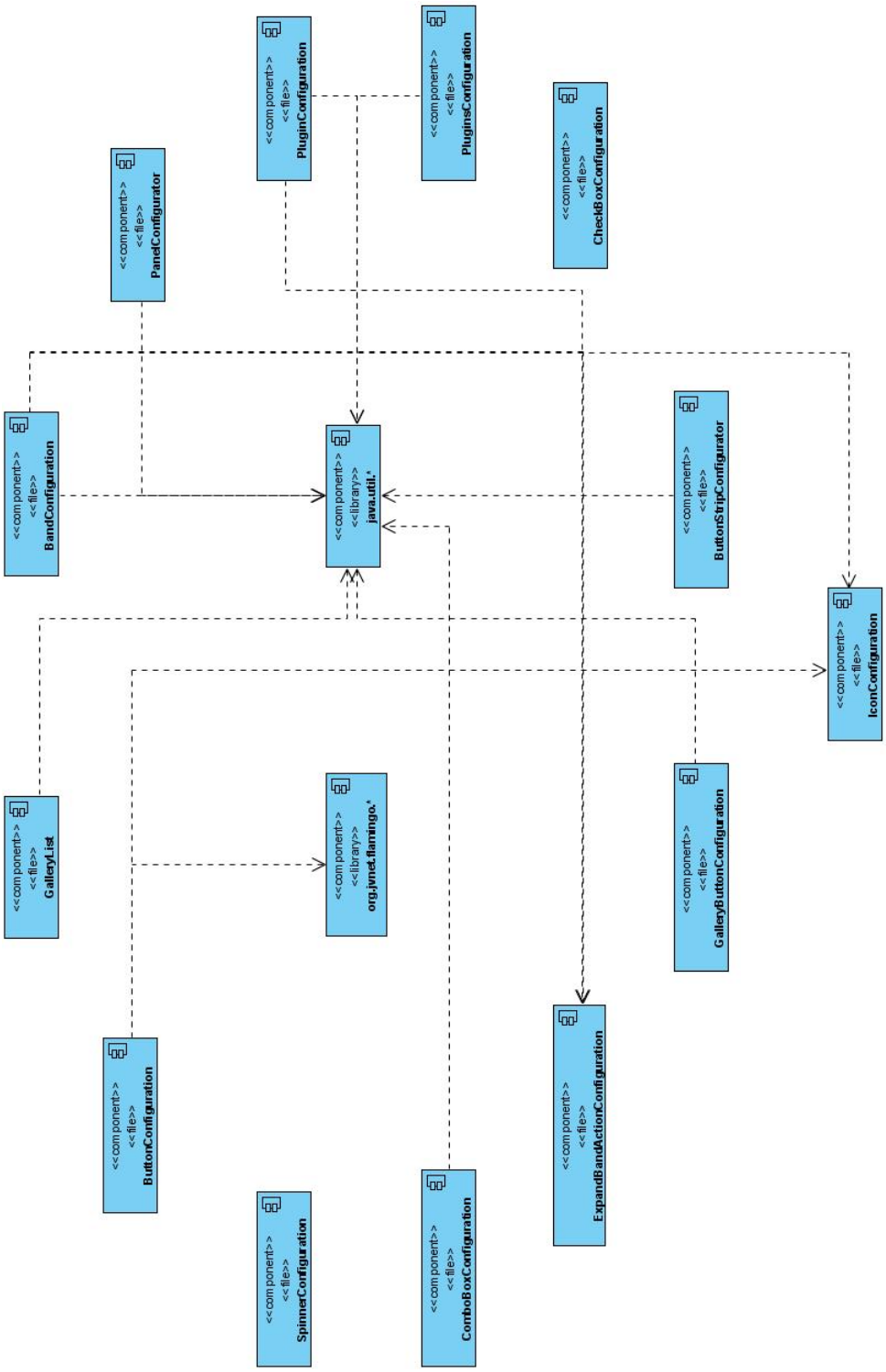
Anexo 13: Diagrama de Componentes del Front-End (Continuación).

