UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD # 6

TÍTULO: "Plug-in para la integración del módulo Cálculo de Descriptores a la plataforma alasGRATO"

TRABAJO DE DIPLOMA EN OPCIÓN AL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS

AUTORES:

Ludmila Campos García

Dairon Domínguez Vega

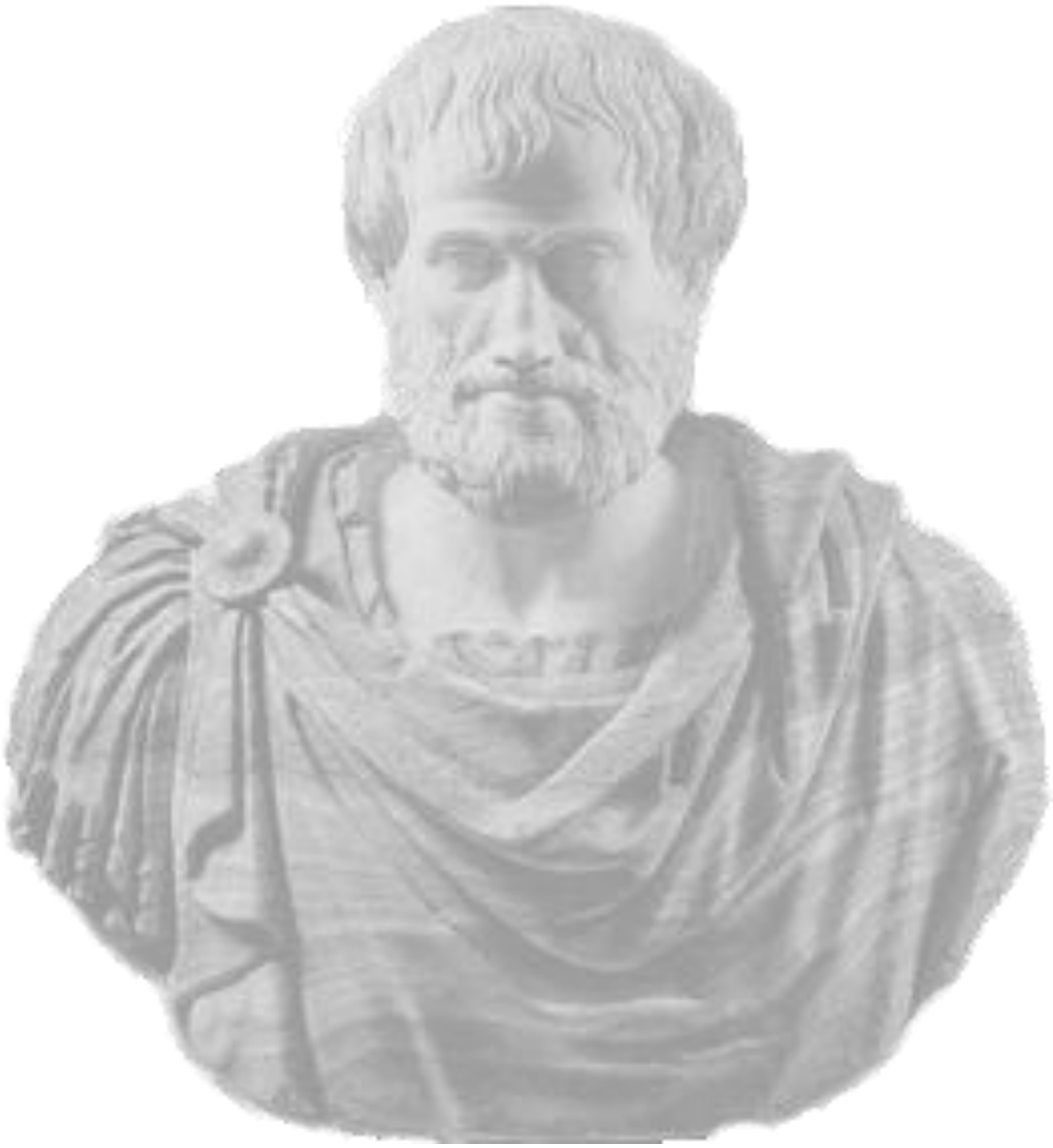
TUTORES:

Dr. Ramón Carrasco Velar

MSc. Alexis René Rodríguez León

CIUDAD DE LA HABANA, CUBA

JUNIO, 2010



Piensa como piensan los sabios, mas habla como lo hace la gente sencilla.

Aristóteles

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 junio del año 2010

Ludmila Campos García

_____

Firma del Autor

Dairon Domínguez Vega

_____

Firma del Autor

Dr. Ramón Carrasco Velar

_____

Firma del Tutor

MSc. Alexis René Rodríguez León

_____

Firma del Tutor

DATOS DE CONTACTO

TUTORES:

Dr. Ramón Carrasco Velar Doctor en Ciencias Químicas

Profesor Asistente

E-mail: rcarrasco@uci.cu

MSc. Alexis René Rodríguez León Máster en Bioinformática

Profesor Instructor

E-mail: arodriguezl@uci.cu

DEDICATORIA

A nuestros padres…

AGRADECIMIENTOS

Dairon:

A mis padres por ser ejemplos y servirme de guía en todo momento. Por confiar en mí, la tarea de hacerme un ingeniero, por todo el apoyo que siempre me brindaron y por todos los consejos que tanto me ayudaron.
A mi novia Arianna por todo el cariño, respeto y amor que siempre me ha brindado y por su infinita paciencia ante la espera de verme como ingeniero.
A mis abuelos que siempre me han dado todo su cariño y todo su apoyo también.
A mi hermano por su confianza, su ayuda, por siempre estar presente cuando lo necesito y por todo el cariño que siempre me ha dado.
A mi tíos Manuel y Alicita, a mis primos Rayko y Oriel por su preocupación y apoyo.
A mi familia de La Habana que siempre me brindaron su casa y su cariño.
A toda mi familia por siempre estar presente y confiar en mí.
A todos mis amigos de aquí y de Matanzas.

A Luis, Roberto, Félix, Andrés, que tanto en las buenas como en las malas se comportaron como mis amigos. Por la ayuda incondicional que siempre me brindaron.
A mis tutores Dr. Ramón Carrasco por las explicaciones de las cuestiones químicas que sin su ayuda no hubiese podido entender. A Msc. Alexis René por su apoyo brindado en todo momento, por sus consejos, y la atención que siempre estuvo dispuesto a dar.
A Aurelio por las dudas aclaradas y el apoyo que siempre nos dio.
A todos aquellos que de una forma u otra aportaron su granito de arena, a Edel, Yadira, Adrian Quintero, Javier, a la profesora Nara, a los profes de ingeniería, al profesor César, al profesor Tonyse.
A todos los profesores que hicieron posible mi formación profesional.

A todos, de corazón, ¡¡Muchas gracias!!

A Dairon, porque sin él, imposible. Porque fue de los primeros que conocí cuando llegué a la UCI y será de los últimos que despida.
A nuestro tutor Alexis, porque a pesar de que lo teníamos “obstinado”, como nos decía sonriendo, siempre nos abrió las puertas con la mayor disposición e inteligencia para hacernos muchos comentarios y sugerencias, sin los cuales no hubiese sido posible conformar este trabajo.
Al Dr. Ramón Carrasco, por sus sabios consejos y a los compañeros del tribunal, porque sus señalamientos ayudaron a mejorar la calidad del trabajo realizado.
A todos los profesores que he tenido durante estos cinco años y a lo largo de mi vida, porque cada granito de su sabiduría ha enriquecido mi formación profesional.
A todos mis compañeros, los que están hoy aquí, los que no pudieron venir. Especialmente a los muchachos del 6106, hasta el 6306. A Themura, Sury, Yili, Lisy, Mary, Karen, Mi Grandote, Rober, Omar, Geo y no me alcanzaría la página para nombrarlos a todos, pero en mi corazón estarán para siempre, porque fueron 3 años llenos de experiencias y alegrías.
A mis hermanitas Yayi y Kaly, porque puedo llenar un álbum con todas las fotos que tenemos juntas, fotos que evidencian los momentos compartidos. Momentos de alegrías, de fiestas, pero sobre todo, momentos de dolor, porque han llorado conmigo muchas veces, porque son especiales para mí.
A mis amigos, My Love, Mi Esposo, Luisi, Frómeta, porque son los varones con los que más he conversado en mi vida, porque siempre estuvieron ahí cuando los necesité.
A todas las Lunas, Yadi, Ana, Tati y por supuesto Kaly y Yayi también, porque son las niñas más loquitas de la UCI, pero además, las más comprensivas y sobre todo buenas amigas.
A todas las niñas con las que he tenido el privilegio de vivir, porque sin una buena convivencia, no logramos nada; y a toda la gente linda que la UCI me ha presentado, a Ene, a Yuly, porque también son amigas. A esos profesores que no me han dado clases, esas niñas que no han vivido conmigo, esos compañeros que no han estado en mi grupo, pero que también son parte de mi vida.

Por último, pero no por ser así es menos importante, al contrario, quiero agradecer especialmente a mi familia.

A mi familia por parte de padre. A mi abuela Alla, mis tíos, primos, especialmente a mi prima Yeyi. A mi papá, mis hermanitos, porque los quiero mucho y de una forma u otra siempre se preocuparon por mí.
A mi familia por parte de madre, mi tesoro.
A mi mamá, a quien quiero con la vida, porque fue quien me la dio y porque dejó todo y vino para La Habana para estar a mi lado, apoyándome como ha hecho siempre.
A mis abuelos, mami y papi, porque son mi vida. Porque a pesar de que estaban lejos, siempre los sentí cerquita de mí, cuidándome, dándome fuerzas.
A mis tías, Barby, Rebe, Goti y mis primas, Ailén, Diana y Claudia, las adoro, porque son la mejor comisión de embullo y la mayor alegría de mi vida.
A mis tíos Abril, July, Lamas, porque hicieron posible muchos de los viajes que hice para estar en casa y siempre pude contar con su apoyo.
A ellos, quiero agradecer y dedicar este trabajo, porque son mi mayor orgullo, mi mayor inspiración, mi mayor felicidad, pero sobre todo, porque son mi Gran Amor.

GRACIAS A TODOS…

Ludmi

RESUMEN

La Faculta 6 de la Universidad de las Ciencias Informáticas (UCI) desarrolla un proyecto denominado: “Plataforma para la Predicción de Actividad Biológica en Compuestos Orgánicos” (alasGRATO). La plataforma consta de varios módulos entre los cuales se encuentra el módulo Cálculo de Descriptores. Estos módulos deberán ser integrados a la misma en forma de plug-in de manera que posibilite la interacción entre todas las partes del sistema. En el presente trabajo, se implementó el plug-in para la integración del módulo Cálculo de Descriptores a la plataforma alasGRATO. Se implementaron, además, dos servicios web, uno para obtener los descriptores disponibles y el otro para realizar el cálculo de descriptores. También, se realizó el diseño de una base de datos local para almacenar los resultados de los descriptores.

Palabras clave: alasGRATO, Descriptores, Plug-in, T-arenal

ÍNDICE

INTRODUCCIÓN

En las condiciones actuales del desarrollo científico técnico es indispensable la búsqueda de novedosos tratamientos o fármacos para las nuevas y diversas enfermedades a las que el mundo se enfrenta. El desarrollo, diseño y obtención de los mismos, se logra y se consolida gracias al conocimiento conjunto en las áreas de la Química, la Biología, la Medicina, la Ingeniería, la Biotecnología y la Bioinformática, entre otras, que están integradas por el trabajo de diferentes grupos multidisciplinarios, y llevan el principio activo desde su origen hasta la presentación farmacéutica que se conoce como medicamento.

El ciclo de desarrollo de un fármaco, tanto en la fase de descubrimiento, como en la fase de desarrollo es un proceso muy complejo y lento, que puede tomar hasta 12 años e incurrir en gastos significativos por cada nuevo medicamento que sale al mercado. Cada nuevo fármaco es fruto de un intensivo proceso de búsqueda durante el cual se examinan todas las posibilidades para desarrollar el fármaco más eficaz que permita mejorar la calidad de vida de quienes sufran alguna enfermedad.

La Facultad 6 de la Universidad de las Ciencias Informáticas (UCI) está desarrollando un proyecto de investigación para el tamizaje virtual de compuestos orgánicos, basado en técnicas de teoría de grafo e inteligencia artificial, denominado: “Plataforma para la Predicción de Actividad Biológica en Compuestos Orgánicos” (alasGRATO).

alasGRATO es una plataforma que pretende centralizar el proceso de obtención de fármacos permitiendo, que un especialista químico que haga estudios en este campo, cuente con la posibilidad de poner en práctica sus investigaciones. Este proceso tan complejo se divide en varios pasos como la edición de estructuras químicas, el cálculo de los descriptores asociados a éstas, la reducción de los espacios muestrales, la predicción de la actividad biológica, entre otros.

Cada uno de estos pasos, constituye un módulo independiente de la plataforma, que como necesidad del proyecto, deberá integrarse como un plug-in para permitir la interacción entre cada parte del sistema que represente una funcionalidad, de manera que resulte más sencillo utilizar las salidas de una operación como entradas para otras. Otra necesidad es realizar todas las operaciones en el servidor de aplicaciones Tomcat del proyecto, mediante servicios web que permitan distribuir los cálculos que demoren mucho

tiempo a través de la plataforma de cómputo distribuido T-arenal. Además, deberá almacenar los resultados obtenidos en una base de datos local para su uso posterior.

Actualmente, alasGRATO cuenta con una aplicación para el cálculo de descriptores atómicos, moleculares y para aminoácidos del módulo Cálculo de Descriptores. Ésta, es una librería que implementa los algoritmos de 11 índices moleculares y 4 atómicos, así como, 4 nuevos índices para aminoácidos. Sin embargo, esta herramienta no permite guardar los resultados en una base de datos. Tampoco trabaja a través del servidor de aplicaciones de la plataforma, ni distribuye los cálculos a través de éste. Además, no está orientada a la arquitectura plug-in, por lo que no puede integrarse a alasGRATO.

Ante tal situación se plantea el siguiente **problema científico:** ¿Cómo lograr la integración del módulo Cálculo de Descriptores a la plataforma alasGRATO?

El problema establecido se enmarca en el **objeto de estudio:** Aplicaciones informáticas basadas en arquitectura plug-in.

El objeto de estudio delimita el **campo de acción:** Aplicaciones informáticas basadas en arquitectura plug-in para la plataforma alasGRATO.

En aras de solucionar el problema planteado se define como **objetivo general:** Desarrollar un plug-in para el módulo Cálculo de Descriptores de la plataforma alasGRATO.

Para cumplir este objetivo se fijan los siguientes **objetivos específicos:**

- Diseñar el plug-in para el módulo Cálculo de Descriptores.
- Implementar el plug-in para el módulo Cálculo de Descriptores.
- Realizar pruebas al plug-in implementado.

Las **tareas de investigación** definidas para cumplir los objetivos específicos fijados son:

1. Revisión de la literatura acerca de los plug-in.
2. Revisión y análisis de la literatura acerca de servicios web.
3. Revisión y análisis de la literatura acerca de bases de datos.
4. Estudiar la arquitectura de la herramienta para el cálculo de descriptores atómicos, moleculares y para aminoácidos.
5. Elaboración del modelo de dominio.

6. Definición de los requisitos funcionales y no funcionales de la herramienta.
7. Desarrollo del diagrama de casos de uso del sistema.
8. Descripción de los casos de uso correspondientes al sistema.
9. Desarrollo del diagrama de clases del diseño.
10. Crear una base de datos local que permite guardar el resultado de los cálculos de descriptores.
11. Implementación de un Servicio Web para obtener los descriptores que se disponen para el cálculo.
12. Implementación de un Servicio Web para realizar los cálculos de descriptores de forma distribuida.
13. Implementación del plug-in para la integración del módulo Cálculo de Descriptores a la plataforma alasGRATO.
14. Realizar pruebas de unidad (caja negra) al plug-in implementado.

El presente documento consta de un Resumen, Introducción, cuatro capítulos, Conclusiones Generales, Recomendaciones, Referencias Bibliográficas, Bibliografía y Anexos.

En el **capítulo I** se realiza el estudio de las herramientas más significativas para la investigación. Así como un resumen de las principales herramientas y las tecnologías empleadas durante el desarrollo del trabajo. En **capítulo II** se realiza una breve descripción de la aplicación a implementar. Se describe la solución propuesta mediante los componentes del dominio. Se detallan las reglas del negocio, requisitos funcionales y no funcionales del sistema. Se muestra el diagrama de casos de uso del sistema y la descripción de éstos para el sistema que se implementó. En el **capítulo III** se describe el estilo arquitectónico utilizado para el desarrollo del sistema y los patrones de diseño empleados. Se realiza el diagrama de clases del diseño e interacción de los casos de uso, así como el modelo de despliegue. Por último, en el **capítulo IV** se realizan los diagramas de componentes, teniendo en cuenta las bases creadas por el flujo de trabajo de diseño además, se describen los algoritmos más significativos y las pruebas que se le realizaron al sistema.

CAPÍTULO I

FUNDAMENTACIÓN TEÓRICA

En este capítulo se realiza el estudio de la herramienta utilizada para el cálculo de descriptores y las características principales de los plug-ins. Se explica el funcionamiento y utilidad de los servicios web y por último, se brinda un resumen de las principales herramientas y tecnologías que se deberán emplear para el desarrollo de la aplicación propuesta.

1.1 Herramienta para el cálculo de descriptores atómicos, moleculares y para aminoácidos

La aplicación realiza el cálculo de índices topológicos y topográficos, tanto atómicos, moleculares y para aminoácidos. Se compone por dos módulos lógicos, uno para el cálculo de los descriptores atómicos y moleculares y otro para el cálculo de los índices para aminoácidos.

El módulo que calcula los descriptores atómicos y moleculares recibe los datos de las moléculas en ficheros con formato ***.mol*** desde un directorio designado por el usuario, luego se optimizan las moléculas utilizando el MOPAC 6. Una vez que se realiza la optimización, se calculan los descriptores seleccionados por el usuario y se ofrecen los resultados en un fichero ***.dsc***, donde se muestra los caminos, clústeres y clúster-camino, así como los valores de los descriptores calculados. Además, asigna la hibridación a cada átomo y la fragmentación molecular que se necesita para calcular los índices.

