Universidad de las Ciencias Informáticas
Facultad de Bioinformática

Título: Interfaz para la visualización y edición de estructuras químicas.

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autores

Yunier René Pérez Valdés

Yanet del Carmen Bravo Rodríguez

Tutores

Dr. Ramón Carrasco Velar

MSc. Aurelio Antelo Collado

MSc. Yanet Villanueva Armenteros

Julio de 2007

“...Todo hombre debe decidir una vez en la vida
si se lanza a triunfar arriesgándolo todo,
o se sienta a contemplar el paso de los triunfadores...”
Benedetti

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas y al Centro de Química Farmacéutica los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Yunier René Pérez Valdés Yanet del Carmen Bravo Rodríguez

_____

Firma del Autor Firma del Autor

MSc. Aurelio Antelo Collado Dr. Ramón Carrasco Velar

_____

Firma del Tutor Firma del Tutor

MSc. Yanet Villanueva Armenteros

_____

Firma del Tutor

AGRADECIMIENTOS

A nuestros padres, por su constante apoyo y dedicación; por su ejemplo de superación incansable; por lo que hemos sido y seremos.

A nuestros tutores por su asesoramiento científico, por su disposición permanente e incondicional en aclarar nuestras dudas y por sus substanciales sugerencias durante la redacción de la Tesis.

A nuestros amigos, tratar de enlistar a todos y cada una de ellos significaría exponernos al riesgo de omitir a alguien importante - todos lo son -, y por ello expresamos aquí nuestro reconocimiento y gratitud a todos.

A la Revolución Cubana, por habernos permitido formar parte de sus recintos universitarios y posteriormente darnos la oportunidad de replicar lo allí aprendido.

DEDICATORIA

A aquellos cuyas vidas puedan ser cambiadas por el resultado de esta investigación. A todos sin distinción de raza, religión, sexo, condición política y social. Respetamos la vida y fomentamos la oportunidad que tenemos de dar y aprender.

RESUMEN

Esta investigación surge en el marco de trabajo del Proyecto “Plataforma Inteligente para la Predicción de Actividad Biológica de Compuestos Orgánicos”, que se desarrolla conjuntamente por el Centro de Química Farmacéutica y la Facultad de Bioinformática de la Universidad de las Ciencias Informáticas, para el cual se han implementado un conjunto de módulos que trabajan independientes. Estos módulos necesitan a su vez de una interfaz amigable con el usuario que permita acceder a cada uno de los módulos del sistema, así como visualizar y editar las estructuras químicas. Se desarrolló una aplicación gráfica que permite visualizar en 3D y editar en 2D las estructuras de moléculas orgánicas, así como cargarlas en diferentes formatos. A esta aplicación se le han incorporado diferentes funcionalidades como son el “Cálculo de las masas moleculares” y el “Cálculo de composición elemental”. También se implementó un “Gestor de plug-ins” para las diferentes funcionalidades de la plataforma, una “Herramienta para generación de archivos de manifiesto” y la posibilidad de crear un “Fichero de almacenamiento de preferencias”. Para garantizar la portabilidad de la aplicación esta se desarrolló bajo ambiente multiplataforma, utilizando fundamentalmente herramientas de software libre.

PALABRAS CLAVE

Visualización molecular, edición molecular,

TABLA DE CONTENIDOS

INTRODUCCIÓN

El descubrimiento de nuevas moléculas de interés farmacéutico está estrechamente relacionado con la búsqueda de información en grandes bases de datos y con la determinación y estimación de estructuras, es por eso que el desarrollo de aplicaciones informáticas para extraer conocimiento de la información generada en los laboratorios se manifiesta como una de las necesidades importantes en la Bioinformática.

Gran cantidad de información experimental se obtiene de los laboratorios por los especialistas, pero para poder compartirla, los investigadores necesitan de un lenguaje común. Por excelencia, la visualización molecular es la mejor manera de comunicación entre los químicos y especialistas afines, los cuales, independientemente de la lengua que dominen, pueden comunicarse a través de los diversos diagramas y visualizaciones de las entidades químicas que proveen las aplicaciones informáticas existentes.

En el mundo existen numerosas de estas aplicaciones para la visualización y edición de estructuras químicas, como por ejemplo el CORINA, el HyperChem, las que poseen múltiples funcionalidades. Estos programas comerciales no tienen en cuenta la predicción de actividades biológicas o el cálculo de descriptores por mencionar algunas limitaciones, sino que se centran en el objetivo para el cual están concebidos, y la edición y visualización de moléculas es solamente una facilidad para los usuarios. Otros como el Dragon y el Codessa, destinados al cálculo de descriptores, carecen de la interfaz de visualización y edición molecular, sin tener en cuenta que eso permitiría un trabajo más cómodo a los usuarios lo cual constituye una desventaja para los mismos.

En la actualidad, la industria químico-farmacéutica, en asociación con las universidades, emplean los métodos de diseño computacional de fármacos como el procedimiento de elección en el desarrollo de nuevos medicamentos. Esto ha promovido a su vez el surgimiento de diferentes métodos y programas que contribuyen a cumplir esos objetivos. Esta política tiene en su mira principal, la disminución de los costos y el tiempo del proceso de investigación. Para poder alcanzar la competitividad frente a esta política actual de los grandes consorcios farmacéuticos, se necesita disponer de un sistema para el tamizaje virtual de compuestos orgánicos con vistas a disminuir también los costos de la investigación-desarrollo de nuevos fármacos, en su fase inicial.

Esta situación condujo al surgimiento del proyecto conjunto CQF-UCI “Plataforma Inteligente para la Predicción de Actividad Biológica de Compuestos Orgánicos”, concebido como aplicación modular multiplataforma, con capacidades para realizar tamizaje virtual de moléculas orgánicas empleando

técnicas de inteligencia artificial para la predicción de actividad biológica, con modelos desarrollados a partir de una base de datos de compuestos con actividad reportada. Pero, teniendo en cuenta lo expuesto anteriormente acerca de las formas de comunicación entre especialistas en química medicinal u otras especialidades afines, así como la necesidad de que los usuarios de dicha Plataforma puedan acceder a los diferentes módulos del sistema, se plantea como **problema científico** de esta investigación: ¿Cómo garantizar que los especialistas que interactúan con la plataforma puedan lograr una mejor utilización de la misma?

La estructura de una molécula puede ser representada de diversas formas, ya sea por su fórmula empírica, fórmula estructural, por su peso molecular, por lenguajes descriptores o códigos que la identifiquen (SMILES, SMARTS, InCHI), pero todas presentarán diferentes limitaciones en cuanto a la capacidad de describir la estructura. Al emplear alguna de estas representaciones se omiten aspectos tan importantes como la topografía de la molécula así como información concerniente a la disposición espacial de los átomos, lo cual puede en ocasiones diferenciar compuestos con la misma fórmula empírica y diferentes actividades biológicas. Es por ello que en el presente trabajo se plantea la necesidad de visualizar y editar moléculas, así como la integración de todos los módulos de la Plataforma.

Por lo que el **objeto de estudio** es la visualización y edición de estructuras químicas, teniendo como **campo de acción**, las aplicaciones para la visualización y edición de estructuras químicas asistida por computadora.

El presente trabajo tiene como **objetivo general** desarrollar una aplicación que permita la visualización y edición de moléculas, así como establecer la comunicación de la interfaz visual con los restantes módulos de la Plataforma.

Para cumplir el objetivo general del trabajo se trazaron los siguientes **objetivos específicos**:

- ✓ Analizar, diseñar e implementar el módulo de visualización.
- ✓ Analizar, diseñar e implementar el módulo de edición.
- ✓ Analizar, diseñar e implementar la comunicación con los restantes módulos de la Plataforma.

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

- Búsqueda bibliográfica acerca de los diferentes visualizadores y editores moleculares que existen y

están probados a nivel mundial, además de las principales tecnologías utilizadas por ellos.

- Estudio de arquitecturas de plug-ins y carga dinámica de clases sobre Java
- Intercambio con especialistas para la determinación de los requerimientos de la plataforma.
- Implementación de los algoritmos de:
    - ✓ Cálculo de por ciento de composición de la molécula.
    - ✓ Cálculo de la masa molecular de la molécula.
- Realización de pruebas a la interfaz del sistema.

El documento está estructurado por un resumen, introducción, 4 capítulos que constituyen el cuerpo fundamental de la tesis, conclusiones generales, bibliografía y referencias bibliográficas. Los capítulos son:

Capítulo 1: Fundamentación Teórica. Se presenta una breve reseña de los resultados del estudio bibliográfico realizado sobre la visualización y edición de moléculas. Se describen las tendencias actuales a tener en cuenta para implementar este tipo de herramientas informáticas, así como las aplicaciones informáticas existentes en la actualidad vinculadas a la visualización y edición de moléculas. Se describen las tecnologías, metodologías y herramientas a utilizar, analizando sus desventajas y ventajas.

Capítulo 2: Características del Sistema. En el se aborda lo referente al funcionamiento del negocio: las reglas y la descripción del mismo. En este caso se desarrolla un modelo de dominio donde se analizan cada uno de los conceptos y entidades presentes en el negocio. Además de describir la solución propuesta, se exponen elementos imprescindibles para una solución exitosa: como lo son los requisitos funcionales y no funcionales, así como la descripción de los Casos de Uso del Sistema.

Capítulo 3: Análisis y Diseño del Sistema. Consiste en traducir los requisitos a una especificación que describe cómo implementar el sistema, en obtener una visión del sistema que se preocupa de ver QUÉ hace y CÓMO cumple el sistema sus objetivos. Mediante el análisis se podrá estructurar los requisitos de manera que facilite su comprensión, preparación, modificación y en general su mantenimiento; y como resultado final y más importante se desarrolla el modelo de diseño.

Capítulo 4: Implementación y Prueba. Comienza con el resultado del diseño, implementando el sistema en términos de componentes, así como una revisión final de las especificaciones del diseño y de la codificación mediante la realización de pruebas, garantizando la calidad del software.

CAPÍTULO 1

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

En la actualidad existen diferentes procedimientos computacionales para la visualización y edición de estructuras químicas a partir del elevado grado de desarrollo que tiene la bioinformática. Tanto las necesidades de esta ciencia emergente como de la propia informática han sufrido cambios que han conducido al avance de esas disciplinas y de las tecnologías, metodologías y herramientas a utilizar para el desarrollo de aplicaciones mas potentes ajustadas a los objetivos de cada proyecto en particular.

1.1 La Bioinformática, herramienta en el descubrimiento de nuevos medicamentos.

El interés en el uso de computadoras para afrontar problemas biológicos comienza a finales de los años 60 y principios de los 70, primeramente en los laboratorios de los Álamos. Dentro del grupo inicial se cuentan nombres como el de Michael Waterman, Temple Smith, Paul Stein y otros. Hacia los 80 se empieza a contar con las herramientas informáticas a nivel de laboratorio para la generación masiva de datos, y en los años 90, de la mano con la explosión de Internet y dentro de la biología moderna surge la Bioinformática como ciencia.

La Bioinformática cobija básicamente estudios relativos a enfoques sistémicos sobre datos biológicos. No se restringe únicamente a aquellos generados como parte del proyecto GENOMA, sino que pretende brindar una herramienta al biólogo a través de la cual él pueda afrontar la integración coherente de grandes cantidades de datos, al permitir la organización del conocimiento en sí mismo. Hasta el momento, su área de mayor desarrollo ha estado ligada a la información proveniente de los distintos proyectos GENOMA, es así como dentro de las áreas que más apoyo y demanda tienen, se pueden citar:

1. Bases de datos
2. Programas para visualización de estructuras moleculares
3. Sistemas de manejo integrado de laboratorios de I&D a nivel de Industria Farmacéutica.
4. Programas para estudios de redes genéticas (Genetic Networks).
5. Desarrollo de nuevos algoritmos para estudios de estructura proteica.

Las áreas antes mencionadas describen de manera clara el interés general. Se tienen grandes cantidades de datos que en sí mismos no dan una idea acerca del comportamiento del sistema como tal. Es por ello que, dentro de la creación de bases de datos, se pretende integrar totalmente las distintas fuentes de información que existe mediante el desarrollo de interfases comunes tanto a nivel de usuario final como a nivel de plataformas de desarrollo.

La Bioinformática y su relación con la Industria Farmacéutica

El auge que experimenta la Bioinformática es visible con el surgimiento de compañías orientadas a prestar servicios en el manejo de la información biológica. Los datos en sí mismos no son comercializables, pero la información implícita en ellos si lo es.

A nivel de la Industria Farmacéutica, donde los procesos de Investigación y Desarrollo (I&D) además de largos son sumamente costosos, la Bioinformática está llamada a jugar un papel preponderante en el descubrimiento de nuevos medicamentos.

Dentro de los procesos de desarrollo de nuevas moléculas con propiedades farmacológicas, es usual identificar primero el blanco de la molécula, su contraparte. Este proceso usualmente se enfocaba solamente a unas pocas moléculas, pero dado el auge en el acceso a la información de los proyectos GENOMA, el número de potenciales blancos para resolver ciertos problemas se ha incrementado de manera palpable. En tanto más genes son secuenciados y conocidos, los procesos de desarrollo de fármacos tienen más y más posibles rutas para dar solución a sus problemas de manera más directa. Pero al tiempo que las opciones crecen se hace también más necesario el uso de herramientas informáticas para afrontar esta explosión de información. La Bioinformática en este contexto facilita la selección de los mejores blancos.

El éxito en las tareas de I&D al nivel de industria farmacéutica está dado en cómo se entienda el sistema genético y en cómo se sea capaz de extraer de dicho sistema la información necesaria para inhibir ciertos procesos. Es real que una computadora no puede reemplazar un laboratorio y es por esto que los aportes tradicionales químicos y farmacológicos continuarán siendo importantes, sin embargo, es igualmente cierto que el uso de recursos informáticos minimiza tiempo en esta clase de procesos de I&D haciendo más atractiva la relación Costo-Beneficio.

1.2 La visualización de la molécula

Para el fácil entendimiento de la Biología Molecular se necesita transmitir de una forma totalmente correcta desde el punto de vista científico la composición y forma de las biomoléculas, en ausencia del apoyo de novedosas tecnologías. Este tema se podía explicar acudiendo a proyecciones bidimensionales de las estructuras.

La necesidad de representar la información científica de una forma más legible para la mente humana no es algo nuevo. Probablemente se recuerde de las clases de química en secundaria los juegos de bolas y varillas con los que se construían la estructura de diferentes moléculas. Las bolas simbolizaban los átomos y las varas los enlaces. Esta representación 3D en el mundo real tenía interesantes propiedades: permitía hacerse una idea de la estructura 3D de la molécula y comprender mejor la forma en que estaban dispuestos los enlaces y hasta ver que determinadas geometrías no eran posibles. Incluso podías tocar y sentir la molécula. Evidentemente, el diseño de nuevos fármacos es muchísimo más complejo. Aquí resulta de interés no sólo la visualización de las moléculas sino, sobre todo, de sus propiedades como la densidad electrónica, la disposición de las fuerzas de Van der Waals, etc.

Pero visualizar no es el resultado implícito del acto de ver, no es un producto espontáneo del individuo que recibe la información ya visualizada. Visualizar es una tarea que transforma datos abstractos y fenómenos complejos de la realidad en mensajes visibles, haciendo posible que los individuos vean con sus propios ojos datos y fenómenos que son directamente inapreciables, y por tanto comprendan la información que yace oculta [1].

Los logros a nivel de dilucidación de estructura tridimensional llevaron, junto con otros factores, a la creación de bancos de datos en los que se almacenaron los archivos con las posiciones espaciales de cada uno de los átomos de la molécula. Con el fin de permitir el libre acceso a este tipo de información, estos bancos de datos fueron implementados sobre Internet y su administración se volvió entonces, tarea de especialistas, llamados hoy día curadores.

Con la masificación del uso de las computadoras se hizo evidente la necesidad de permitir, a quiénes no contaban con una estación de trabajo, la visualización y el análisis de estos archivos de coordenadas moleculares disponibles a través de la Web. Surgen entonces, programas de gráficos moleculares, capaces de leer y presentar al usuario la estructura tridimensional, para que esta pudiera ser analizada desde sus componentes más simples.

Visualizadores consultados

Rasmol

Permite de manera inicial visualizar moléculas, pero además brinda la posibilidad de manipularlas, inclusive al nivel atómico. Básicamente, lo que hace RasMol, es leer un archivo que contiene la información estructural de la molécula, como por ejemplo un archivo PDB; y presentar en pantalla el modelo gráfico, como resultado de la localización de cada uno de los átomos de la molécula, sobre un plano de coordenadas cartesianas *x, y, z* y de la construcción de un grafo que relaciona las entidades que componen el sistema. Una vez que RasMol, presenta la estructura, permite manipularla mediante un conjunto de comandos sencillos que dan la posibilidad, desde seleccionar un átomo, hasta dejar una cadena titilando con el fin de hacer más evidente una explicación determinada.

Aplicaciones básicas:

- Visualizar diferentes representaciones y modelos de la estructura química y aplicar sobre ellos diferentes esquemas y patrones de color.
- Rotar, desplazar y cambiar el tamaño de la representación de la molécula en pantalla.
- Seleccionar y/o restringir segmentos específicos de la molécula tales como sitios activos, dominios de unión, ligados, cofactores, mutaciones, etc.
- Exportar las representaciones o imágenes moleculares generadas en diferentes formatos gráficos como gif, ppm y bmp, de manera que puedan utilizarse en la edición de documentos y presentaciones de algún tema de interés relacionado con la química, la bioquímica o la biología molecular.
- Construir animaciones moleculares o "scripts" en donde se presentan transformaciones sucesivas de una molécula para explicar sus características estructurales y su relación con la actividad biológica.

Chime

Con el desarrollo de las tecnologías Web se hizo necesaria la implementación de medios que permitieran alguna de las operaciones que permitía RasMol. Nace entonces CHIME, un plug-in para navegadores que permite el manejo de archivos tipo PDB pero a nivel Web. Como visualizador molecular, Chime es mucho

más "amigable" para el usuario ya que con este se puede navegar fácilmente por las diferentes propiedades de una molécula con todas las ventajas que ofrecen los archivos HTML.

Ventajas:

- Manejo interactivo de la molécula durante la ejecución de los scripts o animaciones moleculares. Es posible cambiar la posición y el tamaño de la molécula, y hacerla rotar, utilizando únicamente los botones del mouse.
- Aplicación de la mayor parte de comandos de RasMol a partir de un menú de herramientas emergente y no a partir de una Línea de comandos como en RasMol.
- Posibilidad de incluir imágenes, textos, tablas, sonido, etc, permitiendo la generación de sitios Web muy atractivos para el usuario.
- Facilidad en la edición y ejecución de los scripts. A diferencia de un script RasMol, en Chime es posible que el usuario lo ejecute parcialmente, y es posible también devolverse y repetir algún paso de la secuencia tantas veces como se estime necesario.

Limitantes de RasMol y Chime

RasMol y Chime, aunque muy útiles, son simples visualizadores de moléculas que, independientemente de sus funciones, tienen limitaciones pues fueron concebidos como herramientas educativas y no como herramientas de análisis es decir, son simples navegadores. Al no ser de código abierto, limita poderle adicionar otras funcionalidades.

SwissProt

Una herramienta que permite además de visualizar el analizar, o por lo menos llevar a cabo ciertos análisis básicos sobre estructuras proteicas es el SwissProt.

SwissProt pertenece a otra clase de desarrollo, está íntimamente ligado a los desarrollos que para análisis de proteínas se han implementado y son de libre uso como por ejemplo TINKER. Este programa carece de muchas de las capacidades que RasMol y Chime poseen a nivel de presentación de información y

posibilidades de apoyo a nivel educativo básico, pero es una poderosa herramienta introductoria para el análisis de secuencias proteicas.

Aplicaciones Básicas de SwissProt:

- Representaciones moleculares y esquemas de color básicos.
- Manejo simultáneo de varios archivos de coordenadas moleculares.
- Análisis de sitios activos.
- Análisis de Estructura Secundaría, Análisis de conformaciones proteicas posibles en un modelo dado (Ploteos de Ramachandran).
- Análisis de interacción entre proteínas, cofactores, inhibidores y ligandos.
- Comparación conformacional de 2 o más proteínas. Comparación de proteínas homólogas.
- Construcción de fragmentos "loops". Construcción de simetrías cristalográficas,
- Construcción de un modelo de estructura tridimensional a partir de la secuencia aminoacídica. Alineamientos entre secuencias. Modelación por homología.
- Mutaciones de aminoácidos.
- Interpretación de mapas de densidad electrónica.

SwissProt brinda la posibilidad de evaluar los modelos construidos en una proteína dada, es posible evaluar los modelos cristalográficos, la homología entre proteínas, etc.

Limitantes de SwissProt

A pesar de las posibilidades que brinda, SwissProt carece de un editor de moléculas y no realiza cálculos de descriptores u otros valores útiles al investigador. Al ser este un software propietario no es posible reutilizarlo en proyectos de desarrollo una vez comprado el software, además de ser un programa desarrollado para trabajar solo bajo ambiente Windows.

JMOL

Entre las principales características que distinguen a este programa de los otros visualizadores se puede mencionar la posibilidad de ser empleado como miniaplicación (applet), aplicación y componente para la integración en sistemas.

- JmolApplet es una miniaplicación para el navegador Web que puede integrarse en páginas Web. Es ideal para el desarrollo de material docente a través de la Web y para bases de datos químicas accesibles por Internet. La miniaplicación JmolApplet proporciona una vía de actualización para los usuarios del conector Chime.
- La aplicación Jmol es un programa autónomo que se ejecuta localmente en el ordenador.
- JmolViewer puede integrarse como un componente dentro de otros programas Java.

Jmol es compatible con los principales navegadores, entre los que se encuentran:

- Internet Explorer (Win32)
- Mozilla o Firefox (Win32, OSX, *nix)
- Safari (Mac OS X)
- Opera 7.5.4 (sólo en Win32)

JMOL ofrece una representación gráfica tridimensional de alto rendimiento sin requisitos de hardware. Es capaz de interpretar y representar diferentes formatos de archivos moleculares como:

- CIF/mmCIF - Crystallographic Information File and Macromolecular Crystallographic Information File, formatos estándar de la *Unión Internacional de Cristalografía*
- CML - Chemical Markup Language (lenguaje químico de marcado)
- CSF - estructura química de Fujitsu CAChe, ahora Fujitsu Sygress
- CTFile - Tabla química de *Elsevier MDL*
- GAMESS - General Atomic and Molecular Electronic Structure System output, *Gordon Research Group, Iowa State University*
- Gaussian 94/98/03 formato de salida - *Gaussian, Inc.*

- Ghemical
- HIN - HyperChem de *Hypercube, Inc.*
- Jaguar - *Schrodinger, LLC*
- MM1GP - Ghemical, mecánica molecular
- MOL - estructura, de *Elsevier MDL*
- MOLPRO - formato de salida de Molpro
- MOPAC - formato de salida de MOPAC 93/97/2002 (dominio público)
  - MOPAC 2007 (v.7.101) formato de salida graphf (archivos .mgf) (public domain) [a partir de 11.1.28; lee coordenadas, cargas y orbitales moleculares]
- NWCHEM - formato de salida de NWChem, *Pacific Northwest National Laboratory*
- odedata - datos de Odyssey, *WaveFunction, Inc.*
- PDB - Protein Data Bank, *Research Collaboratory for Structural Bioinformatics*
- QOUT - *Q-Chem, Inc.*
- SDF - estructura de *Elsevier MDL*
- SHELX - *Structural Chemistry Department, University of Göttingen (Germany)*
- SMOL - datos de Spartan, *Wavefunction, Inc.*
- xodedata - datos XML de Odyssey, *WaveFunction, Inc.*
- XYZ - archivo XMol de *Minnesota Supercomputer Institute*
- XYZ+vib -formato XYZ con información de vectores de vibración
- XYZ-FAH - archivo XYZ de Folding@home

* Los archivos comprimidos con gzip se descomprimen automáticamente

Entre otras de las prestaciones que presenta están las representaciones de animaciones, vibraciones, respaldo básico para la celdilla unidad, formas esquemáticas para las estructuras secundarias, así como mediciones de:

- distancia
- ángulo
- ángulo de torsión o diedro.