Paquete Plug-in



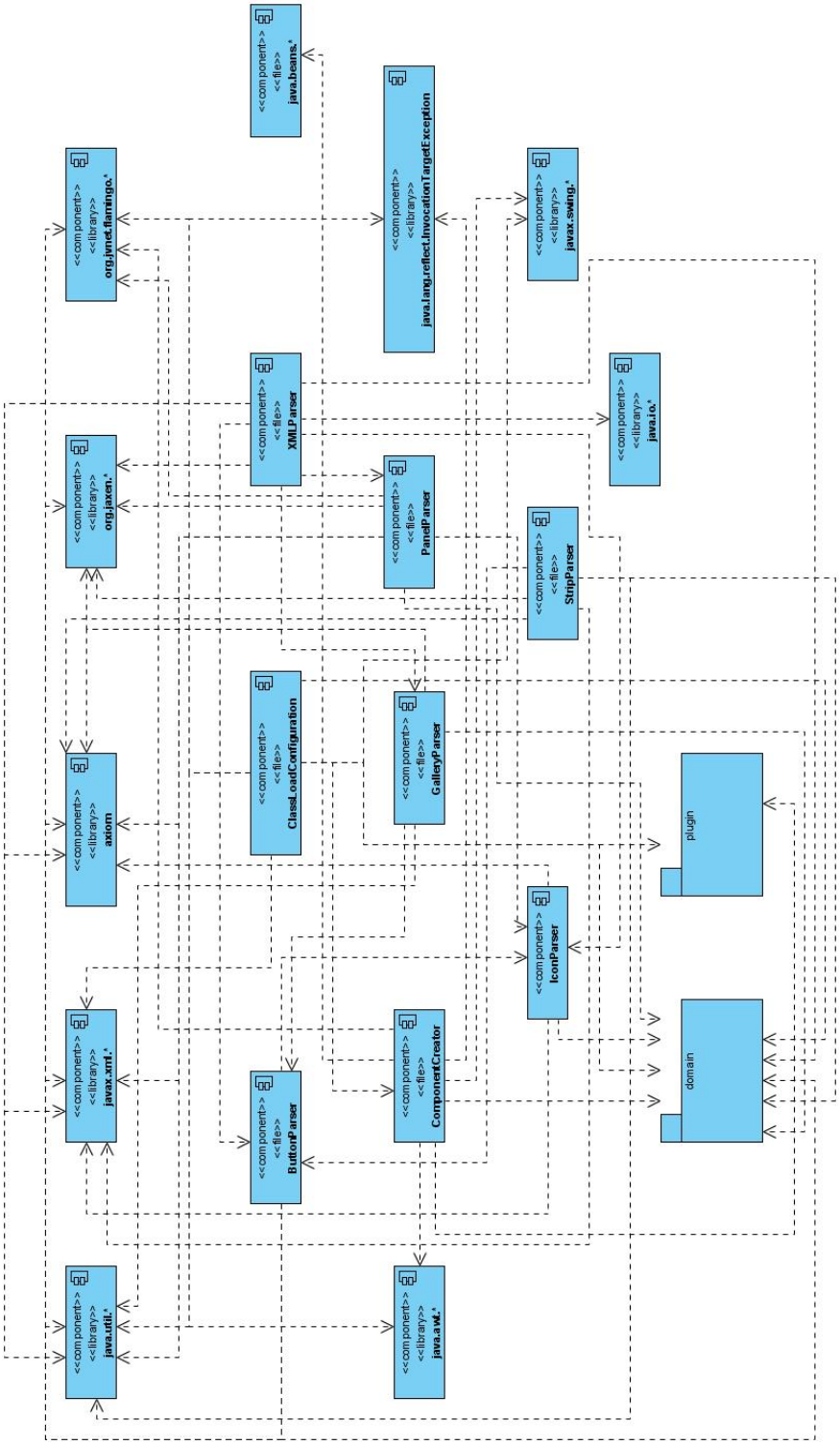
Anexo 13: Diagrama de Componentes del Front-End (Continuación).

Paquete Plug-in-Domain



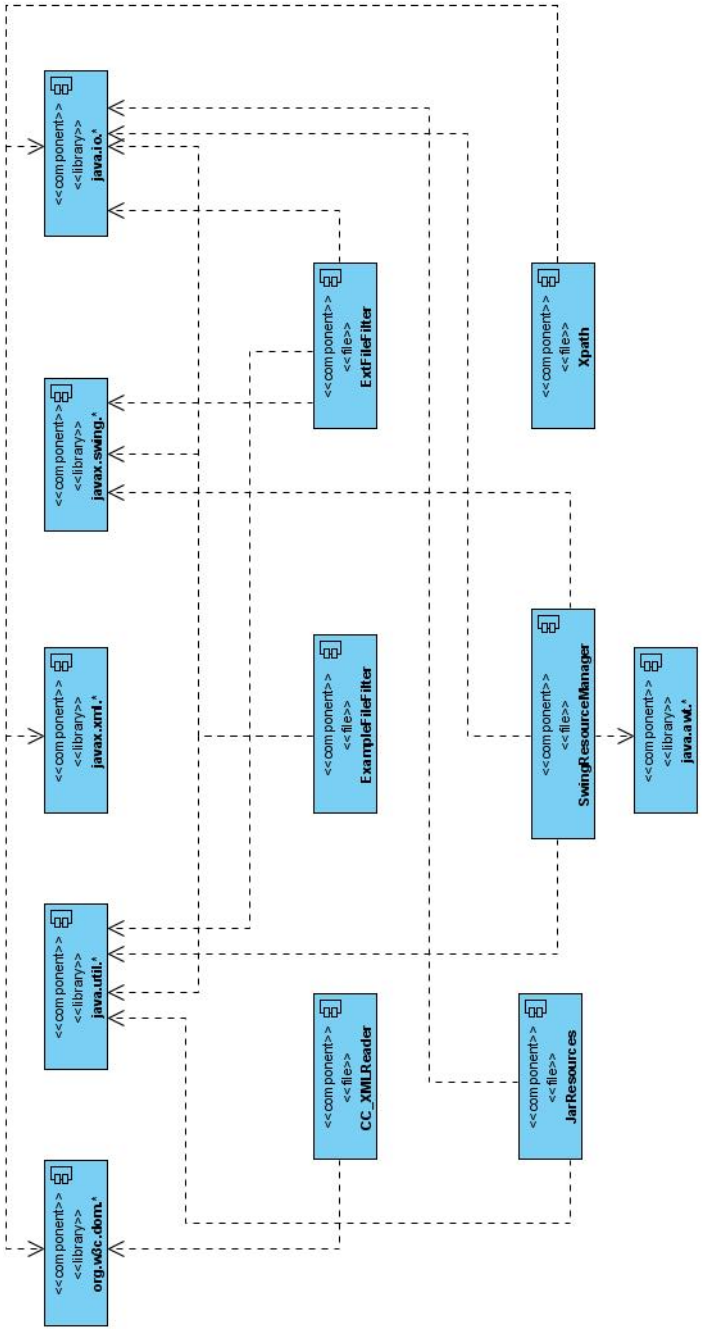
Anexo 13: Diagrama de Componentes del Front-End (Continuación).

