Universidad de las Ciencias Informáticas

Facultad 6

Título: Autodock Vina, Servicio de Acoplamiento Molecular para la Plataforma de Servicios Bioinformáticos de la UCI.

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas.

Autor:

Osniel Medina Pérez

Tutores:

MsC. Alina Agramonte Delgado

Ing. Osvel Chávez Hernández

La Habana, Junio de 2013
"Año 55 de la Revolución"

(...) es incuestionable que el principio de la combinación del estudio y el trabajo es la única fórmula de educación comunista. No hay otra. Nadie aprenderá a nadar sobre la tierra, y nadie caminará sobre el mar. Al hombre lo hace su medio ambiente, al hombre lo hace su propia vida, su propia actividad. Y aprenderemos a respetar lo que crea el trabajo creando. Enseñaremos a respetar esos bienes enseñándolo a crear esos bienes. Y no hay otro camino.

Fidel Castro

Declaración de Autoría

Yo Osniel Medina Pérez declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

Osniel Medina Pérez

_____

Firma del Autor

MsC. Alina Agramonte Delgado

_____

Firma del Tutor

Ing. Osvel Chávez Hernández

_____

Firma del Tutor

Datos de Contactos

Tutores:

MsC. Alina Agramonte Delgado

Universidad de las Ciencias Informáticas, La Habana, Cuba

Email: alinaad@uci.cu

Ing. Osvel Chávez Hernández

Universidad de las Ciencias Informáticas, La Habana, Cuba

Email: ochavez@uci.cu

Agradecimientos

Hoy me miro en el espejo y no puedo creer en lo que me he convertido, nunca imagine llegar hasta aquí, no pensé convertirme en un Ingeniero en Ciencias Informáticas. Particularmente, creo que es debido a la magia que posee una persona para darte fuerzas para seguir adelante y luchar contra cualquier barrera que se imponga, sin importar las condiciones materiales o sentimentales que nos prepare la vida, alguien quien nos trae a este mundo dispuesto a darlo todo por nosotros. Quiero dar mi primer agradecimiento a la persona más importante en mi vida, mi madre, quien ha batallado junto a mí día a día durante estos 5 años y quien más que nadie, sabe todos los problemas que enfrentamos juntos para poder alcanzar esta meta.

Quiero agradecer a mi padre, quien desde yo ser muy pequeño, cultivó en mi persona excelentes valores que me impulsaron a poder lograr cualquier objetivo que me propusiera en la vida.

A quien considero que es mi segundo padre, Diosdado, quien está pendiente en todo momento de mis resultados alcanzados y me brindo la vocación más adecuada para actuar ante la vida.

Agradecer a mi hermanita Leydis, a mi novia Rosmery quien apareció en mi vida en el momento más oportuno y me demostró que más que una pareja, fue la amiga que me ayudó a desahogar todos mis momentos de estrés de la tesis, quien me supo comprender y animar para no caer en este largo camino.

A mis suegros Luis Alberto y Magalis por preocuparse por mis estudios de manera extraordinaria.

A mi amigo Fernando, ese gruñón que más que un amigo, fue un hermano, fue mi maestro.

A mis tutores, Alina y Osvel quienes estuvieron pendientes todo el tiempo del desarrollo de mi trabajo y me guiaron para forjarme como un mejor ingeniero.

A mis amigos del antiguo apartamento 75103, y especialmente a los del 96203 como Eudel “el magnífico”, Raulito, Jeovany “el Pótin”, Reisel el gordo, Ernesto “el largo”, todos en general, quienes ayudaron a que me sintiera como en mi hogar en la universidad y estuvieron siempre presente, tanto para fiestas como para ayudarme a resolver algún ejercicio de programación o cualquier otra materia.

A mis compañeros del grupo 6102, 6202, y 6303 quienes también aportaron su granito para estar hoy aquí.

Por último, con una importancia especial a la revolución cubana y especialmente a su máximo líder Fidel Castro Ruz.

Dedicatoria

A mi madre Barbara Pérez, por ser la mejor mamá del mundo y estar junto a mí apoyándome en todo momento.

A mis futuros hijos, por los que hoy me esfuerzo para servirles como ejemplo para una buena orientación en sus vidas.

A la revolución cubana y en especial, a su máximo líder Fidel castro Ruz, quien tuvo la excelente idea de crear la Universidad de Ciencias Informáticas.

Resumen

El método de Acoplamiento Molecular o *Docking*, es un método de simulación por computadora muy utilizado para el desarrollo de fármacos con resultados exitosos el cual consiste en calcular computacionalmente cuál es la posición más favorable que tendría una molécula (ligando) en el blanco molecular (Proteína). Para aplicar este proceso es necesario cribar bases de datos de cientos de ligandos en interacción con una misma proteína, haciendo que esta tarea requiera de un gran despliegue de recursos puesto que es necesario obtener, procesar y almacenar cantidades colosales de datos. En la UCI contamos con la Plataforma de Servicios Bioinformáticos donde existe un módulo dedicado al acoplamiento molecular que brinda este servicio a través del programa Dock 6.5, utilizando la plataforma T-arenal como alternativa para el cálculo distribuido. El presente trabajo tiene la tarea de integrar el programa Autodock Vina a la Plataforma de Servicios Bioinformáticos de la UCI por sus óptimas condiciones en tiempo de ejecución y resultados obtenidos, utilizando T-arenal para las ejecuciones distribuidas del programa y con el objetivo de incrementar las prestaciones del Módulo de Acoplamiento Molecular. Al concluir contaremos con un módulo de servicios de acoplamiento molecular con mayores prestaciones, ya que brindará a los usuarios especializados la posibilidad de realizar ensayos masivos de *Docking* a partir de dos herramientas disponibles, lográndose una disminución significativa del tiempo de respuesta.

Palabras claves: Acoplamiento Molecular, Sistema Distribuido, Servicios Bioinformáticos.

Índice general

Índice de Figuras

Índice de Figuras

Índice de Tablas

Introducción

Un fármaco, de acuerdo con la farmacología, es cualquier sustancia que produce efectos medibles o sensibles en los organismos vivos y que se absorbe, puede transformarse, almacenarse o eliminarse. Esta definición se acota a las sustancias de interés clínico refiriéndonos a las usadas para la prevención, diagnóstico, tratamiento, mitigación y cura de enfermedades.

Muchos de los medicamentos actualmente en uso deben su descubrimiento tanto al estudio de remedios tradicionales como al aislamiento de principios activos de productos naturales. En el desarrollo moderno de estos se utiliza tecnologías avanzadas para realizar búsquedas sistemáticas a gran escala con el propósito de encontrar sustancias con mayor potencial medicinal y menor toxicidad.

Una etapa fundamental en el complejo proceso de desarrollo de fármacos es la elección del sujeto de estudio, también conocido como molécula blanco, sobre la que actuará el medicamento para lograr el efecto terapéutico deseado. Diversas investigaciones sobre los fármacos empleados en las últimas décadas han revelado que la acción terapéutica de estos incide principalmente en proteínas, ya sea como receptores moleculares o como enzimas. Por esta razón en el diseño racional de medicamentos, las proteínas pueden considerarse como "la primera opción para fungir como moléculas blanco". La selección de la proteína específica para el trabajo dependerá de la información disponible sobre el mecanismo molecular responsable del desarrollo de la enfermedad molecular que se estudie.

Cuando se trabaja con proteínas y particularmente con enzimas como sustancias blanco, deben considerarse tres factores importantes: la especificidad o selectividad molecular de la enzima, la afinidad o fuerza con que se fija el sustrato a ella, y la geometría del sitio de unión. Muchos fármacos son inhibidores de la función de una enzima a través de un bloqueo efectivo del sitio activo o de las zonas coadyuvantes de la catálisis. En estos casos es deseable desarrollar compuestos que se unan con gran firmeza y alta especificidad a los sitios funcionales, y para desarrollar un fármaco con ambas características es imprescindible conocer con detalles la geometría del sitio de unión.