El programa es compatible además con el lenguaje de instrucciones (scripts) de RasMol/Chime para facilitar su uso a usuarios de estas aplicaciones informáticas.

Su biblioteca de apoyo en JavaScript permite a los usuarios de la Web controlar el applet de manera sencilla. Permite también exportar las representaciones a .jpg, .png, .ppm, .pdf y PovRay.

JMOL fue seleccionado para formar parte de la aplicación porque, además de poseer las características antes mencionadas, es un visor de moléculas gratuito y de código abierto autorizado bajo la GNU Lesser General Public License y es multiplataforma, compatible con sistemas Windows, Mac OS X y Linux/Unix.

1.3 La edición de la molécula

Otra de las áreas que dentro de la Bioinformática ha tenido mucha difusión es la de editar moléculas. Debido a la importancia que tiene para los investigadores poder representar un compuesto antes de sintetizarlo, así como hacerle pruebas, cálculos y ensayos teóricos, se ha incrementado el auge de los editores moleculares en el mundo moderno. La implementación de programas para la edición molecular ha permitido dibujar moléculas en dos dimensiones, transformarla en 3D, importar moléculas de otros visualizadores y exportarlas a los diferentes formatos.

El aspecto tridimensional es especialmente importante en la química y en la bioquímica, donde el tamaño, la forma y la polaridad de las [macro] moléculas determinan su función.

Se torna entonces, absolutamente necesario, el análisis del comportamiento de un compuesto dado en un medio específico, así como la variación, por adición o eliminación, de algunas de las partes que lo componen, como por ejemplo, la adición, eliminación o cambio de cadenas, anillos, u otros componentes químicos. La modelación “in sílico” se está convirtiendo en un enfoque esencial para el descubrimiento de fármacos optimizados en cuanto a su actividad biológica, efectos colaterales, entre otros. Su capacidad para descartar avances falsos en las etapas iniciales del descubrimiento de fármacos utilizando la modelación y la informática ahorra costos de oportunidad y dólares. Mediante la utilización de la misma se reducen los riesgos en las pruebas preclínicas y clínicas, impulsando la aceptación de las tecnologías “in

sílico” de la modelación. Con el uso de estas tecnologías alternativas, se han logrado reducir las pruebas en animales debido a las preocupaciones éticas presentadas por grupos de presión. Con todos estos factores a favor, los editores moleculares se han vuelto una funcionalidad indispensable en las herramientas de laboratorios virtuales para modelación de fármacos.

Editores consultados

ChemSketch

Permite dibujar moléculas en dos dimensiones y, con un simple click, las transforma en 3D. El programa consta realmente de dos partes: la de diseño, con múltiples bases de datos de prediseños, que incluso se pueden ampliar en la misma Web y, la parte de visualización 3D, así como otros grupos de moléculas ya diseñadas: alcaloides, esteroides, terpenos, entre otros.

Visualiza una interfaz de dibujo químicamente inteligente que proporciona un portal a una gama entera de herramientas analíticas, y facilita la transformación de estructuras o los datos analíticos en un modo grafico profesional.

La versión gratuita (freeware) contiene las herramientas para la predicción de tautómeros, limpieza de la estructura, optimización y visión 3D, dibujo de polímeros, estructuras organometálicas y de Markush. Incluye también la capacidad de nombrar, según el sistema IUPAC, las moléculas con menos de 50 átomos y 3 anillos. Las capacidades de ChemSketch pueden ser ampliadas y modificadas para requisitos particulares a partir de una programación más a fondo. Incluye además, varias características avanzadas como un diccionario de 140.000 nombres triviales, comunes, y comerciales con sus estructuras correspondientes.

HyperChem

HyperChem es un poderoso programa que permite hacer cálculos moleculares de alta calidad. Usando HyperChem se puede:

- Dibujar estructuras moleculares simples y complejas, incluyendo cristales, carbohidratos, péptidos y secuencias de ADN.

- Mostrar estas estructuras en varias representaciones diferentes incluyendo bolas y cilindros y llenado de espacios.
- Aplicar reglas simples de valencia para generar geometrías ideales.
- Realizar cálculos de minimización de energía por mecánica molecular para producir geometrías más realistas.
- Realizar diferentes cálculos semiempíricos y ab initio.
- Visualizar los resultados de los cálculos de orbitales moleculares con visualizaciones 2D y 3D de los diferentes orbitales, densidades de electrones, carga total y densidad de spin y potencial electrostático
- Calcular y visualizar el espectro vibracional.
- Calcular el espectro electrónico.
- Realizar simulaciones de dinámica molecular.
- Estudiar mecanismos de reacción.
- Incorporar los resultados de los cálculos anteriores en páginas Web interactivas.

Limitantes de ChemSketch e HyperChem

Tanto ChemSketch y otros como HyperChem poseen un alto valor en el mercado. De ahí que su obtención y reutilización se dificulte, además de no poseer otros cálculos necesarios para los diversos módulos de la plataforma. Fueron diseñados para trabajar bajo ambiente Windows, limitando a todos aquellos que trabajen bajo otro sistema operativo.

JME

JME Molecular Editor© es una aplicación Java que permite dibujar, editar, y mostrar moléculas y reacciones en modo de aplicación o como un applet embebido en una página HTML. El editor puede generar SMILES de la estructura creada, de manera que pueda ser utilizado como interfaz para las bases de datos moleculares y de reacción o para los servicios de cálculo de propiedades. El editor posee además las siguientes funcionalidades:

- Fácil creación y edición de moléculas, incluyendo estereoquímica. Generación de SMILES.
- Creación de consultas de subestructura. Generación de SMARTS
- Creación de reacciones, también posible con mapeo de átomos. Generación de códigos SMILES o SMIRKS de las reacciones.
- El applet puede ser usado en modo de dibujo (sin los botones de edición), para visualizar moléculas. Pueden ser procesados ficheros JME internos o MDL.

A pesar de sus características, el editor JME es relativamente pequeño (37 kB) lo que garantiza la rápida carga sobre la Web. Como el programa está escrito en Java 1.0, es compatible con todos los navegadores Web comunes.

JME fue seleccionado para formar parte de la aplicación por poseer las características antes mencionadas.

Otras aplicaciones relacionadas con esta rama

Corina

- Soporte para ficheros SD/RDFile, SMILES, SYBYL MOL/MOL2, macromodelos, CTX.
- Exporta ficheros SD/RDFile, SYBYL MOL/MOL2, PDB, macromodelos, CIF, CTX.
- Genera conformaciones de alta calidad, de mínima energía, y opcionalmente, conformaciones múltiples para sistemas de anillos.
- maneja de manera apropiada la información estereoquímica y de manera opcional genera estereoisómeros.
- Las estructuras pueden ser orientadas como funciones de sus principales momentos de inercia.
- Incluye funcionalidades de limpieza de estructuras
- Maneja un amplio rango de compuestos de química orgánica y organometálica con átomos de un máximo número de coordinación de 6.
- Procesa un conjunto de 100100 moléculas de pequeño a medio tamaño en 2,301 seg en una

estación de trabajo de 1.0 Ghz.

Limitantes del Corina

Solo se limita a la optimización de la geometría de la molécula, no edita ni visualiza las moléculas que procesa, no calcula sus propiedades químicas o actividades biológicas. Su código no puede ser reutilizado, además de tener un alto precio en el mercado.

1.4 Importancia del desarrollo de la aplicación

Después de haber realizado un estudio sobre las aplicaciones informáticas que son utilizadas para la visualización y edición de moléculas se llegó a la conclusión de que ninguna satisface las necesidades planteadas por el proyecto. Las aplicaciones más potentes encontradas, no son gratuitas y tampoco son multiplataforma. Por otra parte, aquellas gratuitas y multiplataforma, como el Rasmol, no cumplen con los requerimientos exigidos.

Con el desarrollo de la aplicación, el trabajo con la Plataforma será más amigable para todo aquel que la utilice. Su interfaz amigable, la posibilidad de visualizar los compuestos en 3D, de editar sus propios compuestos, harán de ella una herramienta que tanto para los biólogos como para los químicos que trabajen con ella, lo hagan de manera interactiva, práctica y agradable. Su posibilidad de incluir otras funcionalidades que se vayan desarrollando hace de la misma un potencial competidor para las soluciones informáticas existentes. La misma permitirá principalmente hacer un poco más corto y ameno el proceso de investigación en la búsqueda de nuevas entidades químicas con potencial actividad biológica, las cuales servirán para el desarrollo de nuevos medicamentos y se abaratarán los costos de investigación y desarrollo en el largo proceso de diseño de fármacos.

Poder contar con código propio brindará además la posibilidad de incorporar nuevos módulos, como por ejemplo el de una base de datos de compuestos orgánicos con actividad biológica asociada, la que interactuará con el módulo de cálculo de descriptores. Algunos de estos fueron desarrollados por investigadores cubanos, por lo que no se encuentran implementados en ninguna aplicación hasta el momento, lo que hace que en ese aspecto la aplicación sea única. El hecho que sea multiplataforma posibilita también su uso bajo cualquier sistema operativo.

Una vez concluida la aplicación será la interfaz gráfica de la Plataforma Inteligente para la Predicción de Actividades Biológicas de Compuestos Orgánicos.

1.5 Metodología, tecnologías y herramientas

Al enfrentarse al desarrollo de un producto de software, se hace indispensable tener básicamente una metodología para su desarrollo.

Una metodología es un conjunto ordenado de pasos a seguir para cumplir un objetivo. Dentro de la Ingeniería de Software, el objetivo es el desarrollo de software de alta calidad que cumpla con las necesidades del usuario (cliente). [2]

Metodologías de desarrollo

XP

La Programación Extrema (XP) es una metodología ligera de desarrollo de software que se basa en la simplicidad, la comunicación y la realimentación o reutilización del código desarrollado.

Es una de las metodologías de desarrollo de software más exitosas en la actualidad utilizadas para proyectos de corto plazo, corto equipo y cuyo plazo de entrega era ayer. La metodología consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo al usuario final, pues es uno de los requisitos para llegar al éxito del proyecto. Por las características de la plataforma, la metodología XP no resultaba eficaz para el desarrollo de la misma, por estar planificada la entrega de los resultados a largo plazo y por poseer un amplio grupo de trabajo.

RUP

La metodología que se emplea para el desarrollo de esta investigación es RUP (Rational Unified Process), analizando que es la que más se adapta a las necesidades de la plataforma. Utilizando UML (Unified Modeling Lenguage), como lenguaje representativo.

El “Proceso Unificado” es el resultado final de tres décadas de desarrollo y uso práctico. Esta es una de las causas que conlleva a que sea la metodología que mejor se ajusta a las necesidades que existen actualmente en el desarrollo de software, pues propone un modelo iterativo e incremental, muy acorde con la naturaleza cambiante de los requisitos en muchos proyectos, guiado por casos de uso y basado en la arquitectura. Es una metodología que está totalmente respaldada por excelentes herramientas CASE como: Rational Rose y Visual Paradigm.

RUP, como ya se había mencionado anteriormente, es una metodología iterativa e incremental que va eliminando los errores cometidos en las iteraciones previas, logrando que al final del proceso se obtenga como resultado un producto de calidad. Define los roles a jugar por cada miembro del equipo de desarrollo en cada una de las etapas por las que transcurre el sistema. Facilita también la comunicación entre los diferentes miembros del equipo de desarrollo.

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 y también puede considerarse como un lenguaje de modelado visual que permite una abstracción del sistema y sus componentes.

Herramientas CASE

Rational Rose

Existen varias herramientas CASE, que dan asistencia a analistas, ingenieros de software y desarrolladores durante el ciclo de vida de desarrollo de un software, pero es Rational Rose líder en el

modelado del desarrollo de los proyectos. La herramienta fue desarrollada por los creadores de UML, utilizando la notación estándar en la arquitectura de software. Esta herramienta integra todos los elementos que propone la metodología RUP para cubrir el ciclo de vida de un proyecto pero, esta presenta limitantes para el desarrollo de la aplicación, ya que se planteó que para su desarrollo, hacer uso de herramientas multiplataforma y sobre software libre.

Visual Paradigm

Como herramienta CASE se empleó Visual Paradigm para el trabajo con UML. La herramienta está diseñada para una amplia gama de usuarios, incluyendo Ingenieros de Software, Analistas de Sistemas, Analistas de Negocios y Arquitectos de Sistemas que estén 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 este a 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.

Programación Orientada a Objeto (POO)

Lamentablemente, los costos de producción de software siguen aumentando; el mantenimiento y la modificación de sistemas complejos suele ser una tarea trabajosa, pues cada aplicación, (aunque tenga aspectos similares a otra) suele encararse como un proyecto nuevo.

Todos estos problemas aún no han sido solucionados en forma completa. Pero como los objetos son portables (teóricamente) mientras que la herencia permite la reutilización del código orientado a objetos, es más sencillo modificar el código existente porque los objetos no interaccionan excepto a través de mensajes y, en consecuencia, un cambio en la codificación de un objeto no afectará la operación con otro objeto siempre que los métodos respectivos permanezcan intactos. La introducción de tecnología de objetos como una herramienta conceptual para analizar, diseñar e implementar aplicaciones permite obtener aplicaciones más modificables, fácilmente extensibles y a partir de componentes reutilizables.

Esta característica del código disminuye el tiempo que se utiliza en el desarrollo y hace que el desarrollo del software sea más intuitivo ya que el ser humano piensa naturalmente en términos de objetos más que en términos de algoritmos de software.

El lenguaje de programación que se propone para esta aplicación es Java, el cual tiene su origen en Sun Microsystems, líder en servidores para Internet, uno de cuyos lemas desde hace mucho tiempo es "the network is the computer" (lo que quiere dar a entender que el verdadero ordenador es la red en su conjunto y no cada máquina individual). Esta firma es quien ha desarrollado el lenguaje Java, en un intento de resolver simultáneamente todos los problemas que se le plantean a los desarrolladores de software por la proliferación de arquitecturas incompatibles, tanto entre las diferentes máquinas como entre los diversos sistemas operativos y sistemas de ventanas que funcionaban sobre una misma máquina, añadiendo la dificultad de crear aplicaciones distribuidas en una red como Internet.

Lenguaje de programación

Java

El lenguaje de programación Java es un lenguaje de propósito general, concurrente, basado en clases y orientado a objetos. Está diseñado para ser lo suficientemente simple 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 la memoria.

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. Además, Java construye sus interfaces de usuario a través de un sistema abstracto de ventanas de forma que las ventanas puedan ser implantadas en entornos Unix, Pc o Mac.

Entornos de desarrollo

Dentro de los entornos de desarrollo integrado (IDE del inglés Integrated Development Environment) para el desarrollo de aplicaciones usando como lenguaje de programación Java se encuentran NetBeans, JBuilder y Eclipse.

NetBeans

El NetBeans IDE es una herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas. Está escrito en Java pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extender el IDE NetBeans, es un producto libre y gratuito sin restricciones de uso.

La versión actual todavía no permite la integración del entorno de desarrollo con un control de versiones subversion, por esa razón no fue seleccionado este IDE para el desarrollo de la herramienta.

JBuilder

JBuilder es un entorno de desarrollo integrado para el lenguaje de programación java desarrollado por Borland. Posee varias ediciones, la Enterprise, para aplicaciones J2EE, Web Services y struts. La Developer, para el desarrollo completo de aplicaciones Java, y la Foundation, con capacidades básicas para iniciarse en Java.

No fue seleccionado para el desarrollo de la herramienta JBuilder como IDE de desarrollo por no ser multiplataforma, solo puede ser ejecutado sobre el sistema operativo Windows.

Eclipse

Eclipse es una poderosa herramienta que permite integrar diferentes aplicaciones para construir un entorno integrado de desarrollo (IDE). Es un potente entorno de desarrollo de Java , usa java como lenguaje de programación aunque permite plug-ins para varios lenguajes más, soporta la programación orientada a objetos (POO), la depuración e implementación de aplicaciones resultan mucho más sencillas. La plataforma esta construida en base a plug-ins. Este mecanismo permite desarrollar, integrar y correr nuevos plug-ins. La misma tiene varios beneficios como son:

- 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.

Por todas estas características y además por 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, fue seleccionado como IDE para el desarrollo de la solución a implementar.

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

Java que ofrece una API portable y una integración muy ligada con la interfaz gráfica de usuario nativa al sistema operativo de cada plataforma. Así, cada plataforma debe adaptar el SWT para los gráficos nativos suyos. Esto parece que entra en contradicción con la filosofía de Java de ser independiente de la plataforma, pero ofrece ventajas en 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.

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

SWT (Standard Widget Toolkit) 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 necesita código nativo para poder funcionar en cada sistema operativo, pero IBM ha sido capaz de adaptar SWT a un buen número de sistemas. Es muy destacable la importancia de SWT porque es el entorno gráfico de desarrollo que viene con Eclipse y debe ser utilizado en los plug-ins desarrollados para el mismo. Eclipse es una herramienta de desarrollo altamente potente, libre distribución, código abierto y adaptable a cualquier lenguaje de programación.

SWT se compone esencialmente de tres elementos:

i. 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.

ii. Por encima de la anterior capa está la clase Display, la interfaz por la que el SWT se comunica con la interfaz gráfica.

iii. El nivel superior lo forma la clase shell, representa el tipo de ventana de más alto nivel y que 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, en cuyo caso es la ventana o marco principal a partir de la cual se construye el resto de la interfaz, o bien hija de otro shell, siendo el ejemplo más común las ventanas de diálogo.

Conclusiones

Es necesario crear una interfaz visual amigable para el proyecto “Plataforma Inteligente para la Predicción de Actividad Biológica de Compuestos Orgánicos”. Dicha aplicación debe presentar las siguientes características:

- Que sea multiplataforma.
- Que permita visualizar y editar moléculas.
- Que facilite la incorporación de nuevos módulos.

Se empleará el lenguaje de programación Java por las posibilidades multiplataforma que este brinda, además por su posibilidad de uso gratuito.

El editor seleccionado para la programación de la solución fue el Eclipse, por sus grandes potencialidades y sus posibilidades de extensión mediante el uso de plug-ins.

Se utilizará JMOL como visualizador molecular, debido a su excelente desempeño, incluso con macromoléculas, además de ser gratuito y de código abierto autorizado bajo la GNU/LGPL.

Se utilizará JME como editor gráfico. Su desarrollo sobre software libre lo hace fácil de modificar e incluir en la aplicación que se plantea.

Se utilizara el lenguaje de modelación UML y como herramienta CASE el Visual Paradigm.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

En el presente capítulo se describe la solución propuesta utilizando los componentes del modelo de dominio de la metodología RUP. De este modelo se tendrán en cuenta la definición de las entidades y los conceptos principales, así como su representación gráfica, también se conocerán las reglas del negocio. Además se describen los requisitos funcionales y no funcionales del sistema, los actores que intervienen en el sistema, así como los diagramas de casos de uso.

2.1 Modelo del dominio

Un modelo del dominio captura los tipos más importantes de objetos en el contexto del sistema. Los objetos del dominio representan las "cosas" que existen o los eventos que suceden en el entorno en el que trabaja el sistema. [3]

La modelación del dominio tiene como objetivo fundamental la comprensión y descripción de las clases más importantes en el sistema.

2.2 Glosario de Términos

Especialista: Persona capacitada para interactuar con la aplicación.

Compuesto Orgánico: sustancias químicas basadas en cadenas de carbono e hidrógeno. En muchos casos contienen oxígeno, y también nitrógeno, azufre, fósforo, boro y halógenos.

Información Estructural: Dato o descripción que permita adquirir la noción de cómo está formado el compuesto en términos de átomos que lo constituyen y relación espacial que existe entre ellos.

Imagen del Compuesto: Representación tridimensional del compuesto teniendo en cuenta la geometría espacial del mismo.

2.3 Diagrama modelo del dominio

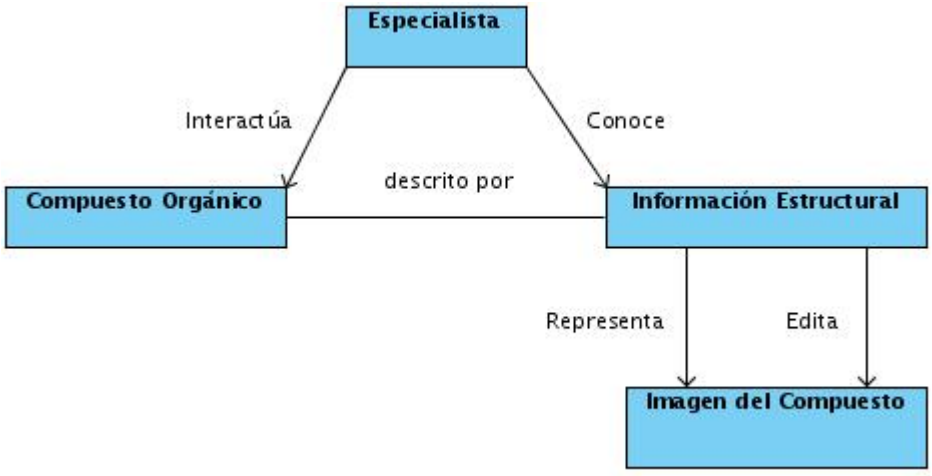


Figura 2.1: Diagrama modelo del dominio.

Descripción textual del modelo del dominio.

El especialista conociendo la estructura química de compuestos orgánicos, puede representarlos, e incluso modificarlos y volverlos a representar.

2.4 Requerimientos del sistema

Los requerimientos son condiciones o capacidades que tienen que tener los sistemas para satisfacer un contrato o documento formal. Los mismos describen el "qué" debe hacer un sistema. [4]

2.4.1 Requerimientos funcionales

Los requerimientos funcionales permiten expresar una especificación más detallada de las responsabilidades del sistema que se propone. Ellos permiten determinar, de una manera clara, lo que debe hacer el sistema. [2]

Los requerimientos funcionales del software propuesto son los siguientes:

R1: Cargar Moléculas

R2: Crear Moléculas

R3: Eliminar Moléculas

R4: Visualizar Moléculas

R5: Editar Moléculas

R6: Incorporar Plug-in

R7: Calcular Masa Molecular

R8: Calcular Por ciento de Composición

2.4.2 Requerimientos no funcionales

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. Representan las características del producto. [2]

Apariencia o interfaz externa:

La aplicación debe estar diseñada con una interfaz amigable, de forma tal que el usuario navegue sin dificultad alguna, ajustándose a los estándares establecidos para el desarrollo de un buen diseño.