El módulo que calcula los índices para aminoácidos parte de ficheros, pero éstos en formato ***.pdb*** que representan una proteína con sus características. Éste módulo trae implícito los valores intrínsecos de electronegatividad y de refractividad de cada aminoácido, calculados a través del módulo descrito anteriormente. Los descriptores seleccionados por el usuario se calculan y se guardan en un fichero ***.dsa***. (1)

1.2 Plug-in

Un Plug-in es una aplicación que incrementa una característica o servicio a un sistema ya existente; y éste se ejecuta por la aplicación principal.

A nivel mundial, existen aplicaciones basadas en arquitectura plug-in que le brindan la capacidad de agregar funcionalidades nuevas en tiempo de ejecución. También tiene varias ventajas reales:

- Permite que los desarrolladores externos colaboren con la aplicación principal, de manera que se extiendan sus funciones.
- Hace posible que sólo cierta funcionalidad esté disponible si alguna librería de o aplicación de terceros no está disponible en el sistema.
- Hace posible el personalizar la aplicación de maneras no pensadas por el autor, o que al menos no tenía la intención de hacer.
- Es posible compartir código entre aplicaciones sin mucho que ver entre ellas si estas comparten una infraestructura de plug-ins entre sí. (2)

Debido a las ventajas y facilidades que brindan los plug-in, a nivel de proyecto se decide integrar todos los módulos basándose en esta arquitectura. De esta manera, posibilita al especialista poder decidir con qué módulos desea trabajar; sin necesidad de tenerlos incorporados a la plataforma. El especialista puede, a través del gestor de la plataforma, reemplazar los módulos por nuevas versiones de éstos sin tener que cambiar la versión de alasGRATO.

1.3 Base de Datos

Una Base de Datos (BD) es un conjunto de datos relacionados entre sí, que tiene las siguientes propiedades implícitas:

- Representan algún aspecto del mundo real.
- Es un conjunto de datos lógicamente coherentes, con un cierto significado inherente.
- Cuenta con un público activamente interesado en el contenido de la BD.
- Su tamaño es variado. (3)

Para la aplicación a implementar se debe contar con una BD que permita guardar el resultado de los cálculos de descriptores, brindando al especialista la posibilidad de consultar los datos y poder utilizarlos como entradas para otros módulos de la plataforma.

1.4 Servicios Web

Un servicio web es una interfaz capaz de recibir una petición, activar procesos y devolver los resultados de una operación, todo esto, en Internet, a través de protocolos de red (HTTP, FTP, SMTP). Mediante XML, es que se realiza la comunicación entre los diferentes entornos del servicio web.

Para establecer un diálogo coherente entre el Servicio Web Cliente, que envía la petición y recibe la respuesta y el Servidor de servicios web, el que ejecuta el proceso y envía la respuesta, se utiliza SOAP (*Simple Object Access Protocol*), que es una codificación basada en XML.

Un Servicio Web, en vez de obtener peticiones desde un navegador y devolver páginas web como respuesta, recibe peticiones, mediante un mensaje formateado con SOAP, desde otras aplicaciones, realiza la labor que le solicitaron y devuelve un mensaje de respuesta también con formato SOAP. (Fig. 1.1)

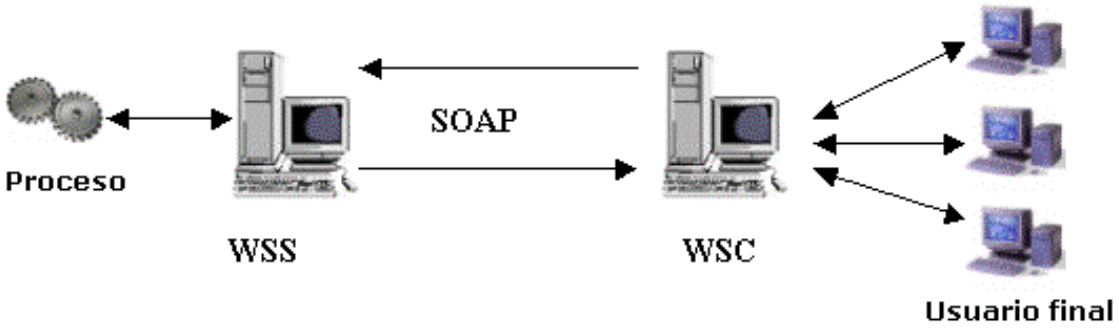


Figura 1.1. Esquema del Funcionamiento de los Servicios Web

Los servicios web se utilizan para tener acceso a información y procesos remotos a través de aplicaciones web o desktop. Esto quiere decir, por ejemplo, que un Servicio Web puede invocarse remotamente como una funcionalidad más dentro de una aplicación desktop, con las ventajas que: es totalmente invisible para el usuario final, al ser un proceso remoto el consumo de recursos se absorbe por el Servicio Web y la aplicación puede desarrollarse en cualquier lenguaje y plataforma. (4) (Fig. 1.2)

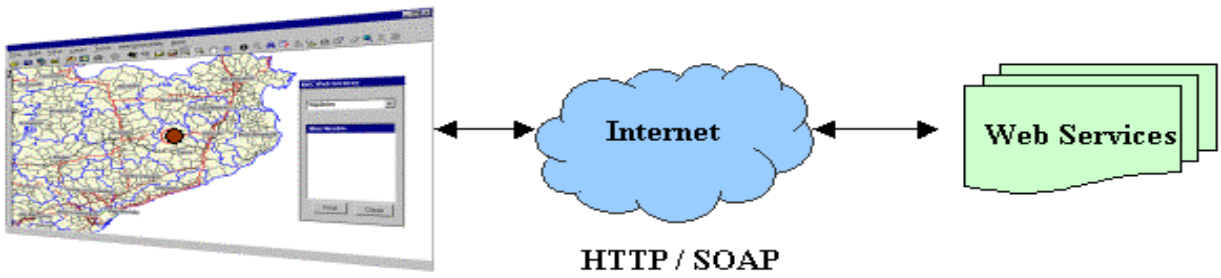


Figura 1.2. Comunicación entre los Servicios Web y la aplicación desktop a través de Internet

1.5 Herramientas y metodologías a utilizar en la aplicación

Uno de los objetivos fundamentales a la hora de realizar una aplicación de software factible, eficaz y de calidad, es definir cuáles son las herramientas y metodologías que serán de mayor utilidad para la implementación. Las metodologías de desarrollo de software describen los pasos para lograr dicho objetivo, puesto que definen detalladamente qué es lo que se debe hacer y quién debe hacerlo. Precisan las tareas fundamentales a desarrollar a lo largo del ciclo de vida del proyecto, los artefactos que deben construirse, el orden en que se deben realizar y designan los responsables de cada tarea.

Para el desarrollo de esta aplicación, se realizó un estudio de las principales herramientas y tecnologías que se seleccionaron para la implementación del mismo, teniendo en cuenta las especificaciones del cliente y siguiendo los estándares empleados en el proyecto alasGRATO.

1.5.1 Metodologías de desarrollo de software: OpenUP

OpenUP es un proceso unificado que aplica acercamientos iterativos e incrementales dentro de un ciclo de vida estructurado. Además, abarca una filosofía pragmática, ágil y que se centra en la naturaleza de colaboración de desarrollo de software. Es una herramienta que se puede ampliar para hacer frente a una amplia variedad de tipos de proyectos.

El esfuerzo personal en un proyecto de OpenUP se organiza en micro-incrementos, los cuales representan unidades cortas de trabajo que producen un paso constante y mensurable en el progreso del proyecto. El proceso aplica la colaboración intensiva como sistema de desarrollo incremental. Estos micro-incrementos proporcionan un lazo de regeneración extremadamente corto, que conducen a decisiones

adaptables dentro de cada iteración. El ciclo de vida de un proyecto en OpenUP está estructurado en cuatro fases: inicio, elaboración, construcción y transición. (5)

1.5.2 Lenguaje de modelado: UML

UML (*Unified Modeling Language o* Lenguaje Unificado de Modelado) es el más conocido y utilizado en la actualidad. Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. Se utiliza para definir un sistema, para detallar los artefactos en el mismo, en otras palabras, es el lenguaje en el que está descrito el modelo. UML no es un lenguaje de programación, sino un lenguaje de propósito general para el modelado orientado a objetos. También puede considerarse como un lenguaje de modelado visual que permite una abstracción del sistema y sus componentes.

Principales diagramas UML:

Diagramas de estructura estática: Describen las propiedades estructurales del sistema.

- Diagrama de clases: Conjunto de clases, interfaces y colaboraciones; así como sus colaboraciones.
- Diagrama de objetos: Conjunto de objetos y sus relaciones.
- Diagrama de casos de uso: Conjunto de casos de uso y actores y sus relaciones.

Diagramas de comportamiento:

- Diagramas de interacción (secuencia y colaboración): Objetos y sus relaciones, incluyendo los mensajes que pueden enviarse entre ellos.
- Diagrama de estados: Muestra una máquina de estado que consta de estados, transiciones, eventos y actividades.
- Diagrama de actividad: Es un tipo especial de diagrama de estados que muestra el flujo de actividades dentro de un sistema.

Diagramas de Implementación:

- Diagramas de despliegue.
- Diagrama de componentes. (6)

1.5.3 Herramientas CASE: Visual Paradigm

Las herramientas CASE (*Computer-Aided Software Engineering* o Ingeniería de Software Asistida por Computadora) se pueden definir como un conjunto de métodos, utilidades y técnicas que dan asistencia a los analistas, ingenieros de software y desarrolladores, facilitando la automatización del ciclo de vida del desarrollo de sistemas de información, completamente o en alguna de sus fases. También se puede decir que es la aplicación de métodos y técnicas a través de las cuales se hacen útiles a las personas comprender las capacidades de las computadoras, por medio de programas, de procedimientos y su respectiva documentación.

Visual Paradigm es una herramienta que soporta el UML como lenguaje de modelado, soporta el ciclo de vida completo del desarrollo de software: análisis y diseño, construcción, pruebas y despliegue. Son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de dinero. Estas herramientas ayudan en todos los aspectos del ciclo de vida de desarrollo del software, en tareas como el proceso de realizar un diseño del proyecto, cálculo de costes, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores, entre otras.

Visual Paradigm es una potente herramienta para visualizar y diseñar elementos de software, posee características gráficas muy cómodas que facilitan la realización de los diagramas de modelación que sigue el estándar UML, proporciona a los desarrolladores una plataforma que les permita diseñar un producto con calidad de forma rápida, facilita la interoperabilidad con otras herramientas; tiene licencia gratuita, es multiplataforma y puede contar con plug-ins que le posibilitan la integración con algunas herramientas Java como son: Eclipse/IBM WebSphere, JBuilder, NetBeans, entre otras. 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”. (7)

Una herramienta CASE muy parecida al Visual Paradigm, es el Rational Rose, sin embargo, ésta no es una herramienta multiplataforma y posee menos facilidades que el Visual Paradigm. Para el desarrollo de esta aplicación se escogió el Visual Paradigm por las características y funcionalidades anteriormente explicadas.

1.5.4 Lenguaje de programación: Java

El lenguaje de programación Java, es un lenguaje muy difundido en la actualidad y cada vez cobra más auge debido a sus potencialidades y facilidad de aprendizaje. Tiene similitud con otros lenguajes como C y C++. Es un lenguaje orientado a objetos y soporta las tres características propias del paradigma de la orientación a objetos: encapsulación, herencia y polimorfismo. Una de las ventajas de este lenguaje es que utiliza el concepto de máquina virtual, es decir, que puede correr en cualquier plataforma o donde esté instalada la Máquina Virtual, pues el código que genera no es específico de una plataforma en particular.
Otra de sus principales características es la seguridad que ofrece, pues su código pasa muchas pruebas antes de ejecutarse en una máquina. Éste 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. Si los byte-codes pasan la verificación sin generar ningún mensaje de error, entonces se sabe que:

- El código no produce desbordamiento de operandos en la pila.
- El tipo de los parámetros de todos los códigos de operación son conocidos y correctos.
- No ha ocurrido ninguna conversión ilegal de datos, tal como convertir enteros en punteros.
- El acceso a los campos de un objeto se sabe que es legal.
- No hay ningún intento de violar las reglas de acceso y seguridad establecidas. (8)

Según el ranking de los lenguajes de programación más populares, del Índice de Programación de la Comunidad de TIOBE[1], desde el año 2002, Java se ha mantenido en primer lugar en la mayoría de los años. (Anexo 1)

1.5.5 Entorno de Desarrollo Integrado (IDE): Eclipse

Un entorno de desarrollo integrado o IDE (*Integrated Development Environment*), es un programa informático compuesto por un conjunto de herramientas de programación. Los IDE pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes.

[1] **Índice de Programación de la Comunidad de TIOBE:** brinda información acerca de la popularidad de los lenguajes de programación, este índice se actualiza mensualmente.

Para el desarrollo del sistema se utiliza el Eclipse, que combinado con el JDT (*Java Development Tools*), permite disponer de un IDE para Java de gran calidad. Eclipse cuenta con un editor muy visual con sintaxis coloreada, ofrece compilación incremental de código, un potente depurador (que permite establecer puntos de interrupción, modificar e inspeccionar valores de variables, e incluso depurar código que resida en una máquina remota), un navegador de clases, un gestor de archivos y proyectos. La versión estándar de Eclipse proporciona también una biblioteca de refactorización de código y una lista de tareas. También incluye una herramienta para completar código. Otra característica de Eclipse es que es muy eficiente para reducir los tiempos de depuración y pruebas. (9)

1.5.6 Sistema Gestor de Base de Datos: SQLite

SQLite es un proyecto de dominio público que implementa una pequeña librería de aproximadamente 500kb, programado en el lenguaje C, y tiene como función hacer de un sistema de bases de datos relacional.

A continuación se exponen algunas razones por las que se elige SQLite:

- **Rendimiento de base de datos:** realiza operaciones de manera eficiente y es más rápido que MySQL y PostgreSQL.
- **Portabilidad:** se ejecuta en muchas plataformas y sus bases de datos pueden ser fácilmente portadas sin ninguna configuración o administración.
- **Estabilidad:** es compatible con ACID, (Atomicidad, Consistencia, Aislamiento y Durabilidad).
- **Interfaces:** cuenta con diferentes interfaces del API (*Application Programming Interface*), las cuales permiten trabajar con diferentes lenguajes de programación como C++, PHP, Perl, Python, Java.
- **Costo:** es de dominio público, y por tanto, es libre de utilizar para cualquier propósito sin costo y se puede redistribuir libremente. (10)

1.6 Front-End GRATO: Herramienta manejadora de plug-in

El Front-End GRATO es dinámico y multiplataforma, y se encarga de la portabilidad de los plug-ins garantizando:

- Unificar la carga de los plug-ins a través de descriptores XML.
- Ganar en soportes de escalabilidad para el ingreso de futuros plug-ins.
- Ganar en el manejo de memoria en ejecución controlando todos los componentes visuales desde un mismo marco.
- Mantener el principio de conservación, pues en la evolución del Front-End, los plug-ins desarrollados siempre serán compatibles. (11)

Se utiliza el Front-End GRATO como herramienta manejadora de plug-in debido a las características y ventajas expuestas anteriormente. Además, su uso es más sencillo pues la interfaz es amigable y configurable. También se tuvo en cuenta, que éste se creó como una necesidad del proyecto alasGRATO con el objetivo de integrarle todos los módulos existentes en la plataforma y de facilitar la interacción con sus usuarios.

1.7 Plataforma de cómputo distribuido: T-arenal

Un sistema distribuido es un conjunto de computadoras autónomas que aparecen integradas ante los usuarios como una única máquina para resolver determinado problema. Los componentes del sistema no son más que hardware y software que se comunican entre sí a través de una red, preferiblemente canales de alta velocidad. En el caso de un sistema de cómputo distribuido el problema a resolver, por lo general, requiere grandes cálculos y lo que se trata es de reducir el tiempo en obtener la respuesta utilizando el procesamiento en paralelo al distribuir los cómputos de forma transparente para el usuario entre varias computadoras. (12)

Al realizar cálculo de descriptores, el tiempo de demora en la obtención de los resultados de los cálculos, depende de la complejidad que presente cada una de las estructuras químicas. A su vez, la gran cantidad de volúmenes de cómputo, en un tiempo moderado, se limita por la incapacidad de ser procesados por un solo ordenador.

Una forma de disminuir el tiempo de obtención de los resultados, es a través de la utilización de la plataforma de cómputo distribuido, T-arenal. Este sistema cuenta, fundamentalmente, con dos módulos, servidor y cliente; y se caracteriza por usar el tiempo ocioso de las computadoras configuradas como clientes.

El módulo servidor almacena los datos y el algoritmo que los procesará y los divide en pequeños sub-problemas, llamados unidades de trabajo. Su responsabilidad es repartir las unidades de trabajo entre los clientes.

El módulo cliente se instala en cada una de las computadoras que conforman la plataforma, conectadas al servidor mediante una red LAN. El cliente realiza una petición de una unidad de trabajo al servidor, hace el procesamiento de la misma y luego retorna el resultado al servidor, y vuelve a pedir otra unidad de trabajo. Finalmente, el servidor almacena el resultado de cada uno de los sub-problemas calculados en cada cliente para finalmente construir el resultado del problema original. (13)

T-arenal ofrece una alternativa de cómputo que agrupa un conjunto de estaciones de trabajo, sin intentar eliminar o aminorar las posibilidades de aplicación de los modelos paralelos y el uso de supercomputadoras, sino de complementar todos los medios disponibles en una gran “supercomputadora virtual”.

1.8 Conclusiones Parciales

Como resultado de la investigación, se aplica la arquitectura plug-in a la herramienta para el cálculo de descriptores atómicos, moleculares y para aminoácidos para su integración a la plataforma. Además, se determinaron las principales herramientas y tecnologías a emplear durante el desarrollo del trabajo, donde se elige, OpenUP como metodología de desarrollo, como lenguaje de modelado UML, el Visual Paradigm como herramienta CASE, el Eclipse como entorno de desarrollo integrado, el lenguaje de programación Java, el Front-End GRATO como herramienta manejadora de plug-in, SQLite para gestionar la base de datos local que se deberá desarrollar y T-arenal como plataforma de cómputo distribuido.