El método de Acoplamiento Molecular o *Docking*, es un método de simulación por computadora muy utilizado para el desarrollo de fármacos con resultados exitosos y consiste en calcular computacionalmente cuál es la posición más favorable que tendría una molécula en el blanco molecular. El objetivo es alcanzar una conformación óptima tanto para la proteína, como para su ligando, haciendo que la orientación entre estos minimice la energía libre.

Durante los ensayos masivos que se llevan a cabo mediante el *Docking*, a veces es necesario cribar bases de datos de cientos de ligandos en interacción con un mismo receptor no haciendo de esta una tarea trivial. Esto requiere de un gran despliegue de recursos, puesto que es necesario obtener, procesar y almacenar cantidades colosales de datos. Frecuentemente este tipo de simulaciones se llevan a cabo en un clúster de computadoras con el objetivo de acortar los tiempos de ejecución y los costos (1).

El método de Acoplamiento Molecular o *Docking,* es ampliamente utilizado alrededor del mundo por compañías privadas con servidores a través de Internet destinados a esta tarea como el *DockingServer* (2) y el *Docking.org* (3), donde se utilizan tecnologías de punta que permiten la constante actualización de nuevas herramientas para hacer acoplamiento molecular, y que establecen una serie de limitaciones por los costos de estas prestaciones para los países subdesarrollados.

Cuba a pesar de ser un país en vías de desarrollo, ha acumulado importantes logros en el campo de la salud pública y la biotecnología, que se comparan con los alcanzados por países más desarrollados. Existen instituciones con recursos computacionales que realizan acoplamiento molecular como el Centro de Inmunológica Molecular (CIM) y el Centro de Ingeniería Genética y Biotecnología (CIGB), sin embargo ninguna brinda este servicio para el resto de la comunidad científica.

La Universidad de las Ciencias Informáticas (UCI) cuenta con varios centros de desarrollo de software, entre los cuales se encuentra el Centro de Tecnologías de Gestión de Datos (DATEC). Uno de sus departamentos es el de Bioinformática que entre otras tareas desarrolla una Plataforma de Servicios Bioinformáticos (PSBio) con el objetivo principal de brindar una serie de servicios relacionados con la especialidad. Existe en el mismo, un módulo dedicado al acoplamiento molecular que brinda este servicio a través del programa Dock 6.5, utilizando la Plataforma de Tareas Distribuidas T-arenal (4), como alternativa para el cálculo a gran escala utilizando estaciones de trabajo no dedicadas.

Con el objetivo de optimizar y ampliar las prestaciones del módulo de servicio de acoplamiento molecular dentro de PSBio, se pretende insertar el programa Autodock Vina que en comparación con otras herramientas similares, incrementa la velocidad de cálculo hasta en dos órdenes de magnitud de tiempo y obtiene resultados estructurales más parecidos a los obtenidos experimentalmente (5). Es por ello que surge la necesidad de brindar a los usuarios del módulo de acoplamiento en PSBio la opción de elegir la herramienta de acoplamiento que desea utilizar según sus intereses o la naturaleza del problema en cuestión. De lo antes expuesto se deriva el siguiente **problema a resolver:**

¿Cómo insertar una nueva herramienta a la Plataforma de Servicios Bioinformáticos de la UCI para incrementar las prestaciones del Módulo de Acoplamiento Molecular?

Determinado como **objeto de estudio:** los métodos que realizan acoplamiento molecular y como **campo de acción**: las herramientas que realizan acoplamiento molecular.

Objetivo General: Implementar un nuevo servicio de acoplamiento molecular con Autodock Vina para incrementar las prestaciones de este módulo en la Plataforma de Servicios Bioinformáticos de la UCI.

Objetivos Específicos:

- Realizar el análisis y el diseño para insertar el Autodock Vina al módulo de acoplamiento molecular de la Plataforma de Servicios Bioinformáticos de la UCI haciendo uso de un conglomerado de computadoras.
- Implementar el Servicio de acoplamiento molecular usando Autodock Vina para la Plataforma de Servicios Bioinformáticos de la UCI.
- Validar el nuevo Servicio de acoplamiento molecular usando Autodock Vina en la Plataforma de Servicios Bioinformáticos de la UCI.

Tareas Investigativas:

1. Estudio y selección del conglomerado de computadoras a utilizar.
2. Generación de los artefactos correspondientes a las fases de análisis y diseño para la adición del Autodock Vina al Módulo de Acoplamiento Molecular de la Plataforma de Servicios Bioinformáticos de la UCI.
3. Diseño de la Interfaz Web que se va a incluir en la Plataforma de Servicios Bioinformáticos de la UCI para que los especialistas interactúen con el nuevo servicio de acoplamiento molecular.
4. Implementación de la Interfaz Web y los componentes de software necesarios para brindar acceso al nuevo servicio de acoplamiento molecular.
5. Realización de los casos de prueba para el nuevo servicio de acoplamiento molecular con Autodock Vina, de la Plataforma de Servicios Bioinformáticos de la UCI.
6. Ejecución de los casos de prueba para validar el funcionamiento del Autodock Vina en el Módulo de Acoplamiento Molecular.

Para darle cumplimiento a las tareas propuestas se utilizaron los siguientes métodos de investigación:

Teóricos:

- **Histórico-Lógico:** este método se usó con el fin de recopilar la información que se posee hasta el momento sobre las herramientas que realizan acoplamiento molecular y resumir los aspectos fundamentales de su evolución para el desarrollo de la investigación en curso.
- **Analítico-Sintético:** este método se usó para analizar y estudiar las teorías recopiladas en la bibliografía y extraer los elementos más importantes que se relacionan con el objeto de estudio.
- **Modelación:** este método se usó para descubrir y estudiar nuevas relaciones y cualidades del objeto de estudio. La modelación es justamente el proceso mediante el cual se crean modelos con vista a investigar la realidad, por lo que es el más importante en el proceso de construcción del software.

Empíricos:

- **Observación:** este método se usó para realizar valoraciones y obtener informaciones a partir de lo observado. Esto se manifiesta principalmente cuando se realizan observaciones sobre el funcionamiento de sistemas similares al desarrollado, lo que da una visión de cómo tiene que ser el sistema a desarrollar en su forma externa.

Este documento está estructurado de la siguiente manera:

Capítulo 1. Fundamentación Teórica: Contiene el estudio sobre las aplicaciones que realizan acoplamiento molecular y los métodos utilizados por estas herramientas para llevar a cabo este fin; también se desarrolla la investigación sobre cada una de las tecnologías, metodologías, lenguajes de programación y herramientas utilizadas para el desarrollo de esta aplicación.

Capítulo 2. Análisis: Contiene los principales conceptos mediante un Modelo de Dominio del Sistema, la captura de requisitos funcionales y no funcionales, así como el diagrama de casos de uso del sistema y la descripción de los mismos.

Capítulo 3. Diseño: Contiene el diseño del sistema obteniendo como resultado principal, los diagramas de clases de diseño, así como los diagramas de interacción para cada realización de los casos de usos. Muestra la descripción de la arquitectura y patrones de diseño utilizados, también se describe el Modelo Entidad Relación que representa las bases de datos utilizadas en el sistema.

Capítulo 4. Implementación y Prueba: Contiene las principales características del flujo de trabajo de implementación, representando el diagrama de despliegue y el de componentes; así como la realización de casos de prueba para mitigar los posibles fallos.

Resultados Esperados:

La Plataforma de Servicios Bioinformáticos de la UCI contará con un módulo de Servicios de Acoplamiento Molecular con mayores prestaciones ya que brindará a los usuarios especializados la posibilidad de realizar ensayos masivos de *Docking* a partir de una nueva herramienta, el Autodock Vina, utilizando un conglomerado de computadoras para acortar los tiempos de ejecución.

Capítulo 1: Fundamento Teórico

Introducción

En el presente capítulo se hace un estudio de las herramientas y servidores más utilizados para las simulaciones de acoplamiento molecular, citando ejemplos y rasgos distintivos en cada caso. Se hace un análisis de las herramientas más factibles a emplear para brindar este servicio en la PSBio, y cuál metodología a emplear para poner en curso la prestación de este servicio en la Plataforma de Servicios Bioinformáticos de la UCI, cuál metodología y lenguaje de desarrollo de software serían los más eficientes para el desarrollo del trabajo.