Usabilidad:

El sistema podrá ser usado por cualquier tipo de personas que posea conocimientos básicos en el manejo de la computadora, solo se necesita contar con conocimientos especializados en química para entender los resultados dados por la aplicación.

Soporte:

El sistema debe propiciar su mejoramiento y la anexión de otras opciones que se le incorporen en un futuro.

Portabilidad:

El sistema puede ser ejecutado sobre los sistemas operativos Linux y Windows, por su característica de ser multiplataforma.

Seguridad:

El especialista tendrá control total de la aplicación, sin restricción alguna.

Confiabilidad:

El sistema debe ser confiable y preciso en la información que le suministra al usuario para evitar cualquier tipo de error.

Ayuda:

La aplicación posee ayuda, en la que se explica de forma clara el uso de las opciones del sistema garantizando así el buen desempeño de los usuarios a la hora de interactuar con el mismo.

Software:

Se debe disponer de sistemas operativos Linux, Windows 95 o superior para la instalación de la aplicación.

Debe tenerse instalado el Java Runtime Environment (JRE) versión 1.5 o superior.

Hardware:

Para el desarrollo y puesta en práctica del proyecto se requieren máquinas con los siguientes requisitos:

- Procesador Pentium 3 o superior
- 256 Mb de RAM
- 50 Mb de capacidad del disco duro

2.5 Actor del sistema a automatizar.

Un actor es una entidad externa del sistema que de alguna manera participa en la historia del caso de uso. Por lo general estimula el sistema con eventos de entradas o recibe algo de él. [5] O sea, es un rol de un usuario, que puede intercambiar información o puede ser un recipiente pasivo de información y representa a un ser humano, a un software o a una máquina que interactúa con el sistema.

Actores	Descripción
Especialista	Representa el usuario que va ha hacer uso del sistema, teniendo la posibilidad de interactuar con todas la funcionalidades de este.

Tabla 2.1: Actor del Sistema.

2.6 Casos de uso definidos.

Los casos de uso se utilizan para obtener información de cómo debe trabajar el sistema, son descripciones de la funcionalidad del sistema independiente de la implementación, describen bajo la forma de acciones y reacciones el comportamiento de un sistema desde el punto de vista del usuario.[2]

Los casos de usos definidos son los siguientes:

- Cargar Moléculas
- Crear Moléculas
- Eliminar Moléculas
- Visualizar Moléculas
- Editar Moléculas
- Incorporar Plug-in
- Calcular Masa Molecular
- Calcular por ciento de composición

2.7 Diagrama de caso de uso del sistema

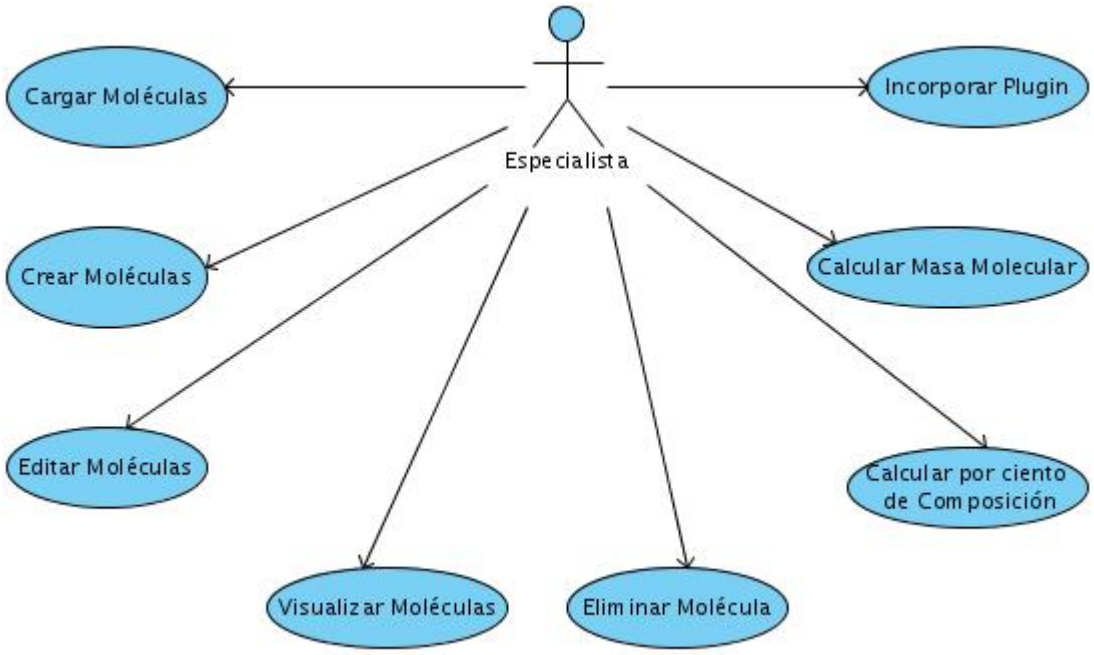


Figura 2.2: Diagrama de casos de uso del sistema.

2.8 Descripción de los casos de uso del sistema.

Caso de Uso:	Cargar Moléculas
Actores:	Especialista
Propósito:	Cargar ficheros de tipo molecular para su uso posterior en la aplicación.
Resumen:	El especialista inicia el caso de uso al seleccionar la opción “Abrir Molécula Existente” del menú o de la Barra de Herramientas, podrá cargar varios ficheros a la vez.
Referencia:	R1
Precondiciones:	Que el o los ficheros existan
Poscondiciones:	El fichero queda cargado

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El especialista selecciona la opción “Abrir Molécula Existente” del menú o de la Barra de Herramientas	1.1 El Sistema abrirá un explorador donde el especialista podrá localizar sin dificultad el fichero.
2. El especialista selecciona el fichero molecular que desea cargar.	2.1 El Sistema verifica que el fichero no se haya cargado ya en la aplicación.

	2.2 El Sistema identificó la extensión del fichero seleccionado por el usuario de tipo mol o de otro formato también molecular 2.3 El sistema carga esta molécula a una lista en la cual estarán todas las abiertas por el usuario con anterioridad y así finaliza el CU.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	2.1 El Sistema emite un mensaje de que el fichero ya existe.
	2.2 El Sistema emite un error de fichero no compatible.
Prioridad	Crítico

Tabla 2.2: Descripción del caso de uso Cargar Moléculas.

Caso de Uso:	Crear Molécula
Actores:	Especialista
Propósito:	Crear moléculas definidas por el especialista para su posterior uso en la aplicación.
Resumen:	El especialista inicia el caso de uso al seleccionar la opción “Nueva molécula” del menú o de la Barra de Herramientas. Estas moléculas se agregarán a la lista de moléculas cargadas en la aplicación para su posterior utilización en los módulos de la misma.
Referencia:	R2
Precondiciones:	
Poscondiciones:	Una nueva molécula es agregada al sistema.
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El especialista selecciona la opción “Nueva Molécula” del Menú o la Barra de Herramientas.	1.1. El sistema mostrará un Editor molecular en el cual podrá insertar átomos y enlaces.
2. El especialista selecciona la opción Guardar Molécula del Menú o la Barra de Herramientas	2.1 El sistema muestra un explorador donde podrá seleccionar el lugar y nombre del fichero donde va a almacenar la molécula.
3. El especialista selecciona el lugar y nombre que tendrá el fichero molecular y presiona Aceptar.	3.1 El sistema crea un nuevo objeto de tipo Molécula. 3.2 El sistema guarda los datos de los átomos y enlaces en la nueva molécula. 3.3 El sistema agrega la nueva molécula a la lista de moléculas cargadas al sistema, finalizando así el caso de uso.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
3. El especialista selecciona Cancelar.	3.1 Se retorna al punto 1.1.

Prioridad:	Crítico

Tabla 2.3: Descripción del caso de uso Crear Moléculas.

Caso de Uso:	Visualizar Moléculas
Actores:	Especialista
Propósito:	Visualizar una molécula dada.
Resumen:	El especialista inicia el caso de uso al seleccionar de la lista de moléculas cargadas en la aplicación la que desee visualizar, luego la representación tridimensional de la molécula queda visualizada en pantalla.
Referencia:	R4
Precondiciones:	Haber cargado o creado la molécula.
Poscondiciones:	La representación tridimensional de la molécula es mostrada en pantalla.
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El caso de uso comienza cuando el usuario selecciona de la lista de moléculas cargadas en la aplicación la que desee visualizar.	1.1 El Sistema localiza la ubicación de la molécula seleccionada por el usuario. 1.2 La molécula es visualizada por la aplicación en una vista de tres dimensiones, cada átomo de la molécula se representará con un color determinado de acuerdo al elemento químico, finalizando así el caso de uso.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
Prioridad:	Crítico

Tabla 2.4: Descripción del caso de uso Visualizar Moléculas.

Caso de Uso:	Editar Moléculas
Actores:	Especialista
Propósito:	Cambiar la estructura de una molécula cargada o creada en el sistema. La misma será utilizada en los procesos de la aplicación.
Resumen:	El especialista inicia el caso de uso al seleccionar la opción Editar molécula del menú o de la Barra de Herramientas. Esto modifica la estructura de la molécula que se encuentra seleccionada y almacenando los cambios permanentemente en el fichero original.
Referencia:	R5
Precondiciones:	Haber cargado o creado la molécula. El plug-in de edición debe ser cargado.
Poscondiciones:	La molécula es modificada.

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El especialista selecciona la opción Editar molécula del menú o de la Barra de Herramientas.	1.1. El sistema mostrará un Editor molecular en el cual podrá modificar la estructura de la molécula seleccionada.
2. El especialista selecciona la opción Guardar Molécula del Menú o la Barra de Herramientas.	2.1 El sistema muestra un explorador donde podrá seleccionar el lugar y nombre del fichero donde va a almacenar la molécula.
3 El especialista selecciona el lugar y nombre que tendrá el fichero molecular y presiona Aceptar.	3.1 El sistema guarda los cambios de la molécula editada y actualiza los datos que ya tenía.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
3. El especialista selecciona Cancelar.	3.1 Se retorna al punto 1.1.
Prioridad:	Crítico

Tabla 2.5: Descripción del caso de uso Editar Moléculas.

Caso de Uso:	Incorporar plug-in
Actores:	Especialista
Propósito:	Incorpora nuevas funcionalidades a la aplicación
Resumen:	El caso de uso se inicia al seleccionar la opción Agregar Extensión de la barra de menú. El sistema muestra una interfaz donde se puede localizar la carpeta donde estará el (los) paquete(s) a incluir (plug-in). Posteriormente la aplicación mostrará los plug-ins disponibles en esta carpeta así como otras informaciones disponibles del mismo. Al seleccionar el plug-in y presionar en aceptar, la aplicación hará una copia del mismo en la carpeta donde se encuentran las extensiones de la aplicación y se cargará en la próxima iniciación de la aplicación.
Referencia:	R6
Precondiciones:	
Poscondiciones:	Se incluirá la funcionalidad seleccionada al sistema.
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El especialista selecciona la opción Agregar Extensión de la barra de Menú.	1.1 La aplicación muestra una interfaz de selección de la carpeta donde se encuentra la funcionalidad a incluir.

2. El especialista selecciona una carpeta.	2.1 La aplicación muestra todos los plug-ins existentes en esa carpeta, así como sus descripciones y otras informaciones de los mismos.
3. El especialista selecciona el plug-in a incluir y presiona Aceptar.	3.1 La aplicación hace una copia del paquete a la carpeta de extensiones de la misma. 3.2 La aplicación incluye la dirección y descripción del nuevo plug-in en la lista de extensiones de la aplicación. 3.3 La aplicación recarga las funcionalidades incluidas para hacer funcionar la nueva funcionalidad. 3.4 La aplicación actualiza las barras de herramientas y de menú para hacer accesibles las funcionalidades dadas. 3.5 Finaliza el caso de uso.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
3. El especialista no selecciona ningún plug-in y presiona Cancelar.	3.1 La aplicación cierra la ventana de selección de plug-ins finalizando así el caso de uso.
3. El especialista selecciona la opción de especificar otra carpeta.	3.1 Se retorna al paso 1.1.
Prioridad:	Secundario

Tabla 2.6: Descripción del caso de uso Incorporar plug-in.

Caso de Uso:	Calcular Masa Molecular
Actores:	Especialista
Propósito:	Calcular la Masa molecular de cada molécula entrada al sistema.
Resumen:	El caso de uso se inicia al interactuar con una molécula. El sistema hace un cálculo de la composición de la molécula y a partir de ahí calcula la masa teniendo en cuenta la masa de sus componentes.
Referencia:	R7
Precondiciones:	Haber cargado o creado la molécula.
Poscondiciones:	La molécula cuenta con su valor de masa molecular.
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El actor carga la molécula al sistema.	1.1 El sistema comprueba la composición de la molécula. 1.2 El sistema calcula la masa de la molécula a través de la siguiente fórmula: $\sum\left(Masa\left(átomo\left[i\right]\right)*CantidadÁtomos\left[i\right]\right)$ 1.3 Finaliza el Caso de Uso.
Flujos Alternos	

Acción del Actor	Respuesta del Sistema
Prioridad:	Secundario

Tabla 2.7: Descripción del caso de uso Calcular Masa Molecular.

Caso de Uso:	Calcular por ciento de composición
Actores:	Especialista
Propósito:	Calcular el por ciento de composición de cada molécula entrada al sistema.
Resumen:	El caso de uso se inicia al interactuar con una molécula. El sistema calcula la composición de la molécula y su por ciento.
Referencia:	R8
Precondiciones:	Haber cargado o creado la molécula.
Poscondiciones:	La molécula cuenta con su por ciento de composición.
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El Actor carga la molécula al sistema	1.1 El sistema determina los diferentes tipos de átomo por los que está compuesta la molécula.
	1.2 El sistema determina la cantidad de átomos de cada tipo que componen la molécula.
	1.3 El sistema calcula el por ciento de composición mediante la fórmula $(CantAtomos_i / CantAtomos_{total})*100$, finaliza el Caso de Uso
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
Prioridad:	Secundario

Tabla 2.8: Descripción del caso de uso Calcular por ciento de Composición.

Caso de Uso:	Eliminar Moléculas
Actores:	Especialista
Propósito:	Eliminar una molécula del sistema

Resumen:	El caso de uso se inicia al seleccionar la opción Eliminar molécula del menú contextual del panel de moléculas cargadas al sistema. El sistema quita la molécula de su lista de moléculas y la elimina de la aplicación, liberando espacio en memoria
Referencia:	R3
Precondiciones:	Debe haber al menos una molécula en el sistema.
Poscondiciones:	La molécula será eliminada del sistema.
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El Actor selecciona Eliminar Molécula del menú contextual del panel de moléculas cargadas al sistema.	1.1 El sistema localiza la molécula a eliminar en la lista de moléculas cargadas en el sistema. 1.2 El sistema elimina la molécula que se encuentra en la posición localizada. 1.3 El sistema actualiza la lista de moléculas cargadas al sistema y finaliza el caso de uso.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
Prioridad:	Crítico

Tabla 2.9: Descripción del caso de uso Eliminar Moléculas.

Conclusiones

Debido a la poca estructuración de los procesos del negocio y para una mejor comprensión se definieron en este capítulo conceptos, que fueron relacionados mediante un diagrama de Modelo del Dominio brindando de forma general cómo se desarrolla este proceso.

Además de brindar una clara definición de los requisitos que debe cumplir el sistema, fueron seleccionados los actores así como los casos de uso con los que constará dicho sistema.

El empleo de los modelos de casos de usos para describir el sistema propuesto permitió una adecuada captación y modelación de los requerimientos, demostrando su importancia en esta etapa. Se ganó claridad en cuanto a la concesión del sistema a construir y se sentaron las bases para las restantes fases del proceso de análisis, diseño e implementación.

CAPÍTULO 3

CAPÍTULO 3: ANÁLISIS Y DISEÑO DEL SISTEMA

El presente capítulo consiste en traducir los requisitos a una especificación que describe cómo implementar el sistema, en obtener una visión del sistema que se preocupa de ver QUÉ hace y CÓMO cumple el sistema sus objetivos. Mediante el análisis se puede estructurar los requisitos de manera que facilite su comprensión, preparación, modificación y en general su mantenimiento; y como resultado final y más importante se desarrolla el modelo de diseño.

3.1 Modelo de análisis

Modelo que se utiliza para obtener una visión del sistema sobre los Requisitos Funcionales, expresados en lenguaje técnico. [6]

3.1.1 Diagramas de clases de análisis

El Diagrama de Clase es el diagrama principal de diseño y análisis para un sistema. En él, la estructura de clases del sistema se especifica, con relaciones entre clases y estructuras de herencia. Durante el análisis del sistema, el diagrama se desarrolla buscando una solución ideal. Durante el diseño, se usa el mismo diagrama, y se modifica para satisfacer los detalles de las implementaciones.

Caso de uso: Cargar Moléculas

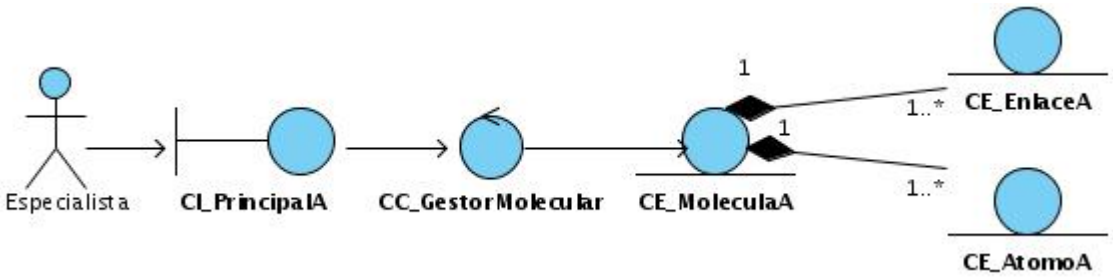


Figura 3.1: Diagrama de clases del caso de uso Cargar Moléculas.

Caso de uso: Crear Moléculas.

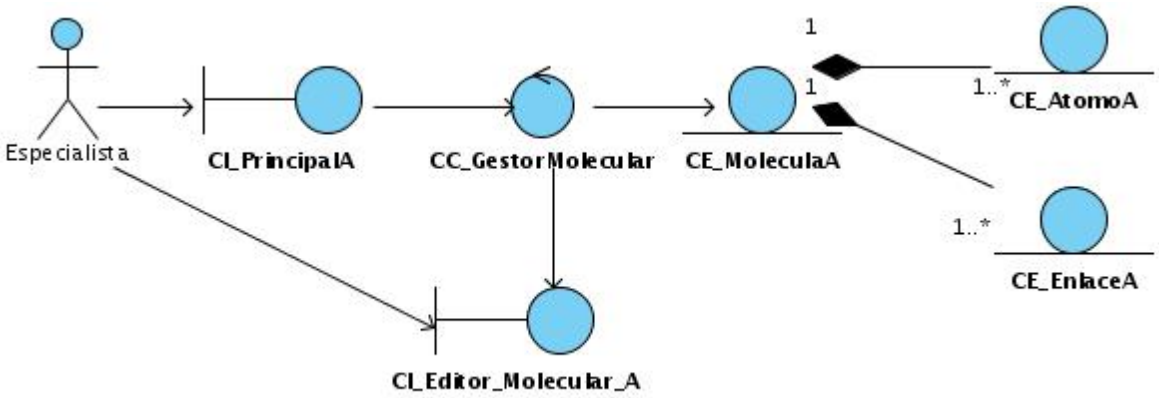


Figura 3.2: Diagrama de clases del caso de uso Crear Moléculas.

Caso de uso: Editar Moléculas.

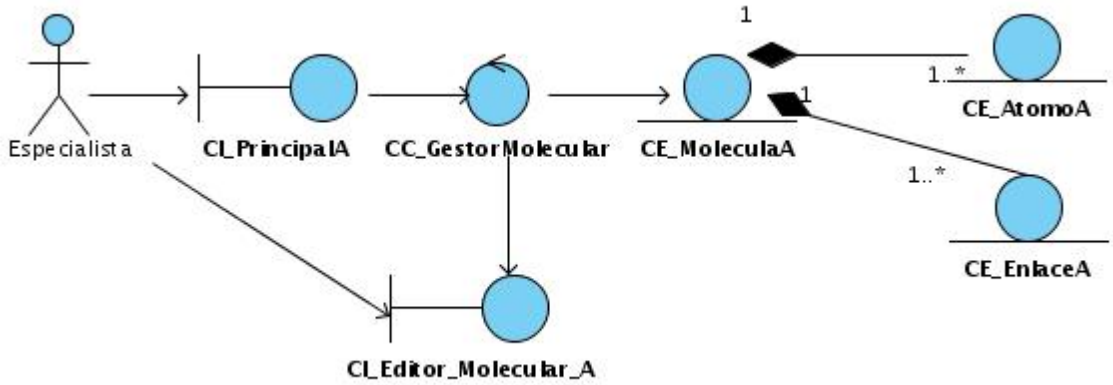


Figura 3.3: Diagrama de clase del caso de uso Editar Moléculas.

Caso de uso: Visualizar Moléculas.

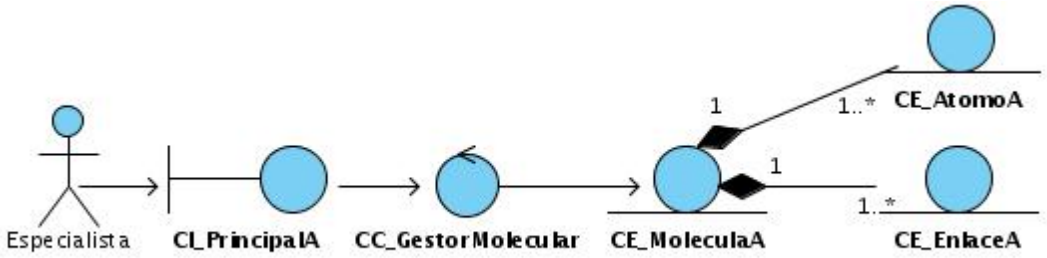


Figura 3.4: Diagrama de clases del caso de uso Visualizar Moléculas.

Caso de uso: Eliminar Moléculas.

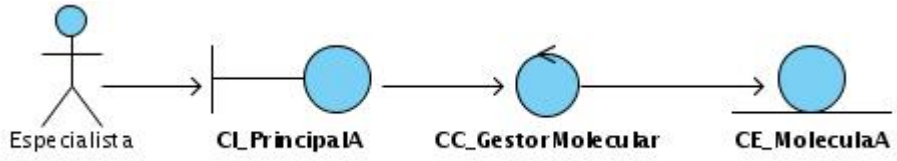


Figura 3.5: Diagrama de clases del caso de uso Eliminar Moléculas.