CAPÍTULO II

CARACTERÍSTICAS DEL SISTEMA

En el presente capítulo se ofrece; una breve descripción de la solución propuesta. Se realiza el modelo de dominio, así como la especificación de los requisitos funcionales y no funcionales de la aplicación. Además, se realiza el diagrama de casos de uso y la descripción textual de éstos.

2.1 Breve descripción de la aplicación

La herramienta a desarrollar será un plug-in para el módulo Cálculo de Descriptores de alasGRATO, que permitirá la integración de éste a la plataforma. En la aplicación se implementarán dos servicios web que se utilizarán cuando el especialista realice los cálculos de descriptores a través de los servidores del proyecto. Uno para obtener los descriptores disponibles y el otro para realizar el cálculo de descriptores de forma distribuida. Se implementará una funcionalidad que posibilite al especialista generar un fichero ARFF (*Attribute-Relation File Format*), a partir de los resultados de los cálculos, que le permitirá utilizarlos en otros módulos. Además, se diseñará una base de datos local para almacenar los resultados que obtiene el especialista luego de realizar el cálculo.

2.2 Modelo de Dominio

Para desarrollar la herramienta se determinó que no es necesario un modelo completo del negocio, debido a que los procesos de éste no están bien definidos, por lo que se decide realizar el modelo del dominio, con el objetivo de identificar los procesos de automatización. Un modelo del dominio captura los tipos más importantes de objetos que existen o los eventos que ocurren en el entorno donde se encontrará el sistema. Éste, se considera un subconjunto del modelo de objetos del negocio y describe las entidades que participan en el sistema y sus relaciones. (14)

Con la realización del modelo de dominio, se describe el funcionamiento de la aplicación mediante conceptos, entidades y sus relaciones agrupadas en este modelo para un mejor entendimiento del contexto en que se enmarca el sistema.

2.2.1 Glosario de términos para el dominio propuesto

Especialista Químico: Persona que se encarga de seleccionar las moléculas y los descriptores, realizar el cálculo de descriptores y generar reportes.

Estructuras Químicas: Archivo de estructura química que puede ser de proteínas o de moléculas.

Descriptores: Números que describen la estructura química o una propiedad de la molécula o fragmento de ésta.

Fragmento: Una porción de la molécula que puede ser un Camino, Clúster o Clúster-Camino.

2.2.2 Descripción del Modelo de Dominio

El modelo de dominio del presente trabajo, representa las entidades y las relaciones entre ellas, donde el ***Especialista Químico*** realiza el ***Cálculo de Descriptores*** que se obtiene a partir de las ***Estructuras Químicas***, los ***Descriptores*** y los ***Fragmentos*** que debió seleccionar con anterioridad. (Fig. 2.1)

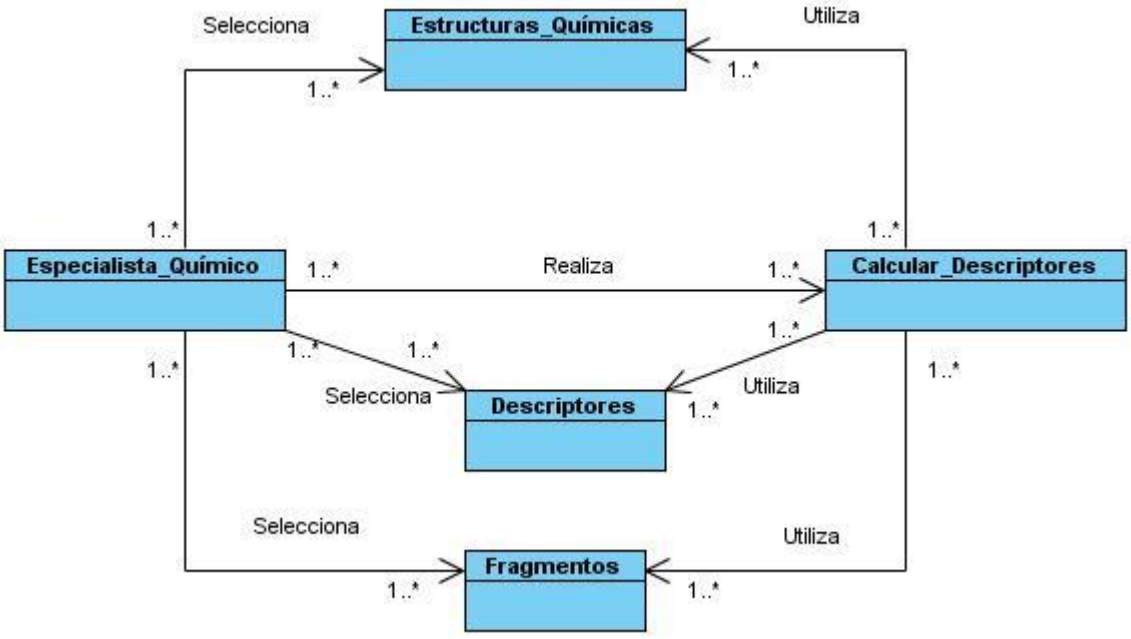


Figura 2.1: Modelo de Dominio del sistema propuesto

2.3 Especificación de requerimientos del sistema propuesto

Los requerimientos del sistema son aquellas condiciones o capacidades que tienen que tener los sistemas para satisfacer un contrato o documento formal. Éstos, describen qué es lo que debe hacer la aplicación.

2.3.1 Requisitos funcionales

Los requerimientos funcionales expresan una especificación detallada de las responsabilidades de la herramienta que se propone. Ellos permiten determinar, de una manera clara, lo que debe hacer el software.

Los requerimientos funcionales del sistema propuesto son los siguientes:

R1 Configurar cálculo

R2 Seleccionar la(s) estructura(s) química(s)

R3 Seleccionar tipos de descriptores a calcular

R4 Seleccionar orden de los fragmentos

R5 Calcular descriptores seleccionados

R6 Guardar los resultados de los cálculos de descriptores en la base de datos local

R7 Mostrar resultado de los cálculos realizados

R8 Seleccionar cálculo realizado

R9 Crear fichero ARFF

2.3.2 Requisitos no funcionales

Los requerimientos no funcionales, son aquellas propiedades o cualidades que el producto debe tener, pues representan las características de éste. A continuación se describen cuáles son algunas de las cualidades más significativas que deberá cumplir la aplicación:

- **Apariencia o interfaz externa:**

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

- **Usabilidad:**

La herramienta podrá utilizarse por cualquier tipo de personas que posea conocimientos básicos en el manejo de la computadora, sólo se necesita contar con conocimientos especializados en química para entender los resultados dados por la aplicación.

- **Soporte:**

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

- **Portabilidad:**

La herramienta es multiplataforma, o sea, puede ejecutarse sobre los sistemas operativos Linux y Windows.

- **Seguridad:**

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

- **Confiabilidad:**

Para evitar cualquier tipo de error, la aplicación debe ser confiable y precisa en la información que le suministra al especialista.

- **Software:**
  - ✓ Se debe disponer de sistemas operativos Linux, Windows 95 o superior para la instalación de la aplicación.
  - ✓ Debe contarse con el *Java Runtime Environment* (JRE) versión 1.5 o superior instalado.

- **Hardware:**

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

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

2.4 Definición del sistema propuesto

2.4.1 Actores del sistema

Los actores, son terceros fuera del sistema, que interactúan con él de alguna manera. Generalmente, estimulan al sistema con eventos de entrada o reciben algo de él. Puede intercambiar información o ser un recipiente pasivo de información. Representa a un ser humano, un software o una máquina que interactúa con el sistema. (14) El actor del sistema, así como una breve descripción de la función que tendrá dentro de éste, se representa en la (Tabla 2.1).

Tabla 2.1: Descripción de los actores del sistema propuesto

Actores	Descripción
Especialista Químico	Representa el usuario que va ha hacer uso de la aplicación, teniendo la posibilidad de interactuar con todas la funcionalidades de ésta.

2.4.2 Casos de Uso del Sistema

Los casos de uso del sistema se relacionan estrechamente con los requisitos funcionales planteados con anterioridad, pues representan las funcionalidades que tendrá el sistema.

En la aplicación propuesta se definieron tres CU, Configurar Cálculo que responde al requisito **R1,** Realizar Cálculo de Descriptores que engloba los requisitos **R2, R3, R4, R5 y R6; y** Generar Fichero ARFF que engloba los requisitos **R7, R8 y R9** definidos en el sub-epígrafe Requisitos Funcionales [ver 2.3.1].

2.4.3 Diagrama de Casos de Uso del Sistema

En el diagrama de casos de uso de la aplicación propuesta, se muestra cómo el especialista químico es quien controla las funcionalidades de la herramienta, que se engloban en los CU, Configurar Cálculo, que puede ser local o distribuido, Realizar Cálculo de Descriptores y Generar Fichero ARFF. La aplicación está configurada por defecto para calcular distribuido, sin embargo, el especialista químico puede configurarla para realizar los cálculos en el ordenador. (Fig. 2.2)

Figura 2.2. Diagrama de casos de uso del sistema propuesto

2.4.4 Descripción textual de los Casos de Uso del Sistema

Con la representación gráfica del diagrama de casos de uso no es suficiente para lograr entender las funcionalidades asociadas a un caso de uso, por lo que es necesario describir textualmente cada uno de éstos, de manera que queden especificadas todas las acciones necesarias que realizan el actor y el sistema. Con las descripciones que se presentan a continuación de los CU, Configurar Cálculo de Descriptores (Tabla 2.2), Realizar Cálculo de Descriptores (Tabla 2.3) y Generar Fichero ARFF (Tabla 2.4), se obtendrá una idea más clara de cómo se realiza el proceso de automatización y quienes intervienen directamente en éste.

Tabla 2.2: Descripción del CU Configurar Cálculo

Caso de Uso	Configurar Cálculo
Actores:	Especialista Químico
Resumen:	El caso de uso se inicia cuando el Especialista Químico desea cambiar el modo de cálculo y selecciona la opción Configurar del menú Configuración del plug-in Cálculo de Descriptores. El mismo cambia la configuración de los cálculos que puede ser local o distribuido, finalizando así el caso de uso.
Referencia:	R1
Prioridad:	Opcional
Precondiciones:	El sistema obtiene la última configuración almacenada en el archivo de configuración de la aplicación.

Poscondiciones:	Se cambia el modo de calcular.
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El Especialista Químico ejecuta la opción Configurar del menú Configuración del plug-in para el módulo Cálculo de Descriptores.	1.1 El sistema muestra la interfaz correspondiente a la Configuración del cálculo; donde el modo de calcular que está por defecto es el que se obtiene del archivo de configuración de la aplicación que puede ser local o distribuido.
2. El Especialista Químico tiene la opción de cambiar la configuración del cálculo de la siguiente forma: • Cambiar para calcular local • Cambiar para calcular distribuido	2.1 Si desea calcular local va a la sección **"Calcular Local".** 2.2 Si desea calcular distribuido va a la sección **"Calcular Distribuido".**
Flujo Normal de Eventos **"Sección Calcular Local"**	
Acción del Actor	**Respuesta del Sistema**
1. El Especialista Químico selecciona la opción calcular local.	1.1 El sistema muestra un campo de texto y un botón para entrar la dirección de la librería para el cálculo.
2. El Especialista Químico presiona el botón para seleccionar la dirección de la librería.	2.1 El sistema muestra una interfaz para buscar la librería.
3. El Especialista Químico selecciona la librería.	3.1 El sistema muestra la dirección de la librería en el campo de texto.
4. El Especialista Químico presiona el botón Aceptar.	4.1 El sistema verifica si hay una librería seleccionada. 4.2 El sistema cambia la configuración para

	calcular local.
Flujo alterno	
Acción del Actor	**Respuesta del Sistema**
	4.1 El sistema muestra un mensaje de error y retorna al flujo normal de eventos 2.
Flujo Normal de Eventos "Sección Calcular Distribuido"	
Acción del Actor	**Respuesta del Sistema**
1. El Especialista Químico desmarca la opción calcular local.	1.1 El sistema muestra la opción desmarcada.
2. El Especialista Químico presiona el botón Aceptar.	2.1 El sistema cambia la configuración para calcular distribuido.

Tabla 2.3: Descripción del CU Realizar Cálculo de Descriptores

Caso de Uso	Realizar Cálculo de Descriptores
Actores:	Especialista Químico
Resumen:	El caso de uso se inicia cuando el Especialista Químico ejecuta el módulo Cálculo de Descriptores de la opción módulos del Visualizador bioGRATOViewer. El mismo realizará los cálculos de los descriptores seleccionados por el especialista y guardará los resultados en la base de datos local de la aplicación, finalizando así el caso de uso.
Referencia:	R2, R3, R4, R5, R6
Prioridad:	Crítico
Precondiciones:	Deben existir moléculas o proteínas cargadas en el árbol de ficheros del Visualizador.
Poscondiciones:	El sistema guarda el cálculo de los descriptores

	seleccionados.
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sitema**
1. El Especialista Químico ejecuta el módulo Cálculo de Descriptores de la opción módulos del Visualizador bioGRATOViewer.	1.1 El sistema obtiene del visualizador, las estructuras químicas que pueden ser, moléculas o proteínas. 1.2 Si obtiene moléculas, va a la sección "**Moléculas**". 1.3 Si obtiene proteínas, va a la sección "**Proteínas**".
Flujo Normal de Eventos "Sección Moléculas"	
Acción del Actor	**Respuesta del Sistema**
	1. El sistema muestra la interfaz principal con las opciones habilitadas para cuando las estructuras son moléculas.
2. El Especialista Químico selecciona las moléculas a las cuales desea realizar el cálculo.	2.1 El sistema marca las moléculas seleccionadas.
3. El Especialista Químico tiene la opción de seleccionar los siguientes descriptores: • Topológicos • Topográficos • Híbridos Tanto Moleculares como Atómicos	3.1 El sistema verifica los descriptores seleccionados y si encuentra alguno Molecular, habilita los fragmentos que puede seleccionar.
4. El Especialista Químico tiene la	4.1 El sistema marca los fragmentos

opción de seleccionar los siguientes fragmentos: • Caminos desde orden 1 hasta orden 15 • Clústeres desde orden 3 hasta orden 4 • Para los Clústeres-Caminos desde orden 1 hasta orden 15.	seleccionados.
5. El Especialista Químico presiona el botón Calcular.	5.1 El sistema verifica si hay moléculas seleccionadas. 5.2 El sistema verifica si hay descriptores seleccionados. 5.3 El sistema verifica si hay fragmentos seleccionados. 5.4 El sistema muestra una interfaz requiriendo un nombre y un comentario para identificar los cálculos.
6. El Especialista Químico introduce el nombre y el comentario.	6.1 El sistema verifica los datos de entrada. 6.2 El sistema guarda los resultados de los cálculos en la base de datos local.
Flujo Alterno	
Acción del Actor	**Respuesta del Sistema**
	5.1 El sistema muestra un mensaje de error y retorna al flujo normal de eventos 2. 5.2 El sistema muestra un mensaje de error y retorna al flujo normal de eventos 3. 5.3 El sistema muestra un mensaje de error y retorna al flujo normal de eventos 4.

	6.1 El sistema muestra un mensaje de error y retorna al flujo normal de eventos 6.
Flujo Normal de Eventos "Sección Proteínas"	
Acción del Actor	**Respuesta del Sistema**
	1. El sistema muestra la interfaz principal con las opciones habilitadas para cuando las estructuras son proteínas.
2. El Especialista Químico selecciona las proteínas a las cuales desea realizar el cálculo.	2.1 El sistema marca las proteínas seleccionadas.
3. El Especialista Químico selecciona los siguientes descriptores: • Topológicos • Topográficos	3.1 El sistema marca los descriptores seleccionados.
4. El Especialista Químico oprime el botón Calcular.	4.1 El sistema verifica que existan proteínas y descriptores seleccionados. 4.2 El sistema muestra una interfaz requiriendo un nombre y un comentario para identificar los cálculos.
5. El Especialista Químico introduce el nombre y el comentario.	5.1 El sistema verifica los datos de entrada. 5.2 El sistema guarda los resultados en la base de datos local.
Flujo Alterno	
Acción del Actor	**Respuesta del Sistema**
	4.1 El sistema muestra un mensaje de error y retorna al flujo normal de eventos 2.
	5.1 El sistema muestra un mensaje de error y

	retorna al flujo normal de eventos 5.

Tabla 2.4: Descripción del CU Generar Fichero ARFF

Caso de Uso	Generar Fichero ARFF
Actores:	Especialista Químico
Resumen:	El caso de uso se inicia cuando el Especialista Químico selecciona Generar Fichero ARFF del menú Convertir del plug-in para el módulo Cálculo de Descriptores. El sistema muestra los resultados de los cálculos realizados y convierte a ARFF, sólo aquel que el especialista seleccione, finalizando así el caso de uso.
Referencia:	R7, R8, R9
Prioridad:	Opcional
Precondiciones:	Deben existir cálculos en la base de datos local.
Poscondiciones:	Se genera el fichero ARFF
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El Especialista Químico ejecuta la opción Generar Fichero ARFF del menú Convertir del plug-in para el módulo Cálculo de Descriptores.	1.1 El sistema muestra los resultados de los cálculos almacenados en la base de datos local.
2. El Especialista Químico selecciona el cálculo que desea convertir.	2.1 El sistema marca el cálculo seleccionado.
3. El Especialista Químico presiona el botón Generar.	3.1 El sistema verifica que exista un cálculo seleccionado. 3.2 El sistema muestra una interfaz para seleccionar dónde guardar el fichero ARFF.
4. El especialista selecciona la dirección donde desea guardar el fichero ARFF.	5.1 El sistema genera el fichero ARFF y lo guarda en la dirección que el Especialista

5. El Especialista Químico presiona el botón Aceptar.	Químico seleccionó.
Flujo Alterno	
Acción del Actor	**Respuesta del Sistema**
	3.1 El sistema muestra un mensaje de error y retorna al flujo normal de eventos 2.

2.5 Conclusiones Parciales

En este capítulo se determinaron las características básicas con las que debe contar un sistema para poder utilizar el producto. Se seleccionó el Especialista Químico como actor del sistema y se definieron nueve requisitos funcionales, de ellos, uno tributó al CU Configurar Cálculo y cinco se englobaron en el CU Realizar Cálculo de Descriptores y tres en el CU Generar Fichero ARFF. Además, se realizó la descripción textual de los CU, para lograr un mejor entendimiento de las funcionalidades de la herramienta a desarrollar.