1.1. Herramientas de Acoplamiento Molecular

Los sistemas que realizan acoplamiento molecular han sido desarrollados con el objetivo de automatizar el complejo proceso de desarrollo de fármacos a través de métodos de simulación por computadora.

Para describir la metodología del acoplamiento molecular implementada en este tipo de aplicaciones es útil dividirla en una serie de etapas. La primera de ellas consiste en disponer de la estructura tridimensional[1] de la molécula blanco, la cual debe ser acondicionada para los cálculos subsecuentes y sobre la que debe identificarse el sitio de unión de una molécula de prueba (generalmente pequeña en relación con la blanco).

La siguiente etapa requiere poseer un archivo o archivos numerosos de ligandos potenciales, estos son, moléculas orgánicas con estructuras tridimensionales conocidas y que se encuentran en condiciones adecuadas para simular su asociación al blanco. La tercera etapa, que es la parte medular del método y particular de cada herramienta, consiste en un algoritmo computarizado que toma cada uno de los ligandos de la base de datos y lo coloca dentro del sitio de unión en una gran cantidad de orientaciones (6). A continuación se muestra una representación esquemática del proceso (Fig. 1):

[1] Que tiene tres dimensiones: altura, anchura y largura.
[2] Es un software que asiste a una aplicación para interactuar o comunicarse con otras aplicaciones, software, redes, hardware y/o

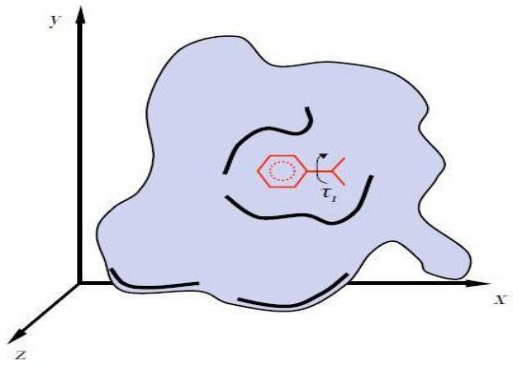

Figura 1: Estructura tridimensional de una molécula blanco (proteína) donde interactúa una molécula de prueba (Ligando) con el objetivo de obtener una conformación óptima para el posterior desarrollo de fármacos.

Finalmente, el programa de cómputo ordena los diferentes compuestos probados de acuerdo al puntaje de su orientación óptima y entonces, el usuario puede analizar estos resultados y planear experimentos para validar los mismos. Aplicaciones como Autodock, Autodock 4, Dock 6.5 y Autodock Vina, son destinadas a esta tarea. A continuación se hace una breve descripción de las características de cada una de ellas:

1.1.1. Autodock

Es una herramienta destinada al acoplamiento molecular automatizado, gratuito y disponible bajo la Licencia Pública General GPL. Está diseñado para predecir cómo los ligandos o moléculas orgánicas, pueden unirse a una estructura tridimensional de una molécula blanco, como candidatos a fármacos. Es uno de los software de acoplamiento más rápido y citados por los usuarios especializados, proporciona predicciones de alta calidad de las conformaciones de los ligandos y una buena correlación entre constantes de inhibición predichas y los experimentales. Autodock también ha demostrado ser útil en acoplamiento ciego donde la ubicación del sitio de unión no es conocida (7). La arquitectura del programa está diseñada para soportar cálculos distribuidos sobre una arquitectura grid para reducir el tiempo de procesamiento.

1.1.2. Autodock 4

Autodock 4 es una versión superior al Autodock y no sólo es más rápido que las versiones anteriores, sino también permite que las cadenas laterales en la macromolécula sean flexibles. Este programa tiene una función de puntuación de energía libre que se basa en un análisis de regresión lineal, un campo de fuerza de Construcción de Modelos Asistida con Refinamiento de Energía (AMBER, por sus siglas en

inglés) y un conjunto aún mayor de diversos complejos de proteína-ligando. Puede ser compilado para tomar ventaja de los nuevos métodos de búsqueda de la biblioteca de optimización "ACRO" (8).

La introducción de Autodock 4 comprende tres mejoras importantes:

- Los resultados del acoplamiento son más precisos y fiables.
- Opcionalmente puede modelar flexibilidad en la macromolécula blanco.
- Permite el uso del Autodock en la evaluación de interacciones proteína-proteína.

1.1.3. Dock 6.5

Es una herramienta para realizar acoplamiento molecular, escrita en lenguaje C++ y de código abierto. Utiliza un modelo 3D de la proteína objetivo y gira el ligando sobre el área elegida de dicha proteína para lograr una mayor conformación entre ellos. Al ser compilado el código fuente para una arquitectura de grid, hace sus ejecuciones de forma paralela sobre el clúster donde esté instalado (9).

Esta herramienta, entre otros algoritmos incorporados, trae los siguientes:

- Opciones de puntuación de reducción al mínimo incluyendo la puntuación electrostática de la malla de Delphi.
- Correcciones de entropía conformacional de los ligandos.
- Desolvatación del ligando y del receptor.
- Puntuación de solvatación con el cribado de sal opcional de Hawkins-Cramer-Truhlar GB/SA.

1.1.4. Autodock Vina

Autodock Vina es un nuevo programa de acoplamiento molecular de código abierto bajo la licencia de Apache, que permite el uso comercial, no comercial y redistribución (8). Brinda un alto por ciento de rendimiento, mayor precisión y facilidad de uso (9). Autodock Vina es aproximadamente dos órdenes de magnitud más rápido que el Autodock 4, además de que mejora significativamente la precisión de las predicciones del modo de unión ligando-receptor. La reducción del tiempo de cálculo se consigue a partir de la ejecución del algoritmo, sobre un clúster, una arquitectura grid o una solución de trabajo distribuido (10).

Incluye algoritmos tales como:

- Métodos estocásticos de optimización global, incluyendo algoritmos genéticos, optimización por enjambres de partículas, recocido simulado, entre otros; combinados con diversos procedimientos de optimización y trucos para acelerar la optimización.

- Método Broyden-Fletcher-Goldfarb-Shanno (BFGS, por sus siglas en inglés) para la optimización local, utilizando no solo el valor de la función de puntuación, sino también su gradiente, es decir, las derivadas de la función de puntuación con respecto a sus argumentos.

1.2. Sistemas computarizados de cálculo a gran escala

La computación distribuida está basada en un nuevo modelo de trabajo para resolver grandes problemas mediante la computación utilizando un gran número de ordenadores organizados en una infraestructura de comunicaciones distribuida.

Cada máquina posee sus componentes de hardware y software que el usuario percibe como un solo sistema. El usuario accede a los recursos remotos de la misma manera en que accede a recursos locales, o un grupo de computadores que usan un software para conseguir un objetivo en común. Los sistemas distribuidos deben ser muy confiables, ya que si un componente del sistema se descompone otro componente debe ser capaz de reemplazarlo, denominado por tolerancia a fallos. El tamaño de los sistemas distribuidos pueden ser variables de acuerdo a los objetivos que se quieran desarrollar con la implementación de estos sistemas y el poder económico que lo pueda respaldar.

1.2.1. Computación Grid

El término Grid se refiere a una infraestructura que permite la integración y el uso colectivo de ordenadores de alto rendimiento, redes y bases de datos que son de propiedad y están administrados por diferentes instituciones. Puesto que la colaboración entre instituciones envuelve un intercambio de datos, o de tiempo de computación, el propósito de la Grid es facilitar la integración de recursos computacionales.

La infraestructura Grid descansa sobre un software denominado “middleware”[2], que asegura la comunicación transparente entre los distintos ordenadores intercomunicados. Estas computadoras englobadas, no están conectadas o enlazadas firmemente, es decir, no tienen por qué estar en el mismo lugar geográfico. También poseen un motor de búsqueda que no sólo encontrará los datos que el usuario necesite, sino también las herramientas para analizar y la potencia de cálculo necesaria para utilizarlas. Al finalizar el proceso, la Grid distribuirá las tareas de computación a cualquier lugar de la red en la que haya capacidad disponible y enviará los resultados al usuario (11).

