**UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS**
**FACULTAD #6**
**BIOINFORMÁTICA**


# MÓDULO PARA EL CÁLCULO DISTRIBUIDO DE MECÁNICA MOLECULAR Y MECÁNICA CUÁNTICA

**TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS**

**AUTORES:**

**ANDRÉS YANIER CAMEJO ISAAC.**

**ADNIER TURRO RODRÍGUEZ.**

**TUTORES:**

**DR. RAMÓN CARRASCO VELAR. CQF**

**MSC. AURELIO ANTELO COLLADO. UCI**

**LIC. LONGENDRI AGUILERA MENDOZA. UCI**

**CIUDAD DE LA HABANA, CUBA**

**JULIO, 2007**

# DECLARACIÓN DE AUTORÍA

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

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

_____ _____

Andrés Yanier Camejo Isaac (Autor) Adnier Turro Rodriguez (Autor)

_____ _____

Dr. Ramón Carrasco Velar (Tutor) MSc. Aurelio Antelo Collado (Tutor)

_____

Lic. Longendri Aguilera Mendoza (Tutor)

# AGRADECIMIENTOS

A nuestros padres quienes engendraron la ética y el rigor que guían nuestro transitar por la vida.

A nuestros tutores por su asesoramiento científico, su profesionalidad y tiempo dedicado.

Al Dr. Ramón Carrasco Velar por las horas de dedicación al trabajo, por sus consejos, su comprensión y por estimularnos para seguir creciendo intelectualmente.

Al MSC. Aurelio Antelo Collado por haber confiado en nosotros para el desarrollo de la tarea.

Al Lic. Longendri Aguilera Mendoza por brindar todas las herramientas y ayuda necesarias en la parte técnica del trabajo.

Al profesor Manuel de Jesús Luis Díaz por su apoyo y preocupación constante por este trabajo.

A todos los que conforman el proyecto BioGrid por su apoyo incondicional, horas de labor incansable para hacer posible el desarrollo de este trabajo y por su amistad.

A nuestros amigos, compañeros de aula, de laboratorio, de cuarto, 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, gracias por confiar en nosotros.

A los que en algún momento nos preguntaron por la tesis y nos brindaron su apoyo incondicional.

A todos: Muchas Gracias.

# DEDICATORIA

A nuestros  padres que con su enseñanza y buenas costumbres han hecho de nosotros  lo que somos.

A Dadis

A Merenia.

A todos aquellos que han contribuido de alguna forma a nuestra formación profesional.

# RESUMEN

En la actualidad dentro del ámbito de las ciencias informáticas, se trabaja bajo presiones a la hora de garantizar la máxima productividad y gestionar de forma eficiente los recursos, sobre todo cuando el escenario tecnológico varía con rapidez. En el marco del proyecto conjunto entre el Centro de Química Farmacéutica y la Facultad de Bioinformática de la UCI, se prevé la tarea de realizar cálculos químico-teóricos a un número considerable de compuestos orgánicos. Estos cálculos son de elevado costo computacional por lo que surge a su vez la idea de diseñar e implementar un módulo capaz gestionar estos cálculos de forma distribuida. Se analizó, diseñó e implementó un módulo para realizar cálculos químico-cuánticos distribuidos utilizando versiones del programa MOPAC para Linux y Windows, con significativos tiempos de respuesta teniendo en cuenta la cantidad de recursos disponibles en una red en el momento en que se envíe el pedido y la magnitud de este. La aplicación se desarrolló en un ambiente multiplataforma, utilizando el lenguaje java para su implementación. Se siguió la línea de utilización de herramientas libres para garantizar el beneficio gratuito de la aplicación. Como resultado se logró una reducción a 6 horas de trabajo el procesamiento de 20000 moléculas de la base de datos del proyecto, distribuidas a través de 125 máquinas en la red.



PALABRAS CLAVES

Sistemas distribuidos, cálculos químico-cuánticos.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

# INTRODUCCIÓN

En Cuba se le presta especial atención al estudio de los problemas que abordan el tema de la medicina y la salud. Para lograr un trabajo más eficiente y con mejores resultados, se cuenta con diversos centros pertenecientes al polo científico cubano dedicados completamente al desarrollo de investigaciones con vistas a enriquecer nuestro desarrollo económico y científico. Se requiere de una ardua labor, tiempo de estudio, y preparación, para lograr resultados satisfactorios. De esta forma nuestros investigadores llevan adelante sus ideas. Día tras día, el Centro de Ingeniería Genética y Biotecnología (CIGB), Centro de Química Farmacéutica (CQF), Centro de Inmunología Molecular (CIM), entre otros centros, se levantan a dar su aporte a nuestros arsenales del conocimiento. Cada paso que avanzamos hoy es una victoria que alcanzamos, más aún si se tiene en cuenta que vivimos en un país bloqueado, que debe valerse por si solo en muchas de las principales ramas que sustentan su economía. Para esto y bajo el calor de una fuerte batalla de ideas, nuestro país, con vistas a ubicarse entre los principales países productores de software y poder así desarrollar herramientas que contribuyan a nuestro trabajo, creó la Universidad de las Ciencias Informáticas, concebida como centro para la formación de especialistas informáticos de alto nivel, así como para el desarrollo y exportación de software. Parte de esta misión es crear conjuntamente con las instituciones científicas y universidades del país, herramientas que agilicen el proceso de desarrollo de la economía.

El CQF, en asociación con la Facultad de Bioinformática de la UCI trabajan en el proyecto “Plataforma Inteligente para la Predicción de Actividad Biológica de Compuestos Orgánicos” el cual permitirá el diseño computacional de principios activos como procedimiento de elección para el desarrollo posterior de nuevos medicamentos. Estos tipos de herramientas son de alto costo y bajo las distintas restricciones a que el país esta sometido, no se pueden obtener fácilmente. Estas aplicaciones permiten disminuir los costos y el tiempo del proceso de investigación. Existen softwares o paquetes especializados en calcular mecánica molecular y cuántica tales como el Mopac, Hyperchem, Nwchem entre otros. Cada uno de estos tiene sus requerimientos que no se ajustan a los objetivos del proyecto que da origen a este trabajo. Los cálculos químico-teóricos resultan demasiado costosos computacionalmente cuando se necesita procesar muchas moléculas, y se convierte en un problema real para las distintas investigaciones, pues trae consigo perdida de tiempo, además de que muchas veces se tiene que optar por suposiciones debido a que

resultados más precisos no se pueden obtener en tiempo.

Por otra parte, la programación de alto rendimiento marca un avance en el mundo informático, y surge para dar solución a necesidades como:

- Repartir grandes volúmenes de información para su procesamiento.
- Compartir recursos computacionales.

De aquí se derivan los sistemas computacionales distribuidos y los sistemas de procesamiento en paralelo. Donde un conjunto de ordenadores conectados por una red son usados para realizar tareas distribuidas. Entonces se opta por la solución distribuida, eficiente y económica para el trabajo que se desea realizar.

Aprovechando las potencialidades que brindan estos sistemas para repartir el volumen de información, así como su capacidad de aprovechamiento de recursos, se plantea: que si se distribuyen los cálculos de mecánica molecular y mecánica cuántica, sobre los bancos de moléculas, será posible obtener los resultados de los mismos en menor tiempo y se aprovecharán mejor los recursos que se posee en la red.

La utilización de sistemas distribuidos representa una solución que aumenta las potencialidades de cómputo de cualquier institución, al no estar sujetos a las restricciones de una máquina, hace posible utilizar y explotar mucho mejor los recursos de toda una red. Ahora, basados en la política de darle explotación racional a nuestros medios y bajo la presión de un considerable número de trabajos a realizar, se decide implementar un módulo para el cálculo distribuido de mecánica molecular y mecánica cuántica que aproveche el estado ocioso de varias máquinas conectadas a través de una red. Por esta razón se fija el **objeto de estudio** del presente trabajo sobre la aplicación de los sistemas distribuidos para el procesamiento y cálculo masivo de información, y como **campo de acción** del mismo los sistemas distribuidos para la realización de cálculos químicos-teóricos.

En este ámbito, se define como **objetivo general** de la investigación: desarrollar un módulo que permita distribuir el proceso de cálculo de mecánica molecular y mecánica cuántica y que explote los recursos existentes en la red.

Para dar cumplimiento a dicho objetivo es necesario alcanzar los siguientes **objetivos específicos**:

- Analizar, diseñar e implementar un módulo que realice cálculos de mecánica molecular y mecánica cuántica sobre grandes bancos de moléculas.
- Evaluar la eficiencia del sistema implementado.

Para alcanzar dichos objetivos se planteó desarrollar las siguientes tareas:

- Realización de búsquedas bibliográficas acerca de los diferentes sistemas distribuidos que existen en el mundo, así como los programas informáticos que realizan cálculos químico teóricos, específicamente de mecánica molecular y mecánica cuántica.
- Estudio de la tecnología Java RMI (Invocación a Métodos Remotos) para el trabajo distribuido.
- Implementación del módulo para realizar los cálculos químicos-teóricos.
- Realización de pruebas al sistema y evaluación de la calidad y eficiencia del mismo.

El presente documento se estructura en un Resumen, varios capítulos que constituyen el cuerpo de la tesis, Conclusiones, Recomendaciones Bibliografía y Referencias bibliográficas. Los capítulos son:

**Capítulo 1: Fundamentación Teórica.**

Se presenta una breve reseña de los resultados del estudio bibliográfico realizado sobre el uso de plataformas distribuidas para la resolución de problemas de grandes volúmenes de cálculo. Se describen las tendencias y tecnologías actuales a tener en cuenta para implementar este tipo de herramientas, así como algunas de las tecnologías más frecuentemente empleadas y se justifica el porque del uso de las escogidas para el desarrollo del proyecto. Se muestran una serie de aplicaciones existentes en la actualidad encargadas de resolver problemas de cálculos químico-teóricos, y se señala aquella que resultó la de mejor ajuste para los recursos y requerimientos que se poseen para los objetivos del proyecto original.

**Capítulo 2: Características del Sistema.**

Aborda lo referente al funcionamiento del negocio: las reglas, 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 los elementos imprescindibles para una solución exitosa: como son los requerimientos funcionales y no funcionales. Se definen y describen los casos de uso funcionales del sistema al igual que los actores que los inician.

**Capítulo 3: Análisis y Diseño del sistema.**

Se muestra la expansión de los casos de uso del módulo presentado, los diagramas de clases y de interacción para cada uno de estos, así como la descripción detallada de las clases. También se definen las clases persistentes. El diseño en este capítulo queda definido y descrito, de modo tal que de define el patrón de diseño y otras técnicas a utilizar para el desarrollo del módulo.

**Capítulo 4 Implementación y prueba.**

En este capítulo se describe el sistema y se implementa en términos de componentes organizados de acuerdo a los nodos específicos en el modelo de despliegue. Se realizan pruebas de caja negra para comprobar que se cumple con el objetivo planeado en cuanto a su funcionamiento. Se evalúa el sistema en una muestra real y se determina la calidad y eficiencia en la ejecución de los cálculos.

# CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

## 1.1 Introducción.

La Bioinformática y la Industria Farmacéutica están estrechamente relacionadas. Tal así es que la primera industria que comprendió el impacto de la Bioinformática fue la farmacéutica. En este capítulo se analiza brevemente la relación entre ambas, así como la importancia del desarrollo de la presente aplicación distribuida para el desarrollo de las investigaciones relacionadas con la búsqueda de nuevos principios activos. En el se definen algunos conceptos que serán de utilidad en su comprensión y de la propuesta de solución. Una pequeña reseña sobre los aspectos más importantes de algunos softwares especializados, tanto en la realización de cálculos químico-teóricos como en distribuir cargas de trabajo para converger de manera urgente a soluciones factibles. Ayudará a precisar el camino del desarrollo del presente módulo, justificará el uso de los sistemas que se seleccionen como referencia para el trabajo. Se describen además, los procesos que serán objeto de automatización, así como la metodología a seguir y la tecnología a utilizar.

## 1.2 La Bioinformática y su relación con la Industria Farmacéutica.

La Bioinformática puede considerarse como un pilar imprescindible en los proyectos de genómica y proteómica en los que es necesario organizar resultados, analizarlos, generar hipótesis y proponer nuevos experimentos. Esta actividad ha hecho que esta se convierta en un componente básico para el desarrollo de la biología molecular, la biotecnología y la biomedicina. La utilización de técnicas bioinformáticas en las fases de descubrimiento de nuevas dianas sobre las que desarrollar nuevos fármacos es potencialmente capaz de acelerar este proceso con el consiguiente ahorro de recursos. Es un componente clave en el desarrollo de la llamada “medicina personalizada”, que implica la posibilidad de desarrollar fármacos específicos para individuos con composiciones genómicas determinadas y así una mejor medicina preventiva (1).

La industria farmacéutica es un sector dedicado a la fabricación y preparación de productos químicos medicinales para la prevención o tratamiento de las enfermedades. Entre los procesos de producción secundaria, altamente automatizados, se encuentran la fabricación de fármacos dosificados, como pastillas, cápsulas o sobres para administración oral, soluciones para inyección, óvulos y supositorios.

Dentro de los procesos de desarrollo de nuevas moléculas con propiedades farmacéuticas, es usual identificar primero el blanco de la molécula, su contraparte. Este proceso se realizaba anteriormente con aquellas pocas moléculas que el químico era capaz de sintetizar o aislar, sin embargo, y dado el auge en el acceso a la información del proyecto 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 para afrontar esta explosión de información. La bioinformática, en lo que atañe al descubrimiento de nuevas moléculas de interés farmacéutico, está estrechamente relacionada con las bases de datos y los sistemas de estimación y diseño de estructuras proteicas, es así como el desarrollo de aplicaciones para extraer conocimiento de la información recabada en los laboratorios se tiene como una de las necesidades principales en la biocomputación (1). El éxito en las tareas 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. Lo real es que una computadora no puede reemplazar un laboratorio y es por esto que los aportes tradicionales químicos y farmacológicos continuarán siendo importantes y necesarios, sin embargo, es igualmente valido que el uso de recursos informáticos minimiza tiempos en esta clase experimentos lo cual hace más atractiva la relación Costo-Efecto. Toma así importancia para el desarrollo de las investigaciones básicas en el campo de la disciplina, la implementación del módulo propuesto, ya que contribuirá a disminuir los costos de la etapa de investigación básica para la búsqueda de nuevos principios con potencial actividad biológica.

### 1.3 Sistemas distribuidos, aplicación en la bioinformática.

Un sistema distribuido consiste en una colección de ordenadores autónomos unidos por una red y con un sistema que les permite compartir recursos de hardware, software y datos (2). La construcción de estos presenta una solución que aumenta las posibilidades, ya no estando sujetos a las restricciones de una máquina, se estará en condiciones de utilizar y explotar mejor los recursos de toda la red.

La necesidad de aprovechar los recursos disponibles en los sistemas informáticos conectados a la red y simplificar su utilización ha dado lugar a una nueva forma de tecnología de la información conocida como *Grid Computing*. De este modo, los sistemas distribuidos se pueden emplear como un único sistema

virtual en aplicaciones intensivas en datos o con alta demanda computacional. Las necesidades básicas de la biología se pueden expresar como una mayor demanda de capacidad de almacenamiento y tratamiento de información, y un crecimiento desmesurado de la capacidad de cálculo. En otras palabras, la forma más efectiva en coste y recursos de resolver los problemas actuales de las Ciencias de la Vida y por extensión responder a las crecientes demandas sociales sobre ellas es recurrir a sistemas masivamente distribuidos de tecnología Grid (3).

### 1.3.1 Posibilidades y limitaciones que brindan los sistemas distribuidos.

**Características:**

- **Concurrencia**.- Esta característica de los sistemas distribuidos permite que los recursos disponibles en la red puedan ser utilizados simultáneamente por los usuarios y/o agentes que interactúan en la red.
- **Carencia de reloj global**.- Las coordinaciones para la transferencia de mensajes entre los diferentes componentes para la realización de una tarea, no tienen una temporización general, esta más bien distribuida a los componentes.
- **Fallos independientes de los componentes**.- Cada componente del sistema puede fallar independientemente, con lo cual los demás pueden continuar ejecutando sus acciones. Esto permite el logro de las tareas con mayor efectividad, pues el sistema en su conjunto continua trabajando.

**En general el desarrollo de sistemas distribuidos intenta poner solución a los siguientes objetivos:**

- Transparencia.
- Fiabilidad.
- Rendimiento.
- Capacidad de crecimiento.
- Flexibilidad.
- Seguridad.

### 1.3.1.1 Ventajas y desventajas de los Sistemas Distribuidos.

**Ventajas**

- Aumento de la disponibilidad.
- Mejora del desempeño.
- Balanceo en la carga de trabajo.
- Compartición de recursos.
- Compartición de información.
- Confiabilidad, disponibilidad y tolerancia a fallas.
- Modularidad en el desarrollo.
- Flexibilidad.
- Crecimiento incremental.
- Reducción de costos.
- Mayor capacidad de modelar estructuras organizacionales.

**Desventajas**

- Uso ineficiente de los recursos distribuidos.
- Capacidad reducida para administrar apropiadamente grupos de procesadores y memoria localizada en distintos sitios.
- Enorme dependencia del desempeño de la red y de la confiabilidad de la misma.
- Debilitamiento de la seguridad.
- Mayor complejidad en la administración y mantenimiento.
- Mayor complejidad en su construcción (4).

### 1.4 Mecánica Molecular y Mecánica Cuántica aplicaciones.

El empleo de herramientas de la química teórica en el campo de la bioquímica tiene un desarrollo elevado en los últimos años gracias a los avances en el campo de la informática, campos estrechamente ligados, de manera que la gente que trabaja en el campo experimental de esta disciplina ha comenzado a ver en la parte teórica y en la llamada bioinformática una herramienta útil y eficiente para la resolución de problemas y que proporciona un campo de acción mucho más amplio y flexible que el campo experimental, lo cual permite el desarrollo y la explicación de conceptos no alcanzables experimentalmente.

Dentro de este marco la posibilidad de estudiar fenómenos biológicos a nivel molecular es una de las grandes aplicaciones que se han desarrollado. Las moléculas biológicas, ácidos nucleicos, proteínas en general están conformadas por átomos y por lo tanto son susceptibles de ser estudiadas teniendo en cuenta las propiedades de estos. La teoría más rigurosa que aborda este estudio es la teoría cuántica ya que los átomos y las moléculas son definitivamente objetos cuánticos. Actualmente el desarrollo tanto de hardware como de software en el campo de la química cuántica permite o hace posible el cálculo de pequeñas biomoléculas en un tiempo razonable. El empleo de la química cuántica y computacional nos permite enfrentar el estudio de estas interacciones con elementos no alcanzables experimentalmente, y que aportan al campo la rigurosidad y riqueza de la aproximación (5).

#### 1.4.1 Una aproximación teórica a la farmacología molecular, modelado molecular.

En los últimos años, los avances en el desarrollo de nuevos modelos matemáticos que describen fenómenos químicos, el desarrollo de programas más intuitivos y la disponibilidad de computadoras más veloces, han provisto a los científicos experimentales de un nuevo conjunto de herramientas computacionales que utilizadas en conjunto con técnicas de investigación tradicionales, permiten tanto examinar las propiedades estructurales de compuestos existentes como predecir propiedades y actividades para nuevas entidades químicas (6).