Paquete Plug-in-Parser



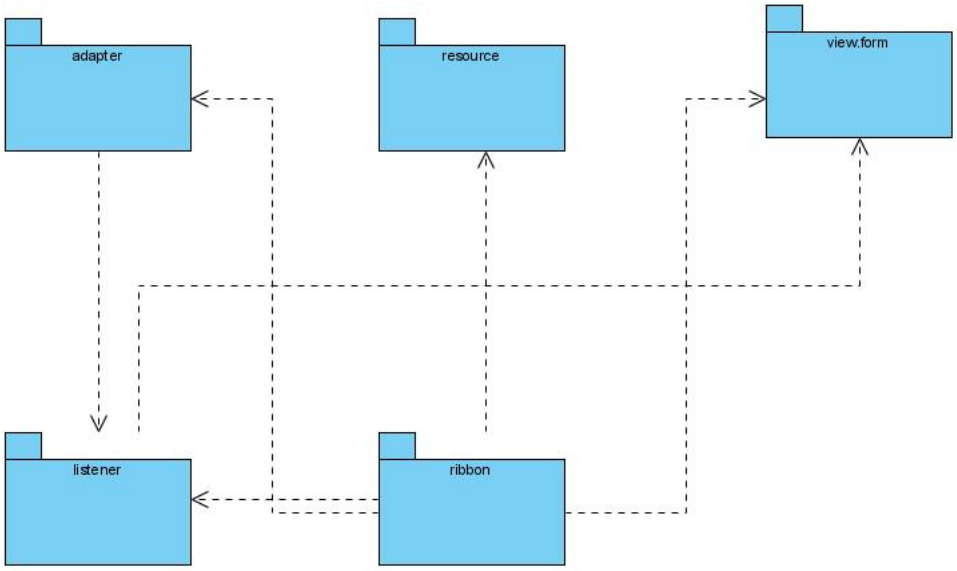
Anexo 13: Diagrama de Componentes del Front-End (Continuación).

Paquete Util

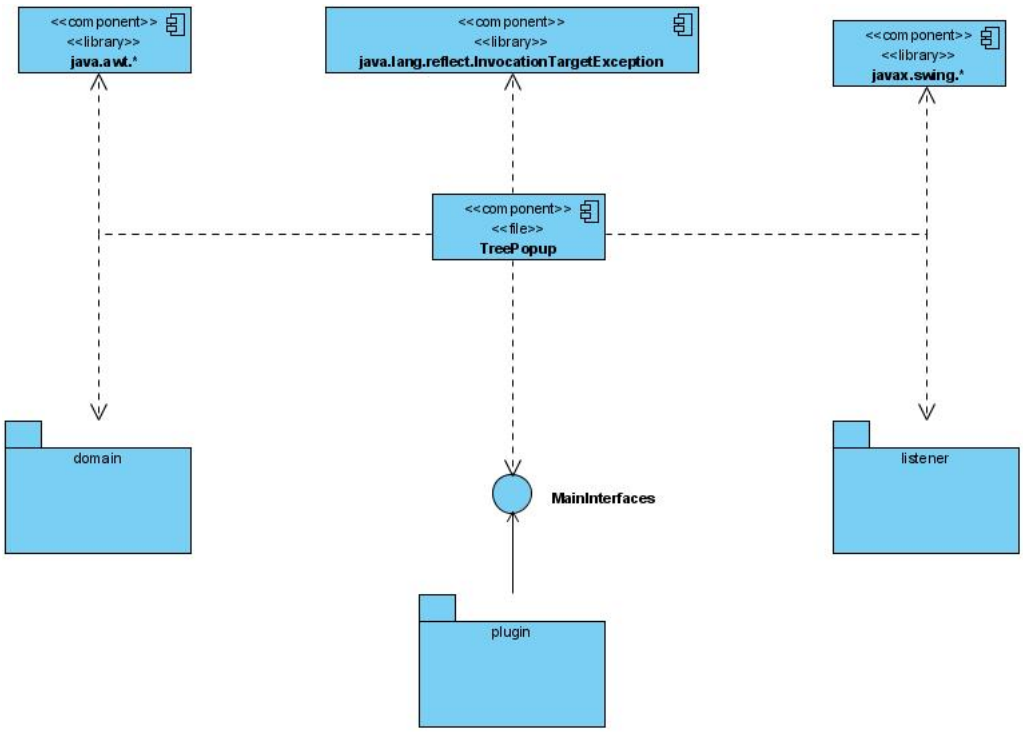


Anexo 13: Diagrama de Componentes del Front-End (Continuación).