El funcionamiento de la Grid se basa en cinco pilares básicos:

[2] Es un software que asiste a una aplicación para interactuar o comunicarse con otras aplicaciones, software, redes, hardware y/o sistemas operativos.

- La posibilidad de compartir recursos.
- La seguridad (acceso seguro).
- El uso eficiente de los recursos.
- Redes de comunicación fiables que eliminen las distancias.
- Estándares abiertos.

1.2.2. Clúster de Computadoras

Un clúster es un conglomerado de computadores construido para la solución de problemas de las ciencias, las ingenieras y el comercio moderno, utilizando componentes de hardware comunes y en la mayoría de los casos, software libre. Los computadores se interconectan mediante alguna tecnología de red. El clúster puede estar conformado por nodos dedicados sin disponer de teclado, mouse, ni monitor, y su uso está exclusivamente dedicado a realizar tareas relacionadas con el clúster. También puede estar compuesto por nodos no dedicados, los cuales disponen de teclado, mouse y monitor, y su uso no está exclusivamente dedicado a realizar tareas relacionadas con el clúster, este último hace uso de los ciclos de reloj que el usuario del computador no está utilizando para realizar sus tareas.

Los computadores del clúster pueden tener, todos, la misma configuración de hardware y sistema operativo (clúster homogéneo), diferente rendimiento pero con arquitecturas y sistemas operativos similares (clúster semi-homogéneo), o tener diferente hardware y sistema operativo (clúster heterogéneo), lo que hace más fácil y económica su construcción (12).

1.2.3. T-arenal

La Plataforma de Tareas Distribuidas (T-arenal) es un producto informático actualmente en su versión 2.0, desarrollado por el departamento de Bioinformática e implementado con el objetivo de ofrecer una alternativa de cómputo y que aglutina en un solo conjunto varias estaciones de trabajo sin intentar eliminar o pretender aminorar las amplias posibilidades de aplicación de los modelos paralelos, sino de complementar todos los medios disponibles en una gran “supercomputadora virtual”. T-arenal está basada en el modelo cliente-servidor y está dividida en tres componentes esenciales: servidor central, servidor de peticiones y cliente. Al servidor central estarán asociados uno o varios servidores de peticiones los cuales son los encargados de solicitar una tarea. A cada servidor de peticiones le pertenecerán uno o varios clientes para realizar el procesamiento de una tarea determinada. El objetivo de un servidor de peticiones es repartir el trabajo entre los clientes correspondientes, gestionar la pérdida de clientes y conformar la solución final a partir de los resultados enviados por cada cliente.

En el servidor central se encuentran almacenados todos los problemas. Un problema es una aplicación desarrollada en el lenguaje de programación Java y está constituido por un manejador de datos (Data-Manager), un algoritmo (Task) y por el resto de los archivos utilizados en su realización. De un problema se pueden crear varias ejecuciones (instancias o tareas), potencialmente con distintos datos de entrada.

De cada ejecución creada se obtendrá una solución. Sólo cuando un servidor de peticiones se encuentra libre es que realizará una solicitud para atender una ejecución.

Cuando los clientes culminan, envían las respuestas obtenidas al servidor de peticiones correspondiente. El servidor de peticiones es el responsable de procesar cada uno de los resultados enviados (13).

1.3. Tecnologías Web

Una aplicación web es un sistema informático que los usuarios utilizan accediendo a un servidor web a través de Internet o de una Intranet. En las tecnologías web se agrupan las técnicas, herramientas y lenguajes de programación que nos permiten crear o modificar estas aplicaciones. En una aplicación web a los usuarios se le permite ejecutar lógica de negocio a través de un navegador, en otras palabras modificar el estado del negocio, utilizando estas tecnologías que generan contenidos dinámicos y permiten a los usuarios del sistema modificar la lógica del negocio en el servidor web.

1.3.1. Apache Tomcat

Apache Tomcat es un servidor web de código abierto y contenedor de servlets[3] desarrollado por la Apache Software Foundation (ASF, por sus siglas en inglés). Tomcat implementa la Java Servlet y JavaServer Pages (JSP, por sus siglas en inglés) de la Oracle Corporation y se desarrolla en un entorno abierto, participativo y publicado bajo la licencia Apache versión 2.0 (14).

1.3.2. Portlet

Un portlet es una tecnología de componentes web desarrollado en el lenguaje de programación Java. Está gestionado por el contenedor de portlet Liferay, procesa las solicitudes y genera contenido dinámico como respuesta. Los portlets son utilizados por portales para la conexión de los componentes de interfaz de usuario (15).

1.3.3. Liferay Portal

[3] Los servlets son objetos que corren dentro y fuera del contexto de un contenedor de servlets (ej: Tomcat) y extienden su funcionalidad.

Es un portal de gestión de portlet, de código abierto e implementado en Java y su contenido está basado en portlet. Incluye muchas aplicaciones como mensajería instantánea, foros y biblioteca de documentos. Es desplegable en servidores como Apache Tomcat y por ser multiplataforma, permite su ejecución en cualquier sistema operativo. Liferay es un proyecto de código abierto que usa licencia LGPL. Provee la personalización del entorno de usuario mediante plantillas y temas que pueden ser instalados a partir de un archivo con extensión .WAR (16).

1.4. Plataforma de Servicios Bioinformáticos (PSBio)

Es un proyecto desarrollado por el Departamento de Bioinformática perteneciente al Centro de Tecnología de Gestión de Datos (DATEC) de la Universidad de las Ciencias Informáticas, su objetivo es integrar sobre una plataforma, un conjunto de herramientas básicas de propósito general en la Bioinformática, así como otros productos especializados, desarrollados en el propio departamento, con el propósito de prestar servicios a usuarios especializados de otros centros y universidades del país donde el acceso a los mismos a través de Internet se ve limitado. La solución informática en desarrollo tiene una arquitectura orientada a servicios y con interfaces web, implementadas en forma de portlet, dentro del portal de gestión de contenidos Liferay. Esta plataforma se desarrolló con el objetivo fundamental de utilizar los recursos computacionales de la universidad para brindar servicios bioinformáticos a la comunidad científica (17).

1.5. Marcos de Trabajo

Un Marco de Trabajo o Framework es una estructura conceptual y tecnológica de soporte definida normalmente con artefactos o módulos de software concretos, en base a la cual otro proyecto de software puede ser organizado y desarrollado. Puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros programas para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

1.5.1. Spring Framework

Spring es un framework de aplicación desarrollado por la compañía Interface 21, para aplicaciones escritas en el lenguaje de programación Java. Spring integra diferentes tecnologías como: J2EE (Java 2 Enterprise Edition), incluyendo EJB (Enterprise Java Beans), Servlets y JSP (JavaServer Page), en un solo framework combinando dichas herramientas y otras más en un solo paquete, para brindar una estructura más sólida y un mejor soporte para este tipo de aplicaciones, además se considera liviano, ya

que es una aplicación que no requiere muchos recursos para su ejecución (18). Este framework se encuentra actualmente en su versión 3.1.2. Una de las características fundamentales es que es una aplicación de código abierto, además que esté disponible todo el código fuente en el paquete de instalación.

1.5.2. Hibernate

Una herramienta de mapeo relacional u ORM (del inglés, *Object Relational Mapping*) es un marco de trabajo para gestionar bases de datos que brinda una mayor abstracción del modelo de base de datos, lo cual permite manejar la información sin tener conocimiento de un determinado lenguaje SQL. Existen muchas herramientas ORM para Java, entre las más populares está Hibernate. Esta provee un puente entre la base de datos y la aplicación para almacenar objetos persistentes, posibilita al desarrollador escribir menos código y gestionar con mucha facilidad el modelo de datos (19).