CAPÍTULO III

DISEÑO DEL SISTEMA

En el flujo de trabajo de Diseño, se modela el sistema a partir de la arquitectura. Se define cuál será el estilo arquitectónico a utilizar para el desarrollo del sistema y se describen los patrones de diseño que se ajusten a la herramienta. Se realiza el diagrama de clases del diseño, y los diagramas de interacción por cada caso de uso, así como el modelo de despliegue.

3.1 Descripción de la arquitectura

La arquitectura de software es previa al diseño y a la implementación del código. Es la organización fundamental de un sistema representada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución. Para que un sistema sea extensible y reusable, tiene que estar diseñado en base a esto. También se puede ver como una vista estructural de alto nivel que ocurre tempranamente en el ciclo y define los estilos o grupo de estilos adecuados para cumplir los requerimientos no funcionales. (15)

Con el objetivo de obtener una solución más confiable, se utilizaron diferentes patrones en la realización del diseño para el plug-in a desarrollar. Un patrón; es una solución a un problema en un contexto, que codifica conocimiento específico acumulado por la experiencia de un dominio. Dichos patrones se mencionan a continuación.

3.2 Patrón arquitectónico Modelo-Vista-Controlador

Para el desarrollo de la aplicación propuesta se utilizó el patrón arquitectónico MVC (*Model-View-Controller*). Éste patrón expresa un esquema organizativo estructural fundamental para sistemas software. Además, especifica un conjunto predefinido de subsistemas con sus responsabilidades y una serie de sugerencias para organizar los distintos componentes. También, separa los datos de una aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos. Indica que se deben establecer tres componentes en la arquitectura, donde cada una de ellas solo se relaciona con la adyacente. (Fig. 3.1) (15)

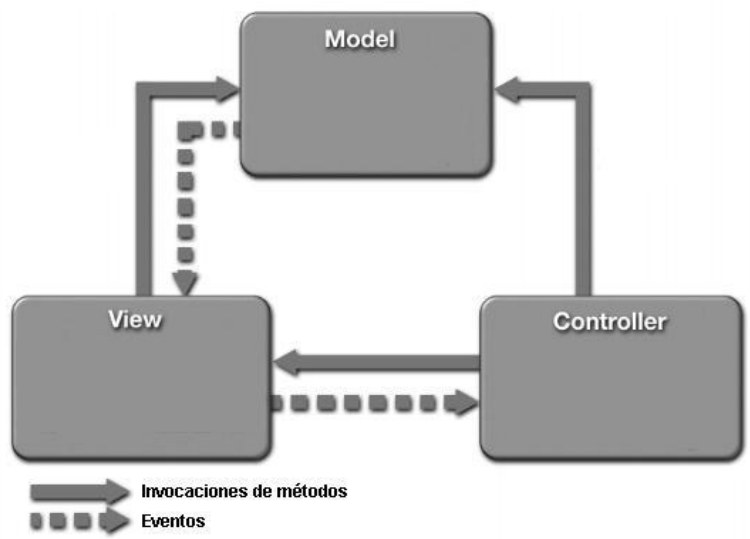


Figura 3.1. Patrón Modelo Vista Controlador

Modelo: es la representación específica de la información con la cual el sistema opera. La lógica de los datos asegura la integridad de éstos y permite derivar nuevos datos. Encapsula los datos y las funcionalidades. Accede a los datos persistentes, a la capa de almacenamiento de datos.

Vista: presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario. Muestra la información al usuario. Pueden existir varias vistas del modelo, cada una asociada a un componente controlador.

Controlador: responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista.

Algunas de las ventajas que presenta este patrón es que soporta múltiples vistas, puesto que la vista se separa del modelo y no hay ninguna dependencia directa entre vista y modelo, es por esto que la interfaz de usuario puede mostrar múltiples vistas de los mismos datos al mismo tiempo. Otra ventaja es que tiene un mayor soporte a los cambios. (15)

3.3 Patrones de diseño de software

Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referente al diseño de interacción o interfaces. Para que esta solución sea factible el patrón debe haber comprobado su efectividad resolviendo problemas similares en ocasiones anteriores y ser reusables, lo que significa que se puede aplicar a diferentes problemas de diseño en diferentes circunstancias.

Para el diseño de la aplicación se utilizaron varios patrones que se describen a continuación:

3.3.1 Patrones GRASP

Los patrones GRASP (*General Responsibility Assignment Software Patterns* o Patrones Generales de Software para Asignar Responsabilidades), describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. (15)

Experto: Se encarga de asignar una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir dicha responsabilidad.

El patrón se evidencia en la clase ***CalculoDescriptoresPluginUI***, puesto que posee un objeto ***GestionarDescriptores*** que contiene las funcionalidades necesarias para la gestión de los cálculos de descriptores. (Fig. 3.2)

Figura 3.2. Aplicación del patrón Experto en la clase ClaculoDescriptoresPluginUI de la herramienta propuesta

Creador: Asigna a la clase A la responsabilidad de crear una instancia de la clase B, si:

A agrega los objetos B

A contiene los objetos B

A registra las instancias de los objetos B

La clase ***GestionLan*** contiene un objeto ***ReceiveCalcDesc*** y un objeto ***ReceiveXMLStringDescriptors***, por ello, el patrón Creador sugieres que ***GestionLan*** es idónea para asumir la responsabilidad de crear las instancias de las clases ***ReceiveCalcDesc*** y ***ReceiveXMLStringDescriptors***. (Fig. 3.3)

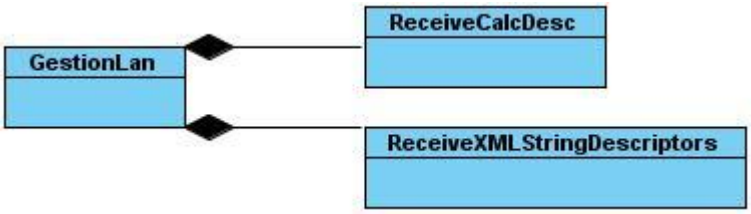


Figura 3.3. Aplicación del patrón Creador en la clase GestionLan de la herramienta propuesta

Controlador: Asigna la responsabilidad del manejo de un mensaje de los eventos del sistema a una clase. Normalmente un controlador debería delegar a otros objetos el trabajo que se realiza mientras coordina la actividad.

El patrón Controlador se evidencia en la clase ***GestionarDescriptores*** pues al poseer un objeto de la clase ***PluginSql*** y otro objeto de la clase ***ICalculo***, tiene la responsabilidad de controlar la ejecución de la lógica del sistema referente al cálculo de descriptores y al almacenamiento en la base de datos. (Fig. 3.4)

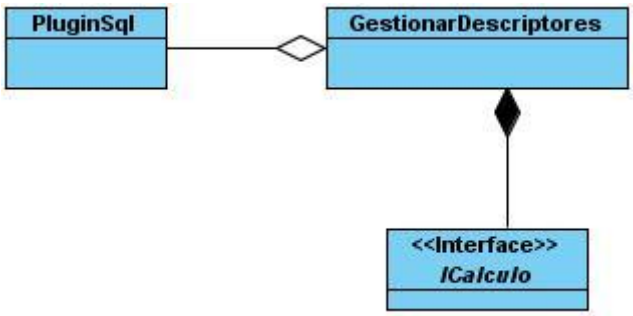


Figura 3.4. Aplicación del patrón Controlador en la clase GestionarDescriptores de la herramienta propuesta

3.3.2 Patrones GOF

El catálogo más famoso es el contenido en el libro *"Design Patterns: Elements of Reusable Object-Oriented Software",* también conocido como LIBRO GOF (*Gang-Of-Four Book*).

Según el libro GOF, existen tres tipos de patrones:

- **De Creación:** son los que abstraen el proceso de creación de instancias.
- **Estructurales:** se ocupan de cómo clases y objetos son utilizados para componer estructuras de mayor tamaño.
- **De Comportamiento:** atañen a los algoritmos y a la asignación de responsabilidades entre objetos. (15)

Para el diseño de la aplicación propuesta se utilizaron los siguientes patrones GOF:

***Singleton* (Instancia única):** es un patrón creacional que garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia.

Este patrón se evidencia en la clase ***PluginSql*** puesto que se requiere de una única instancia de esta clase para almacenar los resultados de los cálculos en la base de datos. (Fig. 3.5)

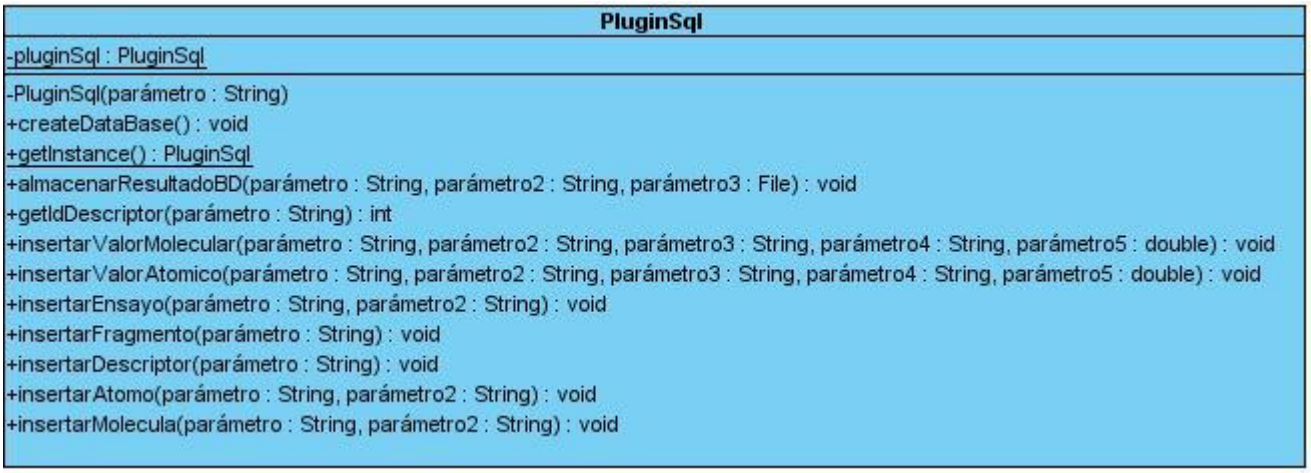


Fig. 3.5 Aplicación del patrón Singleton en la clase PluginSql de la aplicación propuesta

***Template Method* (Método plantilla):** es un patrón de comportamiento que define en una operación el esqueleto de un algoritmo, delegando en las subclases algunos de sus pasos, esto permite que las subclases redefinan ciertos pasos de un algoritmo sin cambiar su estructura.

El patrón Template Method se evidencia en la clase ***Sqlite***, puesto que la clase ***PluginSql*** hereda todas sus funcionalidades. (Fig. 3.6)

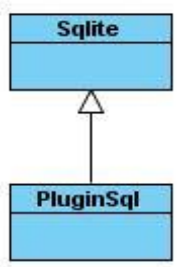


Figura 3.6. Aplicación del patrón Template Method en la clase Sqlite de la aplicación propuesta

3.4 Modelo de Diseño

El modelo de diseño es no genérico, específico para una implementación. Dinámico (centrado en las secuencias). Es un manifiesto del diseño del sistema, incluyendo su arquitectura. Debe mantenerse durante todo el ciclo de vida del software. Además, da forma al sistema, mientras intenta preservar la estructura definida por el modelo de análisis. (16)

3.4.1 Diagramas de clases del diseño

Un diagrama de clases es un tipo de diagrama estático que describe la estructura de un sistema y muestra sus clases y las relaciones entre ellas. Se utilizan durante el proceso de análisis y diseño, cuando se crea el modelo conceptual de la información que se manejará en el sistema. (16)

En el diagrama de clases del plug-in Cálculo de Descriptores (Fig. 3.7), todas las clases de la aplicación están agrupadas por paquetes, siguiendo las especificaciones del patrón arquitectónico MVC, lo que posibilita una mejor interacción entre las clases visuales y los modelos de datos.

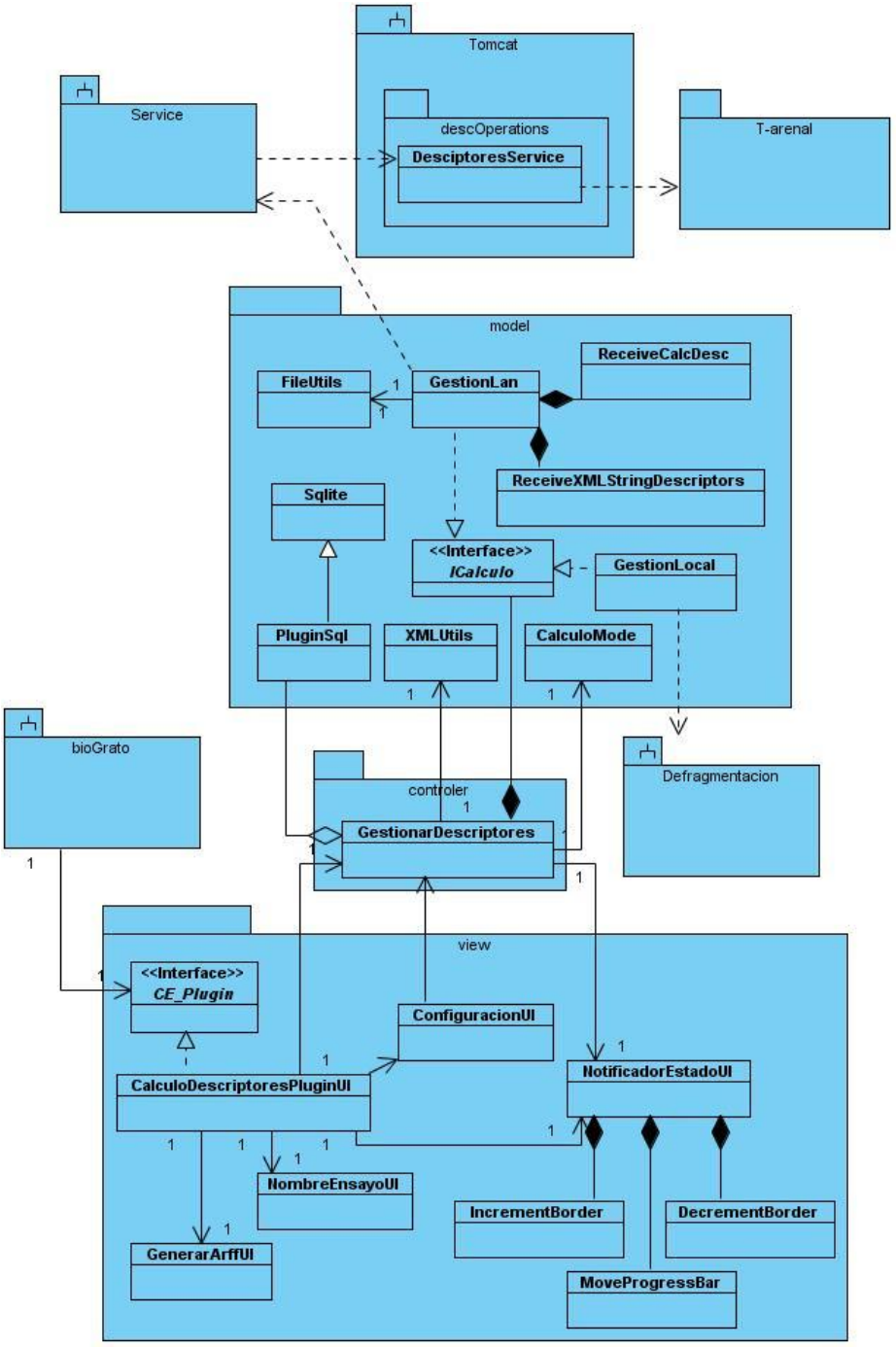


Figura 3.7. Diagrama de clases para el plug-in Cálculo de Descriptores

3.4.2 Descripción de las clases principales del diseño para el plug-in Cálculo de Descriptores

Las clases principales de la aplicación están divididas en paquetes para lograr una mejor representación de éstas y sus relaciones. En el paquete ***model***, se encuentran las clases principales ***GestionLan***, ***GestionLocal*** y ***PluginSql*** que son las que poseen la información con la cual el sistema opera. En el paquete ***controler***, la clase principal es ***GestionarDescriptores***, pues ésta, responde a los eventos o acciones del usuario e invoca cambios en el modelo y en la vista. Por último, en el paquete ***view***, se tiene ***ConfiguracionUI*** y ***CalculoDescriptoresPluginUI***, éstas se encargan de presentar el modelo en un formato adecuado para interactuar, es decir, brindan información al usuario.

La clase ***GestionarDescriptores*** es la clase controladora de la aplicación, la que realiza el cálculo de descriptores y almacena los resultados en la base de datos a través de la clase ***PluginSql***. (Tabla 3.1). Para ver las descripciones de las restantes clases principales de la aplicación, ver (Anexo 2).

Tabla 3.1 Descripción de la clase controladora GestionarDescriptores del plug-in Cálculo de Descriptores

Nombre: GestionarDescriptores	
Tipo de clase: Controladora	
Atributos:	**Tipo:**
configFile	File
PROPERTIES_SAVE	Properties
WORK_DIRECTORY	File
JAR_DESCRIPTOR_FILE	File
XML_FILE_DESCRIPTORS	File
calculoMode	ICalculo
Principales Responsabilidades:	
Nombre:	public File calculateDescriptors(ArrayList<String> fragments, ArrayList<String> files, ArrayList<Integer> descriptors)
Descripción:	Realiza el cálculo de descriptores.

3.4.3 Diagramas de interacción

Se trata de un término genérico que se aplica a varios tipos de diagramas que hacen hincapié en las interacciones entre objetos. Se utilizan no solo para modelar los aspectos dinámicos de un sistema, sino también para construir sistemas ejecutables por medio de ingeniería directa e inversa. Los diagramas de interacción tienen diferentes formas, basadas en una misma información subyacente pero resaltando cada uno en un punto de vista de la misma: diagramas de colaboración, diagramas de secuencia. (16)

Diagramas de secuencias para cada CU del plug-in Cálculo de Descriptores

Un diagrama de secuencia muestra los objetos que participan en la interacción mediante sus líneas de vida y mediante los mensajes que intercambian, organizados en forma de una secuencia temporal. No muestra los enlaces existentes entre objetos. Los diagramas de secuencia tienen distintos formatos, adecuados para propósitos diferentes. (16)

A continuación se muestran los diagramas de secuencias de para los escenarios Configurar Cálculo Local (Fig. 3.8) y Configurar Cálculo Distribuido (Fig. 3.9), del CU Configurar Cálculo. Para ver los restantes diagramas, Diagrama de secuencia del CU Realizar Cálculo de Descriptores, ver (Anexo 3), Diagrama de secuencia del CU Generar Fichero ARFF, ver (Anexo 4).