Paquete View

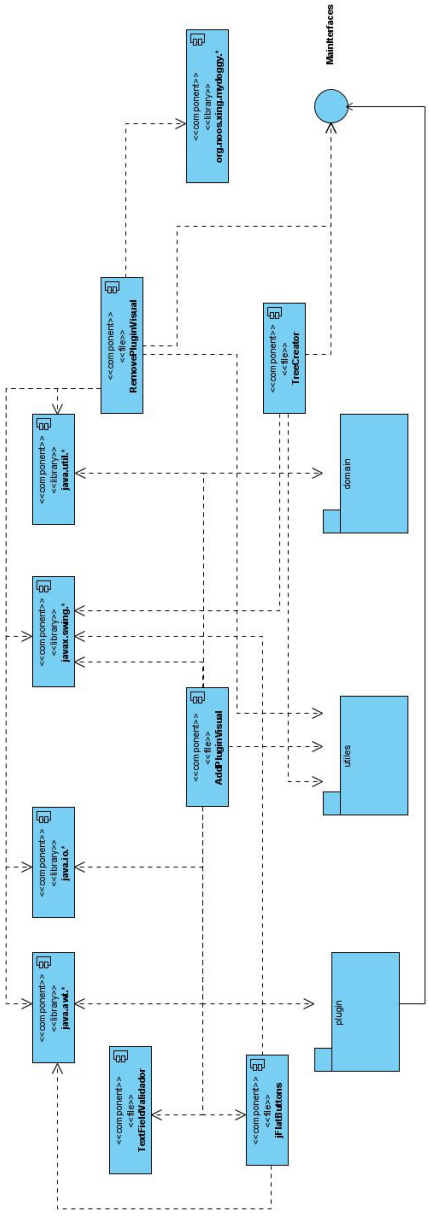


Paquete View-Adapter



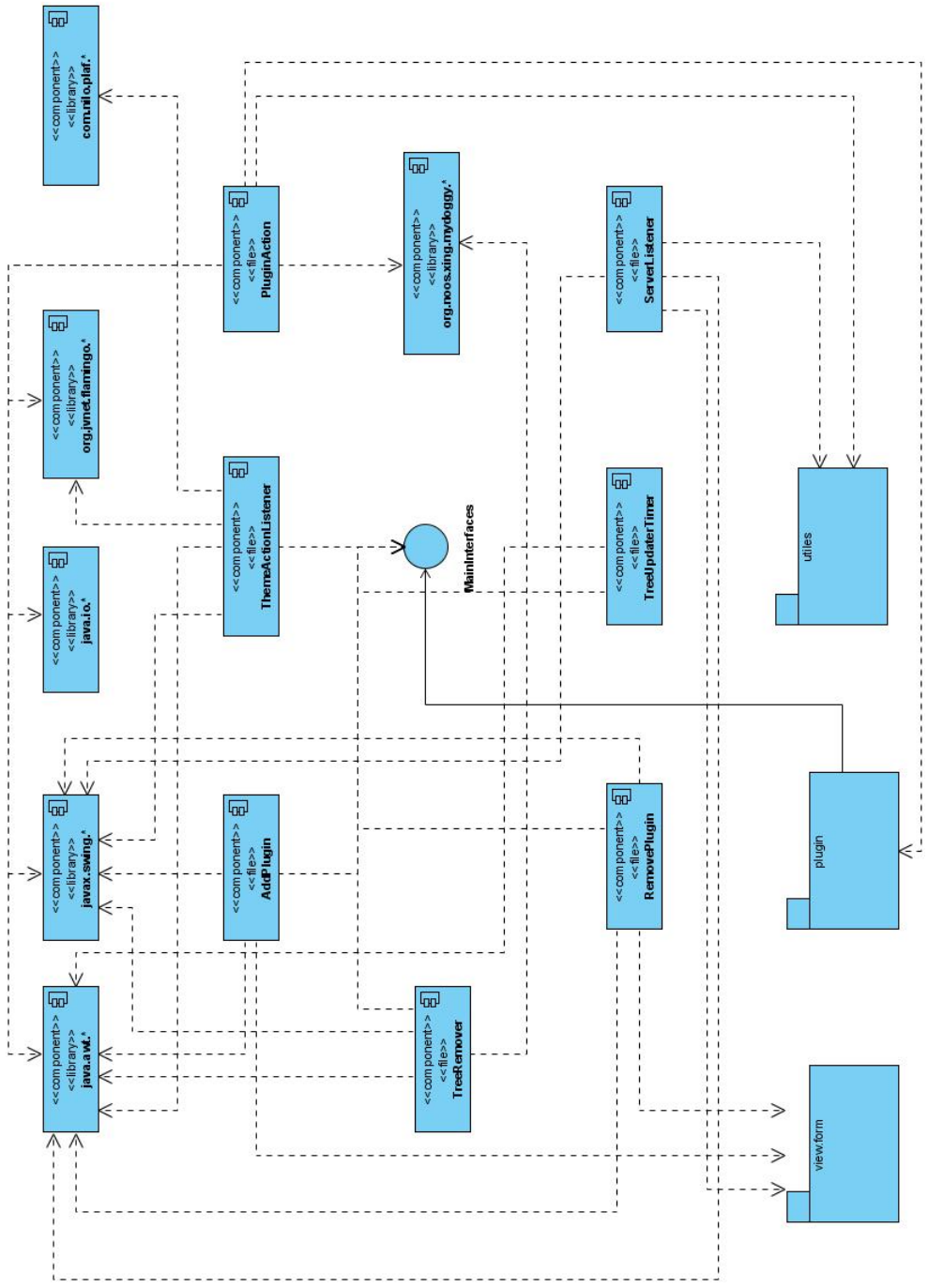
Anexo 13: Diagrama de Componentes del Front-End (Continuación).