Hibernate posee un lenguaje de consultas propio llamado HQL (del inglés, *Hibernate Query Language*). Para utilizar HQL no es necesario tener conocimientos de SQL, además provee un complemento para Eclipse que permite realizar consultas a la base de datos y posibilita mapearlas clases con completamiento de código, además puede generar el código Java a partir del modelo de base de datos y viceversa.

1.5.3. Axis2

Es una nueva generación de servicios middlware de la Apache Web, que rediseña la arquitectura de los servicios en el Apache Web para incorporar cambios en estos. Entre muchas mejoras, Axis2 proporciona mensajería de primera clase y extensión SOAP[4] (del inglés, *Simple Object Access Protocol*). La arquitectura se construye en un núcleo sencillo y extensible que proporciona las abstracciones básicas para el resto del sistema. Presenta el diseño y el proceso, detrás en una arquitectura en tres temas, el modelo de procesamiento de XML, modelo extensible de procesamiento SOAP y un marco de mensajería (20).

1.6. Lenguaje de Programación

[4] Protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML.

Un lenguaje de programación es una técnica estándar de comunicación que permite expresar las instrucciones que han de ser ejecutadas en una computadora. Consiste en un conjunto de reglas sintácticas y semánticas que definen un programa informático (21).

1.6.1. Java

Java es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems. Es un lenguaje bajo una plataforma privativa JDK y una de software libre OpenJDK. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria. Las aplicaciones Java están típicamente compiladas a bytecode (archivo de clase) que puede ejecutarse en cualquier máquina virtual de Java (JVM), independientemente de la arquitectura de computadores.

Entre sus características se pueden destacar:

- Sencillo y familiar: Que no requiere grandes esfuerzos de preparación para los desarrolladores.
- Robusto: Simplificando la gestión de memoria y eliminando las complejidades de la gestión explícita de punteros y aritmética de punteros del C.
- Independiente de la arquitectura y portable: Java está diseñado para soportar aplicaciones que serán instaladas en un entorno de red heterogéneo, con hardware y sistemas operativos diversos. Es además portable en el sentido de que es rigurosamente el mismo lenguaje en todas las plataformas.
- Alto rendimiento: a pesar de ser interpretado, Java tiene en cuenta el rendimiento, y particularmente en las últimas versiones dispone de diversas herramientas para su optimización.
- Interpretado, multi-hilo y dinámico: incorpora capacidades avanzadas de ejecución multi-hilo (ejecución simultánea de más de un flujo de programa) y proporciona mecanismos de carga dinámica de clases en tiempo de ejecución (22).

JSP es un acrónimo de Java Server Pages, es una tecnología orientada a crear páginas web con programación en Java. Con JSP podemos crear aplicaciones web que se ejecuten en variados servidores web, de múltiples plataformas, ya que Java es en esencia un lenguaje multiplataforma. Las páginas JSP están compuestas de código HTML/XML mezclado con etiquetas especiales para programar scripts de servidor en sintaxis Java. Por tanto, las JSP podremos escribirlas con nuestro editor HTML/XML habitual. En JSP se crean páginas de manera parecida a como se crean en ASP o PHP. Se generan archivos con

extensión .jsp que incluyen, dentro de la estructura de etiquetas HTML, las sentencias Java a ejecutar en el servidor.

1.7. Entorno de Desarrollo Integrado

Los Entornos de Desarrollo Integrado (IDE, por sus siglas en inglés) son programas compuestos por una serie de herramientas y que utilizan los programadores para desarrollar software mediante uno o varios lenguajes de programación (23). Los entornos de desarrollo proporcionan un marco de trabajo para la mayoría de los lenguajes de programación existentes en el mercado (por ejemplo C, C++, C#, Java, Python y Visual Basic, entre otros). Además es posible que un mismo entorno de desarrollo tenga la posibilidad de utilizar varios lenguajes de programación, como es el caso de Eclipse.

1.7.1. Eclipse

Es un entorno de desarrollo integrado, de código abierto y multiplataforma. Es una potente y completa plataforma de programación, desarrollo y compilación de elementos tan variados como sitios web, programas en C++ o aplicaciones Java (24). Contiene las herramientas y funciones necesarias para el desarrollo de software, además de una atractiva interfaz que lo hace fácil y agradable de usar. Eclipse posibilita la agregación de componentes ya que su arquitectura está basada en complementos.

1.8. Metodología de desarrollo a utilizar

Las metodologías de desarrollo de software surgen ante la necesidad de utilizar una serie de procedimientos, técnicas, herramientas y soporte documental, a la hora de desarrollar un producto de software.

Dichas metodologías pretenden guiar a los desarrolladores al crear un nuevo software, pero los requisitos de un software a otro son tan variados y cambiantes, que ha dado lugar a que exista una gran variedad de metodologías para la creación del software (25). Estas se podrían clasificar en dos grandes grupos:

- Las metodologías orientadas al control de los procesos, estableciendo rigurosamente las actividades a desarrollar, herramientas a utilizar y notaciones que se usarán. Estas metodologías son llamadas Metodologías Pesadas.
- Las metodologías orientadas a la interactuación con el cliente y el desarrollo incremental del software, mostrando versiones parcialmente funcionales del software al cliente en intervalos cortos de

tiempo para que pueda evaluar y sugerir cambios en el producto según se va desarrollando. Estas son llamadas Metodologías Ágiles o Ligeras.

1.8.1. OpenUp

Es una metodología de desarrollo dirigida a la gestión y desarrollo de proyectos de software basada en desarrollo iterativo, ágil e incremental apropiado para proyectos pequeños y de bajos recursos, como también aplicable a un conjunto amplio de plataformas y aplicaciones de desarrollo. Se trata de una herramienta que puede hacer frente a una amplia variedad de tipos de proyectos. El esfuerzo personal en un proyecto de OpenUP se organiza en micro incrementos, estos representan unidades cortas de trabajo que producen un ritmo constante y medible del progreso del proyecto (normalmente medido en horas o días). El proceso aplica una colaboración intensa de cómo el sistema está desarrollado y aumentado por un compromiso, auto-organizado por el equipo (26).

1.9. Lenguaje Unificado de Modelado

El Lenguaje Unificado de Modelado (UML, por sus siglas en inglés) es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software. Captura decisiones y conocimientos sobre los sistemas que se deben construir. Se usa para entender, diseñar, hojear, configurar, mantener, y controlar la información sobre tales sistemas. Está pensado para usarse con todos los métodos de desarrollo, etapas del ciclo de vida, dominios de aplicación y medios (27).

1.10. Herramienta CASE de modelado

Las herramientas CASE (del inglés, *Computer Aided Software Engineer*), proponen una nueva filosofía del concepto de ciclo de vida basado en la automatización, para lo cual proporciona un conjunto de herramientas bien integradas, que enmarcadas dentro de una determinada metodología, permiten automatizar las fases del ciclo de vida de un sistema software (28).

Estas herramientas definen los siguientes objetivos:

- Automatizar e integrar las tareas de las distintas etapas del ciclo de vida, junto con la gestión de los proyectos de software.
- Mejorar la calidad mediante la automatización de la comprobación de errores.
- Automatizar la generación de la documentación.
- Facilitar que se pueda compartir los proyectos y la reutilización del software.

- Aportar un entorno de desarrollo interactivo.
- Acercar el desarrollo al usuario facilitando la creación de prototipos.
- Simplificar la labor de mantenimiento.

1.10.1. Visual Paradigm

Es una herramienta CASE, de licencia académica y comercial, que soporta todo el ciclo de vida de desarrollo de software denominado por la Ingeniería de Software. La misma propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación (29).

Entre otras características tiene:

- Disponibilidad en múltiples plataformas (Windows, Linux).
- Diseño centrado en casos de uso y enfocado al negocio que genera un software de mayor calidad.
- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- Capacidades de ingeniería directa e inversa.
- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
- Disponibilidad de múltiples versiones, para cada necesidad.
- Licencia: académica y comercial.
- Soporta aplicaciones Web.