El modelado molecular permite representar numéricamente estructuras moleculares y simular su comportamiento con ecuaciones de la física cuántica y de la física clásica. Los programas de química computacional permiten generar y presentar datos moleculares incluyendo parámetros geométricos,

energías y propiedades espectroscópicas de las sustancias. El desarrollo de programas de modelado molecular y su aplicación en la investigación farmacéutica se ha formalizado como un campo de estudio conocido como diseño molecular asistido por computadoras. La mecánica molecular es un método de cálculo basado en la física clásica en el cual los átomos se tratan como partículas newtonianas que interactúan a través de una función de energía potencial. En contraste con los métodos cuánticos, la mecánica molecular se usa para calcular propiedades moleculares tales como geometrías, barreras rotacionales y estabilidad relativa de confórmeros (7). Debido a que los cálculos son rápidos y eficientes, la mecánica molecular puede usarse para estudiar sistemas de cientos de átomos. Así, la mecánica molecular da una excelente aproximación para la búsqueda en el espacio conformacional aunque se debe tener en cuenta que la conformación activa de una molécula puede no detectarse debido a que no siempre es un mínimo en la hipersuperficie.

### 1.4.2 Simulación computacional: Mecánica Molecular.

El propósito de un programa de mecánica molecular es determinar la estructura y energía óptima de una molécula basándose en los modelos mecanísticos definidos por los campos de fuerza, por tanto la entrada de datos en el programa debe de definir la estructura de comienzo para la molécula a estudiar. Esto implica tomar coordenadas cartesianas (x,y,z) para cada uno de los átomos así como definir los enlaces entre ellos. Los métodos de la simulación molecular han llegado a utilizarse ampliamente en las últimas décadas y en diferentes disciplinas de las ciencias exactas; tienen su origen en modelos simples, como los modelos moleculares construidos con alambre, madera o de otros materiales. Estos modelos permitían cambios manuales en las posiciones mutuas de los átomos (7). A pesar de lo burdo de los modelos, contribuyeron al descubrimiento de la estructura de la doble espiral del ADN. Hoy existen softwares con paquetes gráficos muy sofisticados que son estupendos para visualizar moléculas biológicas, pero no dicen mucho acerca de cómo cambian las moléculas cuando están sujetas a su acción mutua. Por tanto, se necesitan modelos físicos que reproduzcan confiablemente la estructura, las conformaciones y las rutas para obtener una conformación favorable bajo ciertas condiciones. La versión moderna de los métodos de simulación para ácidos nucleicos es la mecánica molecular. Ella nos permite calcular la estructura espacial de una molécula con base en la búsqueda de los mínimos locales a través del análisis de la energía de interacción no-Valente. La mecánica molecular se basa en la idea de que una molécula se puede

representar por un conjunto de puntos con carga neta situados en los núcleos de los átomos y distribuidos en una superficie de energía potencial generada por ellos mismos. El cálculo de los valores de la energía se hace utilizando un potencial efectivo para interacciones átomo-átomo, el cual se construye basándose en datos experimentales y en cálculos mecánico cuánticos. El método de Monte Carlo y el de Dinámica Molecular son los dos métodos de la mecánica molecular más utilizados para obtener propiedades promedio de los sistemas moleculares (8). Partiendo de la viabilidad de los métodos de cálculo, la diferencia en la obtención de resultados más acordes con la realidad (el experimento) lo constituyen las interacciones entre los átomos, expresadas a través de las funciones de potencial (5).

### 1.5 Estudio de sistemas existentes.

Existen innumerables sistemas especializados y diseñados para solucionar problemas de modelado y cálculo molecular. Estos han evolucionado proporcionales con la demanda por la necesidad de obtener mejores y más puntuales resultados. La mayoría de estos procesan la molécula deseada, pero solo una a la vez. Lo que da otra tarea a resolver, puesto a que el objetivo es lograr procesar una mayor cantidad de estas en menos tiempo y aprovechando los recursos que poseemos. En la búsqueda de soluciones la investigación entra en el mundo de los sistemas distribuidos; se necesita analizar algunos sistemas distribuidos para tener una idea de cual será la línea a seguir para las condiciones actuales que se tienen. Determinar las posibilidades y limitaciones de estos, deja una idea del por ciento de veracidad que tiene el resultado final de la investigación.

**Softwares para Modelado Molecular**

**1 Hyperchem**

**Mecánica molecular**

**Métodos empíricos**

MM$^{+}$, Amber, BIO$^{*}$ y OPLS.

**Mecánica cuántica**

**Métodos semi-empíricos**

CNDO (Complete Negled of Differential Overlap), INDO (Intermediate NDO), MINDO (Modified INDO) y ZINDO (Dr. M. Zerner INDO).

**Métodos de Hartree-Fock**

Con las bases STO-nG, 3-21G, 4-21G, 5-31G, 6-31G y 6-311G, más los correspondientes polarizados.

CON ESTE PROGRAMA SE PUEDEN REALIZAR CÁLCULOS SOBRE:

- Optimización de la geometría.
- Energía de punto.
- Estado de transición.
- Dinámica molecular.
- Dinámica de Langevin.
- Simulaciones de Monte Carlo.
- Análisis de coordenadas normales y espectros vibracionales.
- Espectros electrónicos.

**Una ventaja** del programa *Hyperchem* respecto a los otros es que se pueden introducir parámetros de constantes de fuerza, de tensión, de deformación, de torsión y de interacciones de Van der Waals para los cálculos de mecánica molecular. También en la última versión de este programa se puede trabajar con clases de bases más potentes para elementos de transición.

**2 Gaussian**

Permite dibujar las moléculas, ver las vibraciones moleculares, representar los orbitales moleculares y las cargas atómicas en las moléculas.

En este programa están implementados los métodos siguientes:

**Mecánica molecular**

UFF, Dreiding, Amber.

**Mecánica cuántica**

**Métodos semiempíricos**

AM1

**Métodos ab inicio**

HF, B3LYP, B3PW91, MP2, MP4, QCISD (quadratic configuration interaction), CCSD (cluster acoplados), CASSCF (complete active space multiconfiguration SCF), compuestos G1 Gaussian 1), G2 (Gaussian 2), CBS-4, CBS-Q, CBS-QB3 (complete basis set).

Clases de bases (basis set)

Con los métodos anteriores se pueden usar las bases siguientes: STO-3G, 3-21G, 6-31G, 6-311G, cc-pvQZ, LANL2DZ y SDD. Puede ser también con funciones polarizadas y difusas (d, df, pd). LANL2DZ es una base doble-zeta conteniendo un efectivo core potencial (ECP).

LOS DIVERSOS MÓDULOS DEL PROGRAMA PERMITEN:

- Optimizar estructuras moleculares, determinar energías, espectros vibracionales y electrónicos así como las cargas atómicas en los sistemas moleculares.
- Calcular estados fundamentales y estados excitados, singletes y tripletes (ZINDO, CIS y TD).
- Otros cálculos posibles son IRC (para recorrido de reacción), SCAN (explora una región de la superficie de energía potencial), STABILITY (reoptimiza la función de onda) y finalmente determina espectros de resonancia magnética nuclear.

**3 Spartan**

Incluye los siguientes métodos:

**Mecánica molecular**

**Métodos empíricos**

MMFF y SYBYL.

**Mecánica cuántica**

**Métodos semi-empíricos**

AM1, PM3 y MNDO.

**Método de Hartree-Fock**

Con las bases STO-3G, 3-21G $^{*}$ 6-31G $^{*}$, 6-31G $^{**}$, 6-31 + G $^{*}$, 6-311G $^{*}$, 6-311+G $^{*}$

**Densidad funcional**

SVWN/DN *, SVWN/DN **, pBP/DN *, pBP/DN **.

CON ELLOS SE PUEDEN CALCULAR:

- Energía de Punto que determina propiedades moleculares como la energía o densidad de spin de una estructura molecular definida.
- Geometría de equilibrio.
- Geometría del estado de transición.
- Distribución de confórmeros.
- Perfil de energía.
- Modos normales y frecuencias vibracionales.

**4 NWChem**

Paquete de química computacional desarrollado por el Laboratorio Molecular Ambiental De las Ciencias (EMSL), diseñado para funcionar con alto rendimiento en las supercomputadoras de manera paralela así como en estaciones de trabajo convencionales, realiza cálculos de energía electrónica molecular y pendientes analíticas utilizando la teoría SFC Hartree-Fock, DFT y la teoría de la perturbación de segundo orden.

**5 MOPAC**

MOPAC es un paquete de Química Cuántica de propósito general (cálculo semiempírico) para el estudio de las reacciones químicas. Implementa los Hamiltonians MNDO, MNDO/3, MNDO-PM3 y AM1.

**Características**

- Usa la física cuántica.
- Utiliza parámetros empíricos derivados experimentalmente.
- Usa aproximaciones extensivas.

**Mejores Aplicaciones**

- Sistemas de tamaño medio (cientos de átomos).
- Sistemas en los que se producen transiciones electrónicas.

### 1.6 Importancia del Desarrollo de la Aplicación.

El desarrollo del presente módulo es de importancia crítica para el desarrollo y funcionamiento general de la plataforma a que pertenece, herramienta concebida para apoyar el avance de los estudios de los clientes; especialistas químicos. La filosofía de este software es servir de ayuda en los estudios básicos que se realizan en la industria farmacéutica en la interpretación de los resultados experimentales y en el diseño de nuevos experimentos en un entorno de uso sencillo, de forma que el usuario pueda llevar a cabo estas tareas sin una excesiva dedicación. Los resultados de los cálculos realizados por este módulo sirven de nutriente para otras funcionalidades de la plataforma y para otros estudios que se realizan, en la línea de estudio para la determinación de posibles medicamentos.

La idea de distribuir los cálculos deseados resuelve conseguir una solución en un menor tiempo y tan solo con los recursos existentes, sin gastos para nueva tecnología, que resulta muy costoso para cualquiera de las entidades beneficiadas.

El estudio de softwares especializados en el tema de la química farmacéutica da una idea de su funcionamiento así como de su adaptabilidad para el trabajo que se quiere lograr. Claro esta, que asumir una tarea de tal magnitud como algunas de las mostradas requieren de años, por muchas razones el país carece de recursos y hay que resolver los problemas con el menor costo posible. La aplicación permite desde un servidor enviarle cálculos a realizar a todas aquellas máquinas que estén listas para cumplirlos, envía así las instrucciones y herramientas necesarias para que el programa realice los cálculos. Se obtienen como respuesta los resultados, y así en toda la red se estará en función de resolver un problema dividido en pequeños fragmentos, con los recursos que se tiene.

### 1.7 Metodologías y herramientas a analizar para el desarrollo del sistema.

Para el desarrollo del sistema se realizó un estudio de las posibles herramientas a utilizar en su construcción. Se tuvo en cuenta la tendencia actual y las novedades de cada una de ellas. Así como la gama de posibilidades que estas brindan para el desarrollo de la presente tarea. Se analizaron algunas de las más utilizadas en la actualidad, para que se disminuyera el dominio de búsqueda.

### 1.7.1 Metodología RUP.

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. 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.

RUP, como ya se mencionó anteriormente, elimina los errores cometidos en las iteraciones previas, logra así que al final del proceso se obtenga como resultado un producto de calidad. Para eso se agrupan las actividades en grupos lógicos definiéndose 9 flujos de trabajo principales divididos en 4 fases. Los 6 primeros flujos son conocidos como flujos de ingeniería y los tres últimos como de apoyo (9). Donde todos y cada uno de ellos cobran vital importancia en el proceso de desarrollo de software. En la siguiente figura se puede encontrar el gráfico conocido por RUP en dos dimensiones, donde se representan los flujos mencionados y la fase en que cada uno de ellos cobra su mayor desempeño.


**Figura 1** RUP en dos dimensiones.

Se pueden resumir entonces como características esenciales y principales de RUP que lo hacen ser una metodología confiable, robusta y utilizable por todos.

- **Proceso dirigido por Casos de Uso:**

Los Casos de Uso son una técnica de captura de requisitos que fuerza a pensar en términos de importancia para el usuario y no sólo en términos de funciones que sería bueno contemplar. Se define un Caso de Uso como un fragmento de funcionalidad del sistema que proporciona al usuario un valor añadido. Los Casos de Uso representan los requisitos funcionales del sistema. En RUP los casos de uso no solo inician el proceso de desarrollo sino que proporcionan un hilo conductor, permiten establecer trazabilidad entre los artefactos que son generados en las diferentes actividades del proceso de desarrollo. En la figura siguiente se hace una representación de esta afirmación (10).

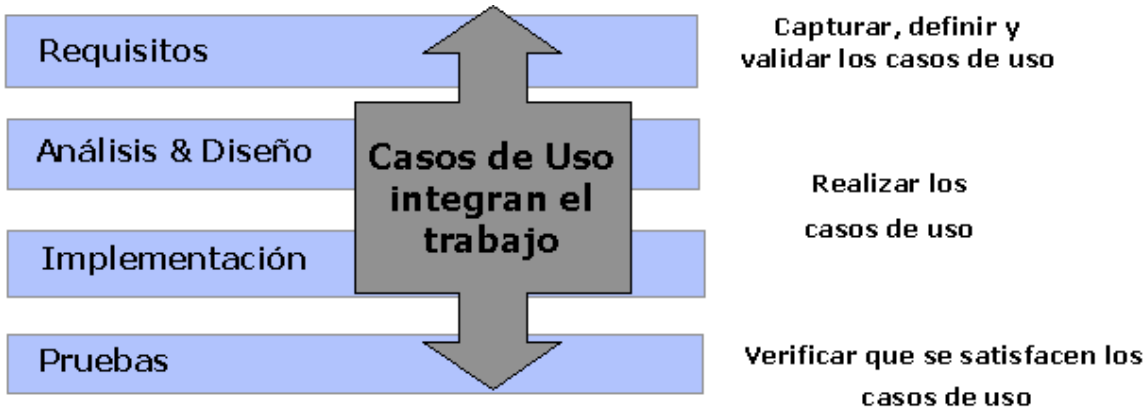


**Figura 2** Los Casos de Uso integran el trabajo.

- **Proceso iterativo e incremental:**

El trabajo se divide en partes más pequeñas o mini proyectos. El equilibrio entre Casos de Uso y arquitectura se logra durante cada mini proyecto, así durante todo el proceso de desarrollo. Cada mini proyecto se puede ver como una iteración (un recorrido más o menos completo a lo largo de todos los flujos de trabajo fundamentales) del cual se obtiene un incremento que produce un crecimiento en el producto. Y en cada iteración se aborda una parte de la funcionalidad total, pasando por todos los flujos de trabajo relevantes y refinando la arquitectura, para obtener un resultado final sin problemas de no haber corregido errores anteriores.

- **Proceso centrado en la arquitectura:**

La arquitectura involucra los aspectos estáticos y dinámicos más significativos del sistema, está relacionada con la toma de decisiones que indican cómo tiene que ser construido el sistema y ayuda a determinar en qué orden. Además la definición de la arquitectura debe tomar en consideración elementos de calidad del sistema, rendimiento, reutilización y capacidad de evolución por lo que debe ser flexible durante todo el proceso de desarrollo. La arquitectura se ve influenciada por la plataforma software, sistema operativo, gestor de bases de datos, protocolos, consideraciones de desarrollo como sistemas heredados. Muchas de estas restricciones constituyen requisitos no funcionales del sistema.

### 1.7.2 Metodología XP.

La programación extrema o eXtreme Programming (XP) es la más destacada de los procesos ágiles de desarrollo de software. La misma se diferencia de las metodologías tradicionales principalmente en que pone más énfasis en la adaptabilidad que en la previsibilidad. Los defensores de XP consideran que los cambios de requisitos sobre la marcha son un aspecto natural, inevitable e incluso deseable del desarrollo de proyectos (10). Creen que ser capaz de adaptarse a los cambios de requisitos en cualquier punto de la vida del proyecto es una aproximación mejor y más realista que intentar definir todos los requisitos al comienzo del proyecto e invertir esfuerzos después en controlar los cambios en los requisitos. Para tener una idea más general de la misma se muestran a continuación algunas de sus características fundamentales.

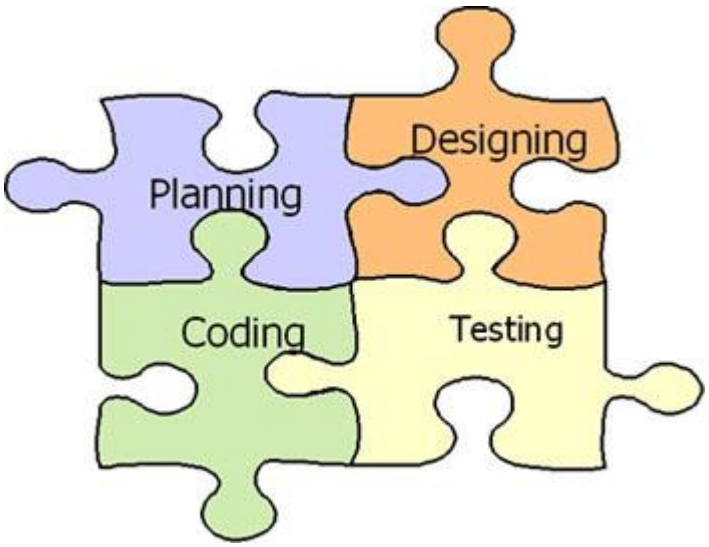


**Figura 3** Metodología Extreme Programing.

**Las características fundamentales del método XP son:**

- **Desarrollo iterativo e incremental:** pequeñas mejoras, unas tras otras.
- **Pruebas unitarias continuas:** frecuentemente repetidas y automatizadas, incluye pruebas de regresión. Se aconseja escribir el código de la prueba antes de la codificación.
- **Programación en parejas:** se recomienda que las tareas de desarrollo se lleven a cabo por dos personas en un mismo puesto. Se supone que la mayor calidad del código escrito de esta manera

el código es revisado y discutido mientras se escribe es más importante que la posible pérdida de productividad inmediata.

- **Frecuente interacción del equipo de programación con el cliente o usuario:** Se recomienda que un representante del cliente trabaje junto al equipo de desarrollo.
- **Corrección de todos los errores antes de añadir nueva funcionalidad:** Hacer entregas frecuentes.
- **Refactorización del código:** es decir, reescribir ciertas partes del código para aumentar su legibilidad y mantenibilidad pero sin modificar su comportamiento. Las pruebas han de garantizar que en la refactorización no se ha introducido ningún fallo.
- **Propiedad del código compartida:** en vez de dividir la responsabilidad en el desarrollo de cada módulo en grupos de trabajo distintos, este método promueve el que todo el personal pueda corregir y extender cualquier parte del proyecto. Las frecuentes pruebas de regresión garantizan que los posibles errores serán detectados.
- **Simplicidad en el código:** es la mejor manera de que las cosas funcionen. Cuando todo funcione se podrá añadir funcionalidad si es necesario. La programación extrema apuesta que es más sencillo hacer algo simple y tener un poco de trabajo extra para cambiarlo si se requiere, que realizar algo complicado y quizás nunca utilizarlo (10).

El objetivo de XP es generar versiones de la aplicación tan pequeñas como sea posible, pero que proporcionen un valor adicional claro, desde el punto de vista del negocio. A estas versiones se las denomina releases. Una release cuenta con un cierto número de historias. La historia es la unidad de funcionalidad en un proyecto XP, y corresponde a la mínima funcionalidad posible que tiene valor desde el punto de vista del negocio. Durante cada iteración se cierran varias historias, lo que hace que toda iteración añada un valor tangible para el cliente.

### 1.7.3 Lenguaje de modelado UML.

El Lenguaje Unificado de Modelado permite modelar, construir y documentar los elementos que forman un producto de software que responde a un enfoque orientado 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. Intenta solucionar el problema de propiedad de código que se da con los desarrolladores, al implementar un lenguaje de modelado común para todos los desarrollos se crea una documentación también común, que cualquier desarrollador con conocimientos de UML será capaz de entender, independientemente del lenguaje utilizado para el desarrollo. En estos momentos se convierte en un Standard, no existe otra especificación de diseño orientado a objetos, ya que es el resultado de las tres opciones existentes en el mercado. Su utilización es independiente del lenguaje de programación y de las características de los proyectos, ya que UML ha sido diseñado para modelar cualquier tipo de proyectos, tanto informáticos como de arquitectura, o de cualquier otro rama (9). Permite la modificación de todos sus miembros mediante estereotipos y restricciones. Un estereotipo nos permite indicar especificaciones del lenguaje al que se refiere el diagrama de UML. Una restricción identifica un comportamiento forzado de una clase o relación, es decir mediante la restricción estamos forzando el comportamiento que debe tener el objeto al que se le aplica.