Caso de uso: Calcular Por ciento de Composición

Figura 3.6: Diagrama de clase del caso de uso Calcular Por ciento de Composición.

Caso de uso: Calcular Masa Molecular.

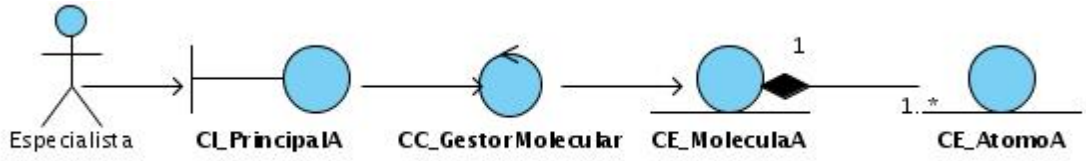


Figura 3.7: Diagrama de clases del caso de uso Calcular Masa Molecular.

Caso de Uso: Incorporar Plug-in.

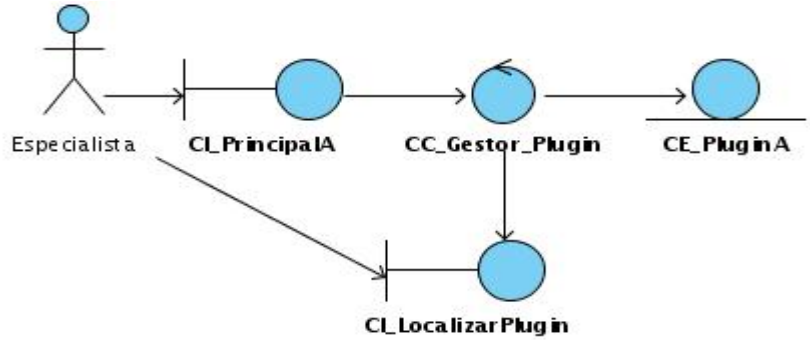


Figura 3.8: Diagrama de clases del caso de uso Incorporar Plug-in.

3.2 Arquitectura empleada

La Arquitectura del Software es la organización fundamental de un sistema formado por sus componentes, las relaciones entre ellos y el contexto en el que se implantarán, y los principios que orientan su diseño y evolución. [7]

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.

Patrón Modelo – Vista – Controlador.

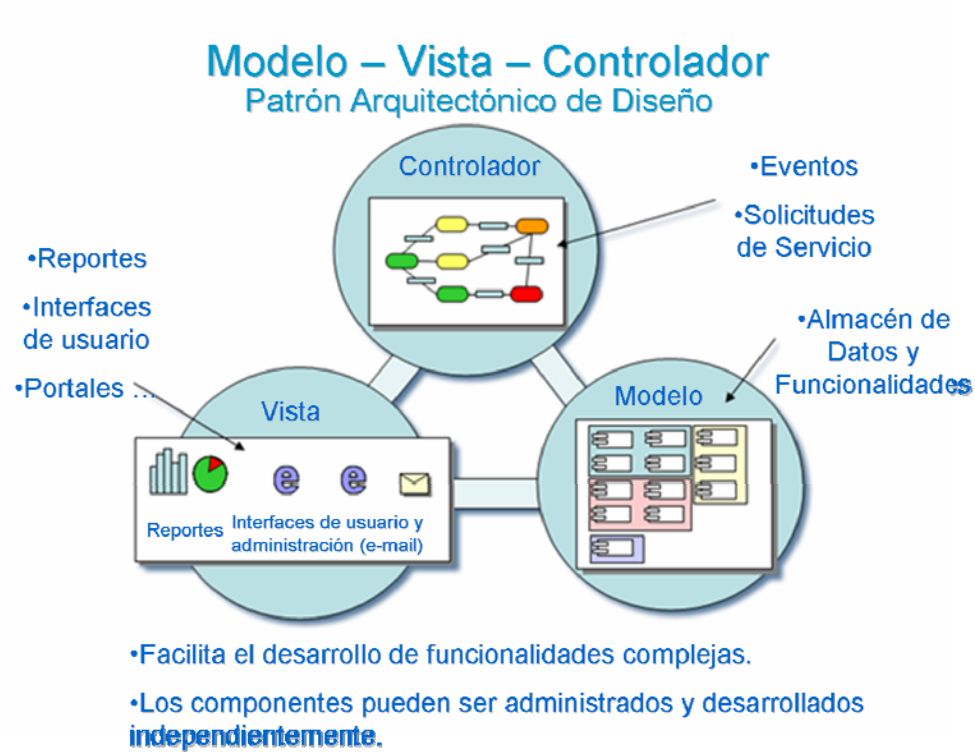


Figura 3.9: Patrón Modelo – Vista – Controlador

El patrón Modelo – Vista – Controlador está catalogado como un patrón de diseño de software donde:

- **Modelo:** Representación específica del 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 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 (“service requests”) para el modelo o la vista.

Muchas aplicaciones utilizan un mecanismo de almacenamiento persistente (como puede ser una base de datos) para almacenar los datos. Modelo – Vista – Controlador no menciona específicamente ésta capa de acceso a datos.

3.3 Modelo de diseño

Es un modelo de objeto que describe la realización de los casos de uso centrándose en el cumplimiento de los Requisitos no Funcionales. [6]

3.3.1 Diagramas de colaboración

Un diagrama de colaboración es una forma de representar interacción entre objetos. A diferencia de los diagramas de secuencia, pueden mostrar el contexto de la operación (cuáles objetos son atributos, cuáles temporales,…) y ciclos en la ejecución. [8]

Caso de uso: Cargar Moléculas.

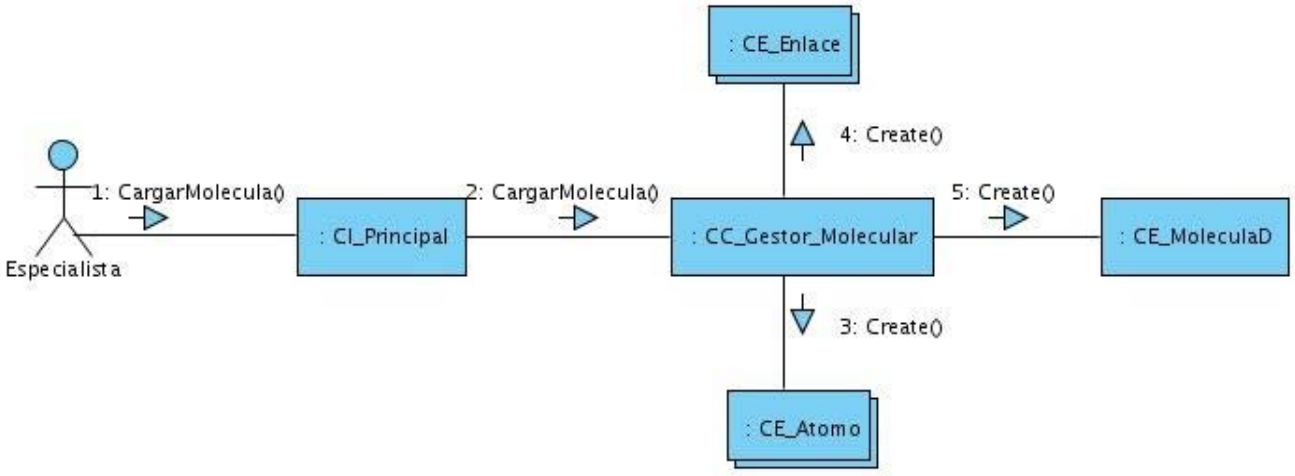


Figura 3.10: Diagrama de colaboración del caso de uso Cargar Moléculas.

Caso de uso: Crear Moléculas.

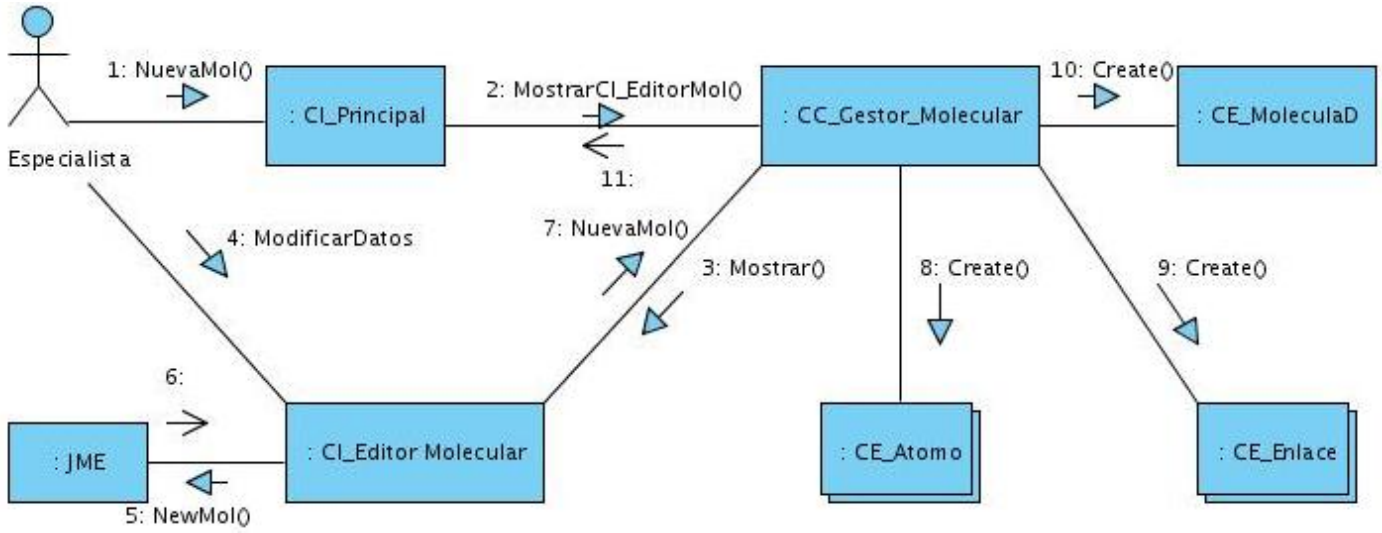


Figura 3.11: Diagrama de colaboración del caso de uso Crear Moléculas.

Caso de uso: Editar Moléculas

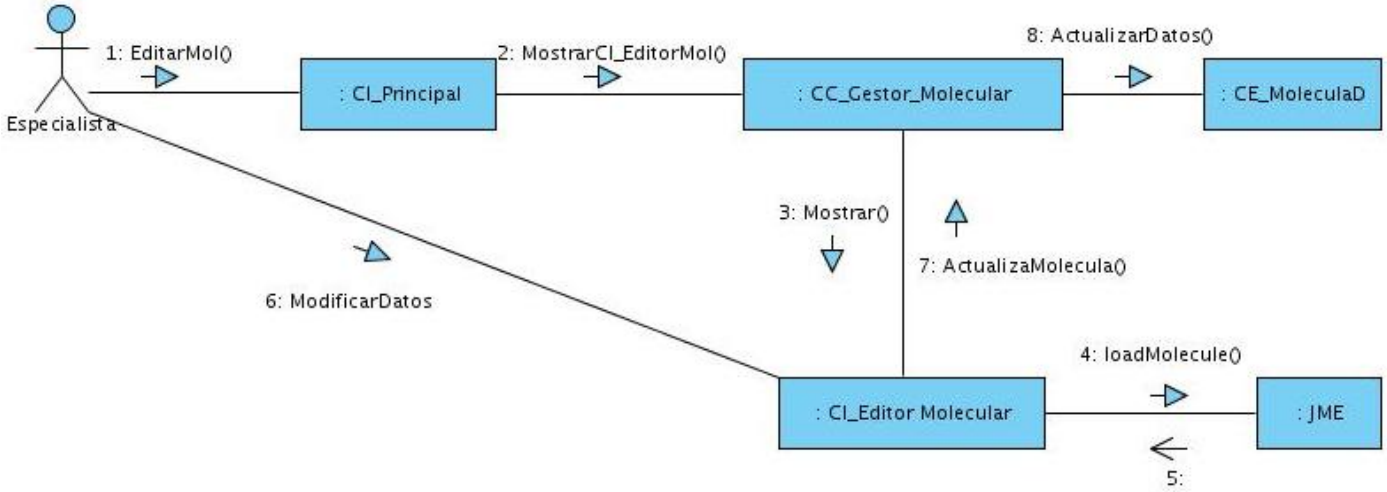


Figura 3.12: Diagrama de colaboración del caso de uso Editar Moléculas.

Caso de uso: Visualizar Moléculas

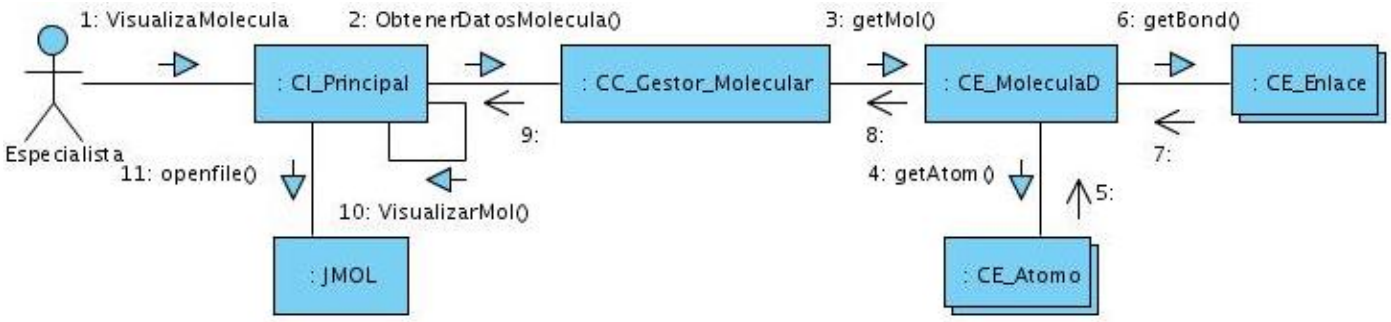


Figura 3.13: Diagrama de colaboración del caso de uso Visualizar Moléculas.

Caso de uso: Eliminar Moléculas.

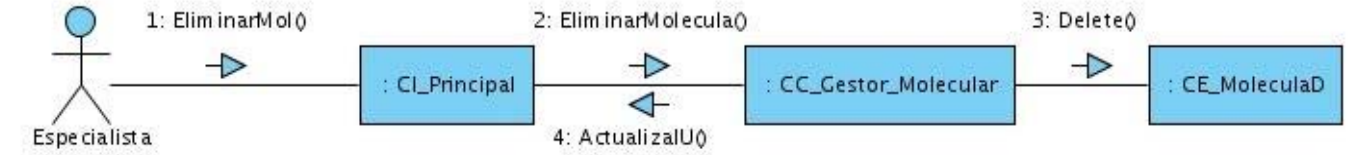


Figura 3.14: Diagrama de colaboración del caso de uso Eliminar Moléculas.

Caso de uso: Calcular Por ciento de Composición.

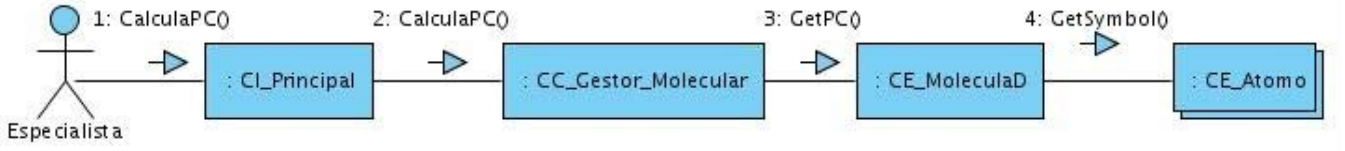


Figura 3.15: Diagrama de colaboración del caso de uso Calcular Por ciento de Composición.

Caso de uso: Calcular Masa Molecular.

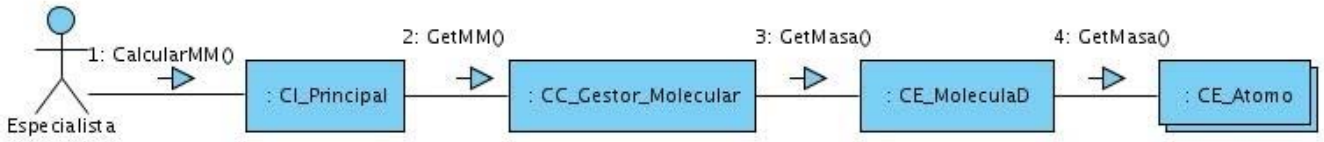


Figura 3.16: Diagrama de colaboración del caso de uso Calcular Masa Molecular.

Caso de uso: Incorporar Plug-in.

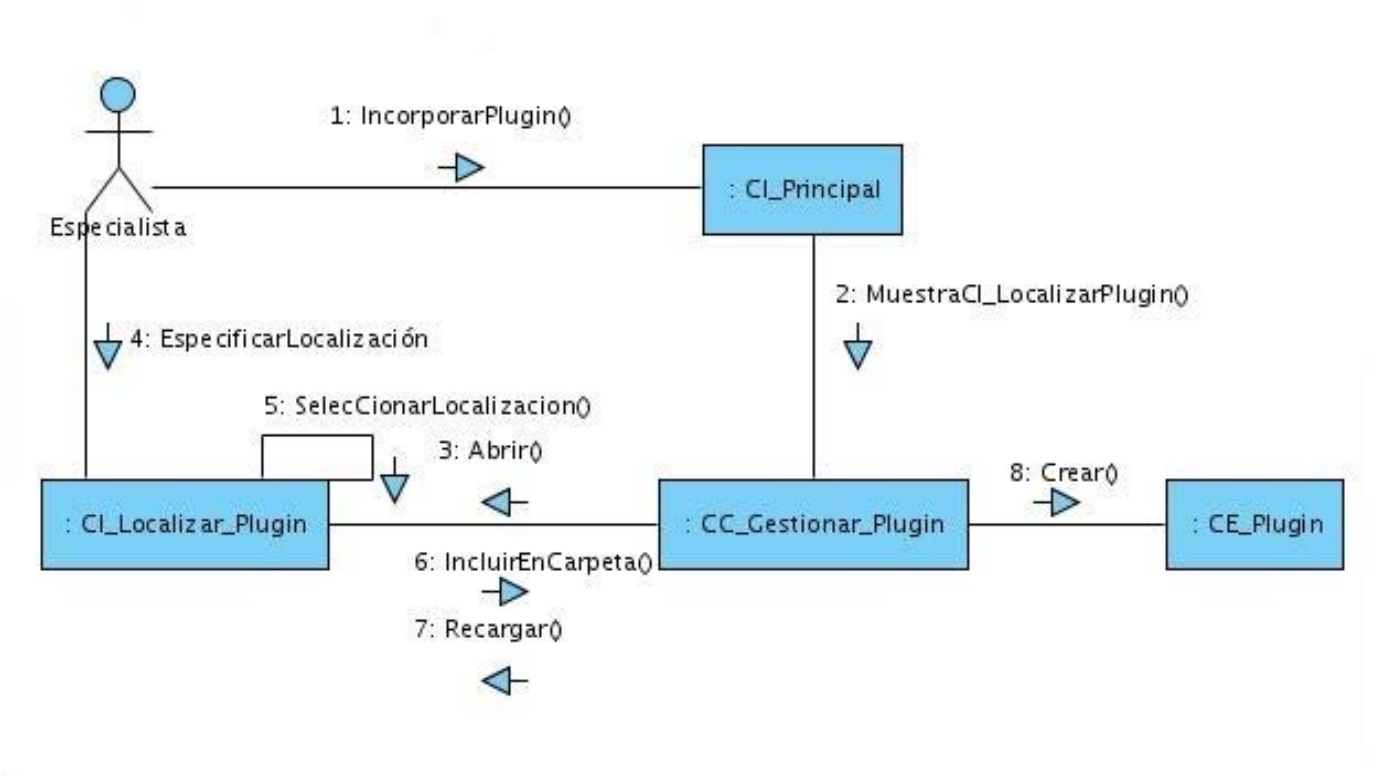


Figura 3.17: Diagrama de colaboración del caso de uso Incorporar Plug-in.

3.3.2 Diagramas de clases

Caso de uso: Cargar Moléculas.

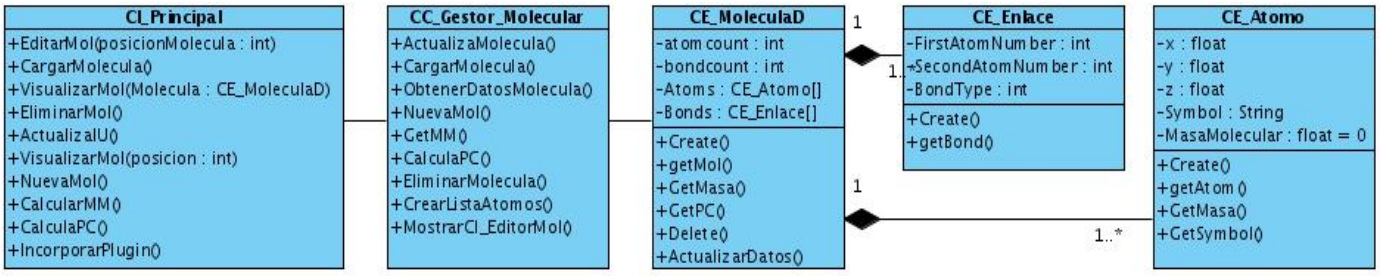


Figura 3.18: Diagrama de clases del caso de uso Cargar Moléculas.

Caso de uso: Crear Moléculas.