1.11. Sistema Gestor de Base de datos

Un Sistema Gestor de Base de Datos (DBMS, por sus siglas en inglés) es un sistema de software que permite la definición de bases de datos, así como la elección de las estructuras de datos necesarios para el almacenamiento y búsqueda de los datos, ya sea de forma interactiva o a través de un lenguaje de programación y permiten a varios usuarios acceder a los datos al mismo tiempo. Brindan facilidades eficientes y un grupo de funciones con el objetivo de garantizar la confidencialidad, la calidad, la seguridad y la integridad de los datos que contienen, así como un acceso fácil y eficiente a los mismos.

1.11.1. PostgreSQL

Es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD y con su código fuente disponible libremente. Es el sistema de gestión de bases de datos más potente de software libre y en sus últimas versiones tiene numerosas condiciones para ser comparada a otras bases de datos comerciales. PostgreSQL utiliza un modelo cliente/servidor y posee una arquitectura multiproceso en vez

de multihilos para garantizar la estabilidad del sistema, es decir, un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando (30).

1.12. Aplicación de diseño y manejo de bases de datos a utilizar

PgAdmin3 es una aplicación gráfica para gestionar las bases de datos en PostgreSQL, siendo la más completa y popular con licencia PostgreSQL. Está escrita en C++ usando la librería gráfica multiplataforma wxWidgets, lo que permite que se pueda usar en Linux, FreeBSD, Solaris, Mac OS X y Windows. Es capaz de gestionar versiones a partir de la PostgreSQL 7.3 ejecutándose en cualquier plataforma, así como versiones comerciales de PostgreSQL. Está diseñado para responder a las necesidades de todos los usuarios, desde escribir consultas SQL simples hasta desarrollar bases de datos complejas. La interfaz gráfica soporta una gran variedad de características de PostgreSQL y facilita enormemente la administración de bases de datos (31).

1.13. Conclusiones Parciales

Después de haber realizado un estudio minucioso de las herramientas pertinentes que pueden ser candidatas a utilizar para dar respuesta al problema planteado, se ha determinado la utilización de la herramienta de código abierto Autodock Vina por su alto rendimiento, mayor precisión, y aceleración ventajosa respecto a otros software con el objetivo de realizar acoplamiento molecular y desplegar sobre la Plataforma de Tareas Distribuidas (T-arenal) aprovechando la arquitectura de trabajo distribuido para sus ejecuciones, con el objetivo de aumentar las prestaciones en el Módulo de Acoplamiento Molecular en la PSBio. Se implementará este servicio web sobre el middleware Aixs2 que estará desplegado sobre un servidor de aplicaciones web Apache Tomcat 6.0.14, trabajando con el contenedor de portlet Liferay 6.0.5. Se utilizará OpenUp como metodología de desarrollo de software por presentar un proceso ágil, iterativo e incremental, lo apropiado para proyectos pequeños y utilizado en el proceso de desarrollo de la Plataforma de Servicios Bioinformáticos. Se hará uso del Visual Paradigm como herramienta de modelado para UML 2.0 ya que su licencia es académica y soporta todo el ciclo de vida de desarrollo de software denominado por la Ingeniería de Software. Como gestor de base de datos se utilizará PostgreSQL 8.4, debido a que este presenta un modelo cliente/servidor lo que permitirá almacenar grandes cantidades de datos. La aplicación gráfica para realizar la administración de la base de datos será pgAdmin3, ya que está diseñado para la realización de bases de datos complejas, puede utilizarse en cualquier plataforma y estar bajo licencia libre. Como marco de trabajo se utilizará el Spring 3.1.2 para el desarrollo de la interfaz web por considerarse ligero y ser un software libre. El Hibernate Core 3.3.1 será la herramienta candidata

para el mapeo objeto–relacional por ser un software libre bajo la licencia LGPL. Java será el lenguaje de programación conveniente para el desarrollo del servicio por estar la plataforma a la que va a integrarse está sobre ese lenguaje, orientado a objetos y multiplataforma, que se implementara sobre el IDE de programación Eclipse Helios 3.6 ya que presenta todas las herramientas y funciones convenientes para el desarrollo de un servicio web, además de resultar menos consumidor en la utilización de los recursos de hardware.

Capítulo 2: Análisis

Introducción

En el presente capítulo se realiza una breve descripción del sistema a desarrollar, como el modelo de dominio, los requisitos funcionales y no funcionales del sistema. Se presenta el diagrama de casos de uso para esta aplicación, detallando textualmente el caso de uso más representativo.

2.1. Modelo de Dominio

El modelo del dominio es un paso fundamental en el diseño del software y es utilizado para modelar los objetos del software, muestra clases conceptuales significativas en el dominio del problema, siendo el artefacto más importante que se crea durante el análisis orientado a objetos. Un modelo de dominio es una representación de las clases conceptuales del mundo real, no de componentes software. No se trata de un conjunto de diagramas que describen las clases programables de un software, u objetos software con responsabilidades. A continuación se muestra en la (Fig. 2) el modelo de dominio para el servicio de acoplamiento molecular en la PSBio de la UCI.

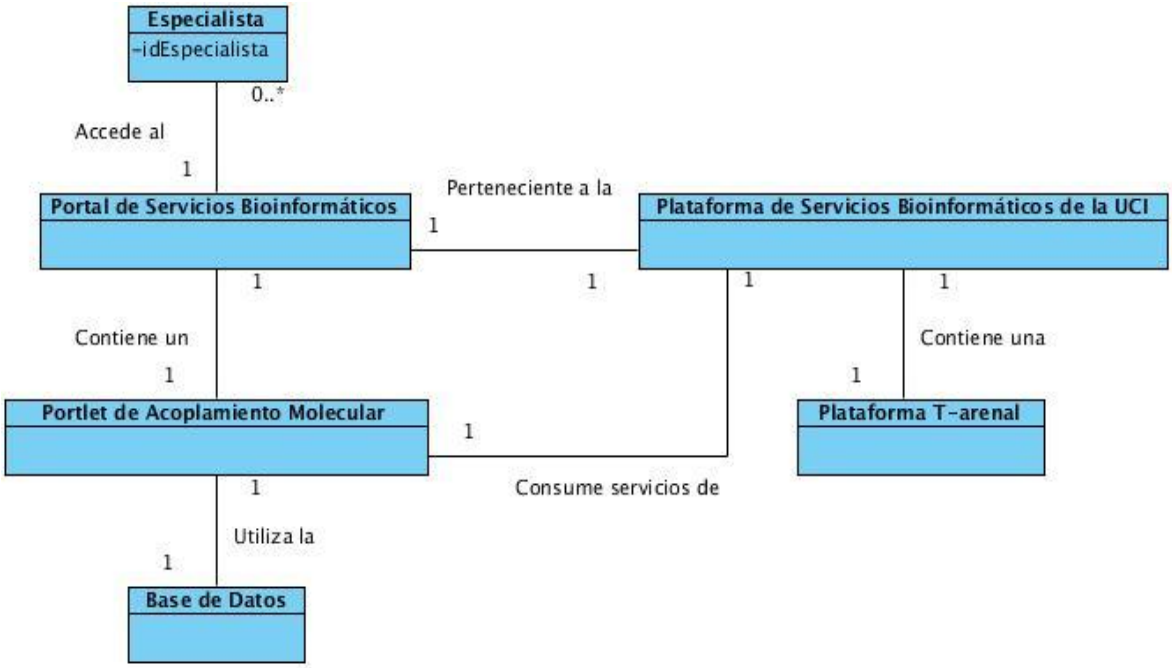

Figura 2: Modelo de Dominio del Sistema. El modelo de dominio es tomado como el punto de partida para el diseño de un sistema.