Dentro de las ventajas que proporciona este lenguaje de modelado podemos destacar las siguientes:

- Permite modelar sistemas utilizando técnicas orientadas a objetos (OO).
- Permite especificar todas las decisiones de análisis, diseño e implementación, construyendo así modelos precisos, no ambiguos y completos.
- Puede conectarse con lenguajes de programación (Ingeniería directa e inversa).
- Permite documentar todos los artefactos de un proceso de desarrollo.
- Cubre las cuestiones relacionadas con el tamaño propio de los sistemas complejos y críticos.
- Existe un equilibrio entre expresividad y simplicidad, pues no es difícil de aprender ni de utilizar.
- UML es independiente del proceso, aunque para utilizarlo óptimamente se debería usar en un proceso que fuese dirigido por los casos de uso, centrado en la arquitectura, iterativo e incremental.

### 1.7.4 Rational Rose.

Es la herramienta Case desarrollada por los creadores de UML que cubren todo el ciclo de vida de un proyecto: concepción y formalización del modelo, construcción de los componentes y certificación de las distintas fases. Nos permite una trazabilidad real entre modelo (análisis y diseño) y el código ejecutable.

Rational Rose domina el mercado de herramientas para el análisis, modelamiento, diseño y construcción orientada a objetos, tiene todas las características que los desarrolladores, analistas, y arquitectos exigen soporte UML incomparable, desarrollo basado en componentes con soporte para arquitecturas líderes en la industria y modelos de componentes, facilidad de uso e integración optimizada.

La corporación Rational Rose ofrece el Proceso Unificado de Racional (RUP), que unifica las mejores prácticas de muchas disciplinas en un consistente y completo proceso del ciclo de vida, que permite al equipo de desarrollo disminuir los tiempos de liberación, además de hacer más predecible el software que ellos producen. Este proceso está basado en el Lenguaje Unificado de Modelación (UML estándar de la industria) y únicamente integrado a herramientas líderes en el desarrollo de software de Racional, el Proceso Unificado de Racional apoya el equipo completo de desarrollo de software con guías detalladas e información crítica aplicable a la mayoría de las aplicaciones de la industria.

### 1.7.5 Visual Paradigm.

Visual Paradigm para el Lenguaje Unificado de Modelado es una herramienta CASE 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. 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, se integra con Eclipse, Borland® JBuilder®, NetBeans entre otros, 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.

### 1.7.6 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, con su nueva línea de utilización Java (RMI), 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 (11).

### 1.7.7 Características Principales de Java.

Java es un lenguaje de cuarta generación orientado a objetos que ofrece ventajas, se caracteriza por su sencillez ya que ofrece toda la funcionalidad de un lenguaje potente, pero sin las particularidades sofisticadas que provocan confusiones. Cabe mencionar que se diseño con la finalidad de que su aprendizaje y utilización resultaran naturales para el programador profesional. Orientado a objetos. Java toma prestadas muchas ideas de entornos orientados a objetos de las últimas décadas, consiguiendo un equilibrio espectacular entre el modelo purista y el modelo pragmático. El modelo de objetos es sencillo y de fácil ampliación. Trabaja con sus datos como objetos y con interfaces a esos objetos. Soporta las tres características propias del paradigma orientado a objetos: encapsulación, herencia y polimorfismo. Distribuido. Se construyó con extensas capacidades de interconexión TCP/IP. Existen bibliotecas que

permiten interactuar con protocolos como HTTP Y FTP. Robusto. El entorno multiplataforma de la Web es muy exigente con un programa, ya que éste se debe ejecutar de manera fiable. Por tal motivo, la capacidad para crear programas robustos tuvo una alta prioridad en el diseño de Java. Para ganar confiabilidad, realiza verificaciones tanto en tiempo de compilación como en tiempo de ejecución y, así, consigue encontrar rápidamente los errores más comunes en el desarrollo del programa. Arquitectura neutral. Para hacer independientes las aplicaciones escritas en Java independientes, se compila su código en un archivo objeto (byte codes) que no depende de la arquitectura de la máquina en que será ejecutado. Cualquier máquina que tenga el sistema runtime puede ejecutar ese código objeto, sin importar la plataforma en que ha sido generado. Seguridad. El código en Java pasa por una serie de pruebas antes de ejecutarse en una computadora. Dicho código se pasa a través de un verificador de (byte codes) que comprueba el formato de los fragmentos de código y aplica un probador de teoremas para detectar fragmentos de código ilegal, el cual viola derechos de acceso sobre objetos o intenta cambiar el tipo o clase de un objeto. Este mismo mecanismo permite la portabilidad entre plataformas. Portabilidad. Implementa estándares para facilitar el desarrollo, tal es el caso de los tipos de datos o interfaces gráficas de usuario que mantienen la independencia de la plataforma. Interpretado. El intérprete Java, que es un sistema runtime, puede ejecutar directamente el código objeto. La independencia de la plataforma tiene un costo, Java es más lento que otros lenguajes de programación, ya que su código debe ser interpretado y no ejecutado, como sucede en lenguajes como C++. Multihilos. Fue diseñado para satisfacer los requerimientos del mundo real, en cuanto a crear programas interactivos en un ambiente de red. Para ello, proporciona la programación multihilo, la cual permite la escritura de programas que realicen varias tareas simultáneamente (12).

#### 1.7.7.1 Java RMI (Remote Method Invocation) Invocación a Métodos Remotos.

**RMI** Fue el primer framework para crear sistemas distribuidos de Java. El sistema de Invocación Remota de Métodos (RMI) de Java permite, a un objeto que se está ejecutando en una Máquina Virtual Java (VM), llamar a métodos de otro objeto que está en otra VM diferente. Esta tecnología está asociada al lenguaje de programación Java, es decir, que permite la comunicación entre objetos creados en este lenguaje. La idea de Java RMI es: extender el modelo de invocación de métodos dentro del espacio de direccionamiento de una sola máquina virtual a invocaciones remotas de forma lo más transparente

posible para el desarrollador. La invocación de métodos se puede realizar entre diferentes VM interconectadas a través de una red de comunicación (11).

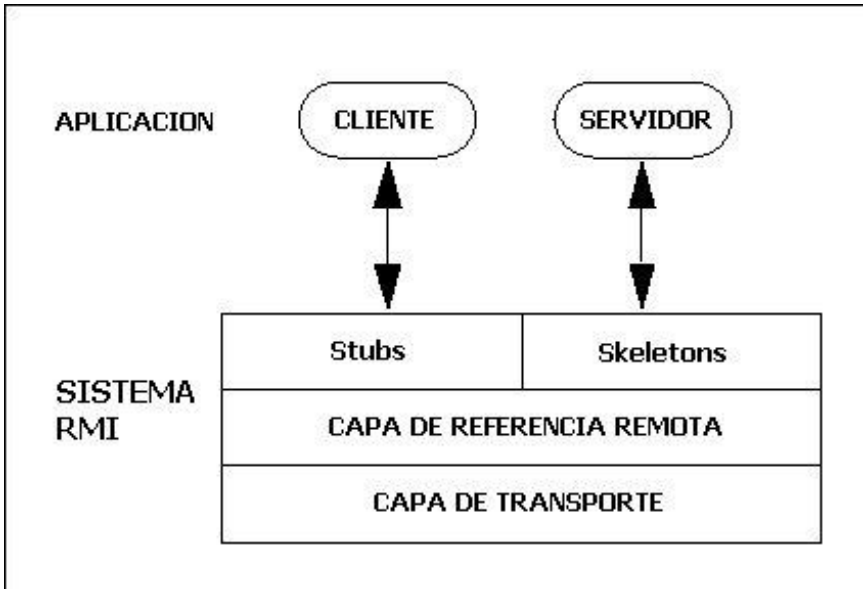


**Figura 4** Arquitectura RMI.

Una característica a destacar de Java RMI es que incluso los objetos pueden pasarse como parámetros de una invocación, o como valores de retorno. Para ello SUN ha desarrollado una metodología para transformar los objetos en flujos de octetos (byte-stream) y así poder ser enviados por los canales de comunicación. A esta metodología se la conoce como serialización de los objetos. Es una característica que también es útil para almacenar en algún medio persistente a los objetos, algo fundamental por ejemplo en las bases de datos de objetos. En SUN ven a Java RMI como una extensión natural al paradigma de los objetos de su RPC (Llamadas remotas a procedimientos). Con esto se logra el objetivo de hacer lo más transparente posible el hecho de que la invocación sea remota, de forma que los programadores tienen un modelo común de objetos tanto para aplicaciones centralizadas como distribuidas, el modelo de objetos de Java. Además, RMI tiene algunas características nuevas que no se encuentran por ejemplo en otras de estas tecnologías distribuidas CORBA o DCOM: la capacidad de distribuir de forma dinámica el código ejecutable gracias a que los objetos pueden viajar como parámetros en las invocaciones. Esta característica se presenta como fundamental en el mundo de los ordenadores que compañías como SUN, Netscape y Oracle entre otras tienen en mente: redes con un servidor central

desde donde se distribuye todo el software, tanto datos como programas ejecutables, a los clientes, que serán los conocidos NC (Network Computer) (11). Esto conlleva las mencionadas ventajas de facilidad de instalación y mantenimiento del sistema.

### 1.7.8 Eclipse.

Eclipse es una poderosa herramienta que permite integrar diferentes aplicaciones para construir un entorno integrado de desarrollo (IDE). Es un entorno de desarrollo de Java , usa java como lenguaje de programación aunque permite plugins 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 plugins. Este mecanismo permite desarrollar, integrar y correr nuevos plugins. Existen por lo general, dos formas de instalar los nuevos plugins en Eclipse. En la mayoría de los casos sólo hay que copiar el plugin en el directorio en el que se encuentra instalado Eclipse. 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++,
- Se ejecuta en varios sistemas operativos incluyendo Windows y Linux.
- Provee a los desarrolladores, herramientas (ej.- PDE) que facilitan la creación de plugins.

### 1.7.9 MOPAC.

MOPAC es un paquete de Química Cuántica de propósito general (cálculo semiempírico) para el estudio de las reacciones químicas. Implementa los Hamiltonians MNDO, MNDO/3, MNDO-PM3 y AM1. Promociona un funcionamiento simple, puede ser ejecutado por consola y de la misma forma pasarle los parámetros necesarios para la realización de las actividades requeridas.

| Estado | Audiencias previstas | Licencia | Lenguaje de Programación |
|---|---|---|---|
| Producto/Estable | Ciencia/investigación | Dominio Público | C FORTRAN |

**Tabla 1** Características de Mopac.

### 1.7.10 Babel.

Babel es un programa diseñado principalmente para el intercambiado de formatos químicos, para la transformación de formatos a otros deseados por los especialistas que lo utilicen. Es capaz de agregar la hibridación, la orden de enlace y la conectividad si es que estos no están presentes en el fichero a procesar. Babel fue diseñado de modo tal que sea flexible para agregarle nuevos formatos de ficheros, permite agregarlos fácilmente. La adición de un nuevo formato consiste simplemente en crear las rutinas del lector y/o escritor y en agregar algunas líneas en el código del programa principal. Existen diferentes versiones para diferentes sistemas operativos. Es una herramienta de libre uso.

## 1.8 Conclusiones.

Al analizar todas las posibilidades y poniéndolas contra las necesidades existentes, se puede arribar a una posible solución que de forma racional haga extensión de los conocimientos que se tienen. Se aprovecha entonces las posibles vías de solución para hacer comparaciones de factibilidad entre ellas, así arribar a una conclusión: para obtener un sistema que sea capaz de funcionar de forma casi autónoma, que cumpliría con todos los requerimientos exigidos, además de que deber ser flexible a cambios, por lo que debe contar con la documentación necesaria para dicha tarea, haría falta realizar una buena selección dentro de las tecnologías existentes. Luego del análisis y comparación se decide seguir la siguiente línea de trabajo:

- Para mantener una compatibilidad con las normas de calidad del centro (UCI) que se encuentran basadas en él y por ser un proceso de desarrollo de software capaz de ser aplicado a cualquier proyecto sin importar la magnitud del mismo, además por las ventajas ya vistas se utilizó como metodología de desarrollo RUP.
- Por consiguiente el lenguaje de modelado será UML, ya que implementa un lenguaje de modelado común para todos los desarrollos y la documentación que crea también es común, por lo que

puede ser entendida por cualquier desarrollador que tenga conocimientos del lenguaje de modelado.

- La herramienta CASE utilizada para modelar el programa es el Visual Paradigm ya que es un producto de Visual Paradigm UML Community que, a su vez, es una de las principales compañías de herramientas CASE.
- Como IDE se escogió eclipse desarrollado por IBM, amplia escala de posibilidades brindadas para esta tipo de trabajo.
- El lenguaje de programación que se utilizó fue Java, aprovechando RMI y sus posibilidades, para la invocación a las tareas remotas, además por su posibilidad de uso gratuito.
- Para realizar los cálculos deseados se utilizó, versiones del Mopac, así como la aplicación Babel para la conversión a los formatos deseados, ambas herramientas con versiones para cualquier sistema operativo y de uso libre.

# CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

## 2.1 Introducción.

En el presente capítulo se hace la descripción del objeto de estudio y las reglas del negocio para un mayor entendimiento del problema. Se especifican los detalles de la construcción del módulo, brindando como parte de estos los requerimientos funcionales y no funcionales que dan marcha al desarrollo del sistema. Se realizó descripciones a los casos de uso que representan las funcionalidades principales del módulo.

## 2.2 Modelo del Dominio.

Las transformaciones de modelos buscan la generación de nuevos modelos en forma idealmente automática, que de otra forma será manual. Esta técnica tiene la potencialidad de ahorrar cantidad de trabajo a los desarrolladores, así como la de evitar errores, entre otras cosas. Así el modelo del dominio tiene como objetivo fundamental la comprensión y descripción de las clases más importantes. Resume todo lo que tenga importancia conceptual para el entorno en que desarrolle el sistema y las relaciones que existan entre estos conceptos.

## 2.3 Reglas del negocio a considerar.

Las reglas de negocio describen políticas que deben cumplirse o condiciones que deben satisfacerse, por lo que regulan algún aspecto del negocio. Todo el proceso de negocio descrito a continuación se desarrolla sobre la plataforma a que pertenece el módulo presente.

- El especialista químico desea procesar un sinnúmero de ficheros, realizar sobre ellos cálculos químicos que le puedan servir para optimizar resultados útiles en su trabajo.
- Se toman una serie ficheros, los que se van a procesar, y se le suministran al software que realiza los cálculos.
- Poder obtener resultados organizados de los cálculos químico-teóricos que se realicen.
- El especialista analiza los resultados, los aplica a sus estudios.

**2.4 Diagrama Modelo del Dominio.**

Descripción Textual del Modelo del Dominio.

El especialista químico presenta una serie de ficheros moleculares que tienen información necesaria para que sus estudios progresen, desea procesarlos para así disminuir el rango de errores en que oscilarán sus estudios, le aplicará técnicas químico-cuánticas. Toma una serie de estos y los procesa con la aplicación necesaria, dependiendo del tipo de optimización que desee realizar. De este proceso se obtienen resultados útiles para continuar con sus estudios. Los resultados se analizan.

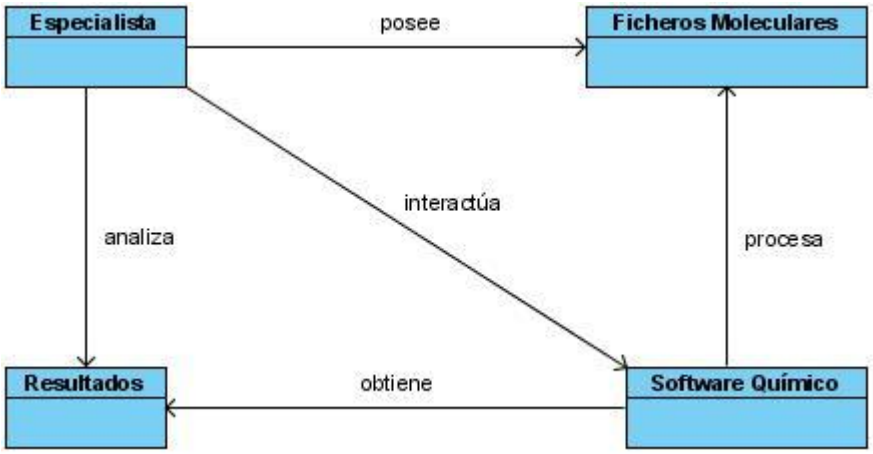


**Figura 5** Modelo del dominio.

**2.5 Requerimientos Funcionales.**

De acuerdo con los objetivos planteados el sistema será capaz de:

**R1**: Gestionar ficheros de entrada.

- **1.1** Reconocer extensiones químicas asociadas a formatos utilizados.
- **1.2** Convertir a formato compatibles con el software que realizará los cálculos químicos.
- **1.3** Agregar a cada fichero, la función personalizada (características del cálculo a realizar).

**R2**: Gestionar set a procesar.

**2.1** Seleccionar set de ficheros desde una fuente, para ser procesados.

**2.2** Copiarlos físicamente hasta la máquina que realizará el cálculo.

**R3**: Realizar cálculos químico-teóricos.

**3.1** Seleccionar ficheros de entrada a los cálculos químico-teóricos.

**3.2** Realizar cálculos químico-teóricos según ficheros de entrada.

**R4**: Gestionar resultados.

**4.1** Seleccionar resultados completados correctamente.

**4.2** Leer cada resultado seleccionado.

**4.3** Almacenar en el servidor de bases de datos cada resultado correcto leído.

### 2.6 Requerimientos No Funcionales.

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener. Representan las características del producto.

**Usabilidad:**

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

**Rendimiento:**

Al estar concebida para un ambiente cliente/servidor, se trata de garantizar la rapidez de respuesta del sistema ante las solicitudes de los usuarios, al igual que la velocidad de procesamiento de la información. Para lo cual se realiza la validación de los datos y la manipulación de eventos en el cliente y en el servidor aquellas que por cuestiones de seguridad, o de acceso a los datos lo requieran. Lográndose así un tiempo de respuesta más rápido, una mayor velocidad de procesamiento, y un mayor aprovechamiento de los recursos.

**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.

**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 plataforma 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, esta incluye la de cada módulo que la conforma.