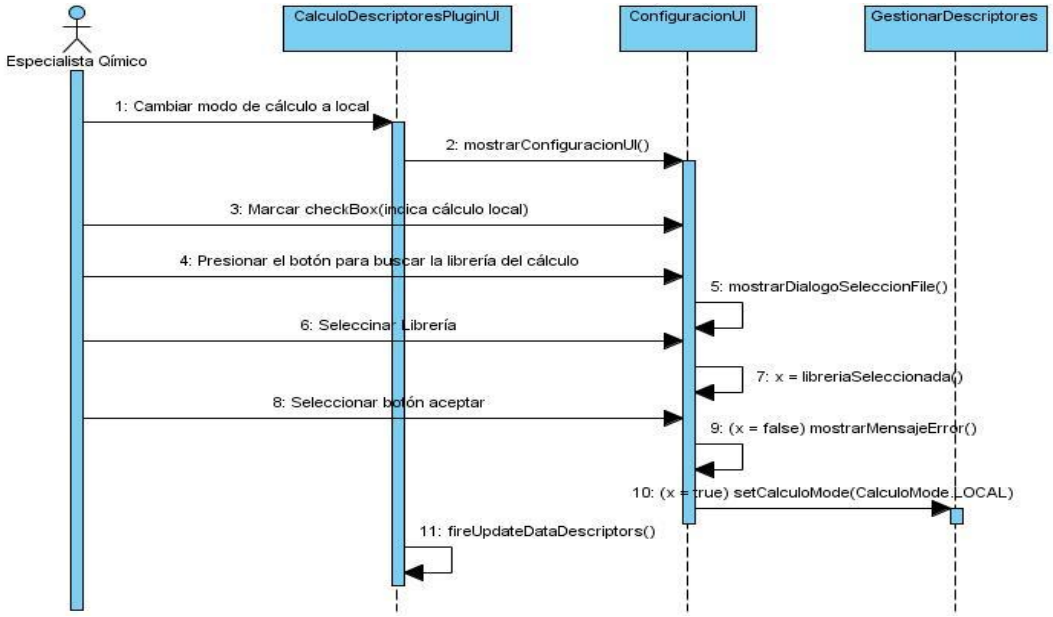


Figura 3.8. Diagrama de secuencia para el escenario Configurar Cálculo Local del CU Configurar Cálculo

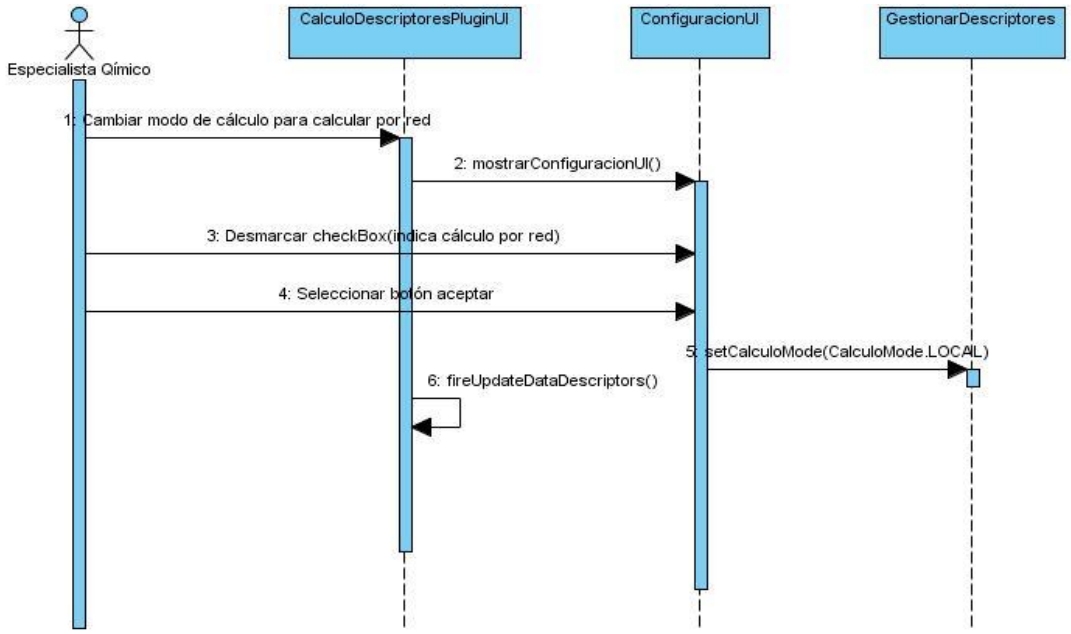


Figura 3.9. Diagrama de secuencia para el escenario Configurar Cálculo Distribuido del CU Configurar Cálculo

3.5 Diseño de la base de datos local de la herramienta propuesta

Para realizar el diseño de la base de datos de la aplicación se realizó el modelo físico de datos (modelo entidad-relación).

"... el modelo ER puede ser usado como una base para una vista unificada de los datos, adoptando el enfoque más natural del mundo real que consiste en entidades e interrelaciones…"[2]

El modelo entidad-relación se basa en una percepción del mundo real que consta de un conjunto de objetos básicos llamados entidades con sus atributos y de las relaciones que existen entre estos objetos. (3)

Modelo físico de la base de datos local implementada

La base de datos consta de 8 tablas, ***molecula***, ***ensayo***, ***atomos***, ***valores_moleculares***, ***valores_atomico***, ***descriptores***, ***fragmentos*** y ***descriptores_fragmentos***, que se relacionan entre ellas

[2] Peter Pen. Chen 1976

como se muestra en la (Fig. 3.10). De esta forma el especialista puede tener almacenados todos los resultados de los cálculos.

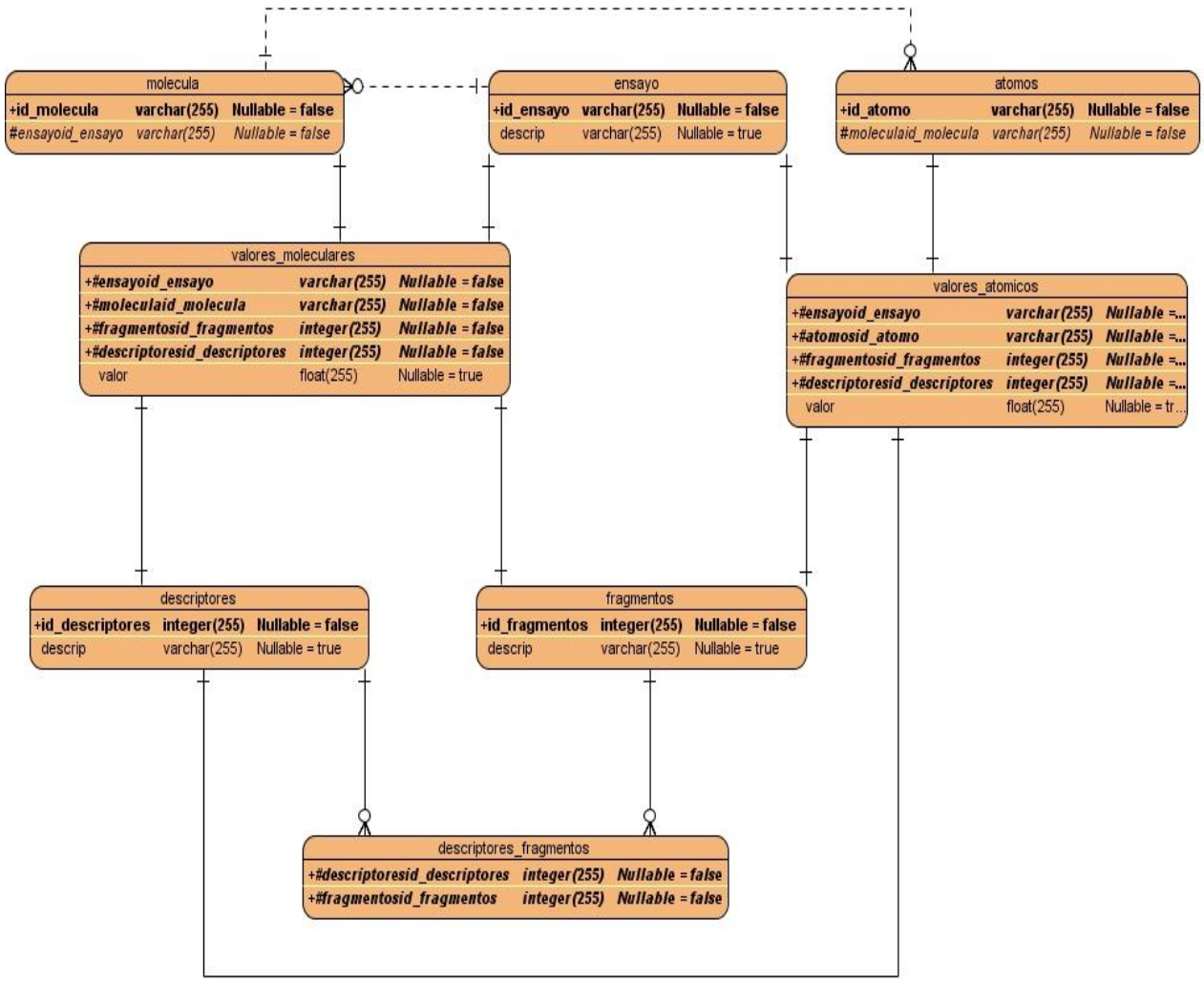


Figura 3.10. Diagrama entidad-relación de la base de datos local del plug-in Cálculo de Descriptores

3.6 Modelo de despliegue

El modelo de despliegue, describe la arquitectura física del sistema durante la ejecución, en términos de procesadores, dispositivos y componentes de software. Describe, además, la topología del sistema, es decir, la estructura de los elementos de hardware y software que ejecuta cada uno de ellos. (14)

Para la aplicación desarrollada se cuenta con la ***PC cliente*** que es donde el especialista químico podrá ejecutar el plug-in Cálculo de Descriptores y realizar los cálculos ya sea de forma local o a través del servidor de aplicaciones, en dependencia de cómo configure el plug-in. La comunicación con el ***Servidor de aplicaciones*** se realiza a través del protocolo SOAP, en éste se instalará el servidor de aplicaciones Apache Tomcat, donde se alojarán los servicios web que tendrá la plataforma para la comunicación con el ***Servidor de cómputo distribuido*** que tendrá desplegada la aplicación T-arenal para realizar los cálculos de forma distribuida, utilizando RMI (*Remote Method Invocation* o Método de Invocación Remota). (Fig. 3.11)

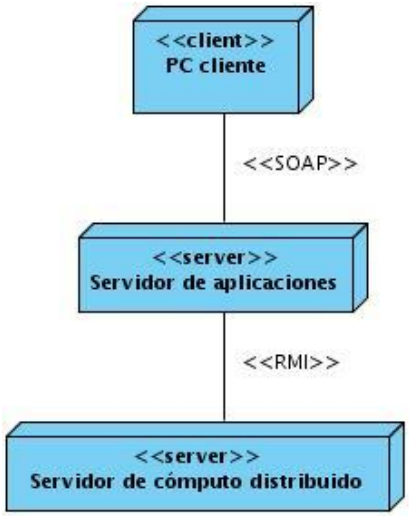


Figura 3.11. Diagrama de despliegue para el plug-in Cálculo de Descriptores

3.7 Conclusiones Parciales

Para el desarrollo de la herramienta se describió el estilo arquitectónico utilizado mediante las especificaciones del patrón Modelo Vista Controlador. Para el diseño de ésta, se utilizaron los patrones de diseño GRASP y los GOF. Se realizó el diagrama de clases del diseño, así como los diagramas de secuencia por cada caso de uso del sistema. Se realizó el modelo de datos para la base de datos local implementada, lo cual permitió definir la cantidad de tablas con que cuenta, así como las relaciones entre ellas.

CAPÍTULO IV

IMPLEMENTACIÓN Y PRUEBA DEL SISTEMA

En este capítulo se describe la implementación del sistema, donde se muestra el diagrama de componentes del plug-in implementado, teniendo en cuenta las bases creadas por el flujo de trabajo de diseño, desarrollado anteriormente. Además, se describen detalladamente las pruebas que se le realizaron al sistema.

4.1 Modelo de implementación

El Modelo de implementación es una visión general de lo que se debe implementar y un apartado para cada iteración, que coincide con la plantilla del plan de integración de constructos, con los componentes y subsistemas a implementar durante esa iteración, así como con los resultados que se deben obtener y las pruebas que se deben realizar sobre ellos. (14)

4.1.1 Diagrama de componentes

Un diagrama de componentes es la representación de la forma en que los componentes físicos de un sistema serán separados, pueden ser: archivos, cabeceras, módulos, paquetes. Muestra la organización y las dependencias que existen entre un conjunto de componentes. Además, se utilizan para modelar la vista estática de un sistema. (14)

En el diagrama de componentes del plug-in Cálculo de Descriptores (Fig. 4.1), se muestran los componentes separados por paquetes, según los requerimientos del patrón MVC y según la función que cumple cada uno de éstos.

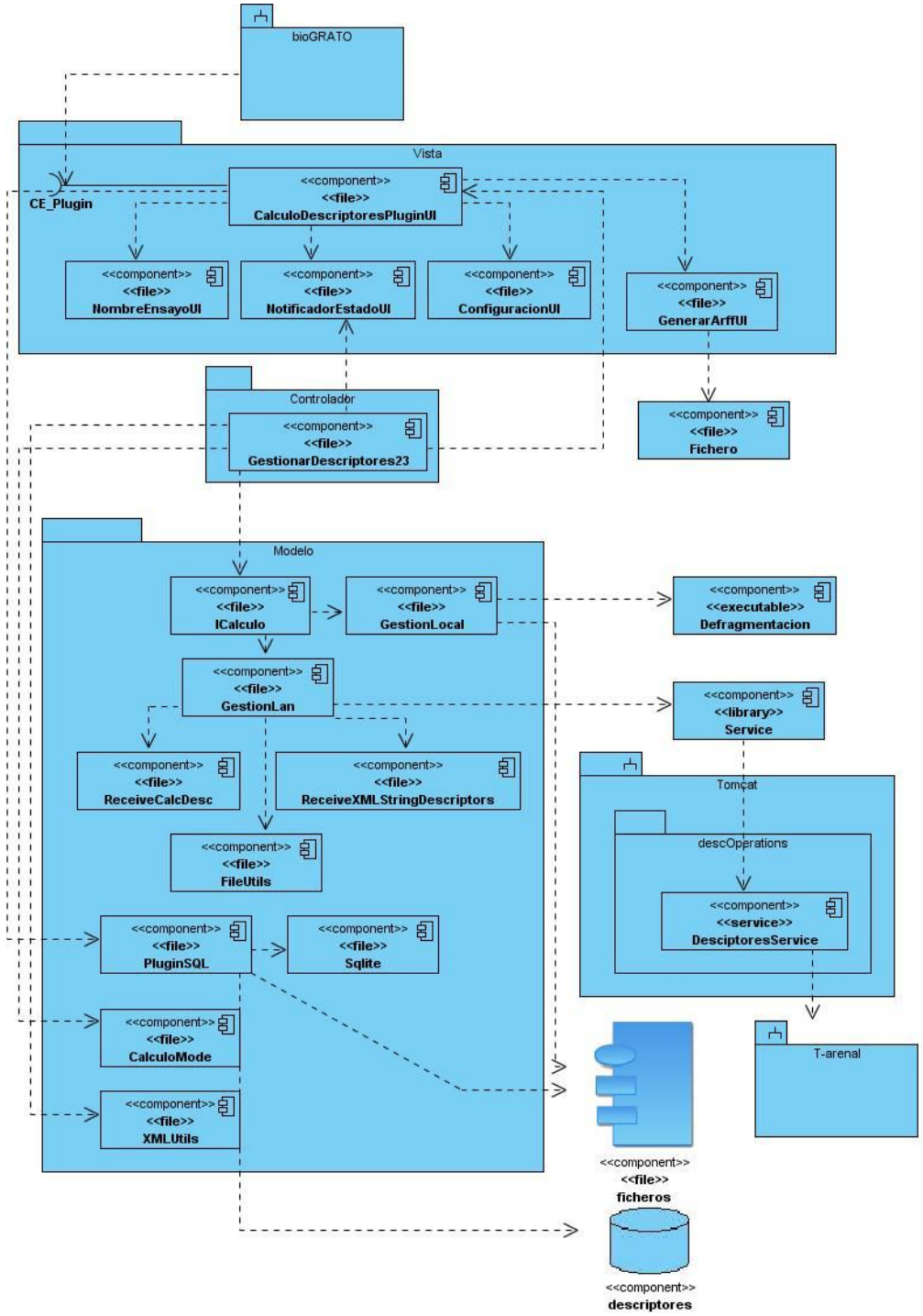


Figura 4.1. Diagrama de componentes del plug-in Cálculo de Descriptores

4.1.2 Integración del plug-in Cálculo de Descriptores a alasGRATO y su funcionamiento

Para integrar el plug-in Cálculo de Descriptores a alasGRATO, el especialista debe ejecutar la plataforma (Fig. 4.2 a), luego selecciona el menú Preferencias que le permitirá añadir el módulo (Fig. 4.2 b). Selecciona la opción Adicionar Módulos y le aparecerá una interfaz Localizador de Extensiones para que el especialista busque el lugar donde se encuentra guardado el plug-in (Fig. 4.2 c) y lo importa (Fig. 4.2 d). La aplicación muestra un mensaje que ratifica que la inserción se realizó con éxito (Fig. 4.2 e). Para trabajar con el módulo, el especialista selecciona, nuevamente, el menú Preferencias y lo ejecuta (Fig. 4.2 f).