2.1.1. Definición de las Clases del Modelo de Dominio.

Tabla 1: Definición de las Clases del Modelo de Dominio

Concepto del dominio	Descripción
Especialista	Usuario que posee privilegios para interactuar con la Plataforma de Servicios Bioinformáticos de la UCI.
Plataforma de Servicios Bioinformáticos de la UCI	Plataforma que integra un conjunto de herramientas básicas de propósito general en la Bioinformática.
Portal de Servicios Bioinformáticos	Portal con interfaces gráficas donde los especialistas pueden consumir los servicios que brinda la Plataforma de Servicios Bioinformáticos de la UCI.
Portlet de Acoplamiento Molecular	Portlet que le permite al especialista gestionar sus ejecuciones referidas a consumir el servicio de acoplamiento molecular.
Base de Datos	Base de datos donde serán almacenados los datos necesarios para realizar cada ejecución del servicio de acoplamiento molecular a consumir, así como sus resultados.
Plataforma T-arenal	Plataforma de Tareas Distribuidas implementada y creada en la UCI.

2.2. Requisitos del Sistema

Los requisitos del software son las capacidades o condiciones que el sistema debe cumplir o alcanzar. A continuación se muestran los requisitos, separados en funcionales y no funcionales.

2.2.1. Definición de los Requisitos Funcionales (RF)

Estos requisitos dependen del tipo de software que se desarrolle, de los posibles usuarios del software y del enfoque general tomado por la organización al realizar un levantamiento. Los RF del sistema describen con detalle la función de éste, como así también sus entradas y salidas. En nuestro sistema se identificaron los siguientes:

RF1: Realizar Acoplamiento Molecular con Autodock Vina.

RF2: Crear Proceso de Acoplamiento Molecular con Autodock Vina.

RF3: Modificar Proceso de Acoplamiento Molecular.

RF4: Eliminar Proceso de Acoplamiento Molecular.

RF5: Listar Ficheros de Salida de Autodock Vina.

RF6: Visualizar Resultados de Autodock Vina.

2.2.2. Definición de los Requisitos No Funcionales (RNF).

Los requisitos no funcionales son las propiedades o cualidades que debe tener el sistema y que deben hacer que el mismo sea atractivo, fácil de usar y seguro, con el objetivo de brindarle al usuario final un software que además de ofrecerle todas las funcionalidades que necesite (Requisitos Funcionales), le sea agradable al usar y le brinde un nivel de confianza y seguridad, logrando así un mayor nivel de aceptación. Estos requisitos son divididos, según su clasificación, en las siguientes categorías:

Software:

Del lado del cliente se debe dispensar de los siguientes requisitos no funcionales:

- Al menos un navegador web capaz de interpretar Java Script y CSS compatible con la W3C como Mozilla o Firefox 10.0 o superior.

Por parte del servidor:

- Se debe instalar el servidor web Apache Tomcat 6.0.14 con la aplicación Liferay Portal 6.0.5.
- En el servidor de base de datos se requiere PostgreSQL 8.4 o superior.
- Se requiere la instalación de la máquina virtual de Java 1.6 o superior.

Hardware

Es necesario para la ejecución de una aplicación por el lado del cliente que las estaciones de trabajo de los especialistas requieran de una PC con microprocesador Intel Pentium III o superior, con un mínimo de 512 MB de RAM y tarjeta de red para la conexión con el Portal de Servicios Bioinformáticos de la UCI.

Por el lado del servidor se recomienda poseer microprocesador Intel Pentium IV o superior, a 2.5 GHz o superior, con un mínimo de 1 GB de memoria RAM, un disco duro con capacidad disponible de almacenamiento de 60GB o superior y una tarjeta de red para poder brindar el servicio a través de ella.

Confiabilidad

El sistema asegura la disponibilidad de la información, aun cuando el especialista abandone el sitio o se interrumpa la conexión y tenga tareas en ejecución. Gestionará los errores correspondientes notificando al usuario la causa de este. Requiere de usuario y contraseña para poder acceder al sistema y la información manejada por la misma será objeto de cuidadosa protección contra la corrupción y estados inconsistentes, de la misma forma será considerada la fuente o autoridad de los datos.

Rendimiento

El sistema está desarrollado para un ambiente cliente-servidor por lo que la rapidez del servicio dependerá de la cantidad de ficheros que se necesiten procesar y de la cantidad de procesos que estén ejecutándose o en espera, debido a esto se necesita la integración de este servicio a la Plataforma de Tareas Distribuidas T-arenal para posibles ejecuciones con gran cantidad de ficheros para un proceso.

Usabilidad

El sistema podrá ser usado por aquellos usuarios que posean conocimientos básicos en el campo de la Bioinformática y en el uso de aplicaciones web.

Apariencia o interfaz externa

La interfaz de usuario estará desarrollada mediante portlets siguiendo las especificaciones JSR 168 y 286, contando con una interfaz amigable que permita al especialista la interacción de forma cómoda, le agilice y facilite el trabajo con el sistema.

Soporte

Una vez terminado el sistema se agregará al Portal de Servicios Bioinformáticos de la UCI para realizar las pruebas piloto y de despliegue, luego quedará instalada en dicha plataforma para ser utilizada.

Restricciones en el diseño y la implementación

El análisis y el diseño de la aplicación serán guiados por la metodología OpenUp utilizando el lenguaje de modelado UML. Como herramienta CASE se utilizara el Visual Paradigm para el modelado y como lenguaje de programación Java.

2.3. Definición del Actor del Sistema

Los actores representan terceros fuera del sistema que interactúan con este. En la tabla 2 se hace la descripción del actor para este sistema:

Tabla 2: Descripción del actor y el rol que juega durante la interacción con el sistema.

Actor	Descripción
Especialista	Rol con conocimientos básicos en Bioinformática que podrá ejercer un conjunto de tareas para ejecutar un proceso sobre el Servicio de Acoplamiento Molecular y obtener la respuesta dada por el mismo.

2.4. Casos de Uso del Sistema

La forma en que los actores usan un sistema es representada a través de los casos de uso, los cuales van a ser fragmentos de funcionalidad que ofrece el sistema para aportar un resultado de valor para sus actores, y especifican una secuencia de acciones que el sistema debe llevar a cabo.

Tabla 3: Casos de Uso del Sistema.

Orden.	Caso de Uso	Justificación
1	Realizar Acoplamiento Molecular con Autodock Vina.	Necesario para la realización del servicio de acoplamiento molecular.
2	Administrar Proceso de Acoplamiento Molecular con Autodock Vina.	Necesario para la administración del servicio de acoplamiento molecular.
3	Listar Ficheros de Salida de Autodock Vina	Necesario para mostrar los ficheros de salida de Autodock Vina.
4	Visualizar Resultados de Autodock Vina	Necesario para visualizar los resultados de Autodock Vina.

Listado de casos de usos y sus requisitos funcionales.

CU1 Realizar Acoplamiento Molecular con Autodock Vina.

RF1 Realizar Acoplamiento Molecular utilizando Autodock Vina.

CU2 Administrar Proceso de Acoplamiento Molecular con Autodock Vina.

RF2 Crear Proceso de Acoplamiento Molecular con Autodock Vina.

RF3 Modificar Proceso de Acoplamiento Molecular.

RF4 Eliminar Proceso de Acoplamiento Molecular.

CU3 Listar Ficheros de salida de Autodock Vina.

RF5 Listar Ficheros de salida de Autodock Vina.

CU4 Visualizar Resultados de Autodock Vina.

RF6 Visualizar Resultados de Autodock Vina.

2.4.1. Diagrama de Casos de Uso del Sistema

Un diagrama de casos de uso del sistema contiene actores, casos de uso del sistema y las relaciones existente entre los mismo. En la (Fig. 3) se muestra el diagrama de casos de uso del sistema a desarrollar.