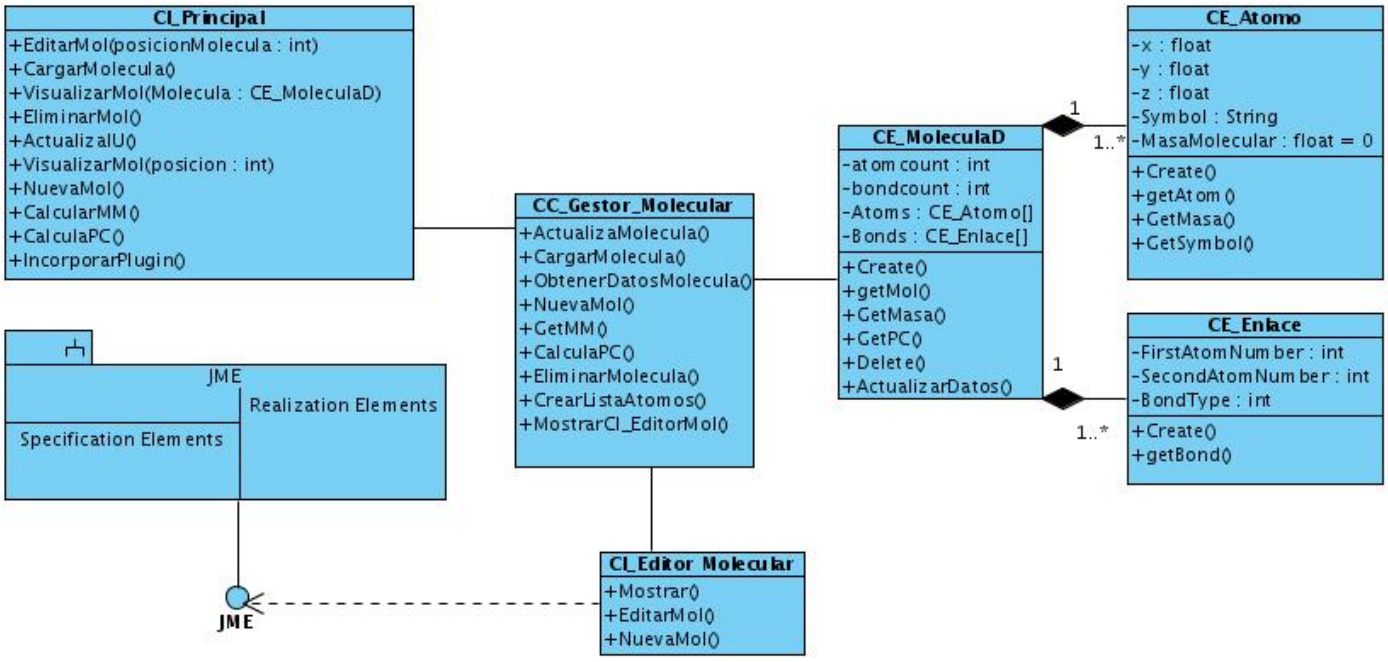


Figura 3.19: Diagrama de clases del caso de uso Crear Moléculas.

Caso de uso: Calcular Por ciento de Composición.

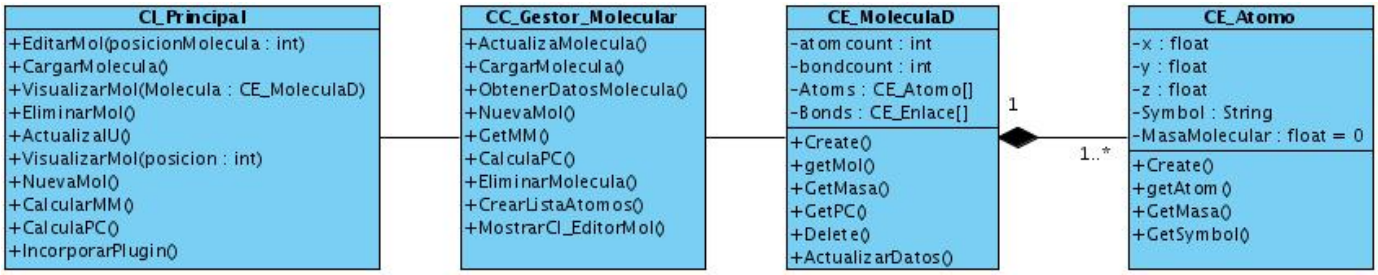


Figura 3.20: Diagrama de clases del caso de uso Calcular Por ciento de Composición.

Caso de uso: Editar Moléculas.

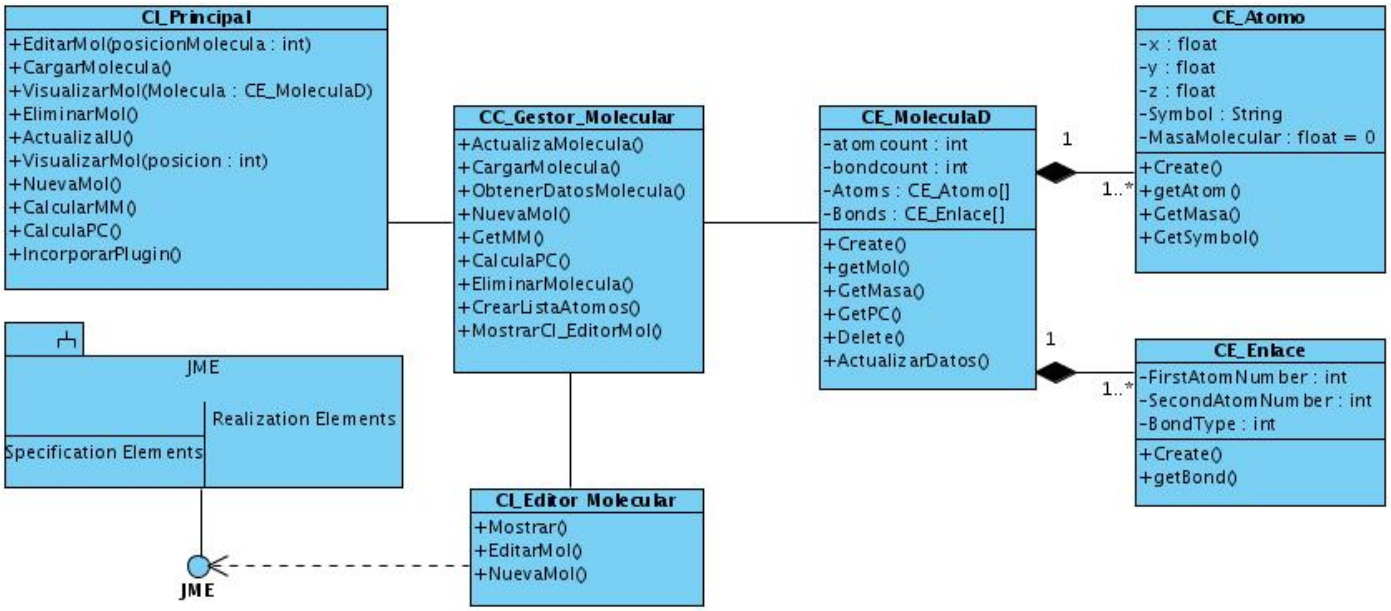


Figura 3.21: Diagrama de clases del caso de uso Editar Moléculas.

Caso de uso: Visualizar Moléculas

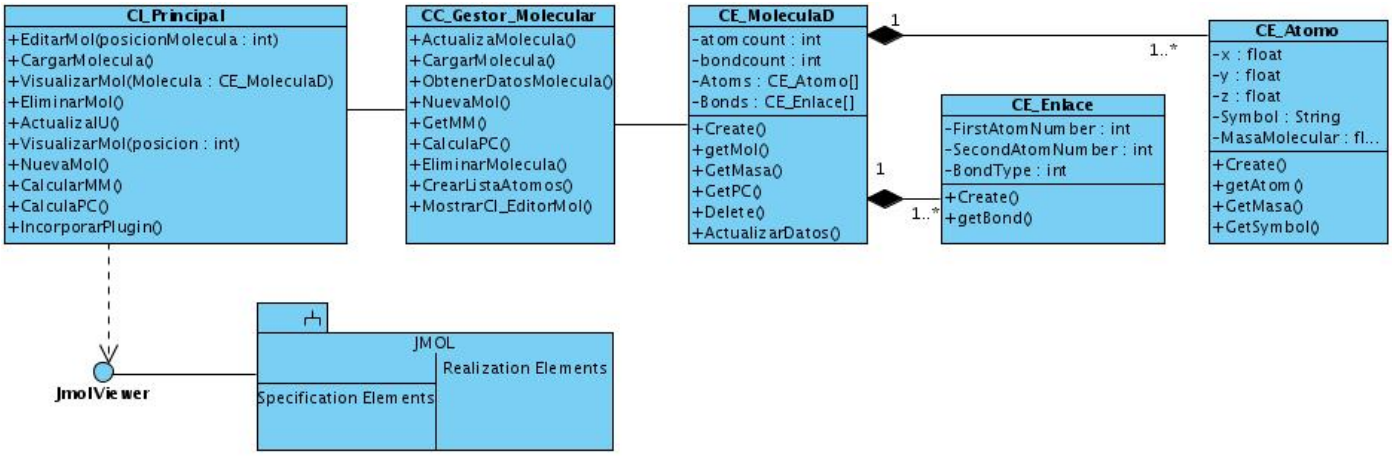


Figura 3.22: Diagrama de clases del caso de uso Visualizar Moléculas.

Caso de uso: Eliminar Moléculas

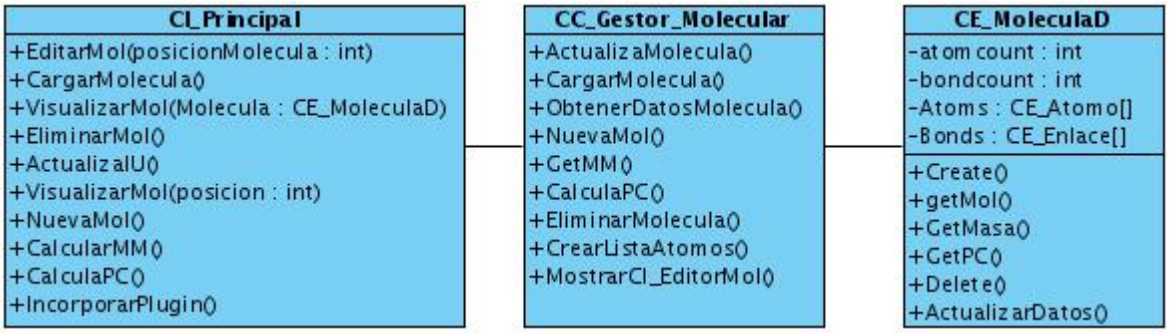


Figura 3.23: Diagrama de clases del caso de uso Eliminar Moléculas.

Caso de uso: Calcular Masa Molecular.

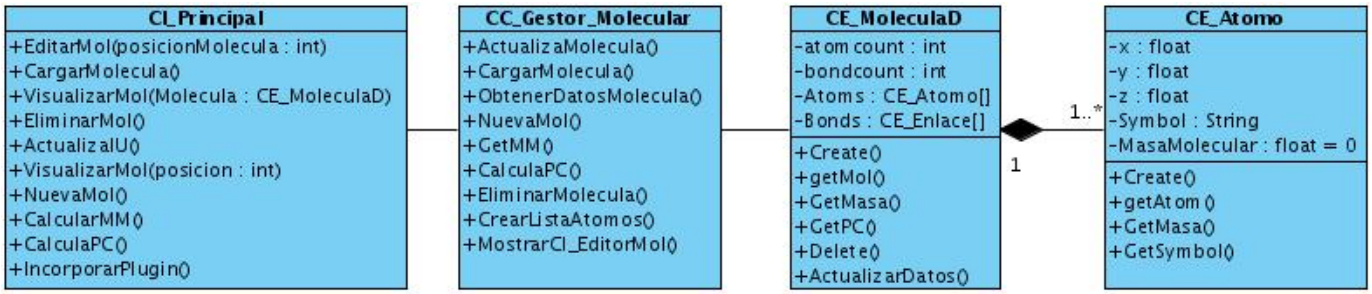


Figura 3.24: Diagrama de clases del caso de uso Calcular Masa Molecular.

Caso de uso: Incorporar Plug-in.

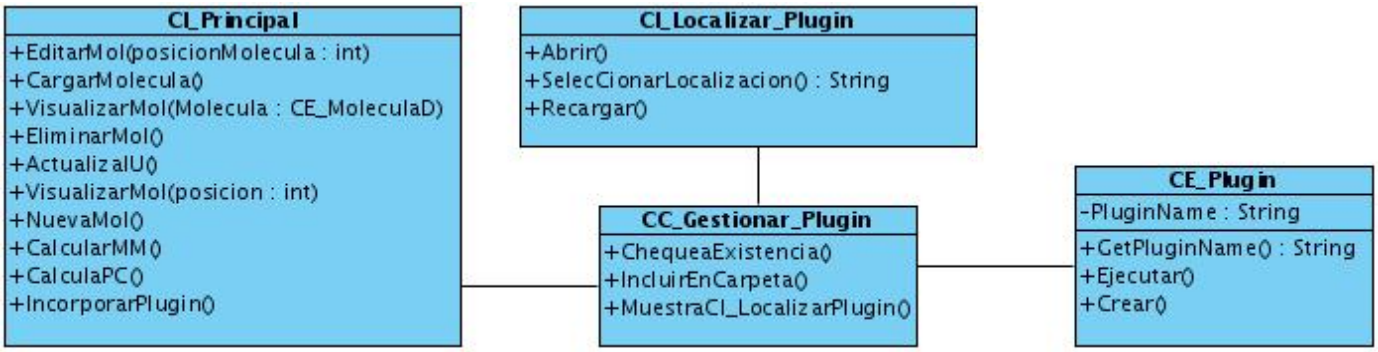


Figura 3.25: Diagrama de clases del caso de uso Incorporar Plug-in.

Diagrama de clases general

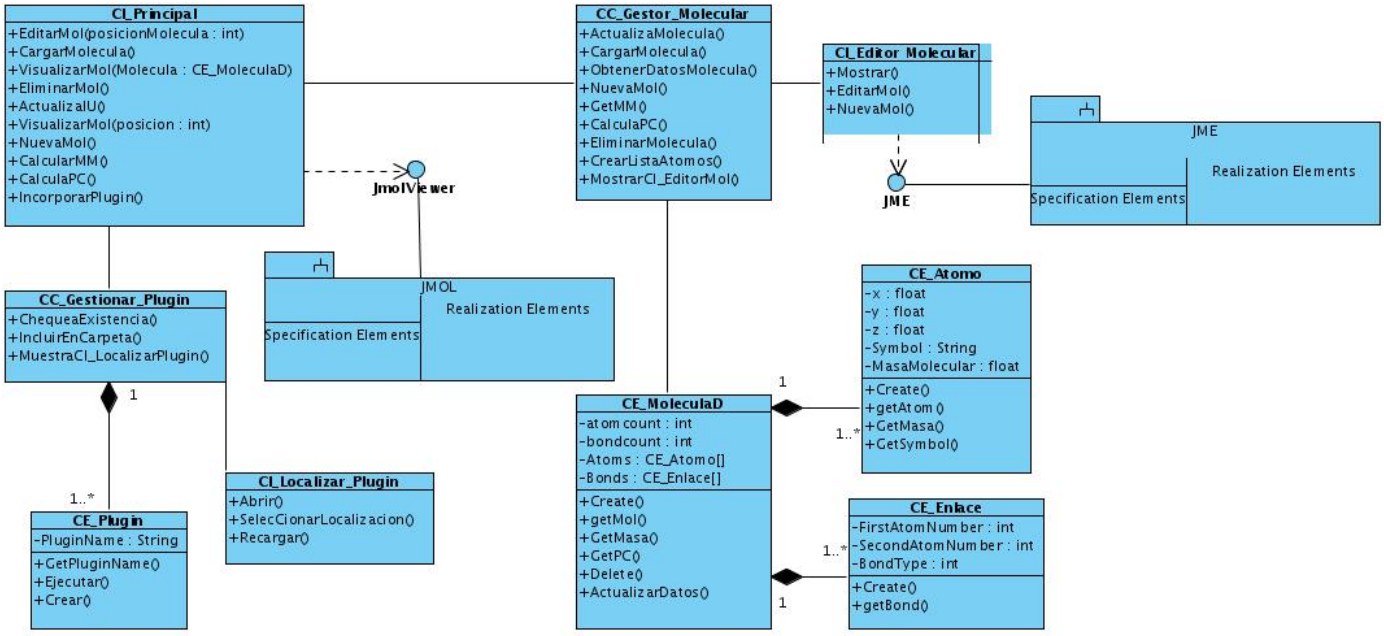


Figura 3.26: Diagrama de clases general

3.3.3 Descripción de las clases

(Ver Anexo 1)

3.3.4 Principios de diseño.

El diseño de la interfaz de una aplicación, la concepción de la ayuda y el tratamiento de excepciones tiene gran influencia en el éxito o fracaso de una aplicación. A continuación se describen los principios de diseño seguidos para el desarrollo del sistema en cuestión.

Estándares en la interfaz de la aplicación

Para garantizar la comodidad del cliente en el ambiente de trabajo al que está acostumbrado, se ha dejado a la aplicación tomar la apariencia que posea el sistema operativo sobre el que se ejecute. De esta manera no presentará contrastes ni dificultades con las personalizaciones de los escritorios de los usuarios así como mantendrá la similitud al resto de las aplicaciones del sistema. Debido a esto, los

colores y tipos de letras de la interfaz estarán de acuerdo a los temas de escritorio que tenga el usuario. No obstante, el usuario tiene la posibilidad de cambiar el color de fondo del área de trabajo como guste.

El tamaño predeterminado de la forma visual es de 800x600 puntos.

Los componentes en las interfaces visuales se encuentran ordenados a través de disposiciones de forma (Form Layout) para garantizar mantener el aspecto de posición de los componentes al cambiar de tamaño la forma visual (maximizar, restaurar).

Se provee al usuario de barras menús y herramientas donde descansan la totalidad de las funcionalidades de la aplicación. Los botones de la barra de herramientas poseen iconos de tamaño 16x16 puntos, por lo que su tamaño es de 24x24 puntos.

La interfaz provee tres áreas fundamentales. Las áreas de ficheros moleculares cargados, área de trabajo y datos del compuesto que se encuentra activo en el momento en las posiciones izquierda, derecha e inferior respectivamente. Dichas áreas pueden ser maximizadas o restauradas a su posición original para facilitar el trabajo en cada una de ellas.

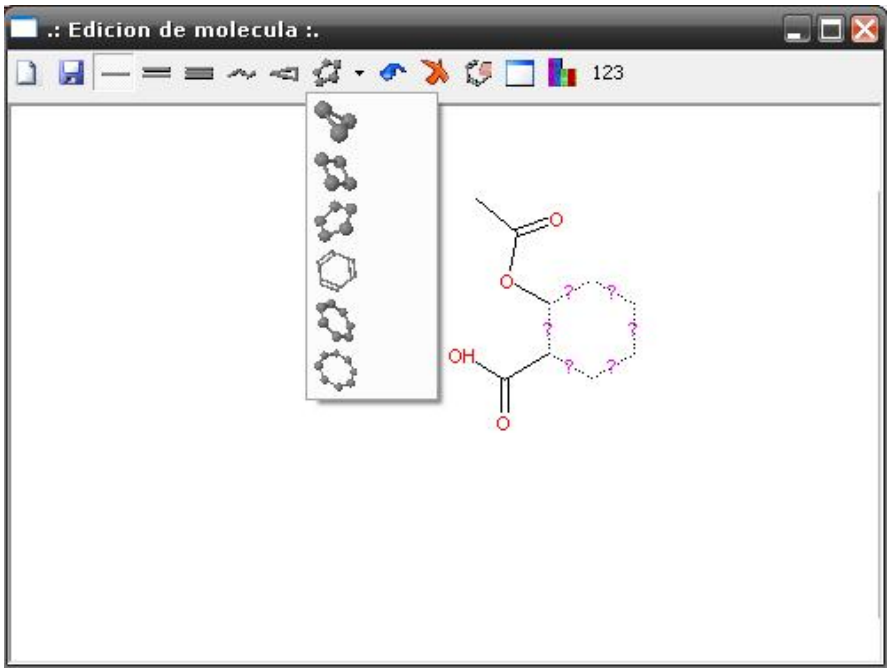


Figura 3.27: Interfaz del Editor molecular

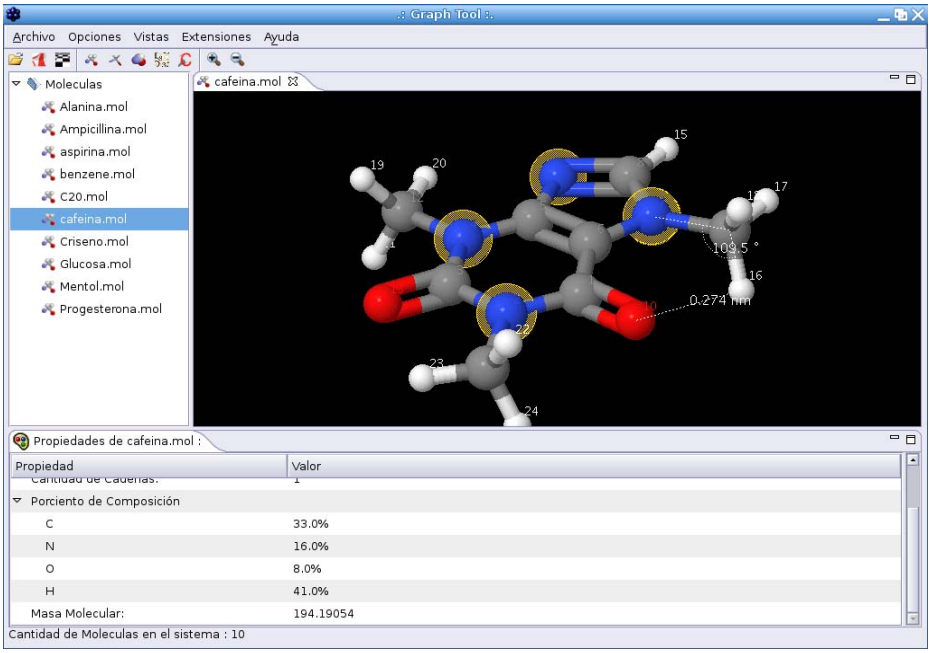


Figura 3.28: Interfaz de la aplicación en sistemas operativos Linux

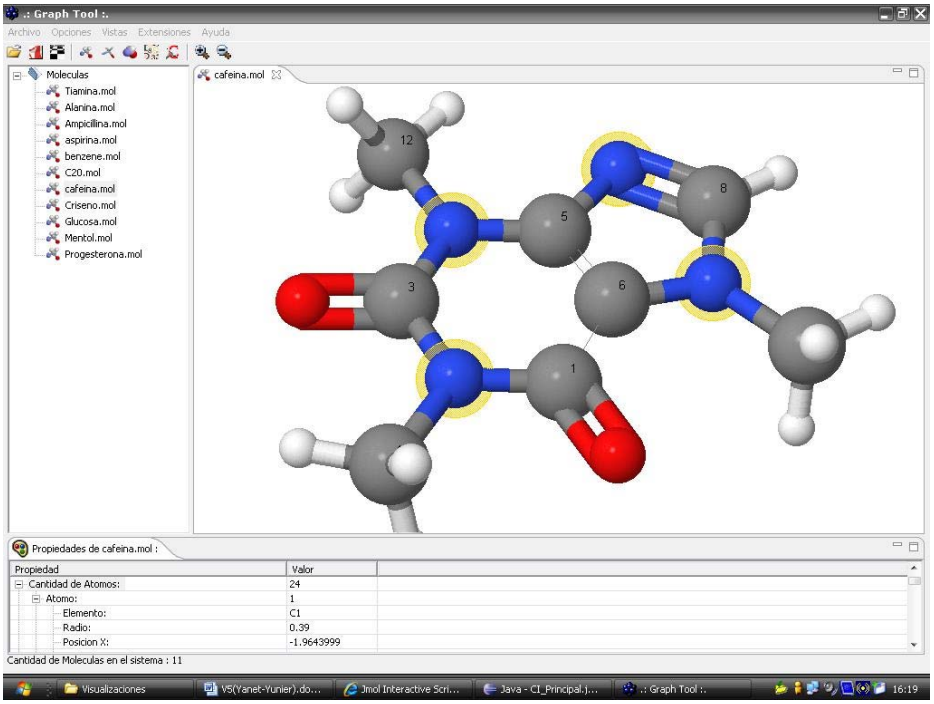


Figura 3.29: Interfaz de la aplicación en sistemas operativos Windows

Tratamiento de errores

En el diseño de la interfaz se debe tener en cuenta el tratamiento de errores logrando que los mensajes de error que emita el sistema sean de fácil comprensión para el usuario y lo más descriptivos posibles y además debe alertarlos de posibles riesgos de las operaciones que realice, debe emitir un error si trata de abrir un fichero que no posee una estructura molecular, esto se hace mediante el levantamiento de excepciones en el lenguaje Java.