**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

### 2.7 Actores del Sistema a Automatizar.

Los actores del sistema pueden representar el rol que juega una o varias personas, un equipo o un sistema automatizado, son parte del sistema, y pueden intercambiar información con él o ser recipientes pasivos de información.

| Actores | Justificación |
|---|---|
| Especialista | Representa el usuario que va a hacer uso del sistema, teniendo la posibilidad de interactuar con todas la funcionalidades de este. |
| Cliente(PC_Ociosa) | Equipo, fuente para la ejecución de los cálculos químico-teóricos. |
| Babel | Programa que realiza las conversiones de formatos de ficheros. |
| Software Químico | Encargado de realizar los cálculos químico-teóricos. |

**Tabla 2** Actores del sistema.

### 2.8 Casos de Usos del Sistema.

| Cod. | Nombre de caso de uso | Justificación de la selección. |
|---|---|---|
| **1** | Gestionar Ficheros de entrada. | Necesario para obtener ficheros compatibles con la aplicación. |
| **2** | Gestionar Set a procesar. | El sistema necesita tener los ficheros en un lugar específico, listos par ser procesados. |
| **3** | Realizar cálculos químico-teóricos. | Objetivo principal del módulo, abarca todo el proceso de los cálculos químicos deseados. |
| **4** | Gestionar Resultados. | Necesidad de obtener y guardar los resultados obtenidos en los cálculos. |

**Tabla 3** Casos de uso del sistema.

## 2.9 Diagrama de Casos de Uso del Sistema.

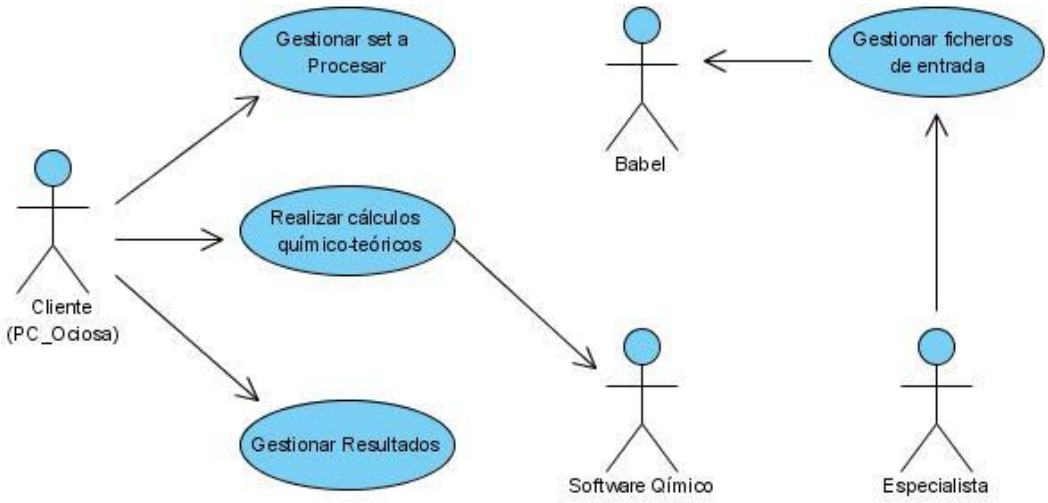


**Figura 6** Diagrama de Casos de Uso del Sistema.

## 2.10 Descripción de los Casos de Uso del Sistema.

| **Caso de Uso:** | **Gestionar ficheros de entrada** |
|---|---|
| **Actores:** | Especialista, Babel. |
| **Propósito:** | Llevar los ficheros que se deseen procesar a un formato compatible con la aplicación, dejarlos en un lugar accesible; listo para realizar los cálculos que se necesitan. |
| **Resumen:** | El especialista inicia el caso de uso, toma la opción de seleccionar ficheros a procesar. El sistema muestra una ventana para que navegue hasta el directorio en que se encuentren. Luego los ficheros compatibles son procesados y guardados en el servidor de bases de datos. |
| **Referencia:** | R1 |
| **Precondiciones:** | Que existan ficheros por procesar. |
| **Poscondiciones:** | Los ficheros quedan salvados en el formato deseado. |
| **Flujo Normal de Eventos** | |
| **Acción del Actor** | **Respuesta del Sistema** |

| | |
|---|---|
| 1. El especialista selecciona la opción procesar ficheros. | 1.1 El Sistema abrirá un explorador donde el especialista podrá localizar sin dificultad el directorio donde se encuentran sus ficheros. |
| 2. El especialista selecciona el directorio. | 2.1 El Sistema filtra dentro del árbol de directorios hijos los ficheros que son compatibles para procesar. |
| | 2.2 El Sistema identificó la dirección y extensión de los ficheros, seleccionando los .mol para procesarlos y llevarlos a la extensión .mop. |
| 3. Babel, toma los ficheros indicados por el especialista, y los procesa. | 2.3 El Sistema lee los ficheros procesados y los envía al servidor de bases de datos para su fácil acceso. |
| **Flujos Alternos** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| | |
| | |
| **Prioridad:** | Secundario |

**Tabla 4** Descripción del CU: Gestionar ficheros de entrada.

| **Caso de Uso:** | **Gestionar set a Procesar** |
|---|---|
| **Actores:** | Cliente (PC_Ociosa) |
| **Propósito:** | Seleccionar y descargar hasta el cliente (PC_Ociosa) un set de ficheros listos para ser procesados. |
| **Resumen:** | El actor Cliente (PC_Ociosa) inicia el caso de uso. El sistema selecciona un número de compuestos, los tiene almacenados en un búfer. Luego a cada pedido cliente le asigna una parte del búfer hasta quedar sin petición o sin compuestos que distribuir. Cada cliente deja listo y escrito en disco cada fichero que se la asignó para ser procesado. |
| **Referencia:** | R2 |
| **Precondiciones:** | Que queden compuestos por procesar en búfer. |
| **Poscondiciones:** | Quedan listos en ficheros los compuestos para ser procesados. |

| Flujo Normal de Eventos | |
|---|---|
| **Acción del Actor** | **Respuesta del Sistema** |
| 1. Notifica al sistema que se encuentra lista para ejecutar una tarea. | 1.1 A través del servidor se le envía un conjunto de herramientas necesarias para que procese un listado de compuestos. |
| | 1.2 Selecciona del búfer de compuestos un set y lo envía al cliente. |
| | 1.3 Copia físicamente y deja listo en el Cliente (PC_Ociosa) el set para ser procesado. |
| **Flujos Alternos** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| 1. Notifica al sistema que se encuentra lista para ejecutar una tarea. | 1.1La aplicación hace nula la respuesta; se terminaron las tareas. |
| | |
| **Prioridad:** | Crítico |

**Tabla 5** Descripción del CU: Gestionar set a Procesar.

| **Caso de Uso:** | **Realizar cálculos químico-teóricos** |
|---|---|
| **Actores:** | Cliente (PC_Ociosa), Software Químico. |
| **Propósito:** | Procesar los compuestos y obtener resultados. |
| **Resumen:** | El actor Cliente (PC_Ociosa) inicia el caso. El sistema procede a través de la interfaz del software químico a procesar el set de ficheros existentes. El procesamiento genera ficheros que son resultados de los cálculos. |
| **Referencia:** | R3 |
| **Precondiciones:** | Que existan ficheros por procesar. |

| | |
|---|---|
| **Poscondiciones:** | Quedan procesados los ficheros. |
| **Flujo Normal de Eventos** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| 1. Ejecuta realizar cálculos. | 1.1 El sistema proporciona al software químico que se utilice los datos necesarios en ficheros para que se realicen los cálculos. |
| 2. El software químico procede a realizar los cálculos químicos indicados. | 2.1 El sistema colecta los resultados de los cálculos. |
| **Flujos Alternos** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| | |
| **Prioridad:** | Crítico |

**Tabla 6** Descripción del CU: Realizar cálculos químico-teóricos.

| **Caso de Uso:** | **Gestionar Resultados** |
|---|---|
| **Actores:** | Cliente (PC_Ociosa) |
| **Propósito:** | Que queden guardados los resultados del proceso de cálculo. |
| **Resumen:** | El actor inicia el caso. El sistema procede a guardar los datos que fueron generados por los cálculos químicos. Para esto son copiados los resultados hasta el servidor distribuidor y el mismo los envía a la fuente origen de los datos (servidos de bases de datos), organizados por el compuesto a que pertenecen. |
| **Referencia:** | R4 |
| **Precondiciones:** | Que existan resultados generados por el proceso de cálculo. |
| **Poscondiciones:** | Los resultados quedan guardados en la fuente indicada. |
| **Flujo Normal de Eventos** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| 1. Ejecuta guardar | 1.1 El Sistema a través del servidor distribuidor procede a recoger todos los |

| | |
|---|---|
| los resultados obtenidos. | resultados, guardados en cada cliente. Los resultados son guardados en la fuente indicada. |
| **Flujos Alternos** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| | |
| **Prioridad:** | Crítico |

**Tabla 7** Descripción del CU: Gestionar Resultados.

### 2.11 Conclusiones.

Siguiendo los pasos que plantea UML, dado a una escasa identificación de los procesos del negocio y una mejor comprensión del problema se definieron en este capítulo conceptos, que fueron relacionados mediante un diagrama de modelo del dominio. El mismo muestra de forma general los distintos objetos existentes en el negocio. Se definieron los requisitos que debe cumplir el sistema para su correcto funcionamiento. Determinado así las diferentes funcionalidades del mismo resumidas como casos de uso, de los que se realizó una descripción detallada para su mejor comprensión.

# CAPÍTULO 3: ANÁLISIS Y DISEÑO DEL SISTEMA

## 3.1 Introducción.

En el siguiente capitulo se muestra el modelado visual del funcionamiento del sistema de modo que se transformaron los requerimientos a diseños del sistema, se presenta la arquitectura del sistema, se adapta el diseño para hacerlo corresponder con el ambiente de implementación para así lograr resultados satisfactorios esperados.

## 3.2 Modelo de análisis.

Expresados los requerimientos funcionales del sistema los diagramas de clases del análisis pasan a dar una especificación de lo que es la estructura de las clases que lo conforman, igual quedan definidas las relaciones existentes entre las ellas.

### 3.2.1 Diagrama de clases del análisis.

**Caso de uso Gestionar Ficheros de entrada.**

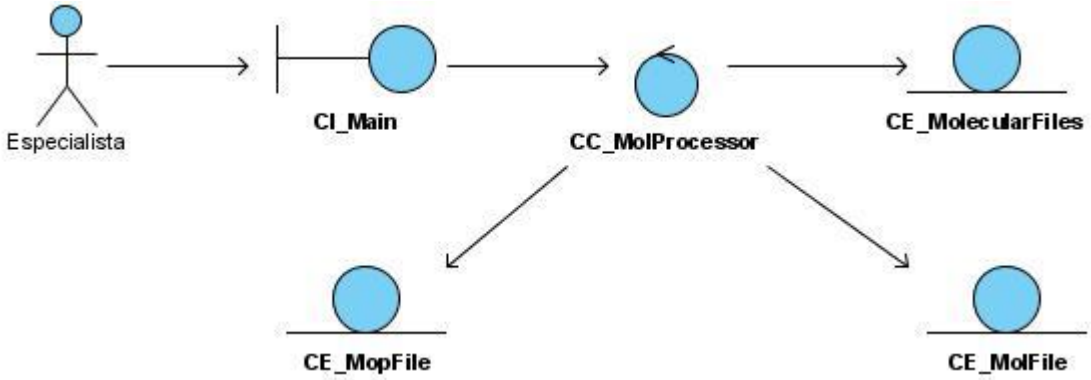


**Figura 7** Diagrama de Clases del Análisis: CU Gestionar Ficheros de entrada.

**Caso de uso Gestionar Set a procesar.**

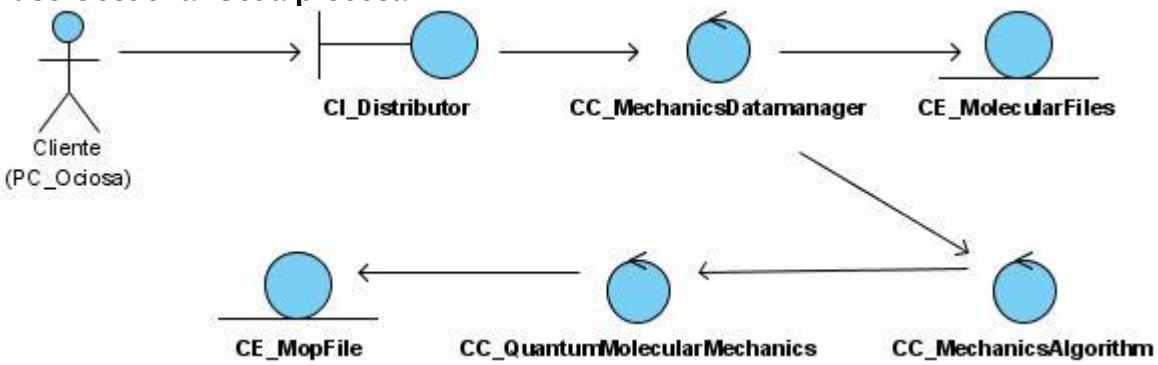


**Figura 8** Diagrama de Clases del Análisis: CU Gestionar Set a procesar.

**Caso de uso Realizar cálculos químico-teóricos escenario # 1.**

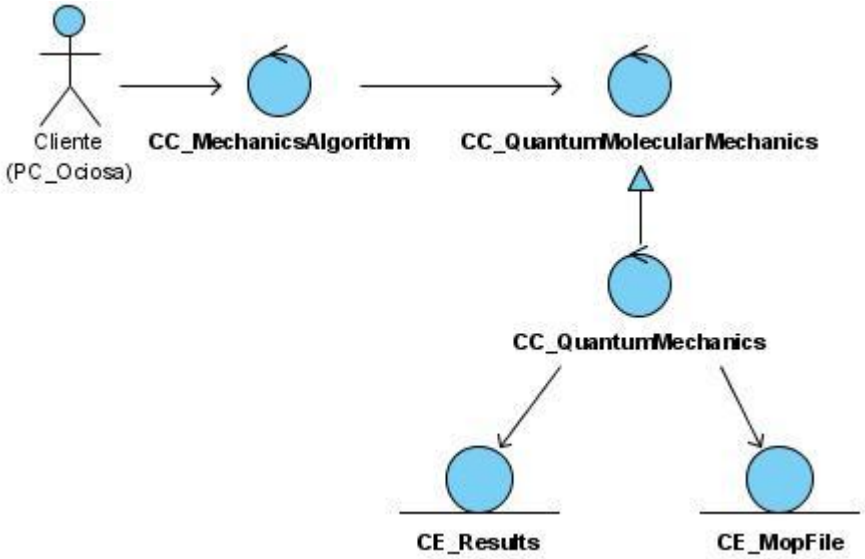


**Figura 9** Diagrama de Clases del Análisis: CU Realizar cálculos químico-teóricos escenario # 1.

**Caso de uso Realizar cálculos químico-teóricos escenario # 2.**

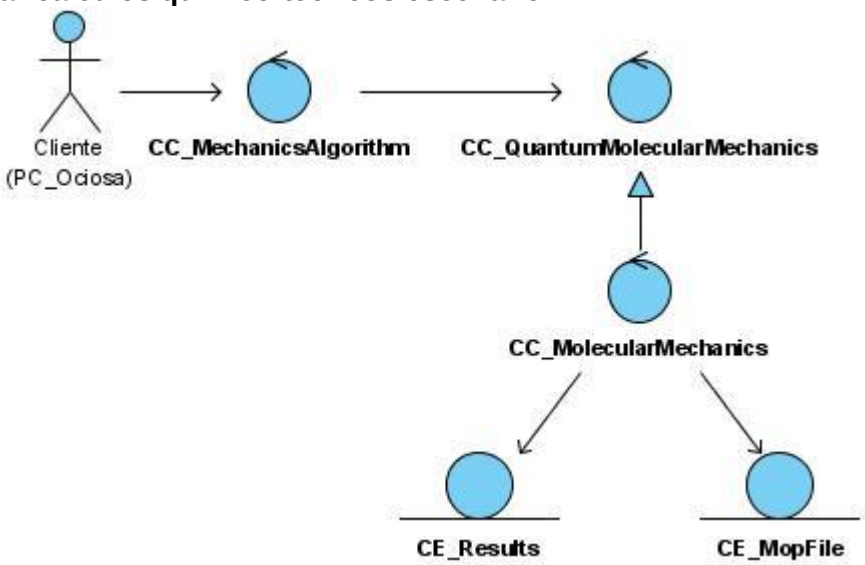


**Figura 10** Diagrama de Clases del Análisis: CU Realizar cálculos químico-teóricos escenario # 2.

**Caso de uso Gestionar Resultados.**

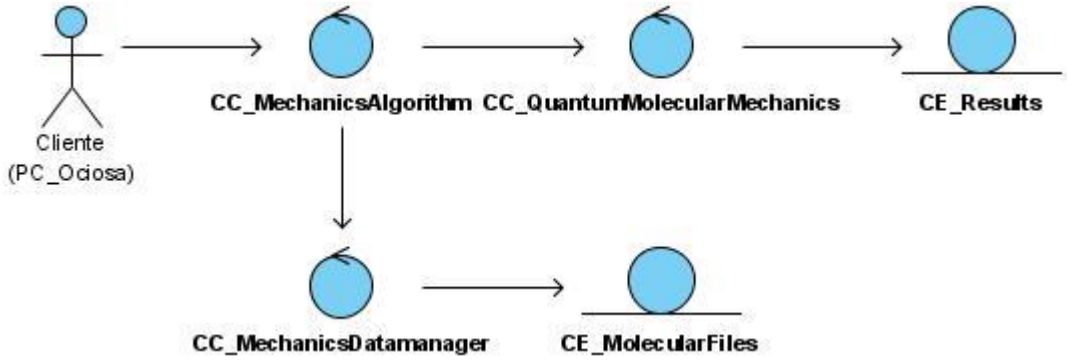


**Figura 11** Diagrama de Clases del Análisis: CU Gestionar Resultados.

**Diagrama de clases del análisis.**

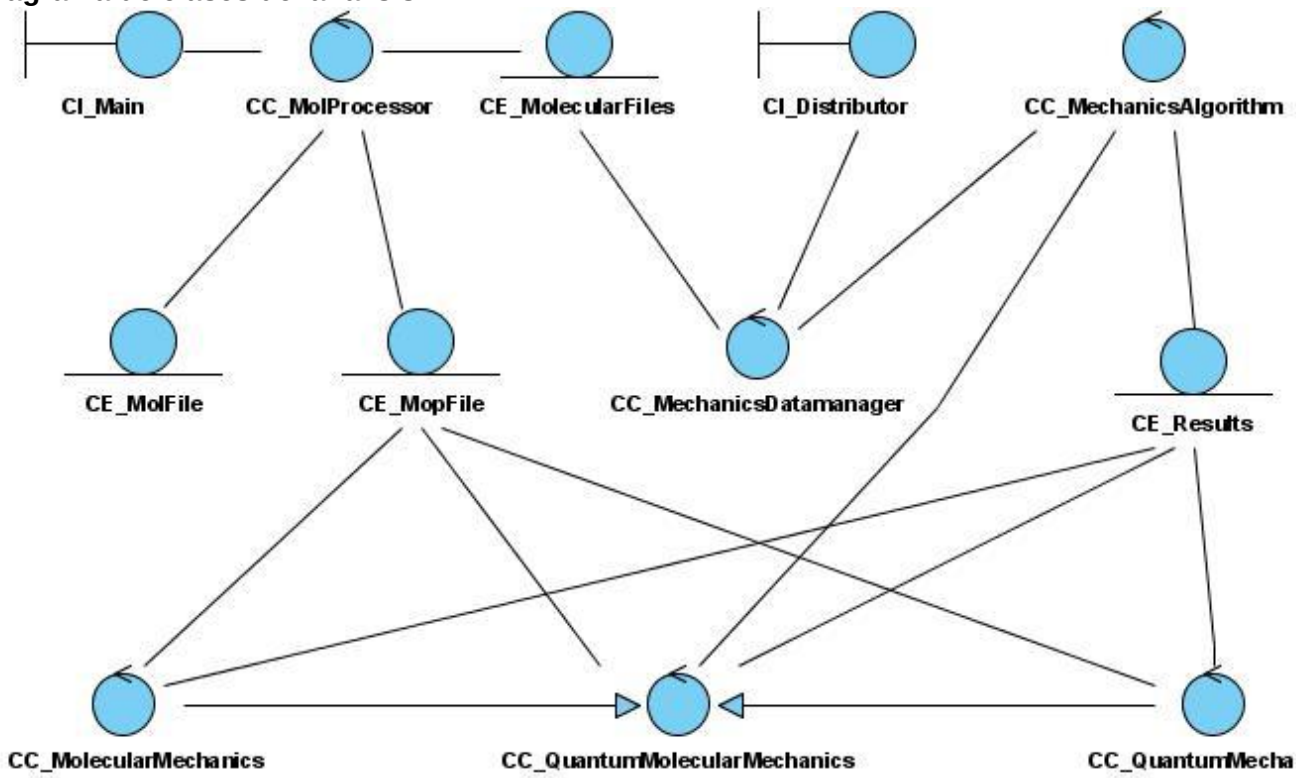


**Figura 12** Diagrama de Clases del Análisis.

### 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.

#### 3.3.1 Diagramas de clases y de interacción del diseño.

Un diagrama de colaboración es una forma de representar interacción entre objetos. A diferencia de los diagramas de secuencia, que proporcionan una forma de ver el escenario en un orden temporal, qué pasa primero, qué pasa después. Los diagramas de secuencia dan un entendimiento simplificado al cliente de los procesos que ocurren.