Paquete View-Form



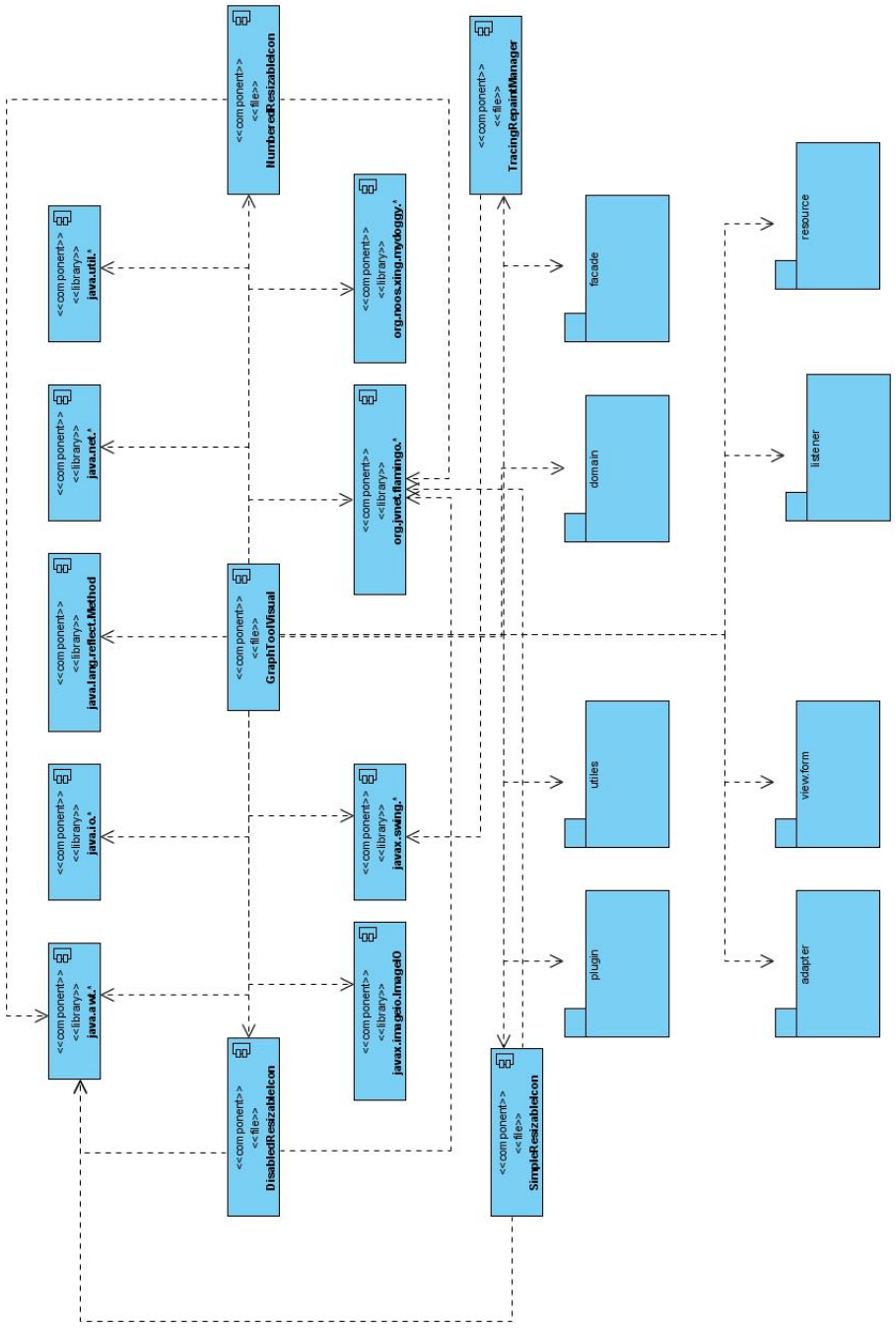
Anexo 13: Diagrama de Componentes del Front-End (Continuación).

Paquete View-Listener



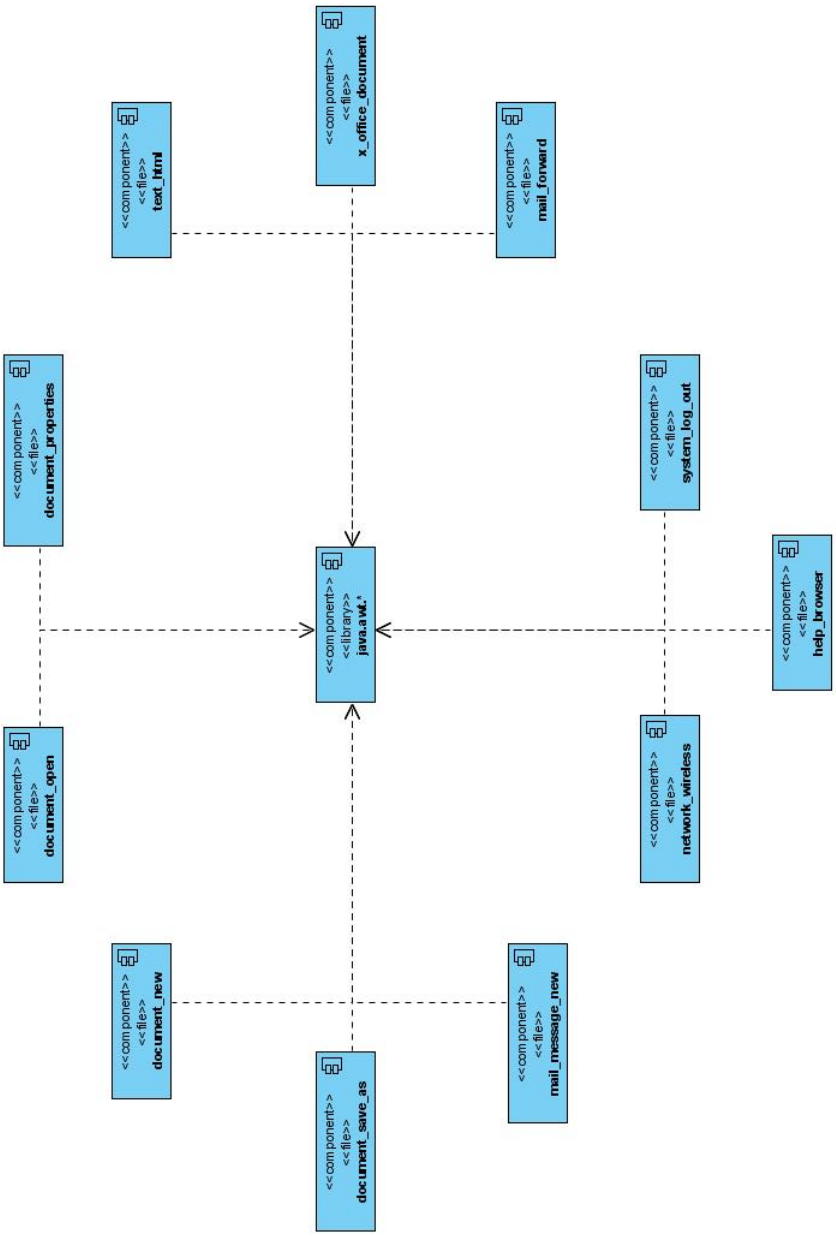
Anexo 13: Diagrama de Componentes del Front-End (Continuación).