Algunos de los errores más probables han sido tratados de manera que sea transparente al usuario la ocurrencia de los mismos.

Concepción de la ayuda.

La ayuda es esencial en cualquier sistema para facilitar su uso. En la barra de menú principal de la aplicación estará disponible la opción de Ayuda, donde se le explicará detalladamente al especialista temas relacionados con el uso de la aplicación. La misma constará de imágenes para una mejor comprensión, así como la posibilidad de acceder a la misma con solo presionar la tecla *F1*.

3.3.5 Diagrama de despliegue

Los Diagramas de Despliegue muestran la disposición física de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos. La vista de despliegue representa la disposición de las instancias de componentes de ejecución en instancias de nodos conectados por enlaces de comunicación.

Para aplicaciones como la que se propone, que se ejecuta en una sola máquina y todos los dispositivos con que se relaciona son los estándares (teclado, mouse), el diagrama de despliegue va a estar constituido por un nodo, por lo que no se considera relevante incluir en la investigación.

Conclusiones

Como resultado del estudio realizado en este capítulo, correspondiente a la etapa de análisis y diseño del sistema, se definieron un total de 9 clases, se desarrollaron los diagramas de clases de la aplicación

organizados por casos de uso, además del diagrama de clases general. Se describieron los principios de diseño seguidos, específicamente, los temas de estándares de la interfaz, concepción del tratamiento de errores y sistema de ayuda.

CAPÍTULO 4

CAPÍTULO 4: IMPLEMENTACIÓN Y PRUEBA

El presente capítulo comienza con el resultado del diseño, implementando el sistema en términos de componentes, así como una revisión final de las especificaciones del diseño y de la codificación, mediante la realización de pruebas, garantizando la calidad del software.

4.1 Implementación

El flujo de trabajo de implementación describe cómo los elementos del modelo del diseño se implementan en términos de componentes y cómo estos se organizan de acuerdo a los nodos específicos en el modelo de despliegue. [9]

4.1.2 Diagrama de componentes

Los diagramas de componentes describen los elementos físicos del sistema y sus relaciones. Muestran las opciones de realización incluyendo código fuente, binario y ejecutable. Los componentes representan todos los tipos de elementos software que entran en la fabricación de aplicaciones informáticas. [10]

Caso de uso: Calcular Masa Molecular.

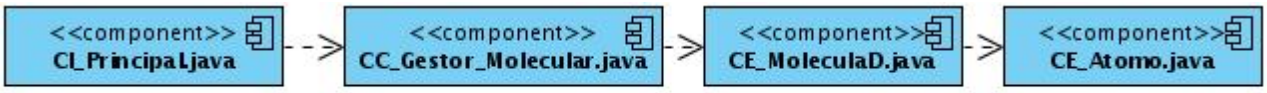


Figura 4.1: Diagrama de componentes del caso de uso Calcular Masa Molecular.

Caso de uso: Crear Moléculas.

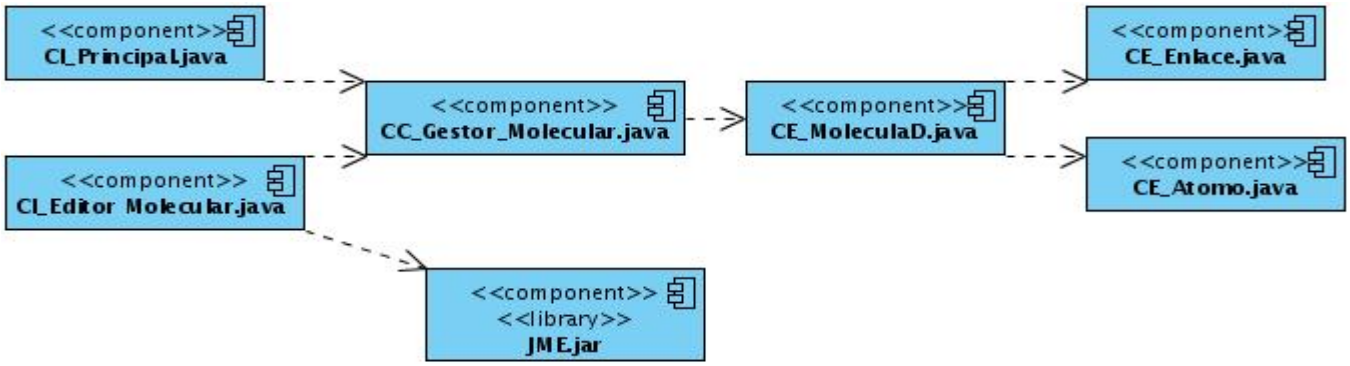


Figura 4.2: Diagrama de componentes del caso de uso Crear Moléculas.

Caso de uso: Cargar Moléculas

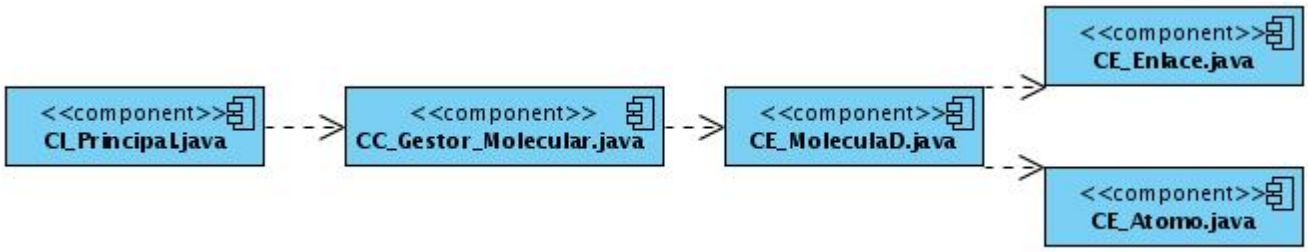


Figura 4.3: Diagrama de componentes del caso de uso Cargar Moléculas.

Caso de uso: Editar Moléculas

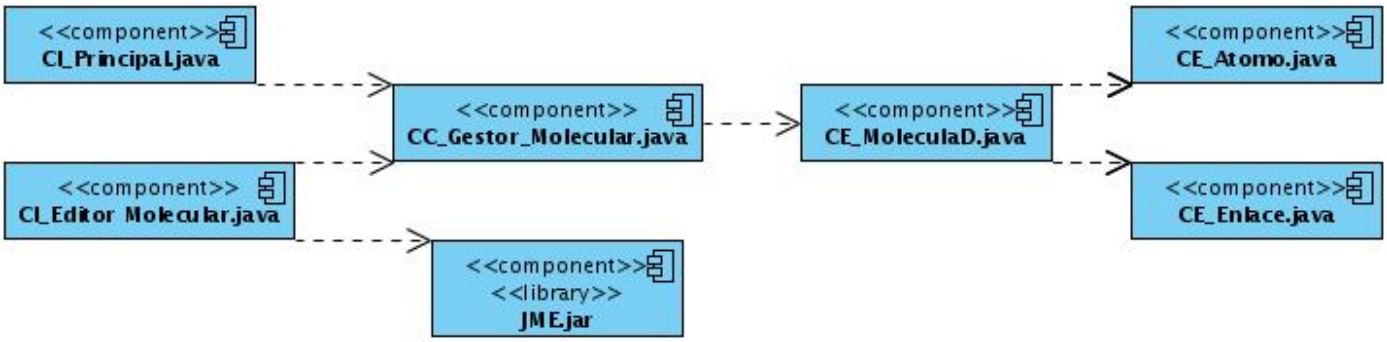


Figura 4.4: Diagrama de componentes del caso de uso Editar Moléculas.

Caso de uso: Visualizar Moléculas.

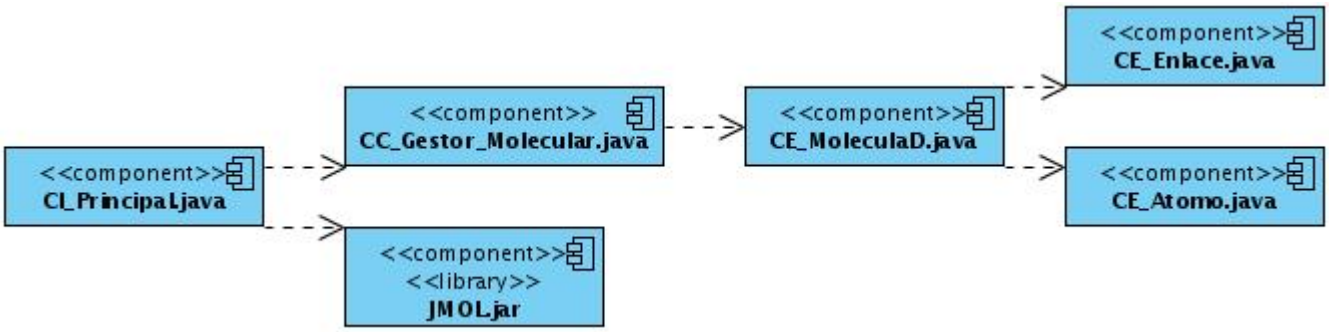


Figura 4.5: Diagrama de componentes del caso de uso Visualizar Moléculas.

Caso de uso: Calcular Por ciento de Composición.

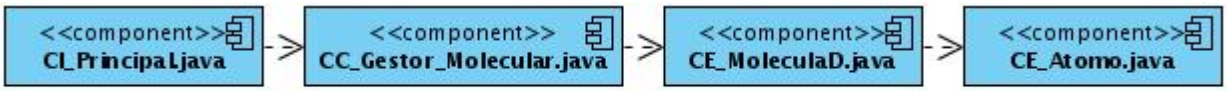


Figura 4.6: Diagrama de componentes del caso de uso Calcular Por ciento de Composición.

Caso de uso: Incorporar Plug-in.

Figura 4.7: Diagrama de componentes del caso de uso Incorporar Plug-in.

Caso de uso: Eliminar Moléculas.

Figura 4.8: Diagrama de componentes del caso de uso Eliminar Moléculas.

4.2 Prueba

Las pruebas son una actividad en la cual un sistema o componente es ejecutado bajo unas condiciones o requerimientos especificados, los resultados son observados y registrados, y una evaluación es hecha de algún aspecto del sistema o componente.

La prueba de software es un elemento crítico para la garantía de la calidad del software y representa una revisión final de las especificaciones del diseño y de la codificación.

4.2.1 Métodos de prueba

Los métodos de prueba fundamentales son: el método de caja negra y de caja blanca. A la aplicación se le realizaron las pruebas de caja negra.

Prueba de caja negra

Las pruebas que se llevan a cabo sobre la interfaz del software, por lo que los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce una salida correcta, así como que la integridad de la información externa se mantiene. Esta prueba examina algunos aspectos del modelo fundamentalmente del sistema sin tener mucho en cuenta la estructura interna del software.

Nombre del caso de uso:	Cargar Moléculas	
Nombre del caso de prueba:	Inserción de un fichero molecular.	
Entrada	**Resultados**	**Condiciones**
Se carga un fichero de tipo molecular.	La molécula cargada aparece en el panel de moléculas cargadas en el sistema.	
Nombre del caso de prueba:	Inserción de varios ficheros moleculares.	
Se cargan varios ficheros de tipo molecular.	El listado de ficheros aparece representado en el panel de moléculas cargadas en el	

	sistema.	
Nombre del caso de prueba:	Inserción de varios ficheros moleculares y no moleculares.	
Se cargan varios ficheros de diferentes tipos.	Se muestran en el panel de moléculas cargadas en el sistema, solamente los ficheros reconocidos como de formato molecular.	

Tabla 4.1 Casos de prueba del caso de uso Cargar Moléculas

Nombre del caso de uso:	Crear Moléculas	
Entrada	**Resultados**	**Condiciones**
Se seleccionó la opción Nueva Molécula de la Barra de herramientas	La aplicación mostró una interfaz para la creación de la molécula, se realizaron cambios estructurales y se guardó en formato .mol	

Tabla 4.2: Caso de prueba del caso de uso Crear Moléculas

Nombre del caso de uso:	Editar Moléculas	
Entrada	**Resultados**	**Condiciones**
Se seleccionó la molécula a editar y se presionó el botón Editar de la barra de herramientas.	El sistema mostró una ventana donde se muestra la estructura bidimensional de la molécula. Se modificó su estructura y se guardó en formato mol.	La molécula seleccionada debe estar en formato mol.

Tabla 4.3: Caso de prueba para el caso de uso Editar Moléculas

Nombre del caso de uso:	Incorporar Plug-ins	
Entrada	**Resultados**	**Condiciones**

Se seleccionó el menú Añadir extensión de la barra de menús	La aplicación muestra una interfaz para la localización de extensiones.	
Se selecciona la extensión a agregar y se presiona el botón Importar	La aplicación muestra un mensaje confirmando que la inclusión fue satisfactoria y que la extensión estará disponible la próxima vez que se inicie la aplicación.	

Tabla 4.4: Caso de prueba para el caso de uso Incorporar Plug-ins

Nombre del caso de uso:	Calcular Masa Molecular	
Entrada	**Resultados**	**Condiciones**
Se selecciona la molécula a calcular masa molecular.	El sistema devuelve un valor float correspondiente al valor de la masa molecular de la molécula.	

Tabla 4.5: Caso de prueba para el caso de uso Calcular Masa Molecular

Nombre del caso de uso:	Eliminar Moléculas	
Nombre del caso de prueba:	Eliminar usando delete.	
Entrada	**Resultados**	**Condiciones**
Se selecciona la molécula a eliminar y se presiona la tecla delete.	La molécula se elimina del listado de moléculas cargadas en el sistema, así como su visualización y sus cálculos.	
Nombre del caso de prueba:	Eliminar usando menú contextual.	
Se selecciona la molécula a eliminar y se selecciona la opción eliminar del menú contextual de la misma.	La molécula se elimina del listado de moléculas cargadas en el sistema, así como su visualización y sus cálculos.	

Tabla 4.6: Caso de prueba para el caso de uso Eliminar Moléculas

Nombre del caso de uso:	Visualizar Moléculas	
Nombre del caso de prueba:	Visualizar nueva molécula.	
Entrada	**Resultados**	**Condiciones**
Se selecciona la molécula a visualizar.	La molécula aparece en el área de trabajo.	
Nombre del caso de prueba:	Visualizar molécula previamente visualizada.	
Se selecciona una molécula previamente visualizada.	La aplicación enfoca el panel donde se encontraba visualizada la molécula anteriormente.	

Tabla 4.7: Caso de prueba para el caso de uso Visualizar Moléculas

Nombre del caso de uso:	Calcular por ciento de Composición	
Entrada	**Resultados**	**Condiciones**
Se selecciona la molécula a calcular por ciento de composición.	El sistema devuelve una lista de elementos con la composición correspondiente a la molécula.	

Tabla 4.8: Caso de prueba para el caso de uso Calcular por ciento de Composición

Conclusiones

Los diagramas de componentes fueron usados para estructurar el modelo de implementación en términos de subsistemas de implementación y mostrar las relaciones entre los elementos de implementación. El uso más importante de estos diagramas fue mostrar la estructura de alto nivel del modelo de implementación, especificando los subsistemas de implementación y sus dependencias a la hora de importar código.

Se le realizaron pruebas al sistema para evaluarlo y determinar la calidad del mismo.

CONCLUSIONES GENERALES

1. Se diseñó e implementó una aplicación multiplataforma para la visualización y edición de moléculas a partir de la librería de código abierto JMOL para la representación espacial de moléculas y el editor molecular JME.
2. La visualización de moléculas cuenta con una elevada resolución y diferentes prestaciones gráficas. Al dar acceso a las funcionalidades de representar no sólo la molécula sino también las superficies de contacto que pueda tener, la solución propuesta se convierte en un producto de gran fortaleza. Además, se implementó el acceso a la presentación de imágenes estereográficas, permitiendo al usuario adentrarse en un mundo virtual a partir del uso de gafas rojo – verde, rojo – cian y rojo – azul. Las representaciones de cintas, cohetes, cadena principal e hilos permiten una mejor representación y visualización de macromoléculas; estas visualizaciones no se encontraban disponibles en versiones anteriores de la aplicación.
3. Se implementó una arquitectura basada en plug-ins para comunicar la aplicación con el resto de los módulos, haciendo extensible la solución propuesta.
4. Se analizaron, diseñaron e implementaron algoritmos de cálculo de por ciento de composición y masa molecular.
5. Se implementó una herramienta para manejar los ficheros Manifest.mf de los paquetes .jar de Java, permitiendo a los desarrolladores de la Plataforma crear módulos integrables a la aplicación.
6. Se definió e implementó un formato para el almacenamiento de las preferencias de visualización. Dicho formato se almacena en un fichero texto, con compatibilidad para cadenas scripts de Rasmol/CHIME y cadenas propias de JMOL.

RECOMENDACIONES

Para incrementar las prestaciones de la solución propuesta se recomienda:

- Definir un conjunto de teclas de acceso rápido a las funcionalidades de visualización.
- Poner a prueba el sistema para garantizar el ajuste del mismo a los requerimientos solicitados.
- Implementar un editor molecular en tres dimensiones.
- Implementar métodos de conversión molecular para permitir guardar las moléculas en diversos formatos.

Referencias Bibliográficas

[1]. CALFA, J. B. *Modelamiento OO*, 2004. [2007]. Disponible en: http://lte.disc.ucn.cl/~jbekios/Recursos/Pregrado/IngSoftware/desarrollo/analisis/clase03.pdf

[2]. FIGUEROA, P. *Conceptos básicos en un Diagrama de Colaboración*, 2005. [2007]. Disponible en: http://www.cs.ualberta.ca/~pfiguero/soo/uml/colaboracion01.html

[3]. JACOBSON, I.; G. BOOCH*, et al. El proceso unificado de software*. Primera edición. Pearson Educación, S.A, 2000. p.

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

[5]. MICROSOFT, C. *Introducción a la Arquitectura de Software*, 2000. [2007]. Disponible en: http://www.microsoft.com/spanish/msdn/arquitectura/roadmap_arq/intro.asp

[6]. MOYA-ANEGÓN, F. D.; B. VARGAS-QUESADA*, et al. Visualización de la Información Científica Mediante Redes PFNET*, 2007]. Disponible en: http://www.cindoc.csic.es/info/fesabid/29.htm

[7]. RUMBAUGH, J.; I. JACOBSON*, et al. El Lenguaje Unificado de Modelado. Manual de Referencia.* . Rational Software Corporation, 2000. p. 84-7829-037-0.

[8]. TOROSSI, G. *El Proceso Unificado de Desarrollo de Software*, 2003. [2007]. Disponible en: http://www.ecomchaco.com.ar/UTN/disenodesistemas/apuntes/oo/ApunteRUP.pdf

[9]. UCITEVE, P.; D. D. C. AUDIOVISUAL*, et al. Fase de Elaboración. Análisis - Diseño*, 2005.

[10]. ---. *Fase de inicio. Levantamiento de requisitos.*, 2005.

BIBLIOGRAFÍA

[1]. *ACD/ChemSketch: Overview*. 2006. [Disponible en: http://www.acdlabs.com/products/chem_dsn_lab/chemsketch/

[2]. *Chemistry-Software.com: Specializing in organic chemistry software, mass stereoscopy, quality control software and general chemistry software.* Disponible en: http://www.chemistry-software.com

[3]. *Diseño Molecular*. 2006]. Disponible en: http://personal5.iddeo.es/pefeco/dibmol.html

[4]. *Herramientas computacionales para la Visualización Molecular. Una posibilidad educativa*. Disponible en: http://www.accefyn.org.co/bioinfo/pdf/articulo-acad2.PDF

[5]. *JMOL*. 2007. [2007]. Disponible en: http://jmol.sourceforge.net/

[6]. *UCB Enhanced Rasmol*. 2000. [2006]. Disponible en: http://mc2.cchem.berkeley.edu/Rasmol

[7]. ABAGYAN, R.; E. RAUSH*, et al. ICM User's Guide: Molecular Editor*, 2006. [2006]. Disponible en: http://www.molsoft.com/gui/molecule-editor.html

[8]. ALVAREZ, M. A. *Qué es la programacion orientada a objetos*, 2003. [2007]. Disponible en: http://www.desarrolloweb.com/articulos/499.php

[9]. CALFA, J. B. *Modelamiento OO*, 2004. [2007]. Disponible en: http://lte.disc.ucn.cl/~jbekios/Recursos/Pregrado/IngSoftware/desarrollo/analisis/clase03.pdf

[10]. COLEMAN, F. *Hyperchem Help*, 1998. [2006]. Disponible en: http://www.wellesley.edu/Chemistry/chem241/hchemlab.html

[11]. CORTÉS, P. F. *Los 20 Mejores programas para la docencia en ESO y bachillerato*, 2006]. Disponible en: http://www.computerhuesca.es/ fvalles/software/software.htm

[12]. DUQUE, M. V. M. *Análisis de Mercados para la utilización de plataformas de biodiversidad en la región andina mediante aplicaciones de tecnología.* Caracas, Venezuela, Biotecnology Center of Excellence Corporation, 2003. 528.

[13]. DÜRSTELER, J. C. *Visualización Molecular*. *Inf@Vis!*, 2002. 76.

[14]. ERL, P. *JME Molecular Editor*, Novartis Institutes for BioMedical Research, 2004. [2006]. Disponible en: http://www.molinspiration.com/jme/

[15]. FERNÁN, J. M. *Rasmol en Español*. Disponible en: http://www.ugr.es/gebqmed/esrasmol.html

[16]. FIGUEROA, P. *Conceptos básicos en un Diagrama de Colaboración*, 2005. [2007]. Disponible en: http://www.cs.ualberta.ca/~pfiguero/soo/uml/colaboracion01.html

[17]. FOUNDATION, T. E. *Eclipse - an open development platform*, 2007. [2007]. Disponible en: http://www.eclipse.org/

[18]. ---. *SWT: The Standard Widget Toolkit*, 2007. [2007]. Disponible en: http://www.eclipse.org/swt/docs.php

[19]. GMBH, M. N. *Generation of 3d Coordinates for Rational Drug Design*, 2006]. Disponible en: http://www.mol-net.de/software/category/gen3dcoord.html

[20]. INC., H. *HyperChem 7.5: What's New*, 2003. [2006]. Disponible en: http://www.hyper.com/products/Professional/index.htm

[21]. JACOBSON, I.; G. BOOCH*, et al. El proceso unificado de software*. Primera edición. Pearson Educación, S.A, 2000. p.