**Caso de uso Gestionar Ficheros de entrada.**

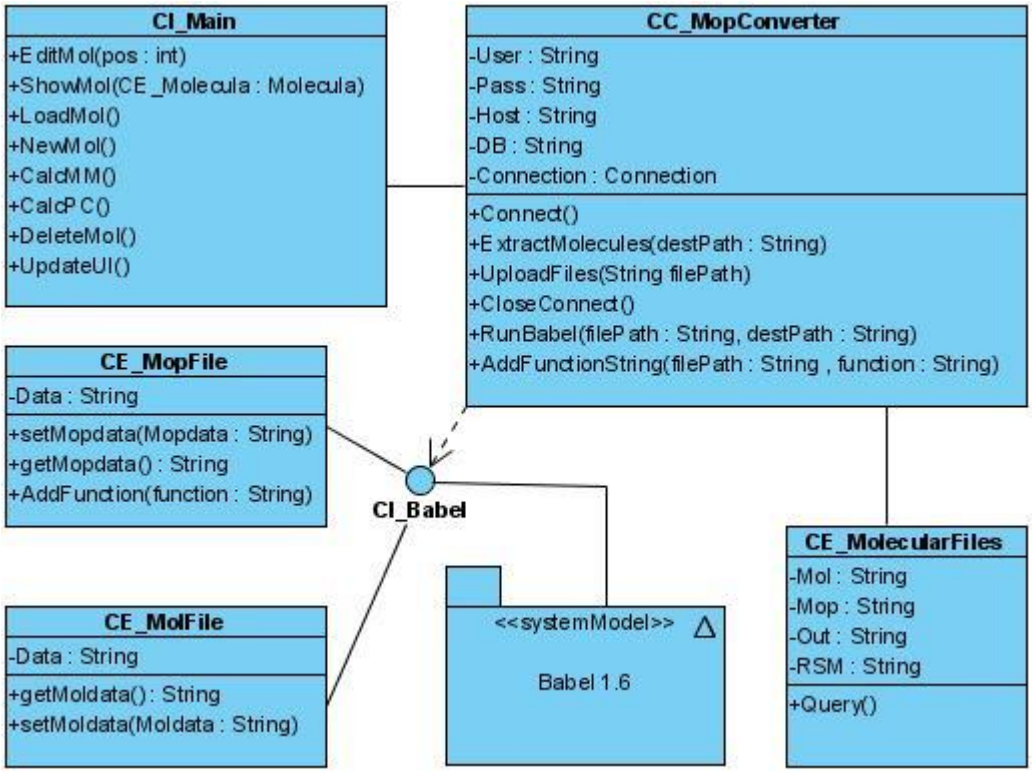


**Figura 13** Diagrama de Clases del Diseño: CU Gestionar Ficheros de entrada.

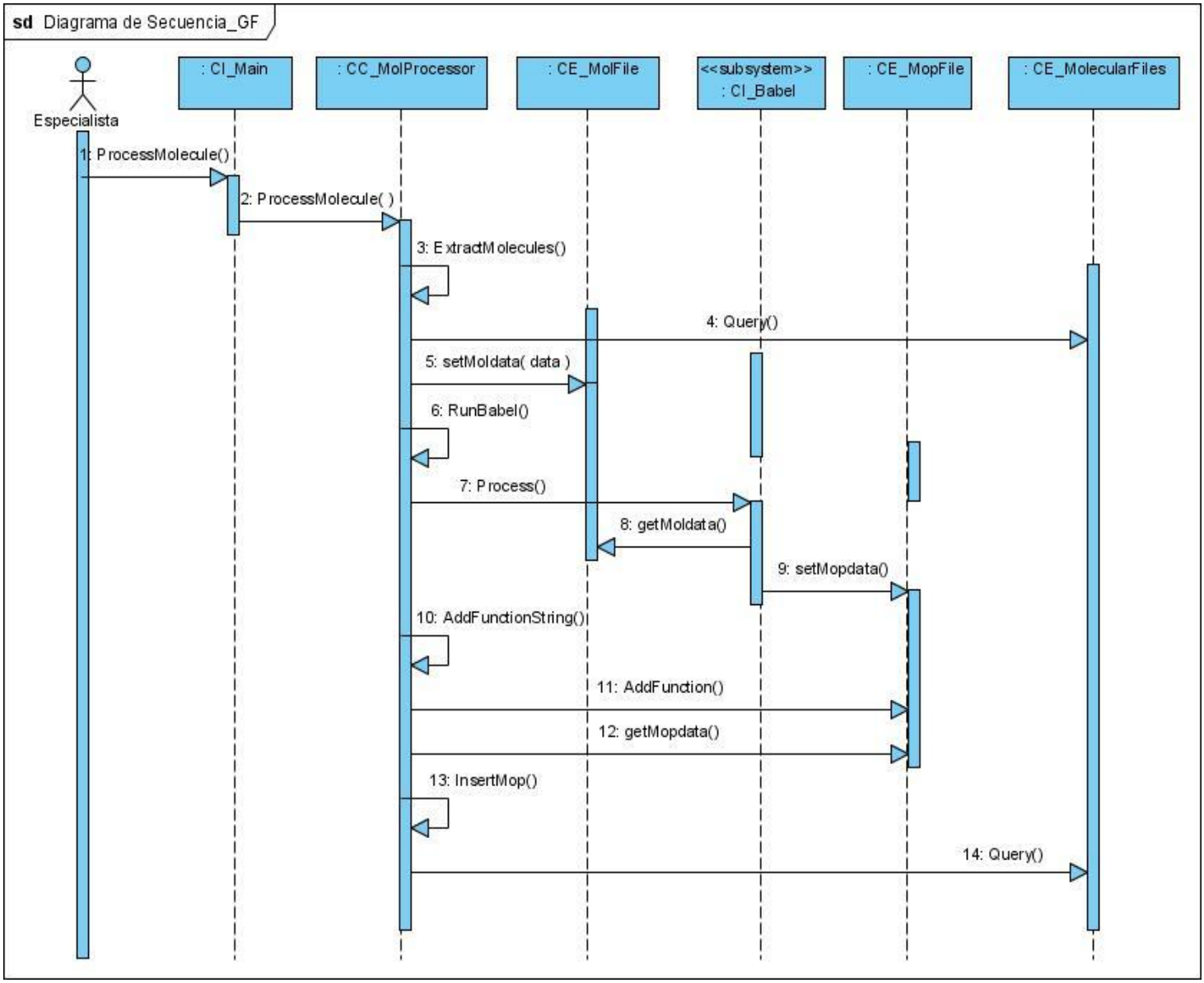


**Figura 14** Diagrama de Interacción del Diseño: CU Gestionar Ficheros de entrada.

**Caso de uso Gestionar Set a procesar.**

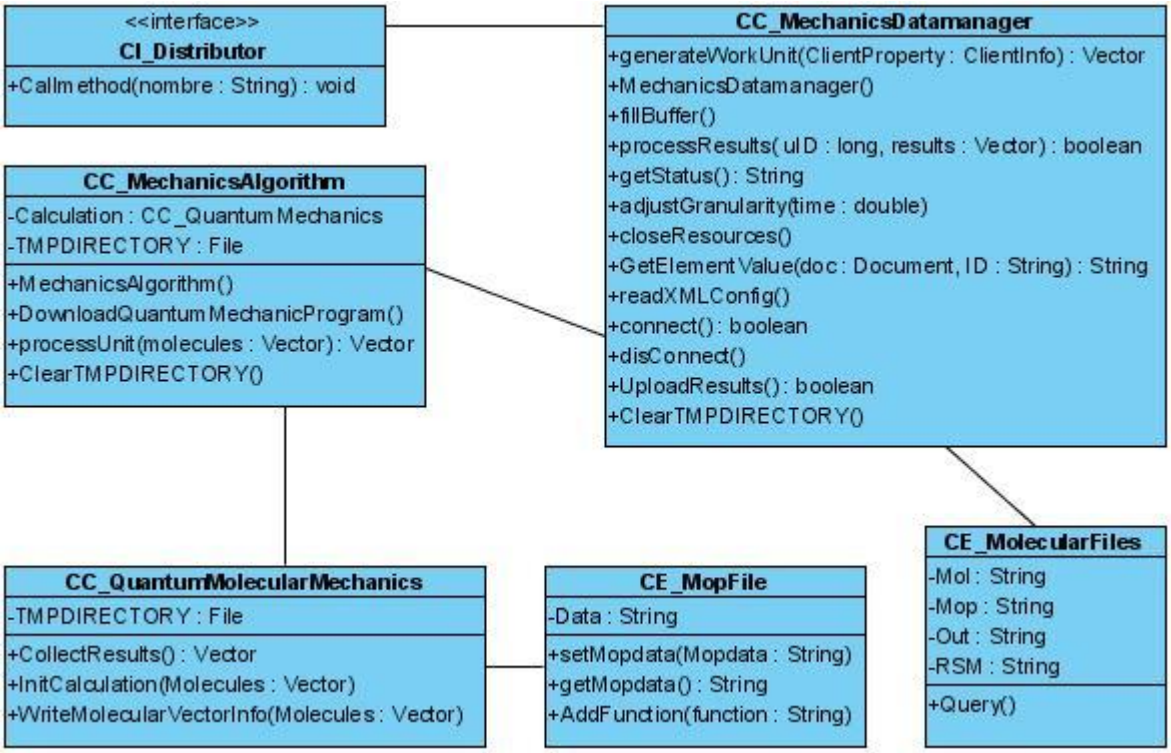


**Figura 15** Diagrama de Clases del Diseño: CU Gestionar Set a procesar.

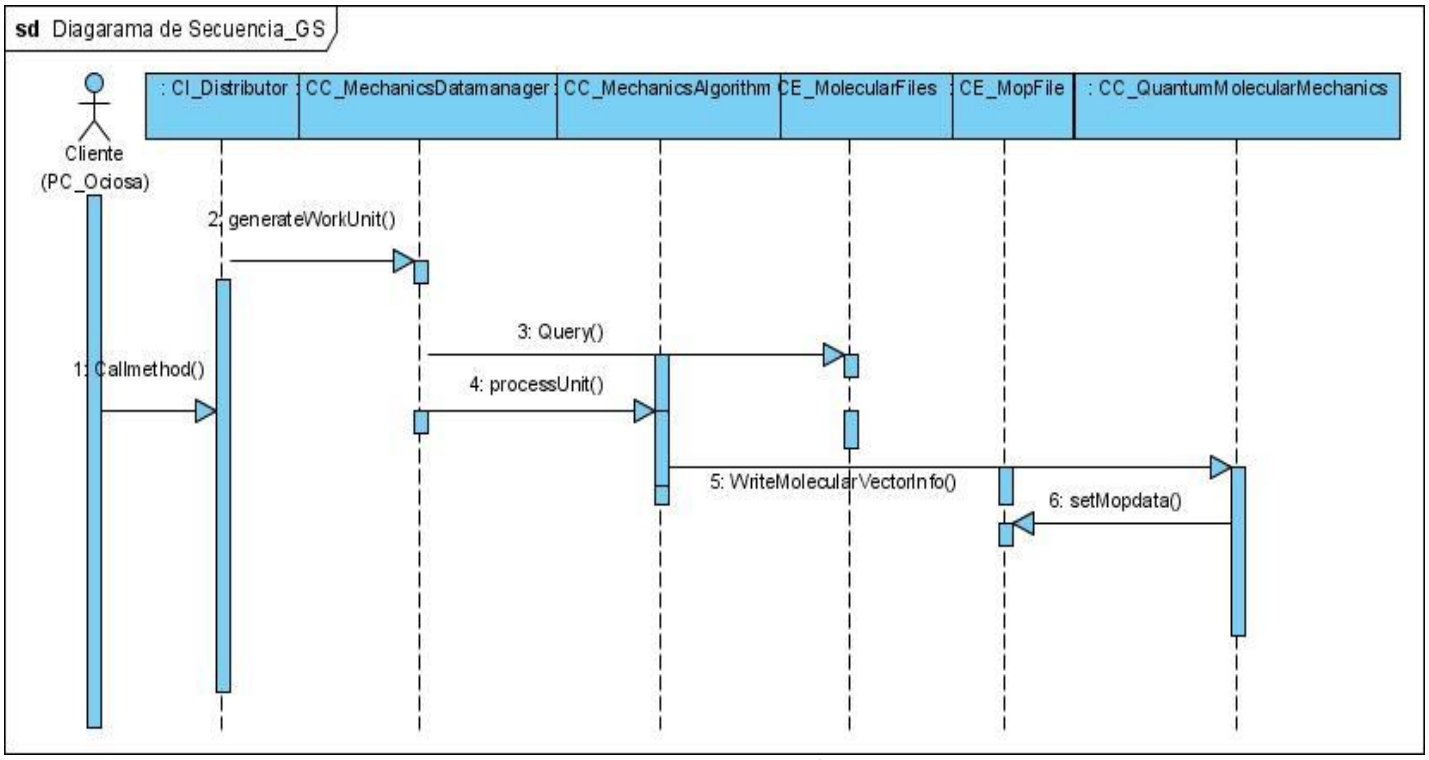


**Figura 16** Diagrama de Interacción del Diseño: CU Gestionar Set a procesar.

**Caso de uso Realizar cálculos químico-teóricos.**

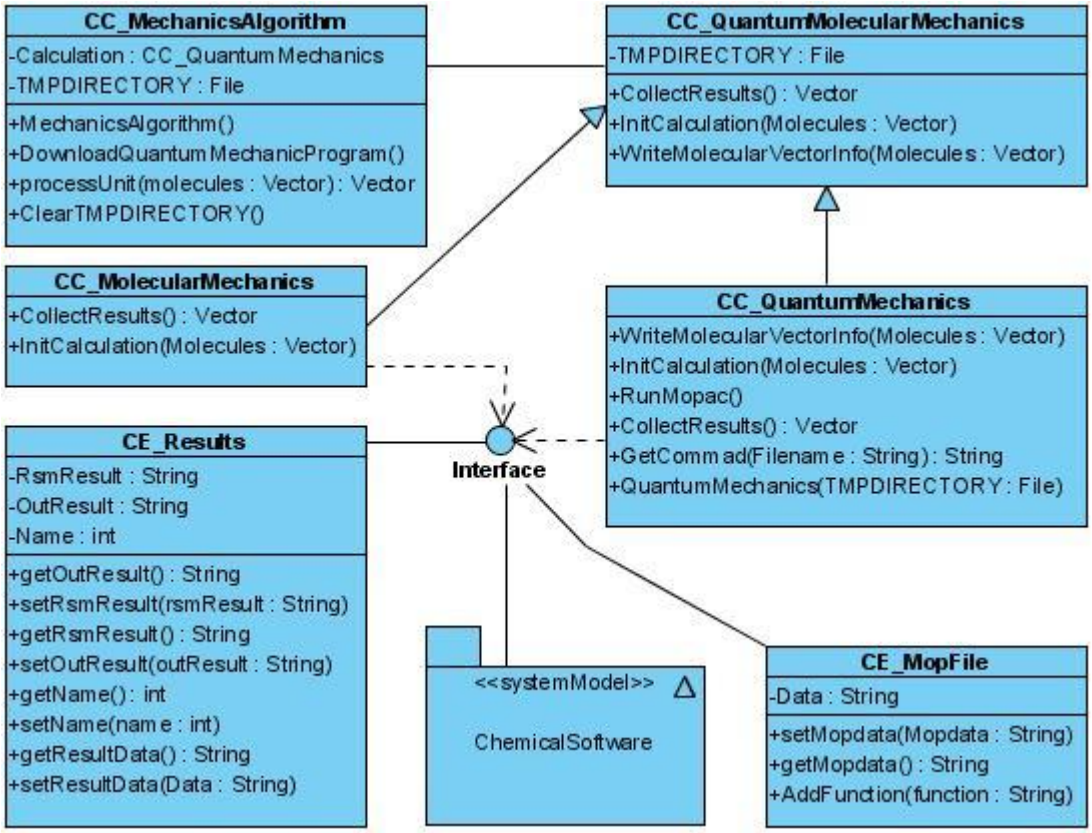


**Figura 17** Diagrama de Clases del Diseño: CU Realizar cálculos químico-teóricos.

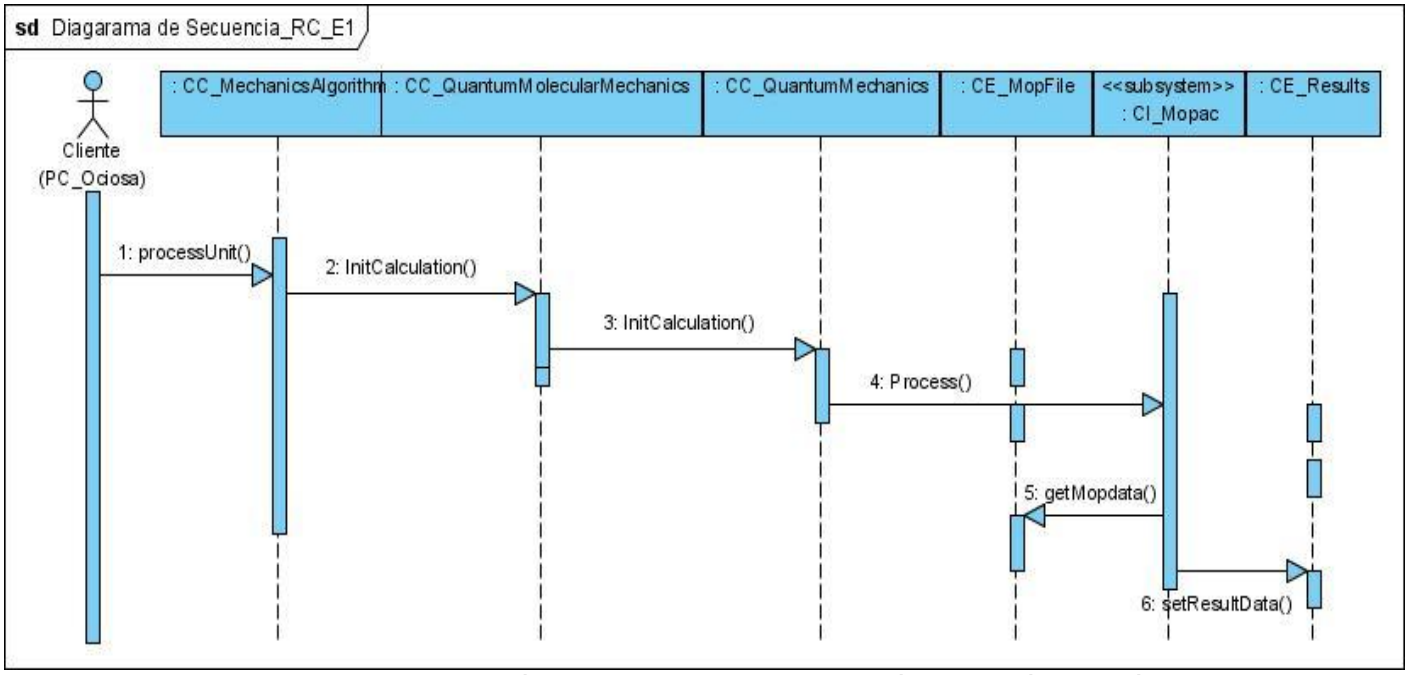


**Figura 18** Diagrama de Interacción del Diseño: CU Realizar cálculos químico-teóricos escenario #1.

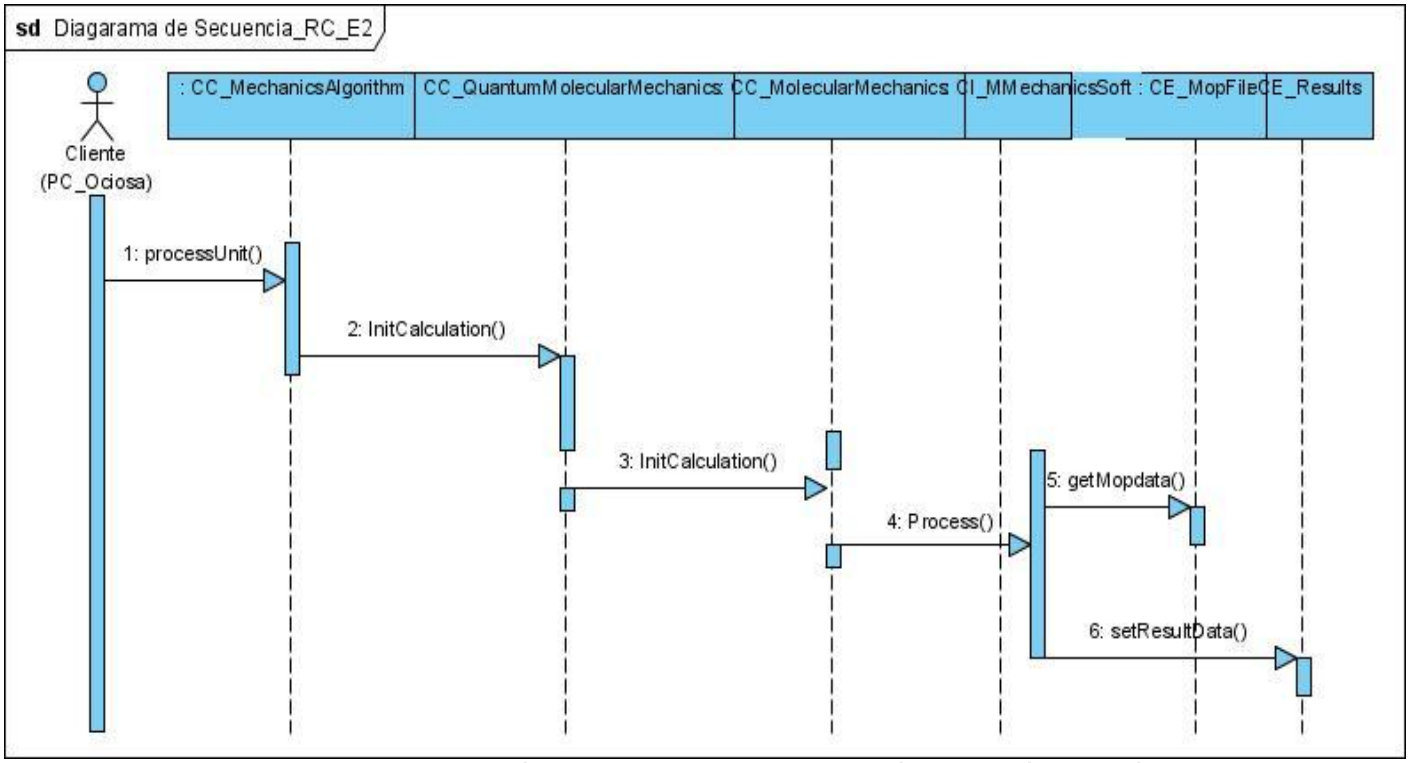


**Figura 19** Diagrama de Interacción del Diseño: CU Realizar cálculos químico-teóricos escenario #2.

**Caso de uso Gestionar Resultados.**

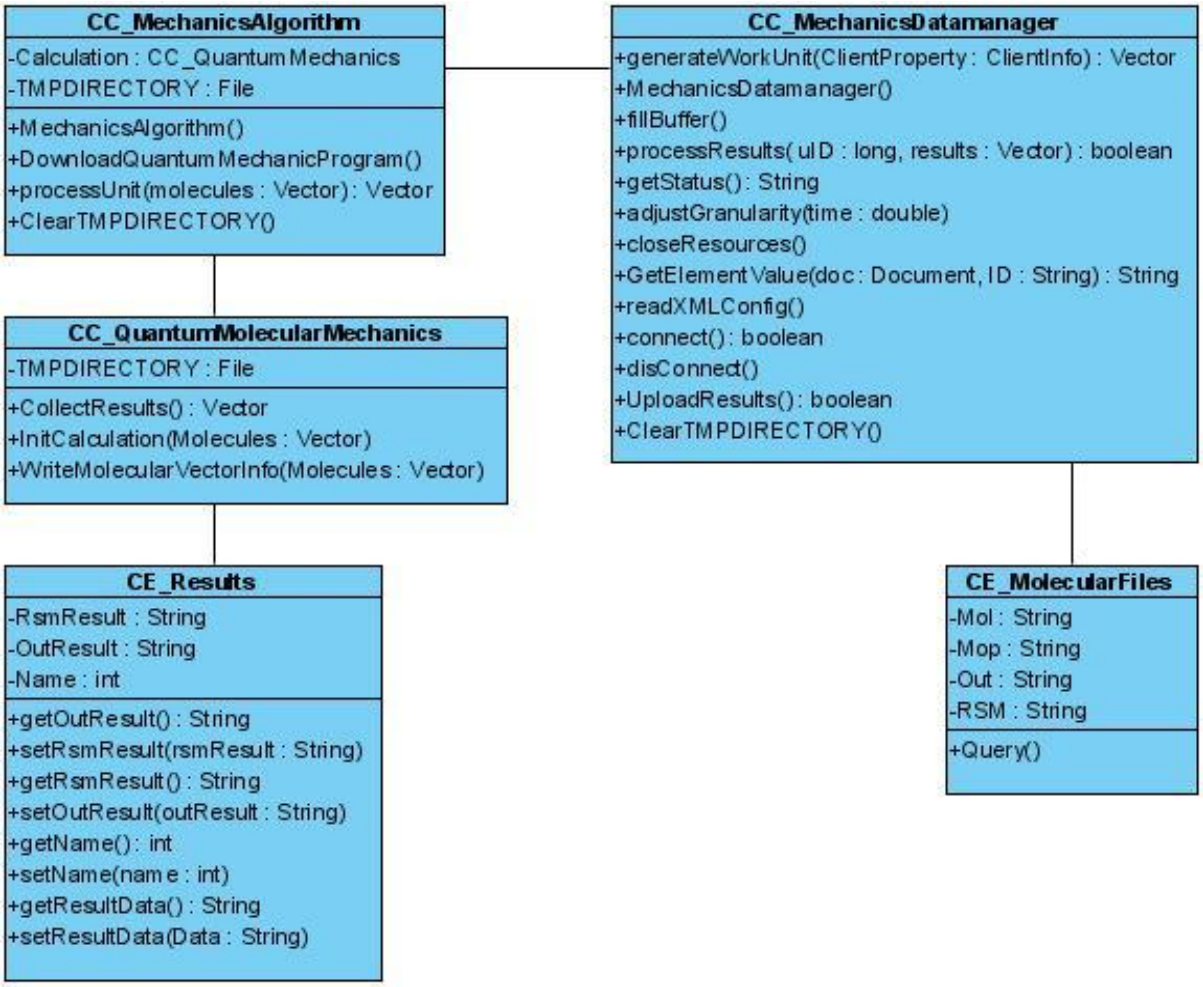


**Figura 20** Diagrama de Clases del Diseño: CU Gestionar Resultados.

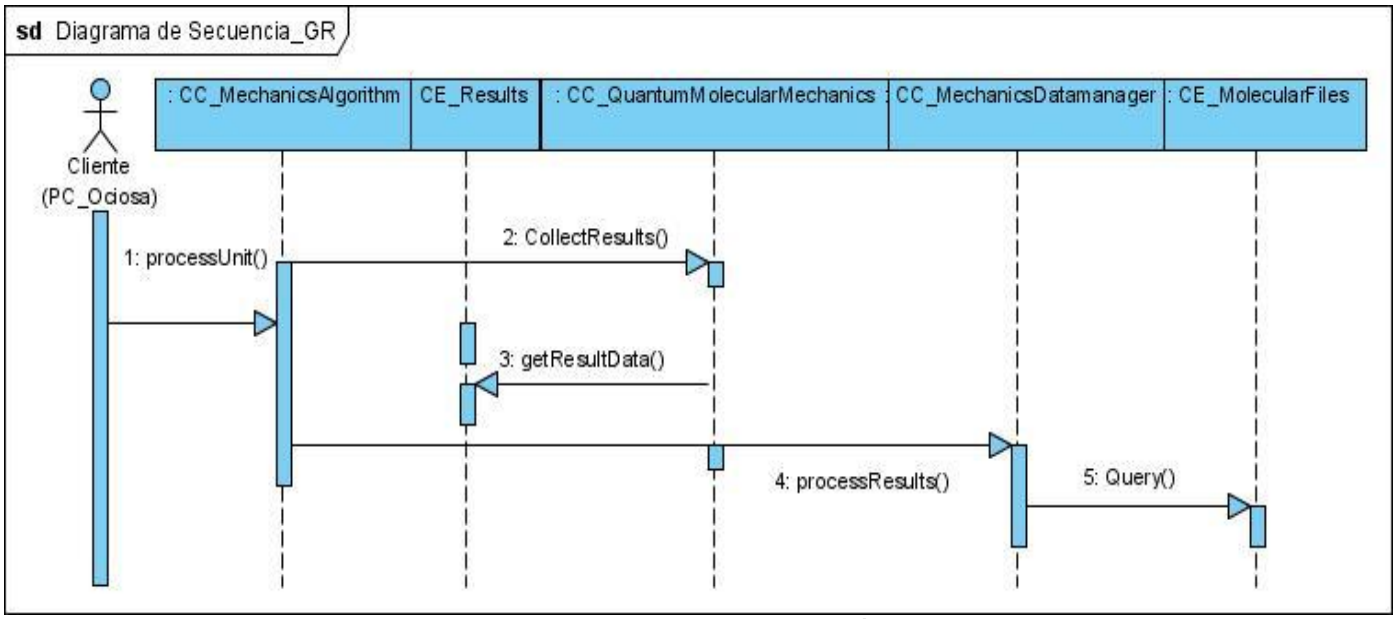


**Figura 21** Diagrama de Interacción del Diseño: CU Gestionar Resultados.

**Diagrama de clases del diseño.**

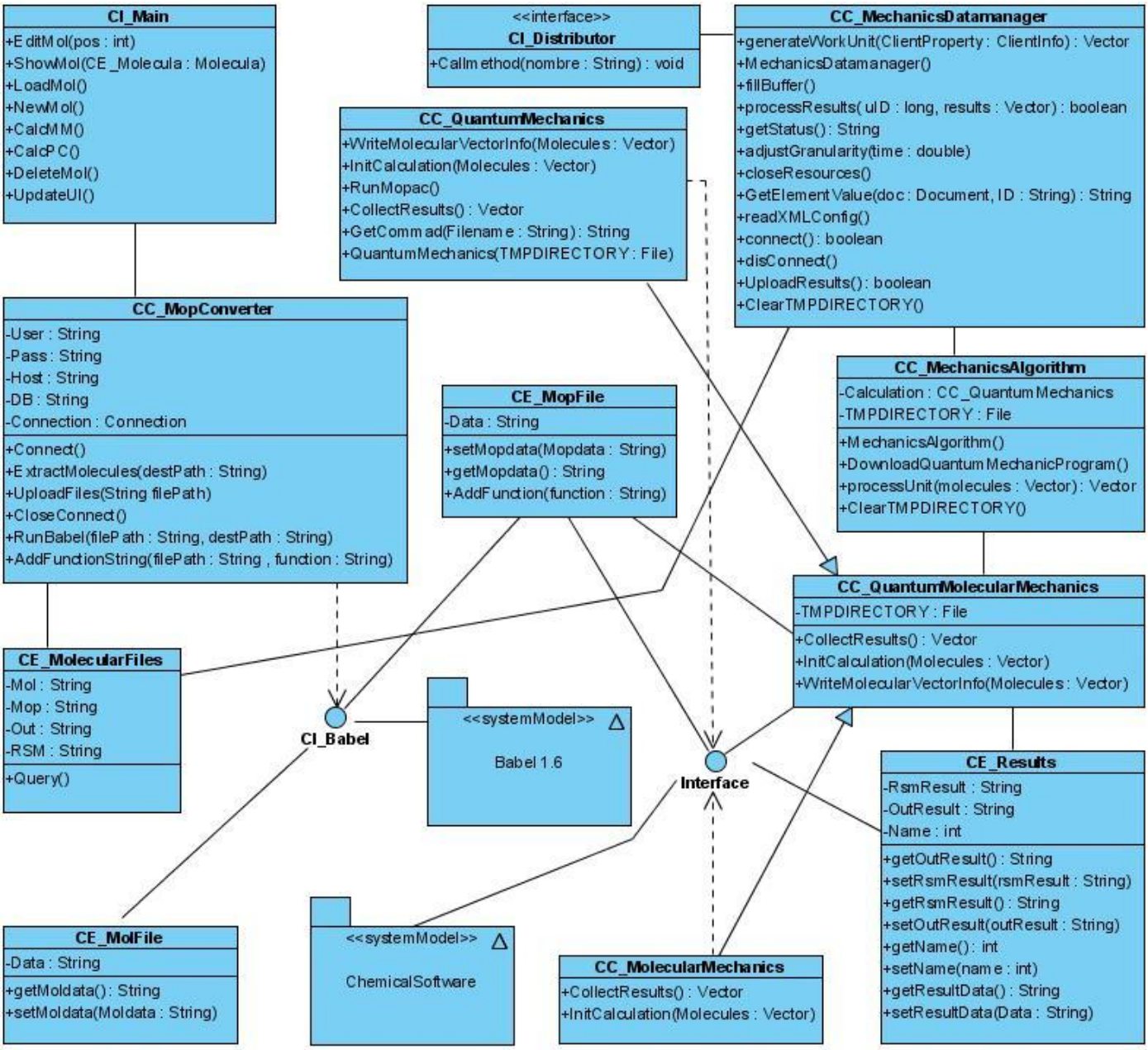


**Figura 22** Diagrama de Clases del Diseño.

### 3.4 Descripción de las clases.

(Ver anexo #1.)

### 3.5 Modelo de Datos.

Un modelo de datos es aquel que describe de una forma abstracta cómo se representan los datos, sea en una empresa, en un sistema de información o en un sistema de gestión de base de datos. Básicamente consiste en una descripción de algo conocido como contenedor de datos, algo en donde se guarda la información, así como de los métodos para almacenar y recuperar información de esos contenedores.

Es preciso conservar los datos resultados que genera la aplicación, y de la misma forma es necesario poder acceder a estos de forma rápida y organizada. Sin dejar de considerar que el presente trabajo constituye un módulo de la plataforma GRATO y que el modelo de datos presentado por la plataforma recoge las posibles entidades a almacenar, por lo que como integración a la misma, se decide añadirle las entidades imprescindibles para que persistan los datos de los cálculos químico-teóricos que gestiona el presente módulo. La entidad MolecularFiles representa la tabla contenedora de los mismos, y la única concerniente al presente trabajo dentro del modelo de datos presentado por la plataforma mencionada.

| **Nombre: MolecularFiles** | | |
|---|---|---|
| **Descripción:** Contenedor de los resultados de los cálculos químico-teóricos. | | |
| **Atributo** | **Tipo** | **Descripción** |
| Incompuesto | INTEGER | Identificador del compuesto químico. |
| Mol | TEXT | Información de la molécula. |
| Mop | TEXT | Contenido del fichero de entrada del Mopac. |
| Out | MEDIUMTEXT | Salida de los cálculos (resultados). |
| Rsm | MEDIUMTEXT | Resumen del proceso de cálculo. |
| | | |

**Tabla 8** Tabla MolecularFiles.

### 3.6 Definiciones de diseño.

Para el desarrollo de una aplicación se llevan de la mano una serie de responsabilidades que dan un funcionamiento correcto y acorde a los requerimientos y gustos de los clientes. Lograrlo es cuestión de seguir un principio de diseño que recoja de forma detallada todo el tratamiento de errores, que abarque el tema de la seguridad de los datos con que se interaccione, así como el diseño correcto de una interfaz para la interacción con la aplicación. En el caso a referencia no se cuenta con una interfaz de usuario bien