Paquete View-Ribbon



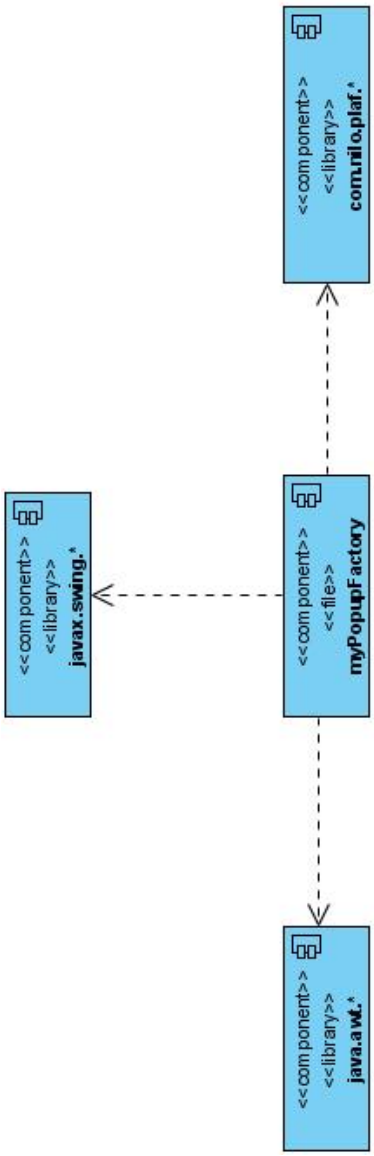
Anexo 13: Diagrama de Componentes del Front-End (Continuación).

Paquete View-SVG-Transcode



Anexo 13: Diagrama de Componentes del Front-End (Continuación).

Paquete View-Util



Anexo 14: Diseño de las pruebas por cada sección del caso de uso Gestionar Plug-ins

Variables

1. - Plug-in alasGRATOViewer

SC1: Adicionar Plug-ins

Id del escenario	**Escenario**	**Var 1**	**Respuesta del Sistema**	**Resultado de la Prueba**
EC 1.1.	Adicionar un plug-in.	V	El sistema adiciona el plug-in a la carpeta de extensiones, muestra el mensaje "Plug-in importado satisfactoriamente" y elimina de la lista de plug-in disponibles el plug-in importado.	El sistema adiciona el plug-in a la carpeta de extensiones y elimina de la lista de plug-in disponibles el plug-in importado.
EC 1.2.	Adicionar un plug-in que no exista.	F	El sistema muestra el mensaje "Debe seleccionar un plug-in".	El sistema muestra el mensaje "Debe seleccionar un plug-in".

SC2: Eliminar Plug-ins

Id del escenario	**Escenario**	**Var 1**	**Respuesta del Sistema**	**Resultado de la Prueba**
EC 2.1.	Eliminar un plug-in.	V	El sistema eliminará totalmente de la plataforma el plug-in indicado.	El sistema elimina el plug-in.

SC3: Activar Plug-ins

Id del escenario	**Escenario**	**Var 1**	**Respuesta del Sistema**	**Resultado de la Prueba**
EC 3.1.	Activar plug-in ya activado.	V	El sistema muestra al especialista este plug-in.	El sistema muestra el plug-in.
EC 3.2.	Activar plug-in na activado.	V	El sistema lo activa creando el TaskRibbon correspondiente al mismo y crea la ventana que contiene el área de trabajo del plug-in ,ubicándola en el lado oeste del sistema.	El sistema activa el plug-in.

Anexo 15: Diseño de las pruebas por cada sección del caso de uso Gestionar Preferencias

Variables

1. - Servidor: 10.34.20.5 puerto: 3389

2. - Alas Green Theme

SC1: Servidor

Id del escenario	**Escenario**	**Var 1**	**Var 2**	**Respuesta del Sistema**	**Resultado de la Prueba**
EC 1.1.	Cambiar preferencias del servidor.	V	F	El sistema recoge los datos introducidos por el especialista, los valida y configura con ellos dos variables globales para el uso de todos los plug-ins de la aplicación.	Se cambia el servidor de servicios de la aplicación.

SC2: Temas de Colores

Id del escenario	**Escenario**	**Var 1**	**Var 2**	**Respuesta del Sistema**	**Resultado de la Prueba**
EC 2.1.	Cambiar temas de colores.	F	V	El sistema realiza el cambio de color de acuerdo a la selección del especialista.	Se cambia de color de acuerdo a la selección del especialista.