Una vez que se ejecuta el plug-in, éste muestra las moléculas que el Especialista Químico seleccionó con anterioridad en la plataforma, los descriptores disponibles y los fragmentos (Anexo 5). El Especialista Químico selecciona, la estructura química, además de algunos descriptores y fragmentos (Anexo 6). Presiona el botón Calcular y aparece una barra de estado que indica la operación que se está realizando. Una vez terminado el cálculo el plug-in muestra una interfaz para introducir el nombre y un comentario sobre el ensayo, que le permita reconocerlo posteriormente (Anexo 7) y notifica que lo está salvando en la base de datos local (Anexo 8).

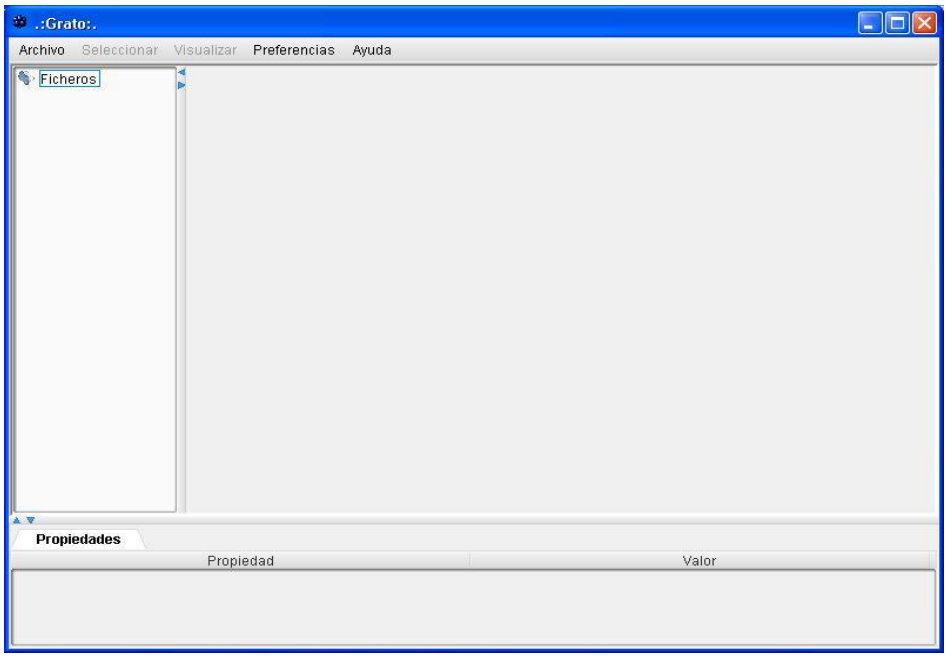


Figura 4.2 a. Interfaz principal de la plataforma alasGRATO sin ningún plug-in incorporado

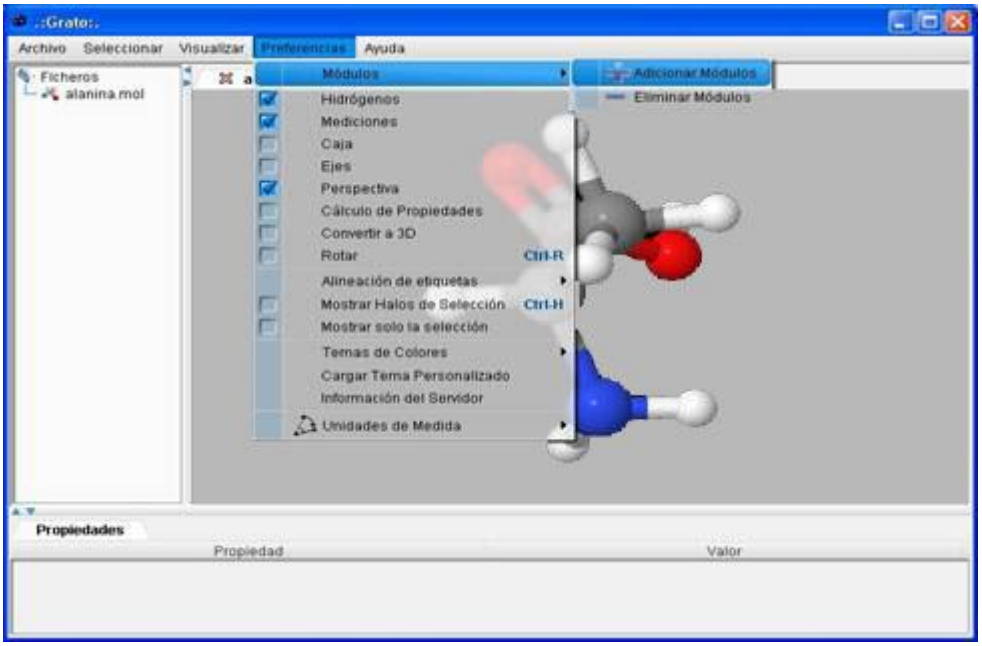


Figura 4.2 b. Opción Adicionar Módulo

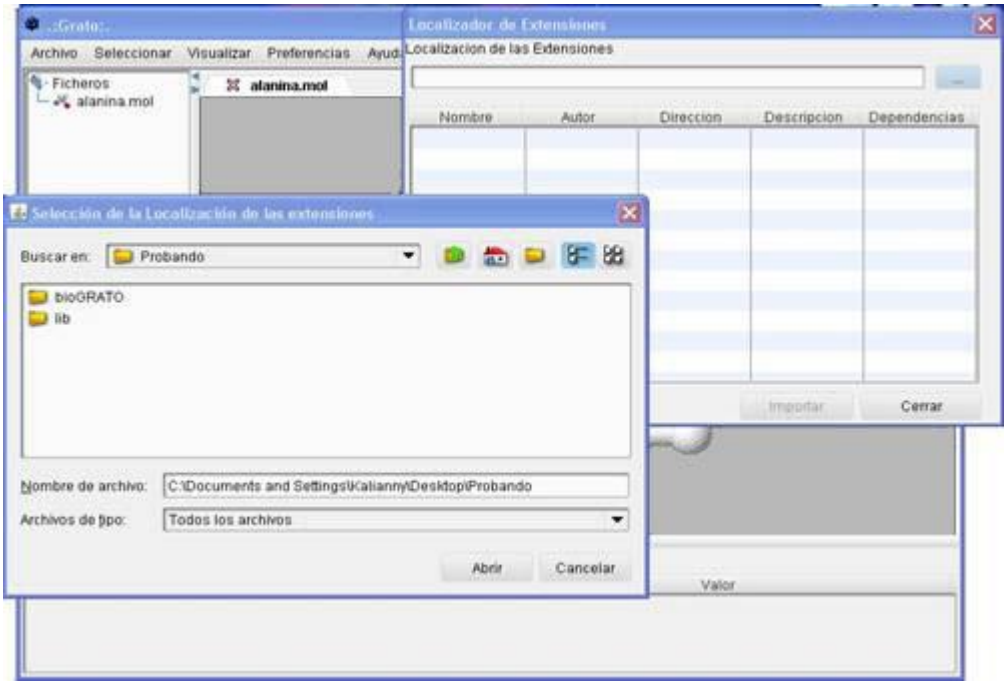


Fig. 4.2 c. Interfaz que permite buscar el plug-in

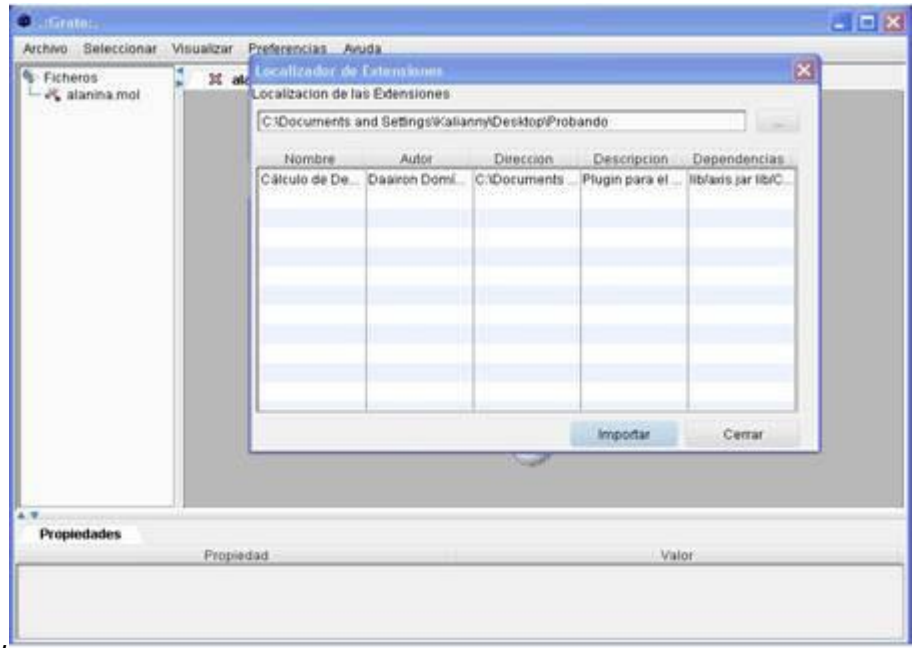


Fig. 4.2 d. Interfaz que permite Importar el plug-in

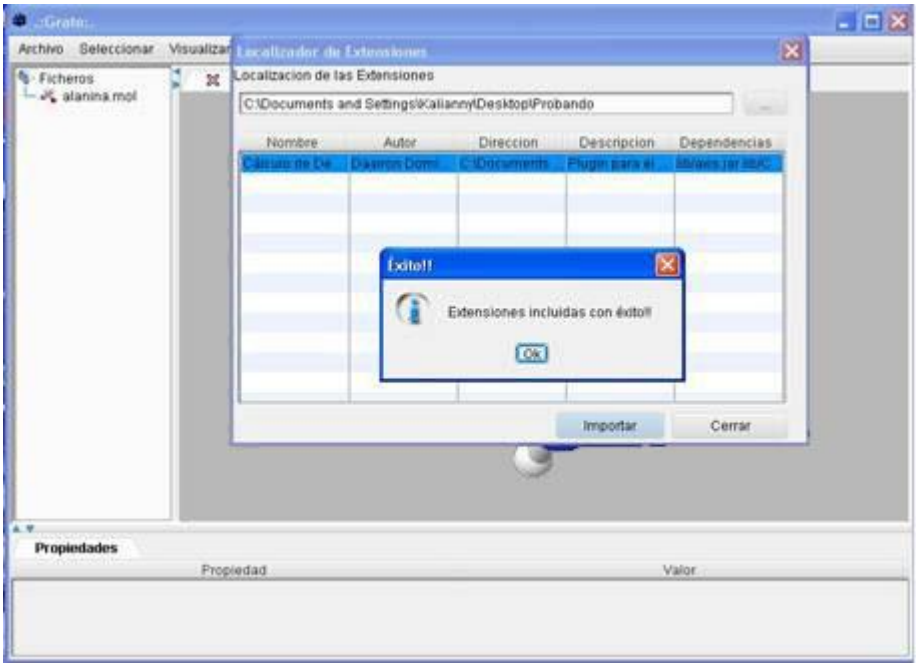


Fig. 4.2 e. Mensaje que ratifica la inserción

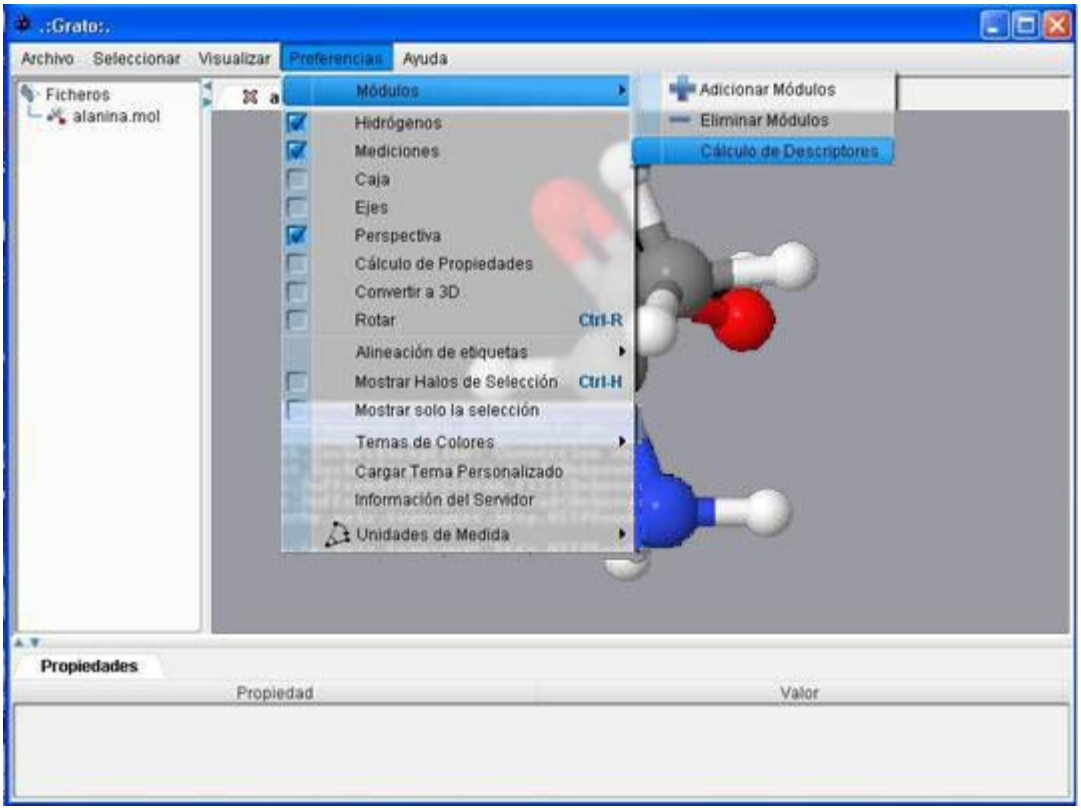


Fig. 4.2 f. Verificar que el plug-in se integró a la plataforma

4.2 Pruebas

El objetivo principal de las pruebas de software es descubrir errores. Para conseguir este objetivo se planifica y se ejecuta una serie de pasos que revisan todos los elementos del software. En la etapa de prueba se refleja la calidad con que se lleva a cabo la proyección del sistema. Existen dos tipos de métodos de prueba, las pruebas de caja blanca y las pruebas de caja negra.

Las pruebas de caja negra son aquellas que se llevan a cabo sobre la interfaz del software, por lo que los casos de prueba tienen como objetivo demostrar que las funcionalidades de la aplicación son operativas, que los datos de entrada se acepten correctamente y que se produce una salida adecuada que garantiza la integridad de la información que se almacena y procesa. (17)

4.2.1 Plan de prueba

Un plan de prueba se constituye por varias pruebas. Cada una de éstas debe especificar que tipo de propiedades quiere probar, cómo se mide el resultado, especificar en que consiste la prueba y definir cuál es el resultado que ese espera.

4.2.2 Configuración del entorno de prueba

La configuración del entorno de prueba es un aspecto muy importante que se debe tener en cuenta a la hora de realizar las pruebas al producto, pues hay que analizar bien los recursos de software y hardware del entorno donde se realizarán las pruebas para que, en el momento de probarlo, cuente con los elementos necesarios que garanticen el éxito del proceso.

Para configurar el entorno de prueba del plug-in Cálculo de Descriptores, se realizó el Plan de Prueba (Tabla 4.1) y se diseñaron las pruebas para los CU, Configurar Cálculo (Tabla 4.2), Realizar Cálculo de Descriptores (Tabla 4.3) y Generar Fichero ARFF (Tabla 4.4).

Requerimientos de Software

- Se debe disponer de sistemas operativos Linux, Windows 95 o superior para la instalación de la aplicación.
- Debe contarse con el *Java Runtime Environment* (JRE) versión 1.5 o superior instalado.

Requerimientos de Hardware

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

Tabla 4.1: Plan de pruebas para el plug-in Cálculo de Descriptores

Fecha de Inicio	Fecha de Fin	Actividades	Personal Implicado
14/04/2010	14/04/2010	Aceptación y firma del Plan de Prueba.	Dairon Domínguez Vega

			Ludmila Campos García Alexis R. Rodríguez León
15/04/2010	16/04/2010	Verificación de las condiciones previas para el inicio de las pruebas.	Dairon Domínguez Vega Ludmila Campos García

Primera iteración del Plug-in Cálculo de descriptores

20/04/2010	20/04/2010	Ejecución de las pruebas de caja negra al caso de uso Realizar Cálculo de Descriptores.	Dairon Domínguez Vega Ludmila Campos García
21/04/2010	21/04/2010	Ejecución de las pruebas de caja negra al caso de uso Configurar Cálculo.	Dairon Domínguez Vega Ludmila Campos García
22/04/2010	22/04/2010	Ejecución de las pruebas de caja negra al caso de uso Generar Fichero ARFF.	Dairon Domínguez Vega Ludmila Campos García
23/04/2010	23/04/2010	Análisis de las no conformidades y las solicitudes de cambio.	Dairon Domínguez Vega Ludmila Campos García Alexis R. Rodríguez León

4.2.3 Diseño de Pruebas de Caja Negra para el plug-in Cálculo de Descriptores

Caso de Uso Configurar Cálculo

Secciones a probar en el caso de uso

Tabla 4.2: Diseño de prueba para la sección Configurar cálculo local del CU Configurar Cálculo

Nombre de la Sección	Escenario de la Sección	Descripción de la funcionalidad	Flujo Central
		El Especialista Químico selecciona la opción Configurar del menú Configuración. Aparece una ventana	Oprime el botón

SC 1: Configurar cálculo	EC 1.1 Configurar cálculo local.	con una opción para seleccionar Cálculo Local. El sistema ofrece la posibilidad de buscar la librería para el cálculo. El Especialista Químico selecciona la librería y oprime el botón Aceptar. El sistema valida los datos y cambia la configuración a local.	Aceptar.