definida puesto a que la aplicación no la necesita, tomará las de la plataforma a la que pertenece. A continuación se hará referencia a la forma o los principios sobre los que se desarrollo la aplicación.

### 3.6.1 Patrón de diseño.

Los Patrones de Diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces. Existen un innumerable número de estos, todos adecuados a dar soluciones a problemáticas existentes. En el caso a referencia se realizó una extensiva búsqueda y estudio de los patrones existentes sobre estas tecnologías distribuidas. Luego de repasar los más utilizados y flexibles para este tipo de trabajo se decidió seguir los basamentos e ideas de los desarrolladores del patrón amo - esclavo (Master – Slave) o jerárquico, donde un componente amo distribuye el trabajo a los componentes auxiliares esclavos esperando entonces que retornen respuestas para formular un resultado final (13). Por lo que se utilizó como patrón de diseño.

### 3.6.2 Tratamiento de errores.

El tratamiento de errores en los leguajes más avanzados como por ejemplo java, c++, c#, entre otros, se basa en un nuevo mecanismo, excepciones, como la aplicación esta desarrollada en java se brinda una pequeña descripción de cómo funcionan estas excepciones (14).

- Permiten separar claramente el tratamiento de errores del código normal.
- Evitan que haya errores que pasen inadvertidos.
- Permiten agrupar en un lugar común el tratamiento de errores que ocurren en varios lugares del programa.

Las excepciones en Java son objetos de una clase extendida de otra clase especial llamada Throwable. Cuando ocurre un error:

- Se crea una instancia de la excepción, y se lanza (throw) la excepción.
- El bloque donde ocurre la excepción puede decidir tratarla (catch) o dejarla pasar.
- Si se deja pasar, el siguiente bloque puede a su vez tratarla o dejarla pasar.
- Si se trata, se ejecutan las instrucciones de un manejador.
- Si nadie la trata, el programa se interrumpe y aparece un error.

Así funciona el sistema de gestión de errores en java, y de esta forma quedaron garantizados los fallos del sistema, de este modo se trataron los posibles errores.

### 3.6.3 Seguridad.

La aplicación desarrollada en java y utilizando su tecnología RMI, consta de todas las políticas de seguridad establecidas y controladas por el lenguaje. Los datos que se envían a los clientes son controlados desde el servidor distribuidor de forma organizada. En caso de que un cliente se retrase o no cumpla con la unidad de trabajo que se le envíe, pasará la tarea como expirada, entonces esa misma tarea será reasignada a otro cliente que tenga las mismas características que el anterior o sea que este listo para cumplirla además de que cumpla con los requerimientos para hacerlo. De esta forma se tiene un buen por ciento de probabilidad de que se concluya con el procesamiento sin pérdidas de unidades de trabajo. Los datos viajan de forma compacta además de serealizados, lo que permite que lleguen a su destino sin problemas por los canales establecidos. Java brinda la clase SecurityManager que establece una serie de políticas de seguridad típicas del lenguaje y que se encarga de lanzar excepciones en caso de cualquier tipo de violaciones de las mismas.