Anexo 16: Diseño de las pruebas por cada sección del caso de uso Eliminar fichero

Variables

1. - 143.mol
2. - 2371.mol

SC1: Eliminar Fichero

Id del escenario	**Escenario**	**Var 1**	**Var 2**	**Respuesta del Sistema**	**Resultado de la Prueba**
EC 1.1.	Eliminar fichero existente.	V	F	El sistema busca qué plug-ing se encuentra haciendo uso del fichero a eliminar y lo elimina del este y del árbol de la plataforma.	Se elimina el fichero de todos los plug-ins y del árbol de ficheros cargados.
EC 1.1.	Eliminar fichero existente.	F	V	El sistema busca qué plug-ing se encuentra haciendo uso del fichero a eliminar y lo elimina del este y del árbol de la plataforma.	Se elimina el fichero de todos los plug-ins y del árbol de ficheros cargados.

Anexo 17: Diseño de las pruebas por cada sección del caso de uso Abrir Fichero

Variables

1. - 143.mol
2. - 2371.mol
3. - pd1.pdb

SC1: Abrir

Id del escenario	**Escenario**	**Var 1**	**Var 2**	**Var 3**	**Respuesta del Sistema**	**Resultado de la Prueba**
EC 1.1.	Abrir una o varias moléculas o Proteínas.	V	F	F	El sistema verifica si la extensión es distinta de mol o pdb, de ser así, le solicita al servicio de conversión de formato que convierta el fichero a mol, lo copia en la carpeta tmp y lo adiciona al final del árbol de ficheros y a la lista de ficheros recientes y por último lo visualiza en el área de visualización.	Se carga el fichero y se visualiza en el área de visualización.

Id del escenario	**Escenario**	**Var 1**	**Var 2**	**Var 3**	**Respuesta del Sistema**	**Resultado de la Prueba**
EC 1.1.	Abrir una o varias moléculas o Proteínas.	V	V	F	El sistema verifica si la extensión es distinta de mol o pdb, de ser así, le solicita al servicio de conversión de formato que convierta los ficheros a mol, los copia en la carpeta tmp, los adiciona al final del árbol de ficheros cargados. y por último adiciona los 10 últimos en la lista de ficheros recientes.	Se cargan los ficheros y se adicionan al árbol de ficheros cargados y los 10 útimos son adicionados a la lista de ficheros recientes.

Id del escenario	Escenario	Var 1	Var 2	Var 3	Respuesta del Sistema	Resultado de la Prueba
EC 1.1.	Abrir una o varias moléculas o Proteínas.	V	V	V	El sistema verifica si la extensión es distinta de mol o pdb, de ser así, le solicita al servicio de conversión de formato que convierta los ficheros a mol, los compia en la carpeta tmp, los adiciona al final del árbol de ficheros cargados. y por último adiciona los 10 últimos en la lista de ficheros recientes.	Se cargan los ficheros y se adicionan al árbol de ficheros cargados y los 10 útimos son adicionados a la lista de ficheros recientes.

SC2: Abrir Reciente

Id del escenario	Escenario	Var 1	Var 2	Var 3	Respuesta del Sistema	Resultado de la Prueba
EC 2.1.	Abrir una fichero reciente.	V	F	F	El sistema carga el fichero seleccionado y lo visualiza en el área de visualización.	Se carga el fichero y se visualiza en el área de visualización.

Id del escenario	Escenario	Var 1	Var 2	Var 3	Respuesta del Sistema	Resultado de la Prueba
EC 2.1.	Abrir una fichero reciente.	F	V	F	El sistema carga el fichero seleccionado y lo visualiza en el área de visualización.	Se carga el fichero y se visualiza en el área de visualización.
EC 2.1.	Abrir una fichero reciente.	F	F	V	El sistema carga el fichero seleccionado y lo visualiza en el área de visualización.	Se carga el fichero y se visualiza en el área de visualización.

Variable

1. - Fichero 1.fpv
2. - Fichero 2.fpv

SC3: Abrir Preferencias de Visualización

Id del escenario	Escenario	Var 1	Var 2	Respuesta del Sistema	Resultado de la Prueba
EC 3.1.	Abrir un fichero de Preferencia de Visualización.	V	F	El sistema verifica si es un fichero de Preferencias de Visualización, carga las Preferencias de Visualización seleccionada y visualiza las preferencias en el área de visualización.	Se carga el fichero y se visualizan las preferencias e el área de trabajo.

Id del escenario	**Escenario**	**Var 1**	**Var 2**	**Respuesta del Sistema**	**Resultado de la Prueba**
EC 3.1.	Abrir un fichero de Preferencias de Visualización que no exista.	F	V	El sistema muestra un mensaje de error “Error en selección de tipo de fichero.”	Se muestra un mensaje de error “Error en selección de tipo de fichero.”

Anexo 18: Diseño de las pruebas por cada sección del caso de uso Gestionar Fichero

Variables

1. - 143.mol
2. - 2371.mol
3. - pd1.pdb

SC1: Cerrar Fichero

Id del escenario	**Escenario**	**Var 1**	**Var 2**	**Var 3**	**Respuesta del Sistema**	**Resultado de la Prueba**
EC 1.1.	Cerrar una moléculas o proteínas.	V	F	F	El sistema cierra el fichero que esté visualizado sin quitarlo de la lista de ficheros abiertos.	Se cierra el fichero que esta visaulizado, sin quitarlo de la lista de ficheros abiertos.
EC 1.1.	Abrir una o varias moléculas o Proteínas.	V	V	F	El sistema cierra todos los ficheros visualizados sin quitarlos de la lista de ficheros abiertos.	Se cierran los ficheros que estan visaulizados, sin quitarlos de la lista de ficheros abiertos.