El diseño de las pruebas de la sección definida anteriormente se encuentra en el anexo 9

Caso de Uso Realizar Cálculo de Descriptores

Secciones a probar en el caso de uso

Tabla 4.3: Diseño de pruebas para cada escenario del CU Realizar Cálculo de Descriptores

Nombre de la Sección	Escenario de la Sección	Descripción de la funcionalidad	Flujo Central
SC 1: Calcular descriptores	EC 1.1 Calcular descriptores correctamente.	El Especialista Químico selecciona las estructuras químicas, los descriptores, los fragmentos y oprime el botón Calcular. El sistema valida los datos, realiza el cálculo y guarda los resultados en la base de datos local.	Oprime el botón Calcular.
	EC 1.2 Calcular descriptores incorrectamente.	El Especialista Químico selecciona las estructuras químicas, los descriptores, los fragmentos y oprime el botón Calcular. El sistema valida los datos, si no ha seleccionado al menos 1 de cada tipo, muestra un mensaje de error.	Oprime el botón Calcular.

El diseño de las pruebas de la sección definida anteriormente se encuentra en el anexo 10

Caso de Uso Generar Fichero ARFF

Secciones a probar en el caso de uso

Tabla 4.4: Diseño de pruebas para el CU Generar Fichero ARFF

Nombre de la Sección	Escenario de la Sección	Descripción de la funcionalidad	Flujo Central
SC 1: Generar Fichero ARFF	EC 1.1 Generar fichero correctamente	El Especialista Químico selecciona la opción Generar Fichero ARFF del menú Generar. El sistema muestra los resultados de los cálculos almacenados en la base de datos local y convierte a ARFF el seleccionado por el especialista.	Oprime el botón Generar.

El diseño de las pruebas de la sección definida anteriormente se encuentra en el anexo 11

4.2.4 No conformidades detectadas

Tabla 4.4: No conformidades detectadas

Elemento	No.	No Conformidad	Aspecto Correspondiente	Etapa de Detección	Signif.	No Signif.
Interfaz	1	No se actualizaba el árbol de moléculas.	Cuando cargaba desde el Visualizador.	Etapa de diseño y aplicación de pruebas al plug-in implementado.	X	
Interfaz	2	Calculaba sin haber seleccionado estructuras químicas.	Cuando se presionaba el botón Calcular.	Etapa de diseño y aplicación de pruebas al plug-in implementado.	X	
Interfaz	3	Generaba error y no convertía a ARFF.	Cuando se presionaba el botón Generar.	Etapa de diseño y aplicación de pruebas al plug-in implementado.	X	

[Signif. indica Significativo, No Signif. indica No Significativo]

4.3 Conclusiones Parciales

Como resultado de la implementación, se logró integrar el plug-in para el módulo Cálculo de Descriptores en alasGRATO, de manera satisfactoria. Se realizaron pruebas de caja negra al plug-in y se comprobó que sus funcionalidades son operativas, aunque se detectaron tres no conformidades que fueron registradas y solucionadas para mejorar el funcionamiento de la herramienta.

CONCLUSIONES GENERALES

- Se implementó un servicio web para realizar el cálculo de los descriptores en la plataforma de cómputo distribuido que posibilita obtener los resultados de forma más rápida.
- Se implementó un servicio web para obtener los descriptores que utiliza el especialista para realizar los cálculos.
- Se diseñó una base de datos local que permite al especialista guardar los resultados de sus cálculos, los que podrá utilizar en estudios posteriores.
- Se implementó un plug-in para el módulo Cálculo de Descriptores que logra integrarse satisfactoriamente en la plataforma alasGRATO.
- Se comprobó el funcionamiento del plug-in a partir de las pruebas de caja negra, realizadas mediante los casos de prueba.

RECOMENDACIONES

- Implementar una funcionalidad en el servicio web que calcula descriptores que verifique, si los descriptores de las estructuras químicas a calcular, han sido almacenados en la base de datos del proyecto alasGRATO con anterioridad.

REFERENCIAS BIBLIOGRÁFICAS

1. **Rodríguez León, Alexis René.** *Herramienta para el Cálculo de Descriptores Atómicos, Moleculares y para Aminoácidos*. Tesis de Implementación. Universidad de las Ciencias Informáticas. Ciudad Habana : s.n., 2010.

2. **Pedersen, J. K.** [En línea] [Citado el: 23 de Febrero de 2010.] Disponible en: http://developer.kde.org/documentation/tutorials/developing-a-plugin-structure/index_es.html.

3. **J., Date C.** Introducción a los sistemas de bases de datos. La Habana : Félix Varela, 2003.

4. Web Services. [En línea] [Citado el: 24 de Enero de 2010.] Disponible en: http://delta.icc.es/idecwebservices/indexcas.html.

5. Open Up. [En línea] [Citado el: 25 de Enero de 2010.] Disponible en: http://epf.eclipse.org/wikis/openup.

6. UML. [En línea] [Citado el: 16 de Febrero de 2010.] Disponible en: http://www.uml.org.

7. UML CASE Tools -Free for Learning UML. Visual Paradigm. [En línea] [Citado el: 16 de Febrero de 2010.] Disponible en: http://www.visual-paradigm.com.

8. Tutoriales de java. Introducción a AWT. [En línea] [Citado el: 7 de Marzo de 2010.] http://www.ac.uma/localres/manuals.

9. **ABIAN, M. Á.** El archipiélago ECLIPSE. [En línea] [Citado el: 17 de Febrero de 2010.] Disponible en: http://www.javahispano.org/articles.article.action?id=81.

10. SQLite. [En línea] [Citado el: 15 de Enero de 2010.] Disponible en: http://www.sqlite.org.

11. **Villaverde Martínez, Julio.** *Un nuevo Front-End para la plataforma alasGRATO*. Tesis de Implementación. Universidad de las Ciencias Informáticas. Ciudad Habana : s.n., 2009.

12. **Aguilera Mendoza, Longendri.** *Sistema de cómputo distribuido aplicado a la Bioinformática*. Tesis de Implementación. Universidad de las Ciencias Informáticas. Ciudad Habana : s.n., 2008.

13. Telemática, Revista digital de la tecnología de la información y las comunicaciones. 2008, Vol. Artículo18: Infraestructura Grid para integrar varias plataformas Tarenal. 1729-3804. [En línea] [Citado el: 14 de Abril de 2010.] Disponible en: http://www.cujae.edu.cu/revistas/telematica/Publicaciones.aspx?ci=9

14. **BOOCH Grady, RUMBAUGH James, JACOBSON Ivar.** El lenguaje unificado de modelado. Manual de referencia. 2000. Páginas 120-121.

15. **Larman, Craig.** UML y Patrones. México : Prentice Hall, 1999. ISBN/970-1 7-0261-1.

16. **JACOBSON Ivar, RUMBAUG James, BOOCH Grady.** El proceso unificado de desarrollo de software. 2000. Páginas 285-287.

17. **JACOBSON Ivar.** L/T El proceso unificado de desarrollo de software. Capítulo 11 (Prueba) pág. 281 - 299.

BIBLIOGRAFÍA

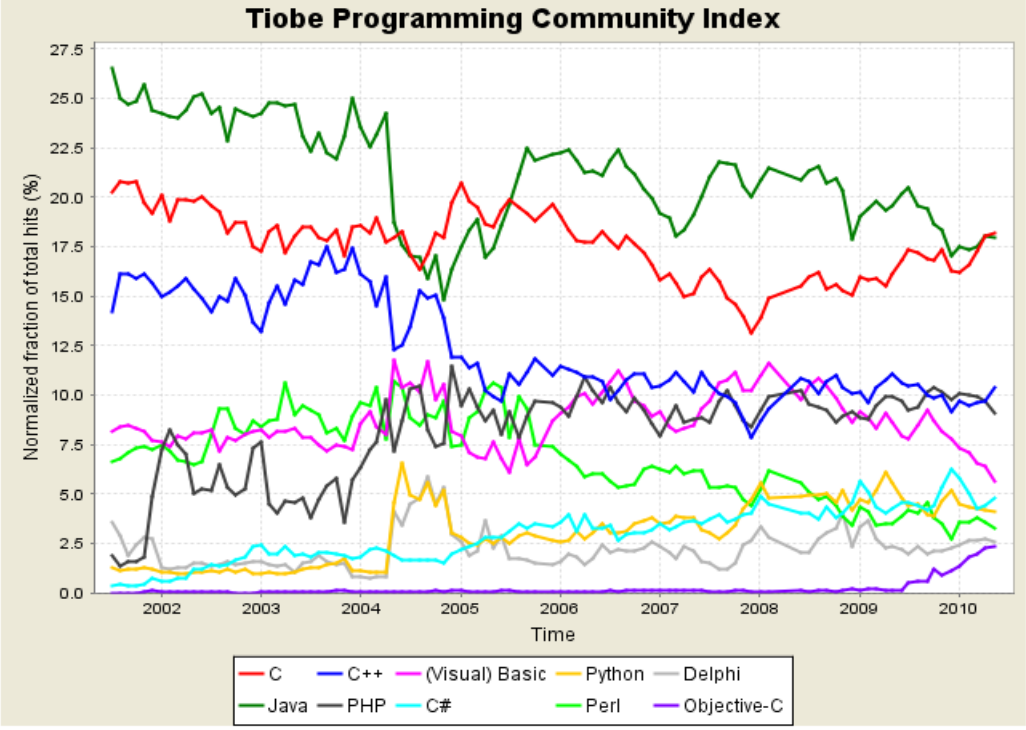
- **ABIAN, M. Á.** El archipiélago ECLIPSE. [En línea] [Citado el: 17 de Febrero de 2010.] Disponible en: http://www.javahispano.org/articles.article.action?id=81.
- **Aguilera Mendoza, Longendri.** *Sistema de cómputo distribuido aplicado a la Bioinformática*. Tesis de Implementación. Universidad de las Ciencias Informáticas. Ciudad Habana : s.n., 2008.
- **BOOCH Grady, RUMBAUGH James, JACOBSON Ivar.** El lenguaje unificado de modelado. Manual de referencia. 2000. Páginas 120-121.
- **G Couloris JD, Kinberg T.** Distributed Systems - Concepts and Design, 4th Edition. 2001.
- **J., Date C.** Introducción a los sistemas de bases de datos. La Habana : Félix Varela, 2003.
- **JACOBSON Ivar.** L/T El proceso unificado de desarrollo de software. Capítulo 11 (Prueba) pág. 281 - 299.
- **JACOBSON Ivar, RUMBAUG James, BOOCH Grady.** El proceso unificado de desarrollo de software. 2000. Páginas 285-287.
- **Larman, Craig.** UML y Patrones. México : Prentice Hall, 1999. ISBN/970-1 7-0261-1.
- **Marrero López, Yadira; Rosales García, Adonis**. Propuesta del diseño arquitectónico de la Plataforma GRAph-TOol. Tesis de arquitectura. Universidad de las Ciencias Informáticas. Ciudad Habana: s.n, 2008.
- Open Up. [En línea] [Citado el: 25 de Enero de 2010.] http://epf.eclipse.org/wikis/openup.
- **Pedersen, J. K.** [En línea] [Citado el: 23 de Febrero de 2010.] Disponible en: http://developer.kde.org/documentation/tutorials/developing-a-plugin-structure/index_es.html.
- **Pressman, Roger**, “Ingeniería de software. Un enfoque práctico.”
- **Rodríguez León, Alexis René.** *Herramienta para el Cálculo de Descriptores Atómicos, Moleculares y para Aminoácidos*. Tesis de Implementación. Universidad de las Ciencias Informáticas. Ciudad Habana : s.n., 2010.
- SQLite. [En línea] [Citado el: 15 de Enero de 2010.] Disponible en: http://www.sqlite.org.
- Telemática, Revista digital de la tecnología de la información y las comunicaciones. 2008, Vol. Artículo18: Infraestructura Grid para integrar varias plataformas Tarenal. 1729-3804. [En línea]

[Citado el: 14 de Abril de 2010.] Disponible en: http://www.cujae.edu.cu/revistas/telematica/Publicaciones.aspx?ci=9

- TIOBE Programming Community Index. [En línea] [Citado el: 23 de Febrero de 2010] Disponible en: http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html.
- Tutoriales de java. Introducción a AWT. [En línea] [Citado el: 7 de Marzo de 2010.] Disponible en: http://www.ac.uma/localres/manuals.
- UML CASE Tools -Free for Learning UML. Visual Paradigm. [En línea] [Citado el: 16 de Febrero de 2010.] Disponible en: http://www.visual-paradigm.com.
- UML. [En línea] [Citado el: 16 de Febrero de 2010.] Disponible en: http://www.uml.org.
- **Villaverde Martínez, Julio.** *Un nuevo Front-End para la plataforma alasGRATO*. Tesis de Implementación. Universidad de las Ciencias Informáticas. Ciudad Habana : s.n., 2009.
- Web Services. [En línea] [Citado el: 24 de Enero de 2010.] Disponible en: http://delta.icc.es/idecwebservices/indexcas.html.

ANEXOS

Anexo 1. Índice de la Comunidad de TIOBE que muestra los lenguajes de programación más populares



Anexo 2. Descripción de las clases principales del diseño para el plug-in Cálculo de Descriptores

Nombre: GestionLan	
Tipo de clase: Entidad	
Atributos:	**Tipo:**
Principales Responsabilidades:	
Nombre:	calculateDescriptors(ArrayList<String> fragments, ArrayList<String> files, ArrayList<Integer> descriptors)
Descripción:	Calcula los descriptores a través del servidor de aplicaciones web.
Nombre:	createXMLDescriptorFile()
Descripción:	Crea el fichero de extensión xml que contiene los descriptores y fragmentos disponibles para el cálculo a través del servidor de aplicaciones web.

Nombre: GestionLocal		
Tipo de clase: Entidad		
Atributos:		**Tipo:**
jarDesc		File
Principales Responsabilidades:		
Nombre:	calculateDescriptors(ArrayList<String> fragments, ArrayList<String> files, ArrayList<Integer> descriptors)	
Descripción:	Calcula los descriptores de forma local.	
Nombre:	createXMLDescriptorFile()	
Descripción:	Crea el fichero de extensión xml que contiene los descriptores y fragmentos disponibles para el cálculo desde la librería local.	

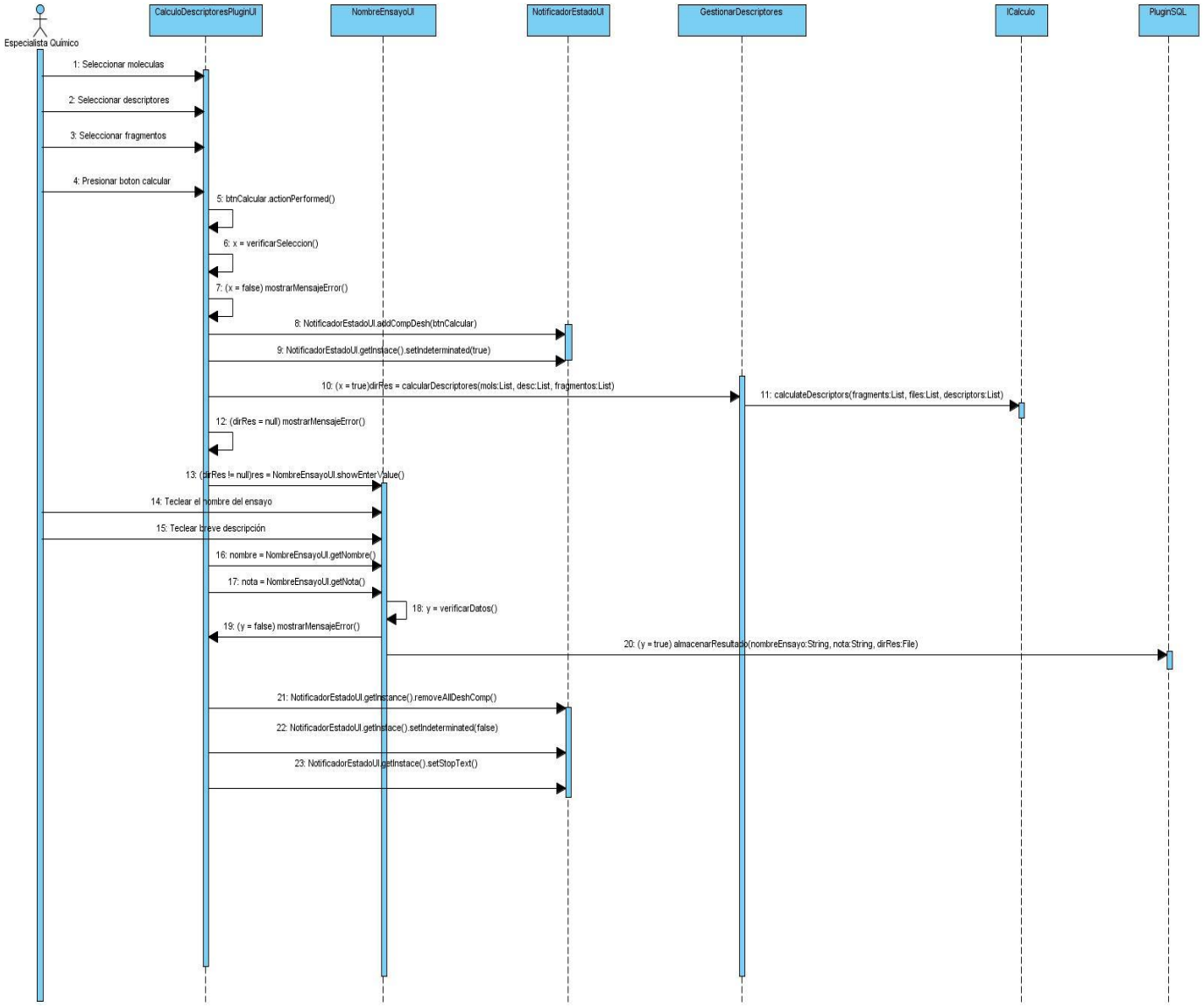
Nombre: PluginSql		
Tipo de clase: Entidad		
Atributos:		**Tipo:**
pluginSql		PluginSql
Principales Responsabilidades:		
Nombre:	public void almacenarResultadoBD(String idEnsayop, String descripcion, File xmlDir)	
Descripción:	Almacena el resultado de los cálculos en la base de datos local.	