### 3.6.4 Concepción de la ayuda.

Toda aplicación debe contar con un manual de usuario, una ayuda, alguna documentación legible que brinde información sobre su correcto funcionamiento. Para eso la plataforma GRAPH TOOL en la barra de menú principal tiene disponible la opción de Ayuda, donde se le explica detalladamente al especialista temas relacionados con el uso de la aplicación. Están adjuntas a la misma las ayudas correspondientes a caca módulo que conformara a la plataforma.

## 3.7 Conclusiones.

En el capítulo se trataron los principios de diseño que se siguieron para el desarrollo de la aplicación. Se brindó una descripción de los mismos describiendo las distintas técnicas de tratamiento de errores que se utilizaron así como las diferentes políticas de seguridad que existen para asegurar la integridad de los datos. Se describieron los diferentes modelos de análisis y diseño para cada caso de uso del sistema, explicando detalladamente su funcionamiento. De la misma manera los diferentes diagramas de clases modelan la forma en que fue implementado el sistema para su correcto funcionamiento.

# CAPÍTULO 4: IMPLEMENTACIÓN Y PRUEBA

## 4.1 Introducción.

En el presente capítulo se desarrolla el modelo de implementación, se analizan los diagramas de implementación y despliegue que lo conforman. Algunos de los diagramas de este flujo de trabajo detallan en componentes las diferentes clases y objetos utilizados en la implementación, para que se tenga una idea del funcionamiento de las mismas. Para mostrar las relaciones físicas entre los componentes hardware y software en el sistema final, la configuración de los elementos de procesamiento en tiempo de ejecución así como procesos y objetos que se ejecutan en ellos se muestra el diagrama de despliegue que conforma la aplicación.

## 4.2 Modelo de Implementación.

### 4.2.1 Diagrama de Despliegue.

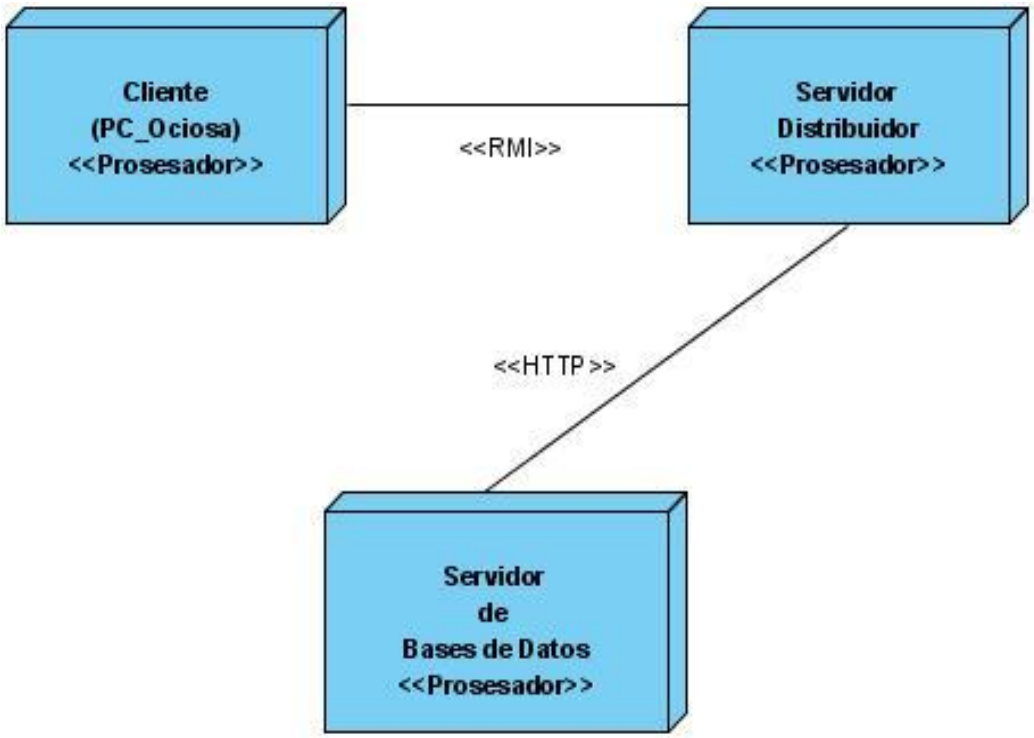


**Figura 23** Diagrama de Despliegue.

### 4.2.2 Diagramas de Componentes.

Un diagrama de componentes representa las dependencias entre componentes software, incluyendo componentes de código fuente, componentes del código binario, y componentes ejecutables. Un módulo de software se puede representar como componente. Algunos componentes existen en tiempo de compilación, algunos en tiempo de enlace y algunos en tiempo de ejecución, otros en varias de éstas (9). Los diagramas de componentes de este trabajo fueron realizados por casos de uso, con el objetivo de simplificar la comprensión de los mismos.

**Caso de uso Gestionar Ficheros de entrada.**

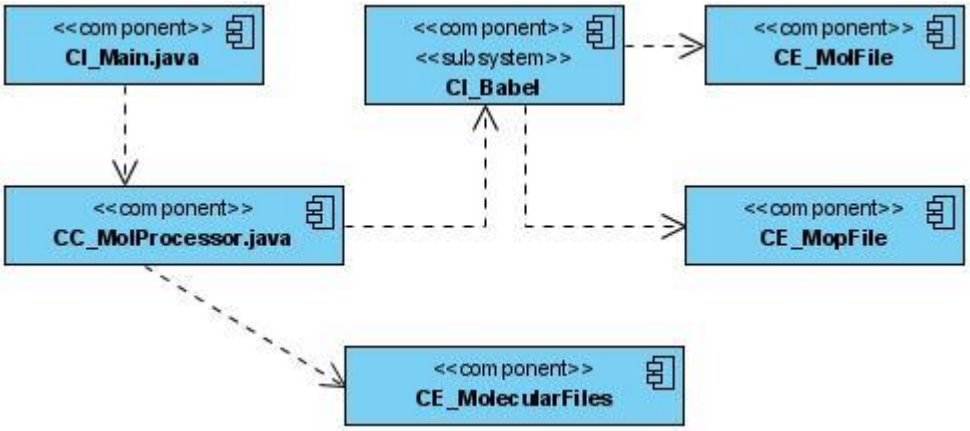


**Figura 24** Diagrama de componentes: CU Gestionar Ficheros de entrada.

**Caso de uso Gestionar Set a procesar.**

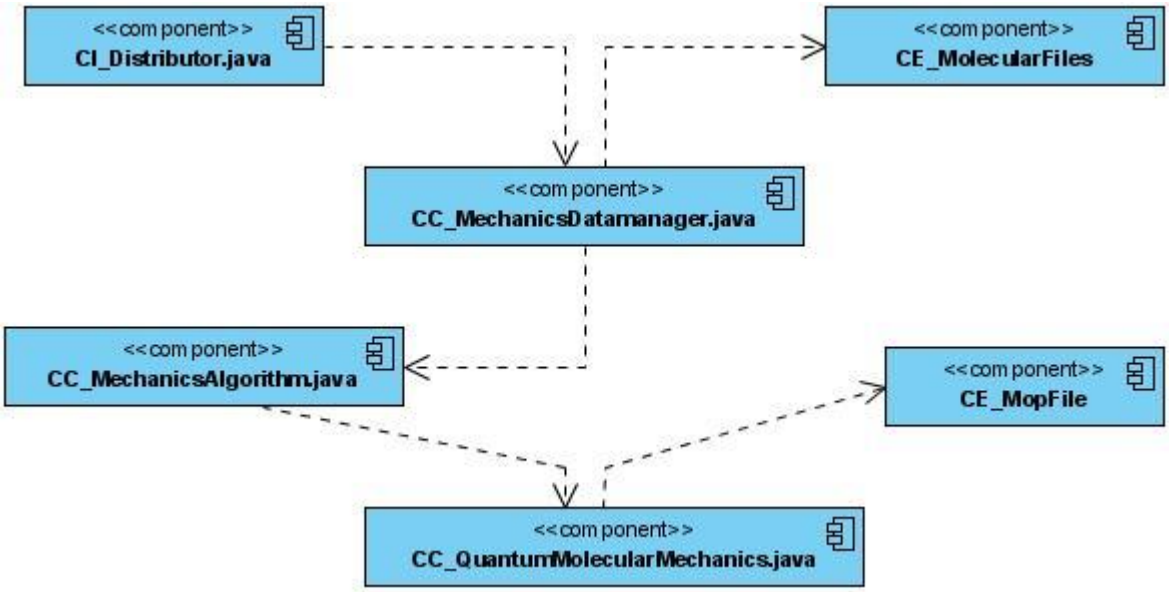


**Figura 25** Diagrama de componentes: CU  Gestionar Set a procesar.

**Caso de uso Realizar cálculos químico-teóricos escenario #1.**

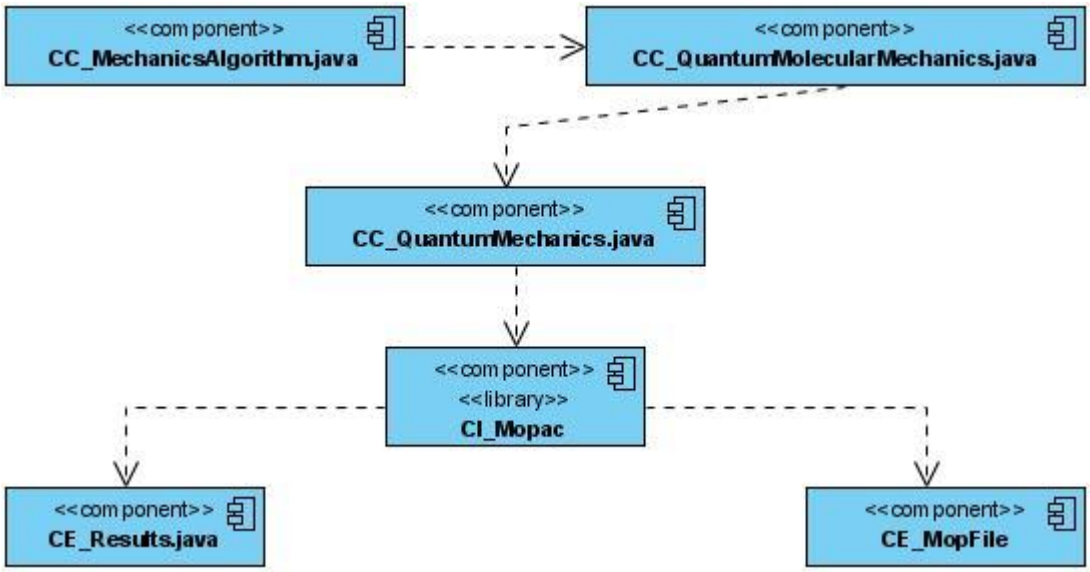


**Figura 26** Diagrama de componentes: CU Realizar cálculos químico-teóricos escenario #1.

**Caso de uso Realizar cálculos químico-teóricos escenario #2.**

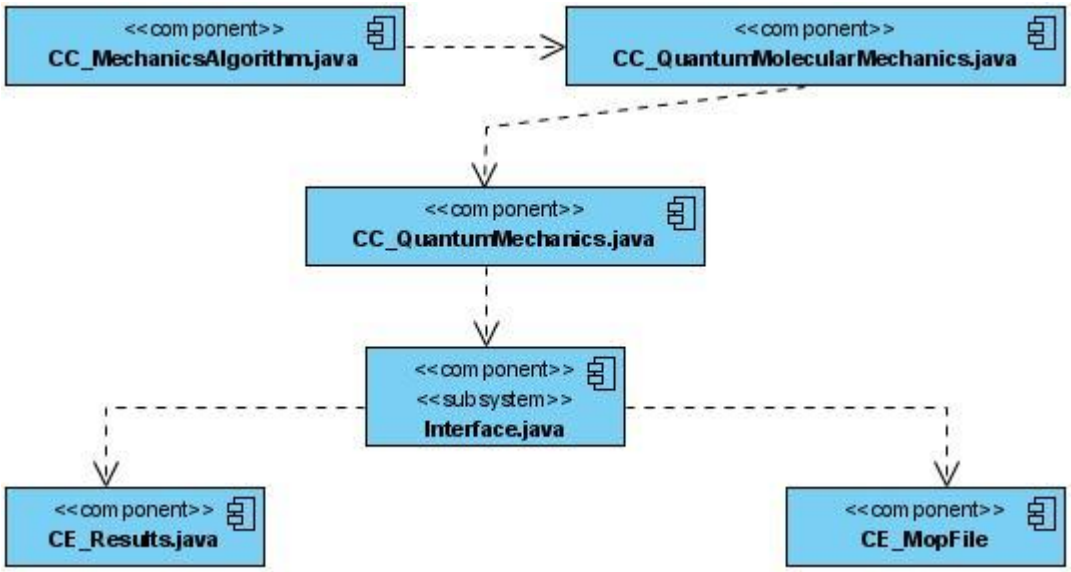


**Figura 27** Diagrama de componentes: CU Realizar cálculos químico-teóricos escenario #2.

**Caso de uso Gestionar Resultados.**

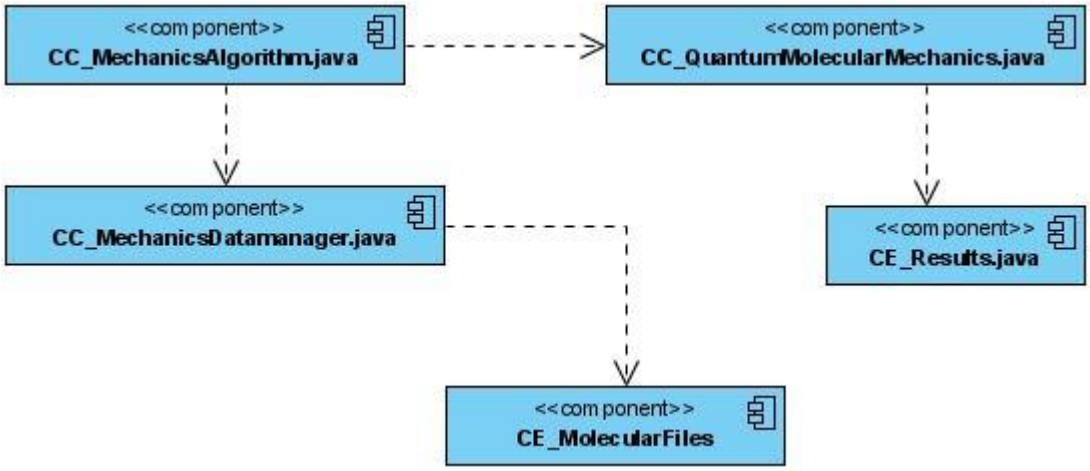


**Figura 28** Diagrama de componentes: CU Gestionar Resultados.

### 4.3 Prueba.

La prueba es el proceso de ejecutar un sistema bajo unas condiciones o requerimientos especificados con la intención de descubrir errores. Las pruebas mejoran la integridad de un sistema, al detectar las desviaciones del diseño y los errores en el sistema (10). Las pruebas tienen como objetivo detectar las áreas propensas a errores. Esto ayuda a la prevención de errores en un sistema. Incrementan el valor del producto, al adaptarlo a las necesidades del usuario.

#### 4.3.1 Prueba de caja negra.

Las pruebas de caja negra son aquellas que se enfocan directamente en el exterior del módulo, sin importar la conformación del código, son pruebas funcionales en las que se trata de encontrar fallas en las que este no se atiene a su especificación. En el caso del módulo presente, se llevan a cabo este tipo de pruebas para comprobar las respuestas del sistema a las diferentes entradas. Para comprobar que el usuario especialista, recibe los resultados esperados. Se realiza una muestra por caso de usos del funcionamiento del sistema, para tener medida de la calidad y eficiencia con que cuenta.

| **Nombre del caso de uso** | **Gestionar Ficheros de entrada.** | |
|---|---|---|
| **Entrada** | **Resultados** | **Condiciones** |
| Se le proporciona a la aplicación un conjunto de moléculas, o ficheros moleculares, que son el contenido de entrada de los cálculos químico-teóricos. | Los ficheros quedan procesados, llevados al formato de entrada del software que posteriormente realiza los cálculos. | Que exista entrada de moléculas. |

**Tabla 9** Caso de prueba: CU Gestionar Ficheros de entrada.

| Nombre del caso de uso | Gestionar Set a procesar. | |
|---|---|---|
| **Entrada** | **Resultados** | **Condiciones** |
| Se introduce la configuración necesaria para que descargue desde la base de datos las moléculas a procesar. | Descarga el set indicado, y lo deja en ficheros listos para la entrada del software químico que realizará los cálculos. | Debe existir conexión hasta el servidor de bases de datos. |

**Tabla 10** Caso de prueba: CU Gestionar Set a procesar.

| Nombre del caso de uso | Realizar cálculos químico-teóricos. | |
|---|---|---|
| **Entrada** | **Resultados** | **Condiciones** |
| Se toman los ficheros y se procesan, se realizan los cálculos necesarios indicados. | Quedan en ficheros resultados los cálculos químicos realizados. | Que existan ficheros y el software químico que realiza los cálculos. |

**Tabla 11** Caso de prueba: CU Realizar cálculos químico-teóricos.

| Nombre del caso de uso | Gestionar Resultados. | |
|---|---|---|
| **Entrada** | **Resultados** | **Condiciones** |
| Recibe como entrada los ficheros que conforman el resultado de los cálculos. | Se almacenan los resultados, se llevan al servidor de bases de datos, cuando llegue a una cantidad indicada. | Que existan resultados de los cálculos. |

**Tabla 12** Caso de prueba: CU Realizar cálculos químico-teóricos.

### 4.3.2 Prueba para demostrar la eficiencia de los cálculos de forma distribuida.

Para demostrar que los cálculos químico-teóricos se desarrollaron con eficiencia, se analizó el tiempo de ejecución del sistema, y se realizaron comparaciones con el tiempo de demora del proceso tal como se realiza actualmente en una computadora personal. Se tomaron muestras de 3000, 5000 y 20000 moléculas para procesar y se distribuyeron en 125 máquinas que simularían clientes-trabajadores. Los cálculos concluyeron en tiempos de 1.54, 4.10, y 6.64 horas respectivamente. Se sumaron los tiempos de ejecución de cada corrida particular leyendo esa cifra en los ficheros de salida correspondientes. Estos resultados muestran la demora del proceso clásico pues la suma de los valores reportados por el Mopac disminuyó en un 98% (Tabla 13).

| Moléculas procesadas | ∑ Tiempos en Clientes | Tiempo JDS |
|---|---|---|
| 3000 | 110.68 horas | 1.54 horas |
| 5000 | 236.25 horas | 4.10 horas |
| 20000 | 464.31 horas | 6.64 horas |

**Tabla 13** Pruebas de tiempos de ejecución para 125 clientes conectados.

Por tanto, se puede afirmar que la distribución de los cálculos químico-teóricos realizada por el presente módulo cumple su objetivo con mayor eficiencia que el proceso ejecutado en una sola computadora personal.

## 4.4 Conclusiones.

En el presentado capítulo se mostró y comprobó el funcionamiento específico del módulo desarrollado. Los diagramas de componentes describieron la organización y dependencia entre nodos físicos en los que funciona la aplicación. Se realizaron pruebas generales al sistema para comprobar su correcto funcionamiento, en las que se determinaron ciertas condiciones para que se obtuvieran los resultados esperados.

# CONCLUSIONES GENERALES

- Se analizó, diseñó e implementó una aplicación capaz de gestionar cálculos químico-cuánticos a grandes bancos de moléculas a través de una red, tanto sobre el sistema operativo Linux como Windows.
- La aplicación implementada permite el acceso a una base de datos de estructuras químicas, la distribución de los cálculos a realizar y el almacenamiento de los resultados en la misma.
- Se demostró la eficiencia del sistema ya que reduce en más del 98% el tiempo de procesamiento y cálculo masivo de mecánica cuántica, en pruebas realizadas a tres lotes de 3000, 5000 y 20000 moléculas que disminuyeron sus tiempos de cálculo de 110, 236 y 464 horas a 1.5, 4.1 y 6.6 horas respectivamente.