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

[23]. 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

[24]. MARTZ, E. *Rasmol Quick Start*, 1996. [2006]. Disponible en: http://www.umass.edu/microbio/rasmol/rasquick.htm

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

[26]. MICROSOFT, C. *Introducción a la Arquitectura de Software*, 2000. [2007]. Disponible en: http://www.microsoft.com/spanish/msdn/arquitectura/roadmap_arq/intro.asp

[27]. MOYA-ANEGÓN, F. D.; B. VARGAS-QUESADA*, et al. Visualización de la Información Científica Mediante Redes PFNET*, 2007]. Disponible en: http://www.cindoc.csic.es/info/fesabid/29.htm

[28]. MUMFORD, C. *The Chemsketch Windfall*. *School Science Review*, 2003. 84: 19-23.

[29]. NETBEANS. *NetBeans IDE Docs & Support Resources - Tutorials, Guides and Articles*, 2007. [2007]. Disponible en: http://www.netbeans.org/kb/index.html

[30]. ROBLES, G. and J. FERRER. *Programación eXtrema y Software Libre*, 2002. [2007]. Disponible en: http://es.tldp.org/Presentaciones/200211hispalinux/ferrer/robles-ferrer-ponencia-hispalinux-2002.html

[31]. RUMBAUGH, J.; I. JACOBSON*, et al. El Lenguaje Unificado de Modelado. Manual de Referencia.* . Rational Software Corporation, 2000. p. 84-7829-037-0.

[32]. SCUSE, D. *Getting Started with Eclipse 3.1 and the SWT*, Department of Computer Science, University of Manitoba, 2005. [2007]. Disponible en: http://www.cs.umanitoba.ca/~eclipse/

[33]. TOROSSI, G. *El Proceso Unificado de Desarrollo de Software*, 2003. [2007]. Disponible en: http://www.ecomchaco.com.ar/UTN/disenodesistemas/apuntes/oo/ApunteRUP.pdf

[34]. UCITEVE, P.; D. D. C. AUDIOVISUAL*, et al. Fase de Elaboración. Análisis - Diseño*, 2005.

[35]. ---. *Fase de inicio. Levantamiento de requisitos.*, 2005.

[36]. VASQUEZ, J. and A. GARCÍA. *Rasmol y Chime*, 2006]. Disponible en: http://www.accefyn.org.co/rasmol/indice.htm#rasmolchime

ANEXOS

Anexo 1: Descripción de las clases

Descripción de la Clase Interfaz CI_Principal

Nombre: CI_Principal	
Tipo de clase: Interfaz	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	**EditarMol(posicionMolecula: int)**
Descripción:	Llama al método EditarMol de la clase CI_Editor_Molecular.
Nombre:	**CargarMolecula()**
Descripción:	Llama al método CargarMolecula de la clase CC_Gestor_Molecular.
Nombre:	**VisualizarMol(posicion: int)**
Descripción:	Visualiza la molécula que se encuentra en la posición especificada.
Nombre:	**VisualizarMol(Molecula: CE_MoleculaD)**
Descripción:	Visualiza una molécula pasada por parámetros.
Nombre:	**NuevaMol()**
Descripción:	Llama al método NuevaMol de la clase CC_Gestor_Molecular.
Nombre:	**CalcularMM()**
Descripción:	Llama al método GetMM de la clase CC_Gestor_Molecular.
Nombre:	**CalculaPC()**
Descripción:	Llama al método CalculaPC de la clase CC_Gestor_Molecular.
Nombre:	**EliminarMol()**
Descripción:	Llama al método EliminarMol de la clase CC_Gestor_Molecular.
Nombre:	**ActualizaIU()**
Descripción:	Actualiza la interfaz de usuario.
Nombre:	**IncorporarPlugin()**
Descripción:	Llama al método MuestraCI_IncorporarPlugin() de la clase CC_Gestionar_Plugin

Tabla 1: Descripción de la Clase Interfaz CI_Principal.

Descripción de la Clase interfaz CI_Editor_Molecular

Nombre: CI_Editor_Molecular	
Tipo de clase: Interfaz	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	**EditarMol(Molecula):CEMolecula**
Descripción:	Edita moléculas pasadas por parámetros y devuelve la molécula editada a la clase

	interfaz CI_Principal.
Nombre:	**NuevaMol():CE_MoleculaD**
Descripción:	Crea una nueva molécula devolviéndola a la clase interfaz CI_Principal
Nombre:	**Mostrar()**
Descripción:	Muestra la forma visual de la clase.

Tabla 2: Descripción de la Clase interfaz CI_Editor_Molecular.

Descripción de la Clase interfaz CI_Localizar_Plugin.

Nombre: CI_Localizar_Plugin	
Tipo de clase: Interfaz	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	**Abrir()**
Descripción:	Muestra la forma visual de la clase CI_Localizar_Plugin
Nombre:	**SeleccionarLocalizacion() : String**
Descripción:	Selecciona la localización donde se encontrarán los plugins a incluir en la aplicación
Nombre:	**Recargar()**
Descripción:	Recarga la interfaz refrescando la lista de plugins encontrados en la localización seleccionada.

Tabla 3: Descripción de la Clase interfaz CI_Localizar_Plugin.

Descripción de la Clase Controladora CC_Gestor_Molecular

Nombre: CC_Gestor_Molecular	
Tipo de clase: Controladora	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	**ActualizaMolecula(Molecula: CE_Molecula)**
Descripción:	Actualiza los datos de la molécula pasada por parámetros.
Nombre:	**CargarMolecula()**
Descripción:	Carga las moléculas al sistema para su posterior uso.
Nombre:	**ObtenerDatosMolecula(posicion:int): CE_MoleculaD**
Descripción:	Obtiene información de la molécula en la posición especificada
Nombre:	**NuevaMol():CE_MoleculaD**
Descripción:	Crea una CE_Molecula con los parámetros obtenidos del CI_Editor_Molecular.
Nombre:	**GetMM(posicion: int): float [ ]**
Descripción:	Obtiene la Masa Molecular de una molécula en una posición dada.
Nombre:	**CalculaPC(posicion: int): float [ ]**

Descripción:	Obtiene Porciento de Composición de una molécula en una posición dada.
Nombre:	**EliminarMolecula(posicion: int)**
Descripción:	Elimina la molécula en la posición especificada.
Nombre:	**CrearListaAtomos(MolString: char[ ]): CE_Atomo [ ]**
Descripción:	Crea la lista de átomos por los que va a estar compuesta la molécula.
Nombre:	**MostrarCI_EditorMol()**
Descripción:	Llama al método Mostrar() de la CI_Editor_Molecular

Tabla 4: Descripción de la Clase Controladora CC_Gestor_Molecular.

Descripción de la Clase Controladora CC_Gestionar_Plugin

Nombre: CC_Gestionar_Plugin	
Tipo de clase: Controladora	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	**ChequeaExistencia(NombrePlugin : String) : boolean**
Descripción:	Chequea la existencia en la aplicación de un plug-in con el nombre del seleccionado.
Nombre:	**IncluirEnCarpeta(DireccionPlugin : String) : void**
Descripción:	Incluye el plug-in seleccionado a la carpeta de extensiones de la aplicación.
Nombre:	**MuestraCI_LocalizarPlugin()**
Descripción:	llama al método Mostrar de la CI_Localizar_Plugin

Tabla 5: Descripción de la Clase Controladora CC_Gestionar_Plugin.

Descripción de la Clase Entidad CE_MoleculaD.

Nombre: CE_MoleculaD	
Tipo de clase: Entidad	
Atributo	**Tipo**
atomcount	int
bondcount	int
Atoms	CE_Atomo[]
Bonds	CE_Enlace[]
Para cada responsabilidad:	
Nombre:	**Create(atomcount: int, bondcount: int, Atoms: CE_Atomo [], Bonds: CE_Enlace[])**
Descripción:	Se crea la molécula con los parámetros pasados
Nombre:	**getMol(): CE_MoleculaD**
Descripción:	Devuelve los datos almacenados.
Nombre:	**GetMasa(): float**
Descripción:	Llama al método GetMasa de la clase CE_Atomo

Nombre:	**GetPC(): float []**
Descripción:	Devuelve el Porciento de Composición
Nombre:	**Delete()**
Descripción:	Elimina los datos almacenados.
Nombre:	**ActualizarDatos(Molecula : CE_Molecula = Molecula)**
Descripción:	Asigna los atributos de una nueva molécula a los atributos privados de la propia clase

Tabla 6: Descripción de la Clase Entidad CE_MoleculaD.

Descripción de la Clase Entidad CE_Atomo

Nombre: CE_Atomo	
Tipo de clase: Entidad	
Atributo	**Tipo**
x	float
y	float
z	float
Symbol	char[]
MasaMolecular	float = 0
Para cada responsabilidad:	
Nombre:	**Create(x: float, y: float, z: float, Symbol: char []):CE_Atomo**
Descripción:	Crea el átomo con los parámetros pasados.
Nombre:	**getAtom(): CE_Atomo**
Descripción:	Devuelve el átomo.
Nombre:	**GetMasa()**
Descripción:	Devuelve la Masa Molecular.
Nombre:	**GetSymbol() : char []**
Descripción:	Devuelve el símbolo que tiene ese átomo.

Tabla 7: Descripción de la Clase Entidad CE_Atomo.

Descripción de la Clase Entidad CE_Enlace.

Nombre: CE_Enlace	
Tipo de clase: Entidad	
Atributo	**Tipo**
FirstAtomNumber	int
SecondAtomNumber	int
BondType	int
Para cada responsabilidad:	
Nombre:	**Create(FirstAtomNumber: int, SecondAtomNumber: int, BondType: int): CE_Enlace**

Descripción:	Crea el enlace con los parámetros pasados.
Nombre:	**getBond(): CE_Enlace**
Descripción:	Devuelve el enlace.

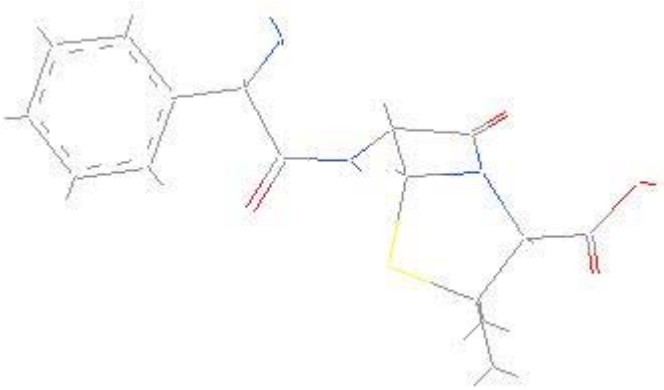
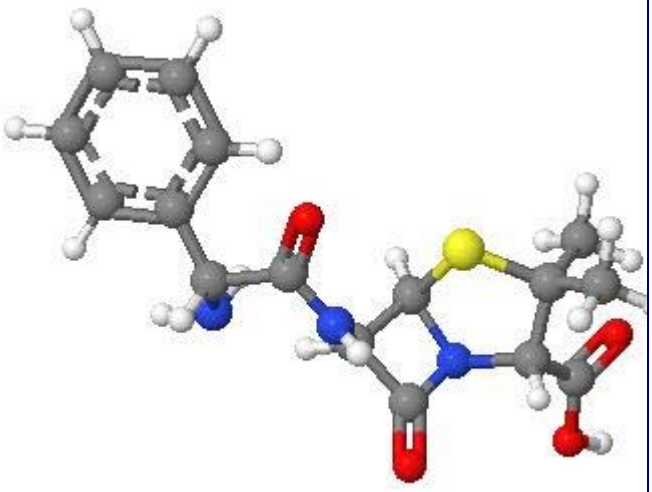
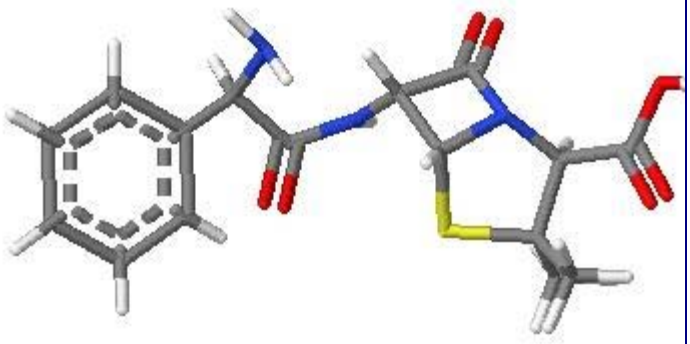
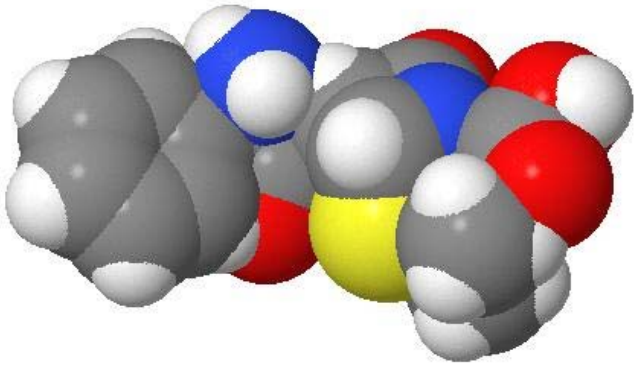
Tabla 8: Descripción de la Clase Entidad CE_Enlace.

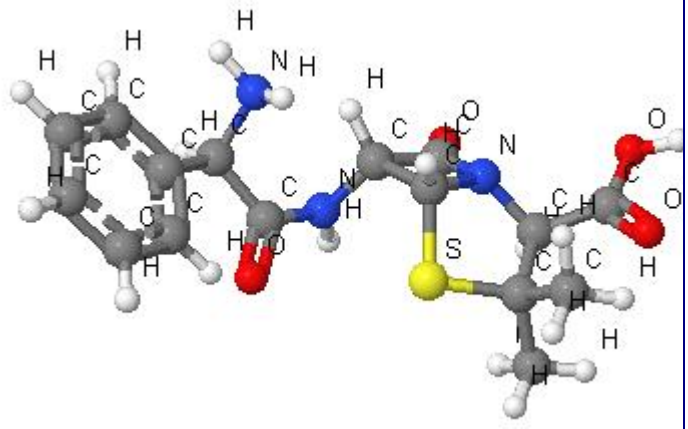
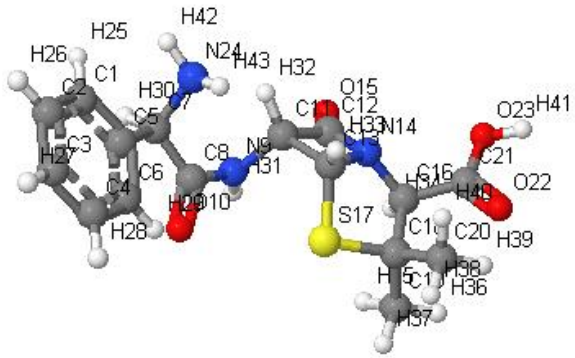
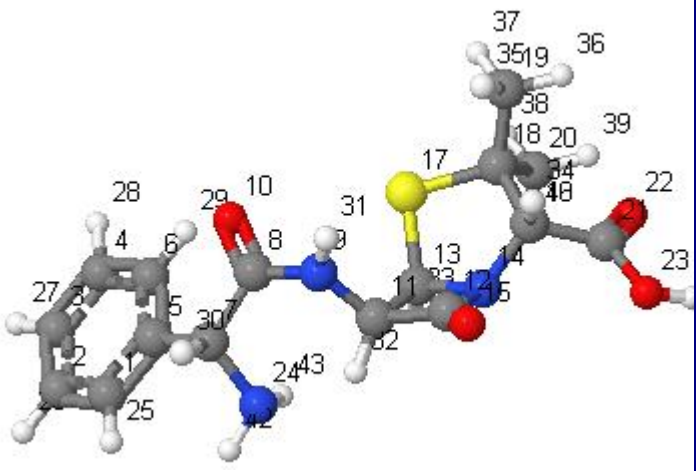
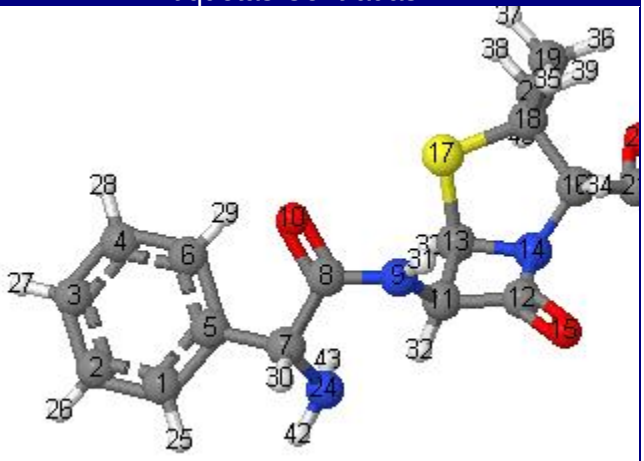
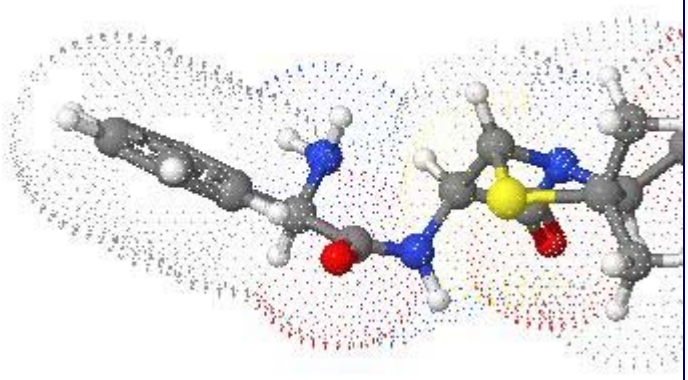
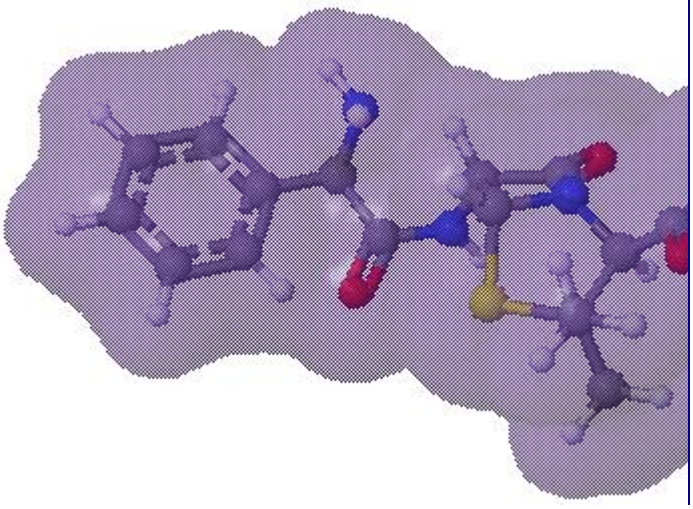
Descripción de la Clase Entidad CE_Plugin

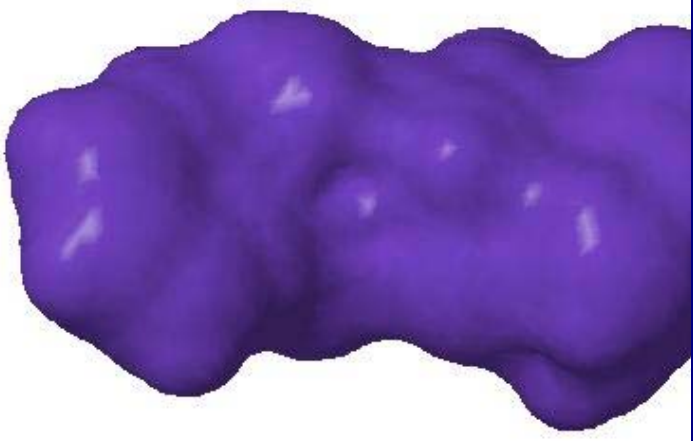
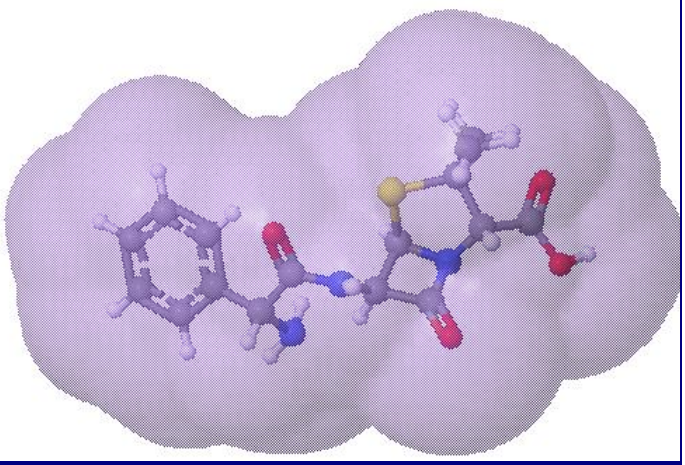
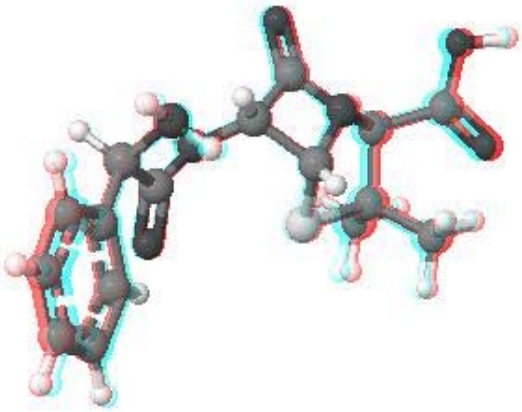
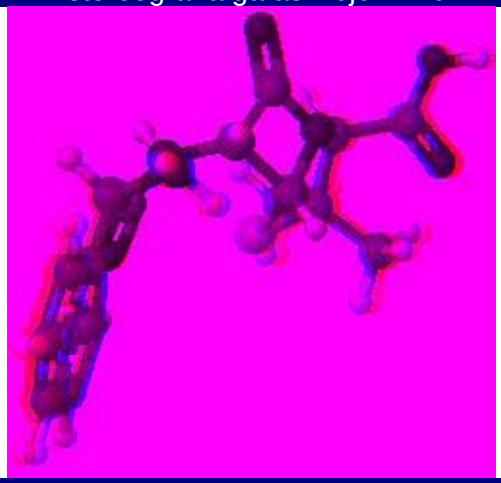
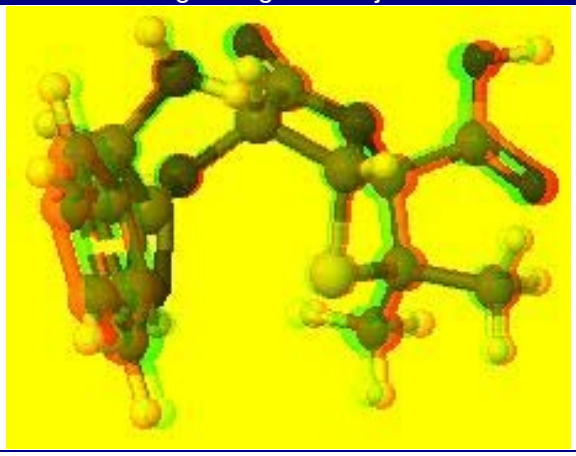
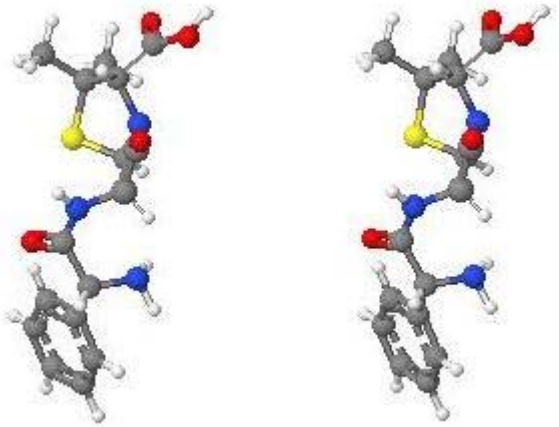
Nombre: CE_Plugin	
Tipo de clase: Entidad	
Atributo	**Tipo**
PluginName	String
Para cada responsabilidad:	
Nombre:	**GetPluginName() : String**
Descripción:	Devuelve el nombre del plug-in.
Nombre:	**Ejecutar()**
Descripción:	Ejecuta las acciones a realizar por el plug-in dependiendo de la funcionalidad que ofrece.
Nombre:	**Crear()**
Descripción:	Crea una nueva clase CE_Plugin para ser asignada a la hora de cargarla dinámicamente.