Nombre: ConfiguracionUI	
Tipo de clase: Interfaz	
Atributos:	**Tipo:**
contentPanel	JPanel
checkBox	JCheckBox
panel_1	JPanel
label	JLabel
label_1	JLabel
label_2	JLabel
txtFieldCorina	JTextField
txtFieldDirMopac	JTextField
txtFieldDirJarDesc	JTextField
btnJarDesc	JButton
btnMopac	JButton
btnCorina	JButton
aceptPanel	JPanel
btnAceptar	JButton

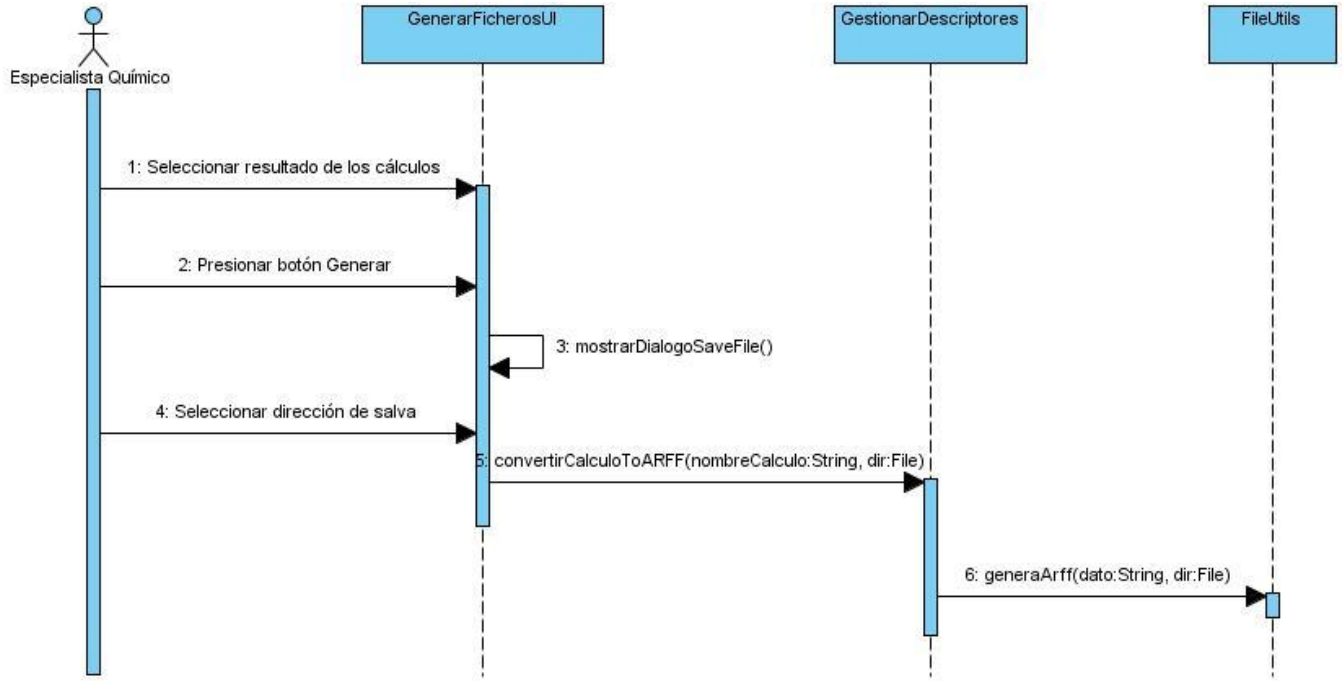
fileChooser	JFileChooser
gestionar	GestionarDescriptores
Principales Responsabilidades:	
Nombre:	setCalculoMode(CalculoMode.LOCAL)
Descripción:	Una vez seleccionado el modo de cálculo, se invoca éste método para fijar el modo de cálculo en el objeto gestionar.

Nombre: CalculoDescriptoresPluginUI	
Tipo de clase: Interfaz	
Atributos:	**Tipo:**
splitPaneInferior	JSplitPane
splitPaneSuperior	JSplitPane
scrollPane	JScrollPane
scrollPaneTreeFiles	JScrollPane
treeFiles	JTree
panelWorkArea	JPanel
panelFragmentos	JPanel
btnCalcular	JButton
panelDescriptores	JPanel
descriptorsTabedPane	JTabbedPane
rootNodeDescriptors	CheckNodeDescriptor
rootNodesTreeMoles	CheckNode
jScrlPaneFragmentos	JScrollPane
menuBar	JMenuBar
resultTabbedPane	JTabbedPane
calculo	GestionarDescriptores
fragsPanel	JPanel
checkFragments	ArrayList<JCheckBox>
spinFragments	ArrayList<JSpinner>
Principales Responsabilidades:	
Nombre:	calculateDescriptors(fragments,files,selectedDescriptors)
Descripción:	Una vez obtenidos los parámetros necesarios para calcular los descriptores se invoca este método para realizar el cálculo.
Nombre:	almacenarResultadoBD(nombreEnsayo, nota, dirResultado)
Descripción:	Una vez realizado los cálculos se invoca este método para almacenar el resultado de los mismos en la base de datos.

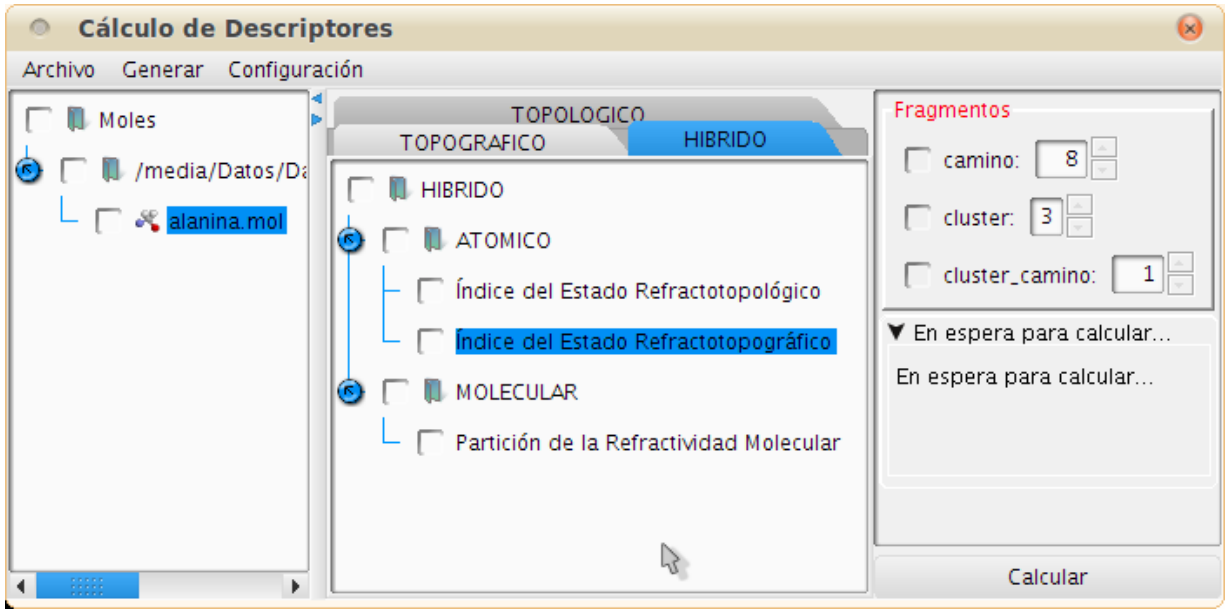
Anexo 3. Diagrama de secuencia del CU Realizar Cálculo de Descriptores de la herramienta propuesta



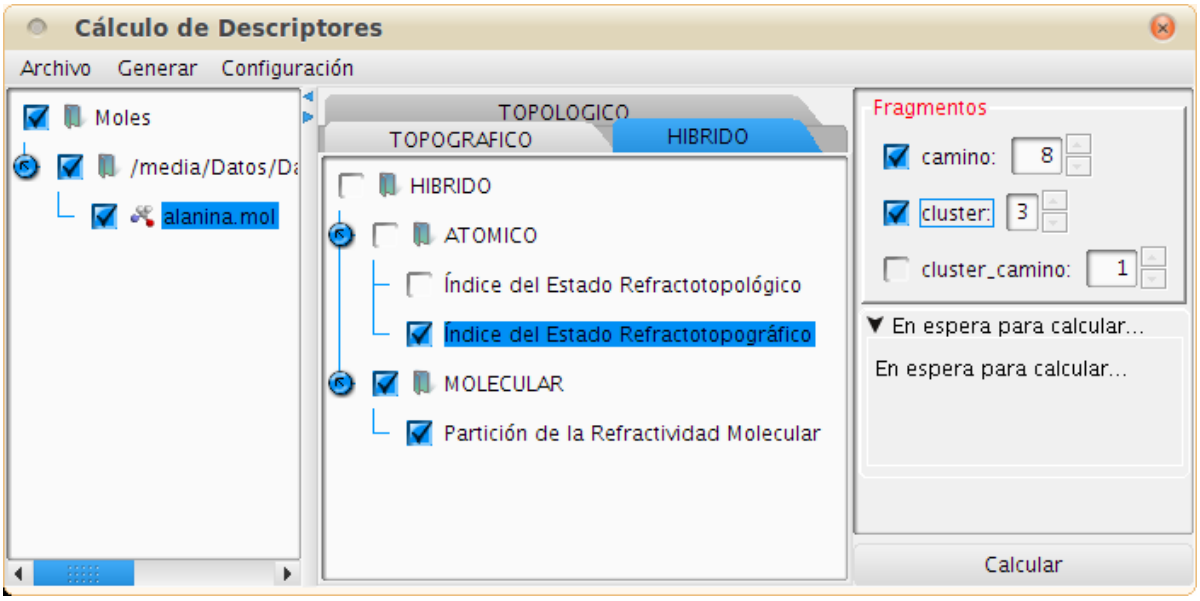
Anexo 4. Diagrama de secuencia del CU Generar Fichero ARFF de la herramienta propuesta



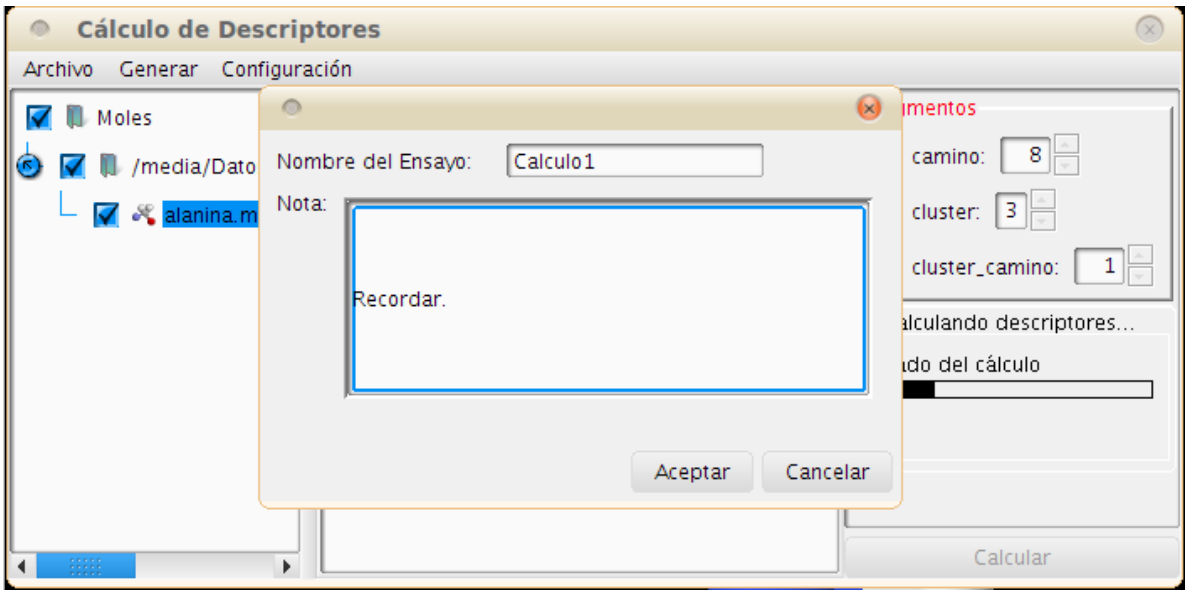
Anexo 5. Interfaz que muestra las moléculas, descriptores y fragmentos



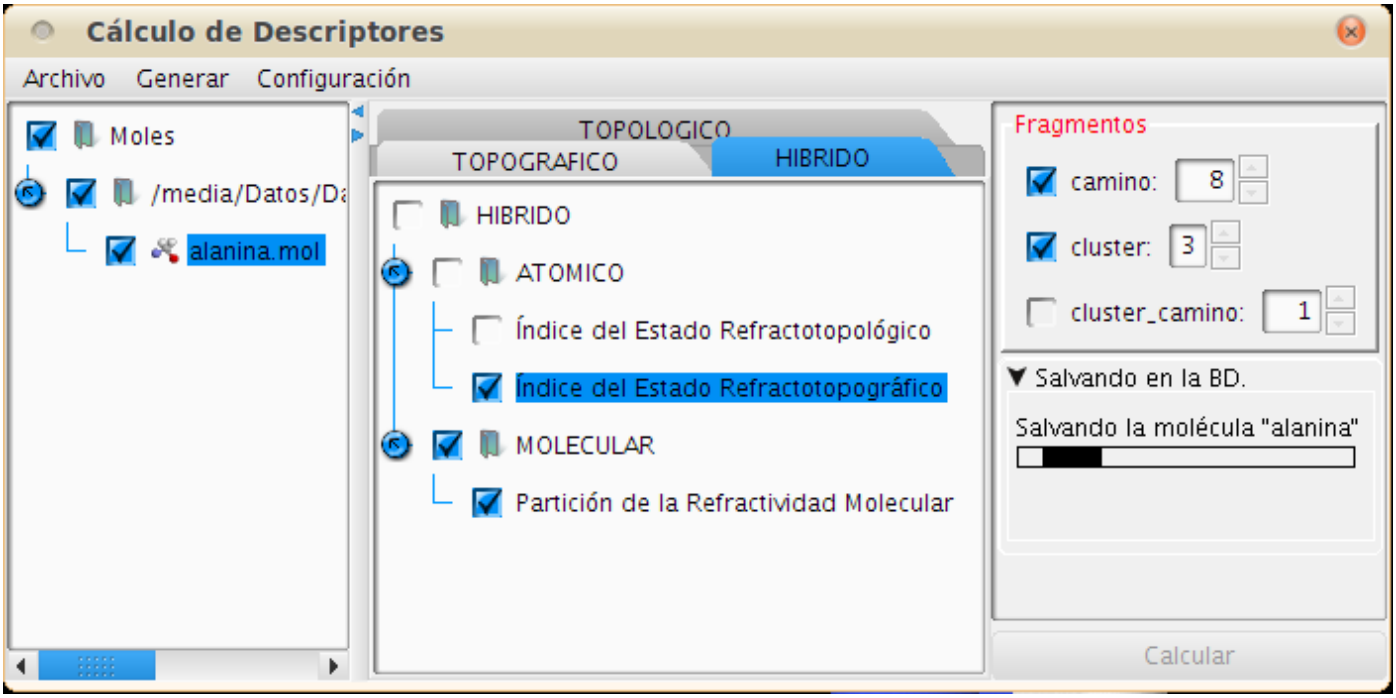
Anexo 6. Interfaz que muestra las moléculas, descriptores y fragmentos que el especialista seleccionó



Anexo 7. Interfaz para introducir nombre y un comentario sobre el ensayo



Anexo 8. Salvando los resultados en la base de datos local



Anexo 9. Diseño de pruebas para el escenario Configurar cálculo local del CU Configurar Cálculo

SC 1: Configurar Cálculo

Variables

Var1: Cálculo Local

Id del escenario	Escenario	Var1	Respuesta del Sistema	Resultado de la Prueba
EC 1.1	Configurar	V	El sistema	Se cambió la

	cálculo local.		muestra una interfaz para seleccionar la librería y valida la selección. Cambia la configuración del cálculo a local.	configuración del cálculo a local.

[V indica válido, I indica inválido]

Anexo 10. Diseño de las pruebas por cada sección del caso de uso Realizar Cálculo de Descriptores

SC 1: Calcular descriptores

Variables

Var1: Estructuras Químicas

Var2: Descriptores

Var3: Fragmentos

Id del escenario	Escenario	Var1	Var2	Var3	Respuesta del Sistema	Resultado de la Prueba
EC 1.1	Calcular descriptores correctamente.	V	V	V	El sistema recoge los datos y realiza el cálculo de los descriptores. Se guardan los resultados en la	Se realizó el cálculo de los descriptores y se guardaron los resultados en la base de datos.

					base de datos.	
EC 1.2	Calcular descriptores incorrectamente.	I	V	V	El sistema recoge los datos y valida la selección. En caso de que alguno sea inválido, muestra un mensaje de error.	Mostró el mensaje de error "Debe seleccionar al menos una molécula".
		V	I	V	El sistema recoge los datos y valida la selección. En caso de que alguno sea inválido, muestra un mensaje de error.	Mostró el mensaje de error "Debe seleccionar al menos un descriptor".
		V	V	I	El sistema recoge los datos y valida la selección. En caso de que alguno sea inválido, muestra un mensaje de error.	Mostró el mensaje de error "Debe seleccionar al menos un fragmento".

[V indica válido, I indica inválido]

Anexo 11. Diseño de pruebas para el CU Generar Fichero ARFF

SC 1: Generar Fichero ARFF

Variables

Var1: resultado obtenido

Id del escenario	Escenario	Var1	Respuesta del Sistema	Resultado de la Prueba
EC 1.1	Generar fichero correctamente.	V	El sistema recoge los datos y genera el fichero ARFF.	Se generó el fichero ARFF.

[V indica válido, I indica inválido]

GLOSARIO

ARFF: *Attribute-Relation File Format*, es un archivo de texto ASCII que describe una lista de casos que comparten un conjunto de atributos.

byte-codes: El byte-code no es exactamente el código fuente ni el código compilado, sino un camino a medias entre la interpretación y la compilación que se interpretan a través de una máquina virtual genérica.

FTP: *File Transfer Protocol*, es un protocolo de red estándar que se utiliza para copiar un archivo desde un host a otro en una red TCP / IP, como Internet.

HTTP: *Hypertext Transfer Protocol*, es un protocolo de capa de aplicación para sistemas distribuidos, de colaboración, los sistemas de información hipermedia.

SMTP: *Simple Mail Transfer Protocol*, es un estándar de Internet para la transmisión de correo electrónico a través del Protocolo de Internet (IP).

XML: *Extensible Markup Language*, es un formato sencillo, texto muy flexible. Desempeña un papel importante en el intercambio de una amplia variedad de datos en la Web y en otros lugares.