Id del escenario	Escenario	Var 1	Var 2	Var 3	Respuesta del Sistema	Resultado de la Prueba
EC 1.1.	Abrir una o varias moléculas o Proteínas.	V	V	V	El sistema cierra todos los ficheros visualizados sin quitarlos de la lista de ficheros abiertos.	Se cierran los ficheros que estan visaulizados, sin quitarlos de la lista de ficheros abiertos.

SC2: Guardar Fichero

Id del escenario	Escenario	Var 1	Var 2	Var 3	Respuesta del Sistema	Resultado de la Prueba
EC 2.1.	Guardar.	V	F	F	El sistema verifica si la extensión es distinta de mol y utiliza el servicio de conversión de formato para convertir a la extensión especificada y se guarda en la dirección especificada por el usuario.	Se guarda el fichero seleccionado.

Id del escenario	**Escenario**	**Var 1**	**Var 2**	**Var 3**	**Respuesta del Sistema**	**Resultado de la Prueba**
EC 2.1.	Guardar.	V	V	F	El sistema verifica si la extensión es distinta de mol y utiliza el servicio de conversión de formato para convertir a la extensión especificada y se guarda en la dirección especificada por el usuario.	Se guarda el fichero seleccionado.
EC 2.1.	Guardar.	V	V	V	El sistema verifica si la extensión es distinta de mol y utiliza el servicio de conversión de formato para convertir a la extensión especificada y se guarda en la dirección especificada por el usuario.	Se guarda el fichero seleccionado.

SC3: Exportar Fichero

Id del escenario	**Escenario**	**Var 1**	**Var 2**	**Var 3**	**Respuesta del Sistema**	**Resultado de la Prueba**
EC 3.1.	Exportar a PDF.	V	F	F	El sistema guarda el fichero en el destino indicado, con el nombre y extensión especificada.	Se exporta a pdf el fichero seleccionado.
EC 3.1.	Exportar a PDF.	F	V	F	El sistema guarda el fichero en el destino indicado, con el nombre y extensión especificada.	Se exporta a pdf el fichero seleccionado.
EC 3.1.	Exportar a PDF.	F	F	V	El sistema guarda el fichero en el destino indicado, con el nombre y extensión especificada.	Se exporta a pdf el fichero seleccionado.

Anexo 19: Diseño de las pruebas por cada sección del caso de uso Convertir a 3D

Variables

1. - 143.mol
2. - 2371.mol

SC1: Convertir a 3D

Id del escenario	**Escenario**	**Var 1**	**Var 2**	**Respuesta del Sistema**	**Resultado de la Prueba**
EC 1.1.	Convertir a 3D.	V	F	El sistema envía la molécula seleccionada al servicio de conversión en 3D, recibe la molécula convertida del servicio y la visualiza en el área de visualización.	Se visualiza la molecula en 3D en el área de trabajo.
EC 1.1.	Convertir a 3D.	F	V	El sistema envía la molécula seleccionada al servicio de conversión en 3D, recibe la molécula convertida del servicio y la visualiza en el área de visualización.	Se visualiza la molecula en 3D en el área de trabajo.

Glosario de Términos

Applet: Es un componente de software, que corre en el contexto de otro programa, por ejemplo un nave-gador web. El applet debe correr en un contenedor, que es proporcionado por un programa anfitrión, mediante un plug-in, o en aplicaciones como teléfonos celulares que soportan el modelo de programación por applets.

Arquitectura Informacional: Es la disciplina y arte encargada del estudio, análisis, organización, disposición y estructuración de la información en espacios de información, y de la selección y presentación de los datos en los sistemas de información interactivos y no interactivos.

Bioinformática: Es la aplicación de los ordenadores y los métodos informáticos en el análisis de datos experimentales y simulación de los sistemas biológicos.

CASE: Computer Aided Software Engineering (Herramientas de ingeniería de software asistida por computadora).

Compuestos Orgánicos: Sustancias químicas basadas en cadenas de carbono e hidrógenos. En muchos casos contienen oxígenos, nitrógenos, azufres, fósforos y halógenos.

Especialista: Persona capacitada para interactuar con la aplicación.

GUI: Graphic User Interface o Interfaz Gráfica de Usuario.

Información Estructural: Dato o descripción que permite conocer cómo está formado el compuesto en términos de átomos y la relación espacial que existe entre ellos.

Interacción Persona-Ordenador: Es la disciplina que estudia el intercambio de información entre las personas y los ordenadores.

Interfaz de Usuario: Es el medio que el usuario utiliza para comunicarse con una máquina, un equipo o una computadora, y comprende todos los puntos de contacto entre el usuario y el equipo, normalmente suelen ser fáciles de entender y fáciles de accionar.

Plug-ins: Aplicación informática que interactúa con otra aplicación para aportarle una función o utilidad específica.

Shell: Clase básica para todas las aplicaciones visuales de Java.

UCI: Universidad de las Ciencias Informáticas.

Usabilidad: Es la característica de un sistema que pretende ser utilizado por: el tipo o tipos específicos de usuario/s, la tarea o tareas que para las cuales el sistema se ha hecho, y el contexto en el que se da la interacción.

Widgets: Pequeña aplicación o programa, usualmente presentado en archivos o ficheros pequeños que son ejecutados por un motor de Widgets o Widget Engine. Entre sus objetivos están los de dar fácil acceso a funciones frecuentemente usadas y proveer de información visual.

XML: Metalenguaje extensible de etiquetas, desarrollado por el World Wide Web Consortium (W3C).