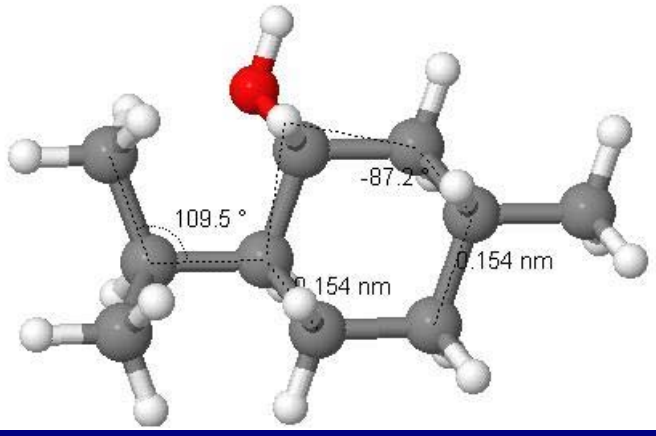
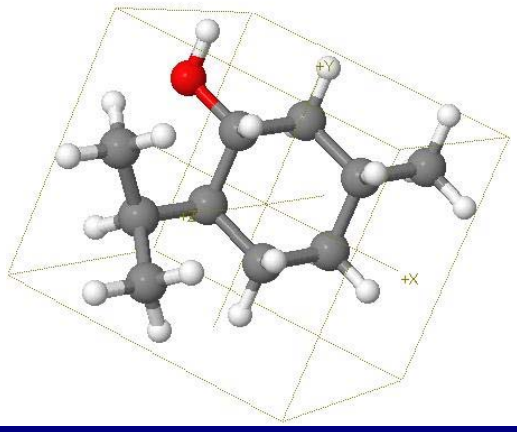
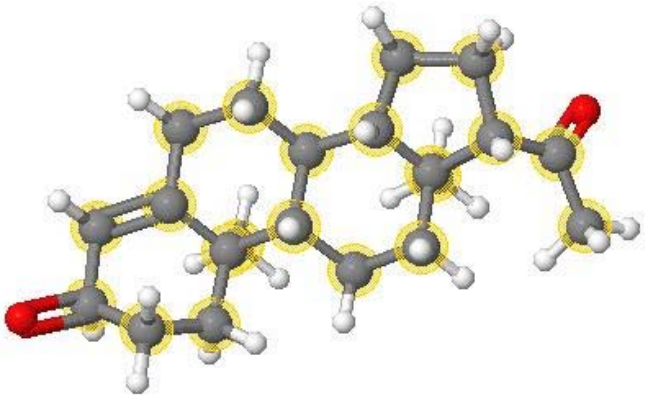
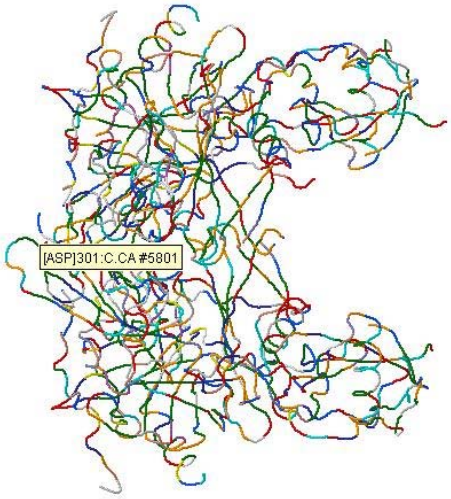
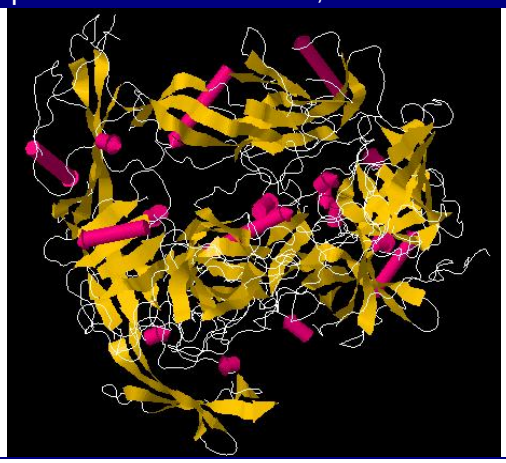
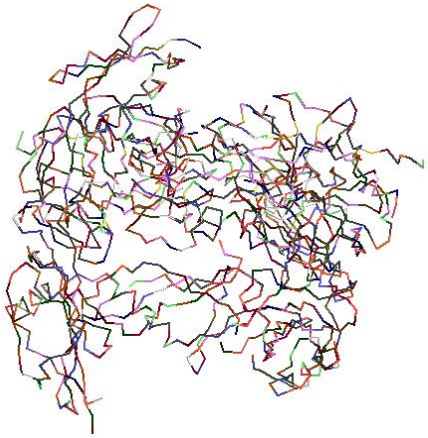
Tabla 9: Descripción de la Clase Entidad CE_Plugin

Anexo 2: Estilos de Visualización alcanzados.

Alambres	Bolas y Cilindros
	
Ocultar átomos	**Radios de Van Der Waals**
	

Etiquetas Elemento	Etiquetas Nombre del Átomo
	
Etiquetas número del átomo en la molécula	**Etiquetas Centradas**
	
Superficie de Puntos	**Superficie de Van der Waals traslúcida**
	

Superficie Molecular Opaca	Superficie Accesible al Disolvente
	
Estereografía gafas Rojo - Cian	**Estereografía gafas Rojo - Azul**
	
Estereografía gafas Rojo - Verde	**Estereografía Visión Bizca (Cross Eyed Vission)**
	

Mediciones de distancia, ángulos y ángulos diédricos	Visualización de ejes y caja contenedora
	
Selección de átomos	**Representación de trazas, Color amino**
	
Representación caricatura, color estructura	**Representación backbone, color Forma**
	

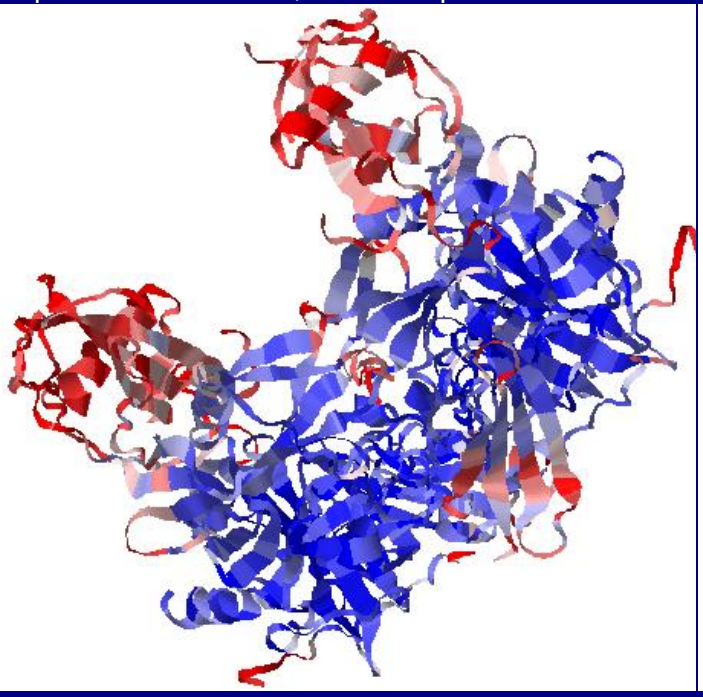
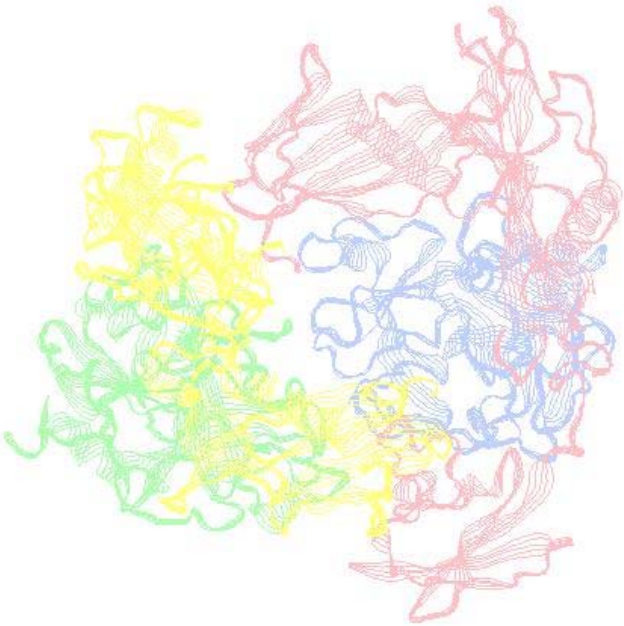
Representación Cintas, color Temperatura Relativa	Representación hilos, color Grupo
	

Tabla 10: Estilos de visualización alcanzados

Anexo 3: Fichero de almacenamiento de preferencias

Para hacer más ameno el trabajo con la aplicación se incluyó una opción para almacenar las preferencias de visualización en un fichero de texto plano, conteniendo código que interpreta el visualizador y que el usuario puede personalizar para lograr mejores visualizaciones. Esta opción permitirá al especialista que interactúe con la aplicación utilizar una configuración visual que prefiera sin necesidad de volver a seleccionar todas las opciones que componen la visualización final.

Ejemplo de fichero Script <preferencias.txt>

```
background [xFFFFFF];
color label [x000000];
Color CPK
select carbon
cpk 25%
wireframe on
label %i
zoom 60
set selectionhalos true
isosurface delete resolution 0 solvent 0 translucent
```

Figura 1: Ejemplo de fichero de preferencias.

Anexo 4: Arquitectura de Plug-ins

Adaptar las aplicaciones a las necesidades de los usuarios es, actualmente, una obligación. Las aplicaciones deben estar dotadas de un mecanismo de extensión que permita modificar o añadir nuevas funcionalidades.

Es fácil encontrar ejemplos reales: los filtros de un programa de retoque como el PhotoShop, los efectos especiales que se le pueden aplicar a la música que genera el WinAmp, los motores en DirectX y OpenGl de un juego en 3D…. y un sinfín más.

Ciertamente no existe ninguna magia detrás de estas piezas de software “enchufables”. Sólo se necesita tener un convenio de llamadas y una manera de establecer un enlace con el código externo.

En orientación a objetos se trata, simplemente, de aplicar el *polimorfismo*. Se necesita una interfaz que defina cómo se debe llamar al código externo, y una clase que nos proporcione objetos concretos que implementan la interfaz. Cada uno de esos objetos concretos será un plug-in – esto se conoce como Factoría en términos de patrones de diseño.

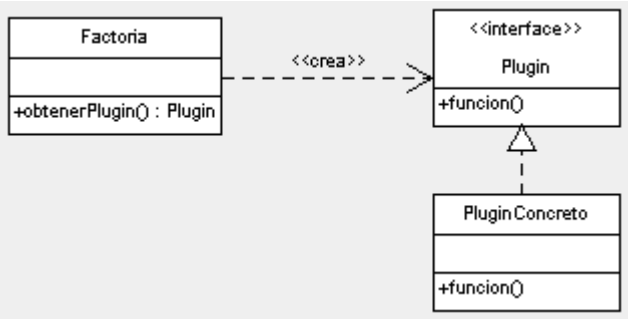


Figura 2: Relación de llamadas entre los plug-ins y la aplicación

Con este diseño se puede cambiar todo el código que realiza las operaciones definidas en la interfaz sin necesidad de recompilación, y en muchos casos en tiempo de ejecución.

Los plug-ins son clases Java empaquetadas en un fichero Jar que habrá que colocar en un directorio determinado para que sean accesibles para la aplicación. Se define una clase CC_Gestor_Plugin, que se ocupa de buscar los plug-ins existentes en ese directorio y permitir su uso, cargándolos bajo demanda.

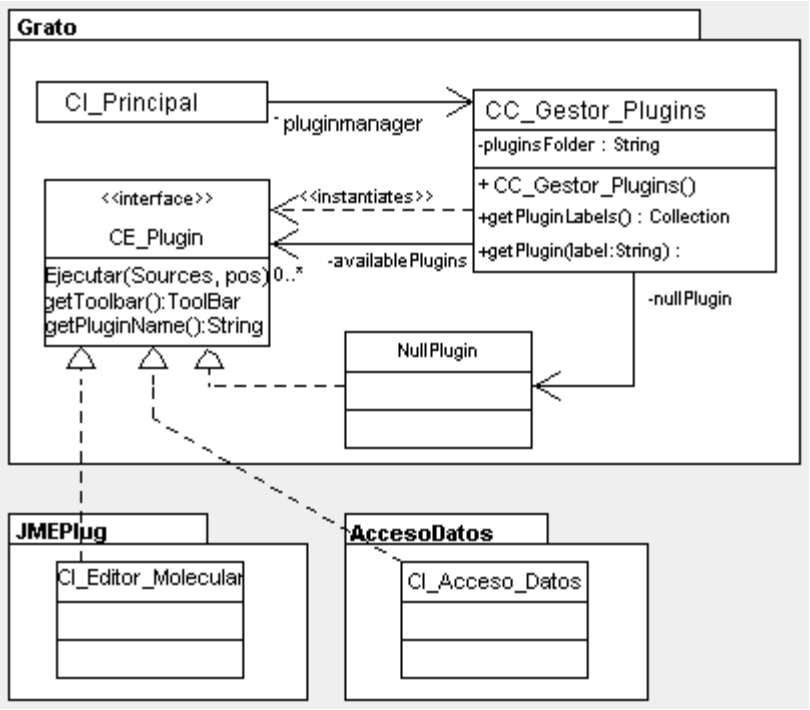


Figura 3: Implementación de la solución

Cuando se invoca un plug-in, se hace a través de la interfaz CE_Plugin y no importa cuan complejo es el procesamiento llevado a cabo o cuantas clases y paquetes se ven involucrados. Todo ese código debe estar empaquetado en un solo fichero para facilitar la instalación, reducida a copiar el fichero en el directorio designado.

Para empaquetar se utiliza la herramienta Jar que se incluye en el paquete de desarrollo de J2SE.

Dentro de un jar debe incluirse un manifiesto (Manifest.mf) con información sobre los contenidos del fichero. Lo que interesa es que se trata de un documento de texto donde se pueden incluir claves, englobadas en secciones, a las que se puede acceder en los programas a través del API de Java, paquetes java.util.jar.

El manifiesto de que se definió debe incluir obligatoriamente una sección nueva con dos claves necesarias para registrarlos. Si no se cumple este requisito, el fichero jar no se registrará. “JMEPlug” tendría esta sección en su manifiesto:

Figura 4: Archivo de manifiesto del editor molecular (JMEPlug)

Cuando la aplicación se inicia, se explora el directorio de plug-ins buscando los ficheros jar. Por cada fichero, se intenta leer su manifiesto buscando las claves de 'nombre' y 'clase que implementa la interfaz'; el nombre se podía haber obtenido del propio plug-in, con un método getPluginName(), pero fijándolo en el manifiesto, se evita cablearlo y se podría, por ejemplo, ponerlo en varios idiomas.

Una vez encontradas se puede suponer que es un plug-in del sistema y se registra, esto es, añadiendo el par [nombre, nombre de la clase que implementa la interfaz] a una tabla de plug-ins disponibles y se añade la URL que apunta al fichero jar a una lista. Se guarda en la tabla el nombre de la clase y no una instancia porque se quiere cargar bajo demanda el plug-in ya que puede tener una inicialización costosa y nadie asegura que vaya a usarse en esa sesión de trabajo.

Para usar el plug-in debe solicitarse al CC_Gestor_Plugin, a través del método getPlugin(nombre). Se chequea si entre los registrados hay uno con ese nombre. Si lo que devuelve es un nombre de clase, implica que no se ha creado la instancia todavía. Se crea a través de un classLoader (cargador de clases) y se modifica la información de registro para que la próxima solicitud devuelva el objeto recién creado (realizando la carga bajo demanda). La dificultad de este proceso aparece a la hora de cargar la clase. Cuando se ejecuta una máquina virtual Java para ejecutar la aplicación queda fijado el classpath (directorio de clases) del cargador de clases del sistema. Por supuesto, no se quiere añadir a mano cada fichero jar del directorio de plug-ins. La solución es crear un cargador de clases propio que incluya en su classpath los jar de los plug-ins registrados. Ese es el motivo por el que se creo la lista de URLs

apuntando a los jars durante el proceso de registro. Al solicitarle la carga de una clase, intentará usar su cargador padre, en este caso el del sistema, si no obtiene resultados busca en su propio classpath, en los ficheros cargados, dando como resultado la aplicación esperada.

Anexo 5: Herramienta para generación de archivos de manifiesto (MANIFEST.MF)

Crear los ficheros de manifiesto de manera manual puede resultar sumamente engorroso. Teniendo en cuenta que el resto de los miembros del equipo de trabajo necesitan crear archivos de manifiesto acordes a sus módulos para poder ser asimilados por la aplicación y no necesitan necesariamente conocer la estructura de los mismos, se desarrolló una herramienta que permite la generación de archivos de manifiesto (MANIFEST.MF) correctamente formados a partir de la información ofrecida por el desarrollador o creador del módulo al que se le está desarrollando el manifiesto.

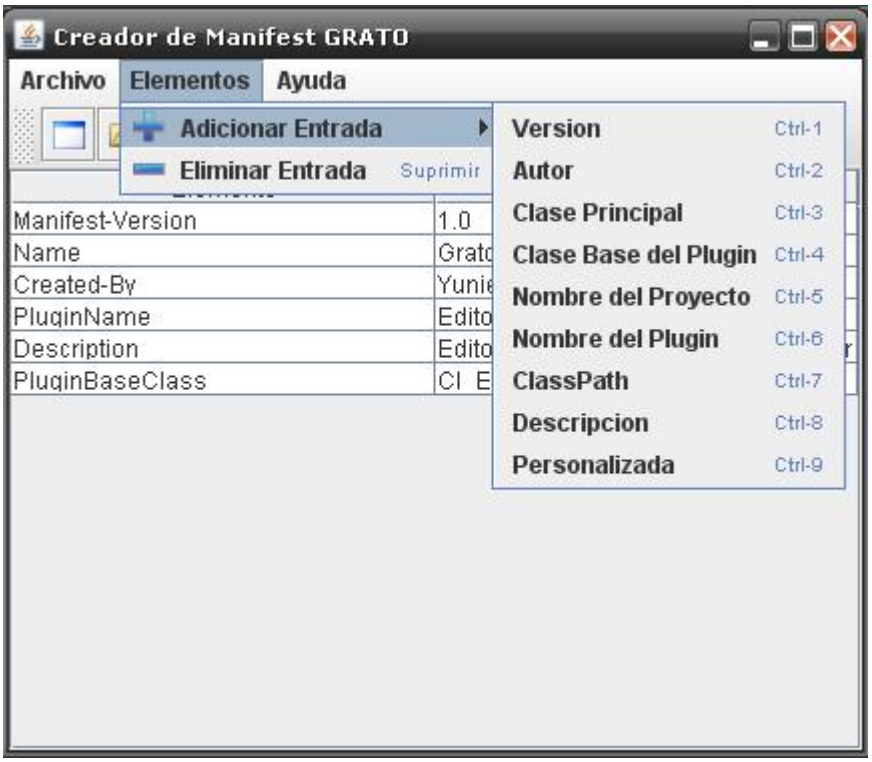


Figura 5: Interfaz y funcionalidades del creador de manifiestos

Con esta pequeña herramienta, los desarrolladores serán capaces de definir la información que consideren necesaria para su módulo, la cual será útil a la hora de registrar y reconocer los paquetes y plug-ins en la interfaz de localización de plug-ins de la solución propuesta.

La herramienta consta con una pequeña ayuda que explica de manera sencilla cada campo y lo que significa, la importancia que tiene y cómo quedaría en el archivo de manifiesto.

GLOSARIO DE TÉRMINOS

Applet: es un componente de software que corre en el contexto de otro programa, por ejemplo un navegador 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.

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)

CQF: Centro de Química Farmacéutica

GENOMA: El Proyecto Genoma Humano comenzó en 1990 en los Estados Unidos, con el objetivo de analizar molecularmente la herencia genética humana.

GNU/LGPL: "GNU Lesser General Public License" (Licencia Pública General Menor). Pretende garantizar la libertad de compartir y modificar el software "libre", esto es para asegurar que el software es libre para todos sus usuarios. Esta licencia pública general se aplica a la mayoría del software de la "FSF Free Software Foundation" (Fundación para el Software Libre) y a cualquier otro programa de software cuyos autores así lo establecen.

GUI: Graphic User Interface o Interfaz Gráfica de Usuario

I&D: Investigación y Desarrollo.

IUPAC: International Union of Pure and Applied Chemistry (Unión Internacional de la Química Pura y Aplicada)

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

script: son un conjunto de instrucciones generalmente almacenadas en un archivo de texto que deben ser interpretados línea a línea en tiempo real para su ejecución, se distinguen de los programas, pues deben ser convertidos a un archivo binario ejecutable para correrlos.

shell: clase básica para todas las aplicaciones visuales de Java.

SMARTS: SMiles ARbitrary Target Specification (Especificación arbitraria de blancos de SMILES)

SMILES: Simplified Molecular Input Line Entry System (Sistema de líneas de entrada molecular simplificada)

SMIRKS: Lenguaje de transformación de reacciones. Súper conjunto de SMILES de reacción. Subconjunto de SMARTS de reacción. Posee comportamientos que no existen en el resto de los lenguajes anteriores, capaz de describir mecanismos de reacción con grados variables de especificación y generalización

UCI: Universidad de las Ciencias Informáticas.

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.