# RECOMENDACIONES

Para un mejor aprovechamiento de las funcionalidades de la aplicación:

- Tener en cuenta que el trabajo fue realizado específicamente para funcionar en la UCI, que en caso de trasladarlo habría que implementar un sistema Grid similar al implementado en el centro.
- Implementar métodos de conversión molecular para permitir guardar las moléculas en diversos formatos moleculares sin tener que depender de otros sistemas existentes y con restricciones propias.
- Que se implemente una mejor forma de integrar la plataforma grato, al sistema BIOGRID, para así poder distribuir los cálculos, y dejarlos guardados donde desee el cliente de la plataforma GRAPH TOOL.

# REFERENCIAS BIBLIOGRÁFICAS


1. **Andalia, Lic. Rubén Cañedo.** *Bioinformática: en busca de los secretos moleculares de la vida.* 2004.

2. **Sagi, Diego J. Bodas.** Programación Distribuida. *Programación Distribuida.* [En línea] 2005. [Citado el: 24 de 12 de 2006.] http://www.sun.com. ISSN.

3. **Martín, Ignacio.** *Propuesta para la Creación de un Programa de e-Ciencia.* 2003.

4. **JARA, OMAR HURTADO.** *NUEVOS PARADIGMAS DE LOS SISTEMAS DE INFORMACIÓN.* [Documento] UNIVERSIDAD CARLOS III DE MADRID : UNIVERSIDAD CARLOS III DE MADRID, 2006.

5. **UNNE., Departamento de Química. Facultad de Agroindustrias.** UNNE. *UNNE.* [En línea] UNNE, 2007. [Citado el: 14 de 03 de 07.] http://www.ua.es/cuantica/docencia/ccem/teoria/node1.html.

6. *COMUNICACIONES CIENTÍFICAS.* **FERNANDEZ, Ing. José Sergio.** Chaco,Argentina : s.n., 1999.

7. **Diez, Maria Helena.** Modelos computacionales y modelos químicos. [En línea] [Citado el: 05 de 03 de 2007.]

8. **Jiménez, Eduardo González.** La simulación computacional de procesos genéticos a nivel molecular. *La simulación computacional de procesos genéticos a nivel molecular.* [En línea] 2002. [Citado el: 16 de 2 de 2007.] http://www.elementos.buap.mx/num47/htm/31.htm.

9. **Addison Wesley Ed. James Rumbaugh, Ivar Jacobson y Grady Booch.** *El Lenguaje Unificado de Modelado. Manual de Referencia.* 2000.

10. **Valencia, Universidad Politécnica de.** *Process, Rational Unified.* Valencia : Departamento de Sistemas Informáticos y Computación., 2005.

11. **Lago, Ramiro.** Introducción a Remote Method Invocation. *Introducción a Remote Method Invocation.* [En línea] Febrero de 2006. [Citado el: 16 de 3 de 2007.] http://www.proactiva-calidad.com/java/rmi/introduccion.html.

12. **froufe@arrakis.es.** Características de Java. *Características de Java.* [En línea] 2007. [Citado el: 12 de 04 de 2007.] http://www.ciao.es/Java__Opinion_420399.

13. **Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, and Michael Stal.** Patrones de diseño. *Patrones de diseño.* [En línea] 2003. [Citado el: 15 de 2 de 2007.] http://www.umlpatterns.com/pages/PatronsDisseny/Pattern%20Master%20Slave/index.html.

14. **Harbour, Michael González.** *Java excepciones.* [PDF] UNIVERSIDAD DE CANTABRIA : s.n., 2007.

15. **wikipedia.** s.l. : wikipedia.

# BIBLIOGRAFÍA


1. **Andalia, Lic. Rubén Cañedo.** *Bioinformática: en busca de los secretos moleculares de la vida.* 2004.

2. **Sagi, Diego J. Bodas.** Programación Distribuida. *Programación Distribuida.* [En línea] 2005. [Citado el: 24 de 12 de 2006.] http://www.sun.com. ISSN.

3. **Martín, Ignacio.** *Propuesta para la Creación de un Programa de e-Ciencia.* 2003.

4. **JARA, OMAR HURTADO.** *NUEVOS PARADIGMAS DE LOS SISTEMAS DE INFORMACIÓN.* [Documento] UNIVERSIDAD CARLOS III DE MADRID : UNIVERSIDAD CARLOS III DE MADRID, 2006.

5. **UNNE., Departamento de Química. Facultad de Agroindustrias.** UNNE. *UNNE.* [En línea] UNNE, 2007. [Citado el: 14 de 03 de 07.] http://www.ua.es/cuantica/docencia/ccem/teoria/node1.html.

6. *COMUNICACIONES CIENTÍFICAS.* **FERNANDEZ, Ing. José Sergio.** Chaco,Argentina : s.n., 1999.

7. **Diez, Maria Helena.** Modelos computacionales y modelos químicos. [En línea] [Citado el: 05 de 03 de 2007.]

8. **Jiménez, Eduardo González.** La simulación computacional de procesos genéticos a nivel molecular. *La simulación computacional de procesos genéticos a nivel molecular.* [En línea] 2002. [Citado el: 16 de 2 de 2007.] http://www.elementos.buap.mx/num47/htm/31.htm.

9. **Addison Wesley Ed. James Rumbaugh, Ivar Jacobson y Grady Booch.** *El Lenguaje Unificado de Modelado. Manual de Referencia.* 2000.

10. **Valencia, Universidad Politécnica de.** *Process, Rational Unified.* Valencia : Departamento de Sistemas Informáticos y Computación., 2005.

11. **Lago, Ramiro.** Introducción a Remote Method Invocation. *Introducción a Remote Method Invocation.* [En línea] Febrero de 2006. [Citado el: 16 de 3 de 2007.] http://www.proactiva-calidad.com/java/rmi/introduccion.html.

12. **froufe@arrakis.es.** Características de Java. *Características de Java.* [En línea] 2007. [Citado el: 12 de 04 de 2007.] http://www.ciao.es/Java__Opinion_420399.

13. **Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, and Michael Stal.** Patrones de diseño. *Patrones de diseño.* [En línea] 2003. [Citado el: 15 de 2 de 2007.] http://www.umlpatterns.com/pages/PatronsDisseny/Pattern%20Master%20Slave/index.html.

14. **Harbour, Michael González.** *Java excepciones.* [PDF] UNIVERSIDAD DE CANTABRIA : s.n., 2007.

15. **wikipedia.** s.l. : wikipedia.

16. **Valencia, Alfonso.** Instituto Roche. *Instituto Roche.* [En línea] 12 de 05 de 2003. [Citado el: 17 de

Enero de 2007.] http://www.institutoroche.es/bioinformatica.php?op=biotecnologia&id=4. ISBN.

17. Centro de Química Farmacéutica. *Centro de Química Farmacéutica.* [En línea] Centro de Química Farmacéutica. [Citado el: 25 de 10 de 2006.] http://www.cqf.sld.cu/.

18. **Cárdenas, Constanza.** CALTECH. *CALTECH.* [En línea] 15 de 01 de 2005. http://atlas.puj.edu.co/PASI/viewabstract.php?id=31.

ANEXOS

**Anexo 1: Descripción de las clases.**

| **Nombre: CI_Main** | |
|---|---|
| **Tipo de clase:** Interfaz | |
| **Atributo** | **Tipo** |
| | |
| | |
| **Responsabilidades** | |
| **1.**Nombre: | EditMol(int pos) |
| **2.** | LoadMol() |
| **3.** | ShowMol(int posmol) |
| **4.** | NewMol() |
| **5.** | CalcMM() |
| **6.** | CalcPC() |
| **7.** | DeleteMol() |
| **8.** | UpdateUI() |
| **9.** | CalcQT() |
| Descripción: | Todas las operaciones mencionadas son brindadas por la clase interfaz principal creada por la plataforma que será soporte del sistema. Solo la operación **CalcQT()** esta relacionada con el módulo presentado. Será la encargada de dar ejecución a los cálculos químico-teóricos. |
| | |

**Tabla 14** Descripción de la Clase: CI_Main.

| **Nombre: CI_Distributor** |
|---|
| **Tipo de clase:** Interfaz |
| **Responsabilidades** |
| Interfaz mediante la cual se comunicarán las clases CC_MechanicsAlgorithm y CC_MechanicsDatamanager para todo el intercambio de información necesaria para la ejecución del problema. |

**Tabla 15** Descripción de la Clase: CI_Distributor.

| **Nombre: CI_Babel** |
|---|
| **Tipo de clase:** Interfaz |
| **Responsabilidades** |
| Interfaz brindada por le aplicación Babel 1.6, encargado en este caso de procesar los ficheros mol, convirtiéndolos en ficheros listos para los cálculos a realizar. |

**Tabla 16** Descripción de la Clase: CI_ Babel.

| **Nombre: CI_Interface** |
|---|
| **Tipo de clase:** Interfaz |
| **Responsabilidades** |
| Interfaz de programador, brindada por la aplicación o por el software químico escogido para que realice los cálculos indicados, en sus versiones para sistemas Linux y Windows en este caso. Encargada de permitir la entrada de datos al software escogido. |

**Tabla 17** Descripción de la Clase: CI_Interface.

| **Nombre: CC_MolProcessor** | |
|---|---|
| **Tipo de clase:** Controladora | |
| **Atributo** | **Tipo** |
| User | String |
| Pass | String |
| Host | String |
| DB | String |
| Connection | Connection |
| **Responsabilidades** | |
| **1.** Nombre | Connect() |
| Descripción: | Encargado de realizar la conexión con el servidor de bases de datos. |
| **2.** Nombre | ExtracMolecules( String destPath ) |
| Descripción: | Encargada de descargar a la fuente, las moléculas que serán procesadas. |
| **3.** Nombre | RunBabel( String Path, String destPath ) |
| Descripción: | Mandará a procesar los ficheros de la dirección especificada. |
| **4.** Nombre | AddFunctionStrig( String Path, String function ) |
| Descripción: | Agrega la función correspondiente al cálculo químico deseado. |
| **5.** Nombre | UploadFiles( String Path ) |
| Descripción: | Salva los nuevos ficheros en el servidor de bases de datos. |
| **6.** Nombre | CloseConnet() |
| Descripción: | Cierra la conexión actual al servidor de bases de datos. |
| | |

**Tabla 18** Descripción de la Clase: CC_MolProcessor.

| **Nombre: CC_QuantumMolecularMechanics** | |
|---|---|
| **Tipo de clase:** Controladora | |
| **Atributo** | **Tipo** |
| TMPDIRECTORY | File |
| | |
| **Responsabilidades** | |
| **1.** Nombre | WriteMolecularVectorInfo ( Vector Molecules ) |
| Descripción: | Escribe en ficheros los objetos (moléculas) que contiene el vector enviado desde la clase CC_MechanicsDatamanager. |
| **2.** Nombre | InitCalculation() |
| Descripción: | Inicia el proceso de cálculo. |

| | |
|---|---|
| **3.** Nombre | CollectResult() |
| Descripción: | Recolectar los resultados obtenidos en el cálculo, los guardas en un vector que sera enviado al servidor y desde el mismo a su destino o fuente de datos. |
| | |

**Tabla 19** Descripción de la Clase: CC_QuantumMolecularMechanics.

| **Nombre: CC_MechanicsDatamanager** | |
|---|---|
| **Tipo de clase:** Controladora | |
| **Atributo** | **Tipo** |
| AcceptJob | Bolean |
| Buffer | Vector<Molecule> |
| Calculation | String |
| CurrentResultCount | Double |
| Database | String |
| Granularity | Int |
| Limit | Int |
| Offset | Int |
| Password | String |
| ResultsCount | Int |
| Server | String |
| TMPDIRECTORY | File |
| Table | String |
| User | String |
| Connection | Connection |
| | |
| **Responsabilidades** | |
| **1.** Nombre | generateWorkUnit( Clientinfo clientProperties) |
| Descripción: | Encargado de repartir la unidad de trabajo a un cliente, mediante un vector con el contenido o el tipo que objeto que procesará. |
| **2.** Nombre | ajustGranularity( double time ) |
| Descripción: | Encargado de ajustar la cantidad de trabajo por cliente. Dependiendo del tiempo de respuesta o de la eficiencia con que lo haga. |
| **3.** Nombre | closeResources() |
| Descripción: | Encargado de terminar el trabajo con un cliente determinado. |
| **4.** Nombre | processResults( long uId , Vector result ) |
| Descripción: | Encargado de procesar los resultados, terminar de realizar las operaciones indicadas con los mismos. |
| **5.** Nombre | ClearTMPDIRECTORY() |
| Descripción: | Tarea de borrar o eliminar el directorio utilizado temporalmente para realizar los cálculos, con el los ficheros que contenga. |
| **6.** Nombre | GetElementValue(doc document, String ID) |
| Descripción: | Trabajará en devolver el elemento que se solicite, seleccionado por su identificador, devolverá entonces el String que conforma el mismo. |

| | |
|---|---|
| **7.** Nombre | UploadResults() |
| Descripción: | Retorna verdadero o falso si es que se ejecutó la acción de guardar todos los resultados del proceso de cálculo. |
| **8.** Nombre | connect() |
| Descripción: | Conectará el servidor distribuidor con le fuente de datos. |
| **9.** Nombre | disConnect() |
| Descripción: | Terminara la conexión con el servidor de bases de datos que se este conectado. |
| **10.** Nombre | fillBuffer() |
| Descripción: | Encargado de dar valores al buffer que contiene los datos a ser procesados. |
| **11.** Nombre | getStatus() |
| Descripción: | Retornara el estado real en que se encuentre la solución del problema. |
| **12.** Nombre | readXMLConfig() |
| Descripción: | Encargado de leer desde el XML de configuración los datos necesarios para estableces una conexión con el servidor de bases de datos. |
| | |

**Tabla 20** Descripción de la Clase: CC_MechanicsDatamanager.

| **Nombre: CC_MechanicsAlgorithm** | |
|---|---|
| **Tipo de clase:** Controladora | |
| **Atributo** | **Tipo** |
| Calculation | class |
| TMPDIRECTORY | File |
| | |
| **Responsabilidades** | |
| **1.** Nombre | processUnit ( Vector workUnit ) |
| Descripción: | Responsable del proceso de los datos que se reciben en los clientes (PC_Ociosa). |
| **2.** Nombre | DownloadQuantumMechanicProgram() |
| Descripción: | Descargara la aplicación necesaria para ejecutar los cálculos al cliente. |
| **3.** Nombre | ClearTMPDIRECTORY() |
| Descripción: | Borrara el directorio en que se realizaron los cálculos, con el sus ficheros. |
| | |

**Tabla 21** Descripción de la Clase: CC_MechanicsAlgorithm.

| **Nombre: CC_MolecularMechanics** | |
|---|---|
| **Tipo de clase:** Controladora | |
| **Atributo** | **Tipo** |
| | |
| | |
| **Responsabilidades** | |
| **1.** Nombre | CollectResults() |
| Descripción: | Encargada de recolectar los resultados de los cálculos. |
| **2.** Nombre | InitCalculation( Vector Molecules) |
| Descripción: | Iniciará el proceso de cálculo de mecánica molecular. |

| | |
|---|---|
| | |

**Tabla 22** Descripción de la Clase: CC_MolecularMechanics.

| **Nombre: CC_QuantumMechanics** | |
|---|---|
| **Tipo de clase:** Controladora | |
| **Atributo** | **Tipo** |
| | |
| | |
| **Responsabilidades** | |
| **1.** Nombre | CollectResults( ) |
| Descripción: | Encargada de recolectar los resultados de los cálculos. |
| **2.** Nombre | InitCalculation( Vector Molecules ) |
| Descripción: | Iniciará el proceso de cálculo de mecánica molecular. |
| **3.** Nombre | GetCommad( ) |
| Descripción: | Tomará el comando a ejecutar para mandar a correr la aplicación que realizara los cálculos. |
| **4.** Nombre | WriteMolecularVectorInfo( Vector Molecules ) |
| Descripción: | Encargado de descargar los datos que se encuentran en el vector. |
| | |

**Tabla 23** Descripción de la Clase: CC_QuantumMechanics.

| **Nombre: CE_MolFile** | |
|---|---|
| **Tipo de clase:** Entidad | |
| **Atributo** | **Tipo** |
| Data | String |
| | |
| **Responsabilidades** | |
| **1.** Nombre | getData() |
| Descripción: | Encargado de devolver el valor de los datos que conforman al MolFile. |
| **2.** Nombre | setData( String data ) |
| Descripción: | Encargado de dar valores a los MolFile. |
| | |

**Tabla 24** Descripción de la Clase: CE_MolFile.

| **Nombre: CE_MopFile** | |
|---|---|
| **Tipo de clase:** Entidad | |
| **Atributo** | **Tipo** |
| Data | String |
| function | String |
| | |

| **Responsabilidades** | |
|---|---|
| **1.** Nombre | getData() |
| Descripción: | Encargado de devolver el valor de los datos que conforman al MopFiles. |
| **2.** Nombre | setData( String data ) |
| Descripción: | Encargado de dar valores a los MopFiles. |
| **3.** Nombre | AddFunction( String function ) |
| Descripción: | Agregar a cada mop, la función que se requiera calcular. |
| | |

**Tabla 25** Descripción de la Clase: CE_MopFile.

| **Nombre: CE_Results** | |
|---|---|
| **Tipo de clase:** Entidad | |
| **Atributo** | **Tipo** |
| resultData | String |
| | |
| **Responsabilidades** | |
| **1.** Nombre | getresultData() |
| Descripción: | Encargado de devolver el valor de los datos que conforman los resultados. |
| **2.** Nombre | setresultData( String resultData ) |
| Descripción: | Encargado de dar valores a los resultados. |
| | |

**Tabla 26** Descripción de la Clase: CE_Results.

| **Nombre: CE_MolecularFiles** | |
|---|---|
| **Tipo de clase:** Entidad | |
| **Atributo** | **Tipo** |
| Incompuesto | Int |
| Mol | String |
| Mop | String |
| Out | String |
| Rsm | String |
| **Responsabilidades** | |
| **1.** Nombre | Operations() |
| Descripción: | Responsabilidad de gestionar las diferentes consultas que se requieran a la base de datos. |
| | |

**Tabla 27** Descripción de la Clase: CE_MolecularFiles.

# GLOSARIO

**ADN:** Abreviatura del ácido desoxirribonucleico. Constituye el principal componente del material genético de la inmensa mayoría de los organismos, junto con el ARN (15).

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

**Cálculos químico-teóricos:** Refiere a los cálculos a realizar por la aplicación, tal es el caso de mecánica cuántica y mecánica molecular.

**Cálculos químico-cuánticos:** Refiere a los cálculos de mecánica cuántica que serán realizados.

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

**Framework:** En el desarrollo de software, un framework es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, un framework puede incluir soporte de programas, bibliotecas y un lenguaje de scripting entre otros softwares para ayudar a desarrollar y unir los diferentes componentes de un proyecto (15).

**Grid Computing:** Es una tecnología innovadora que permite utilizar de forma coordinada todo tipo de recursos (entre ellos cómputo, almacenamiento y aplicaciones específicas) que no están sujetos a un control centralizado.

**JSD:** Sistema distribuido de java.

**PC_Ociosa:** Cliente del servidor distribuidor, máquina encargada de realizar los cálculos químico-teóricos. El proceso de recibir tareas cobrara fuerza en su tiempo ocio, tiempo en que la cpu este en desuso.

**RMI:** Invocación a métodos remotos, tecnología java para el trabajo distribuido.

**Runtime:** Cuando un programa esta en ejecución o corriendo.

**Set:** Conjunto de datos almacenados en forma de lista o vector de longitud variable.

**Sistemas Distribuidos:** Sistemas en que existen varias CPU conectadas entre sí, las cuales trabajan de manera conjunta.