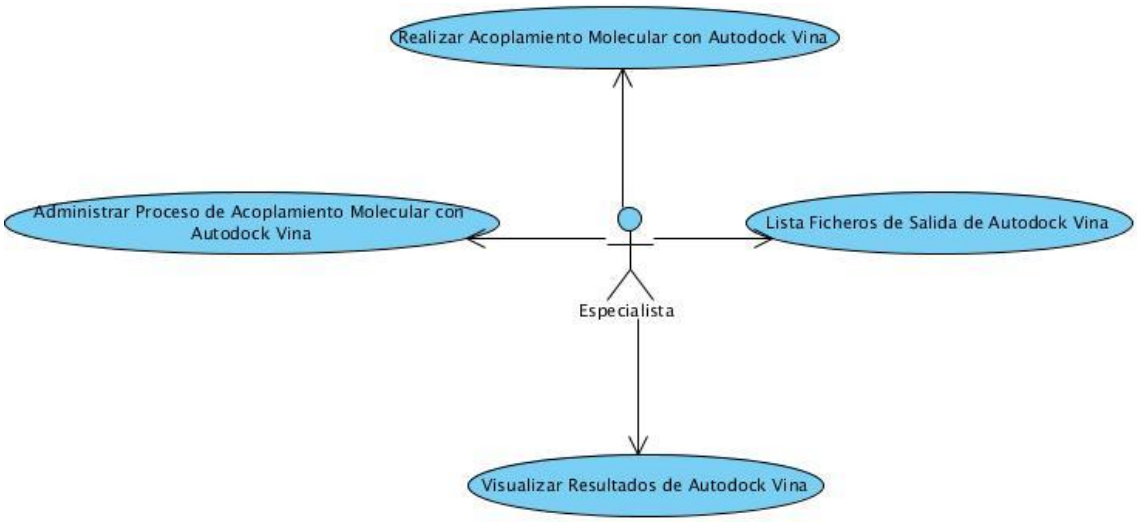

Figura 3: Diagrama de Casos de Uso del Sistema.

2.5. Descripción de los Casos de Uso del Sistema

Cada caso de uso proporciona uno o más escenarios que indican cómo debería interactuar el sistema con el usuario o con otros sistemas para conseguir un objetivo específico. A continuación de describe cómo ocurre el flujo de acciones para el caso de uso "*Administrar Proceso de Acoplamiento Molecular con Autodock Vina*" en el sistema, ya que no se brinda la información detallada en las imágenes gráficas.

Tabla 4: Descripción del flujo de acciones para el caso de uso "Administrar Proceso de Acoplamiento Molecular con AutodockVina".

Nombre del Caso de Uso	Administrar Proceso de Acoplamiento Molecular con Autodock Vina.
Actores:	Especialista
Resumen:	El caso de uso es iniciado cuando el especialista selecciona la opción de "Mis Procesos" en la interfaz principal del Portlet de Acoplamiento Molecular. El sistema muestra la interfaz correspondiente a "Mis Procesos" donde el especialista selecciona la herramienta Autodock Vina para efectuar una ejecución. En el caso de finalizada la ejecución el usuario puede listar los resultados obtenidos o descargarlos.
Referencia:	RF2, RF3, RF4
Precondiciones:	El especialista debe ser autenticado en el Portal.

Poscondiciones:	El sistema crea un proceso de acoplamiento molecular con Autodock Vina.
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del sistema**
1 El especialista selecciona la opción “Mis Procesos” en el portlet de “Acoplamiento Molecular”	1.1 El sistema muestra la interfaz correspondiente y lista los procesos creados anteriormente por el especialista.
2 El especialista escoge una de las siguientes opciones: Crear proceso con Autodock Vina. Modificar proceso con Autodock Vina. Eliminar proceso con Autodock Vina.	Si el especialista escoge: -Crear proceso con Autodock Vina, ver sección Crear proceso con Autodock Vina. -Modificar proceso con Vina, ver sección Modificar proceso con Autodock Vina. -Eliminar proceso con Autodock Vina, ver sección Eliminar proceso con Autodock Vina.
Sección Crear Proceso con Autodock Vina	
1 El especialista selecciona la herramienta Vina en la sección Autodock de la Interfaz “Mis Procesos” correspondiente al módulo.	1.1 El sistema muestra el formulario para que el usuario entre los datos.
2 El especialista entra los siguientes datos: -Nombre del proceso, fichero receptor, ligandos comprimidos (.zip), coordinatethecenterx, coordinatethecentery, coordinatethecenterz, size thedimensionx, size	2.1 El sistema valida los datos insertados por el especialista, de ser correctos, crea un nuevo proceso e inserta los datos del mismo en la Base de Datos.

thedimensiony, size thedimensionz. -El especialista presiona la opción “Aceptar”.	
Sección Flujo alterno Crear Proceso con Autodock Vina	
1 El especialista inserta datos incorrectos.	1.1 El sistema muestra los mensajes de error correspondientes, especificando los campos incorrectos.
Sección Modificar Proceso con Autodock Vina	
1 El especialista selecciona la opción “Modificar proceso” en la ejecución creada anteriormente sin ejecutarse.	1.1 El sistema verifica el proceso a modificar y muestra los campos que el especialista desee actualizar.
2 El especialista modifica los datos del proceso y presiona el botón “Aceptar”.	2.1 El sistema valida los datos insertados por el especialista, de ser correctos, crea un nuevo proceso e inserta los datos del mismo en la Base de Datos.
Sección Fuljo alterno Modificar Proceso con Autodock Vina	
1 El especialista inserta datos incorrectos.	1.1 El sistema muestra los mensajes de error correspondientes especificando los campos incorrectos.
Sección Eliminar Proceso con Autodock Vina	
1 El especialista selecciona la ejecución para ser eliminada en la lista de procesos.	El sistema elimina el proceso creado con anterioridad y los archivos correspondientes del proceso en la Base de Datos.
Prototipo de Interfaz	
	
Prioridad	Crítico

2.6. Conclusiones del Capítulo

En este capítulo se definieron los artefactos correspondientes a los requisitos de la metodología de desarrollo de software OpenUp, utilizada en este sistema, identificando los requisitos funcionales y los no funcionales con los que debe contar el sistema. Se definió el modelo conceptual del sistema y se describieron los objetos de dominio del problema, los RF se agruparon en el modelo de caso de uso del sistema y se realizó una descripción detallada del caso de uso con prioridad "*Critico*" con el objetivo de lograr una mejor comprensión de los mismos.

Capítulo 3: Diseño

Introducción

Una vez comprendido realizado el levantamiento de los requisitos del sistema en el capítulo anterior, se procede a diseñar la solución propuesta. Este capítulo nos brinda un acercamiento a la implementación del sistema, para ello se construye el modelo de clases de diseño, así como los diagramas de interacciones del mismo, los cuales proporcionan la base para proceder a la etapa de construcción.

3.1. Patrones de Desarrollo de Software

Los patrones de desarrollo de software constituyen un conjunto de soluciones a problemas comunes. Estos problemas se encuentran frecuentemente en el desarrollo de aplicaciones de software, por lo que el uso de una solución aplicada anteriormente para resolver un problema parecido puede ser conveniente. Un patrón sería una solución probada para un problema ya conocido en un contexto dado, esta solución debe ser recurrente, logrando así que pueda ser utilizada en otras soluciones.

En la actualidad el uso de los patrones se ha ido intensificando ya que gracias a ellos se puede desarrollar software más resistente a los cambios. Además permiten identificar con facilidad los problemas y la solución conveniente para los mismos.

3.1.1. Patrones Arquitectónicos Utilizados

Los patrones arquitectónicos están orientados al diseño de alto nivel. Están dirigidos a aspectos fundamentales de la estructura de un software. Especifican un conjunto predefinido de subsistemas con sus responsabilidades y un grupo de recomendaciones para organizar los distintos componentes.

Patrón Modelo Vista Controlador (MVC)

El patrón de diseño de software MVC consiste en tres tipos de objetos:

- Modelo: Son los objetos de la aplicación, también conocida como lógica de negocio o lógica de aplicación.
- Vista: Es la que especifica la visualización de los datos, conocida también como lógica de presentación.
- Controlador: Procesa las peticiones que vienen desde la capa de presentación y donde esta encapsulada la lógica del negocio.

La idea fundamental de este patrón es lograr separar responsabilidades entre las personas que trabajan para un proyecto de desarrollo de software; es decir, descomponer el problema en módulos funcionales, (entre ellos el diseño gráfico), lo que se traduce en enfocar de una forma reduccionista la solución de un proyecto de software. En la (Fig. 4) se representa con claridad la función de este patrón arquitectónico.

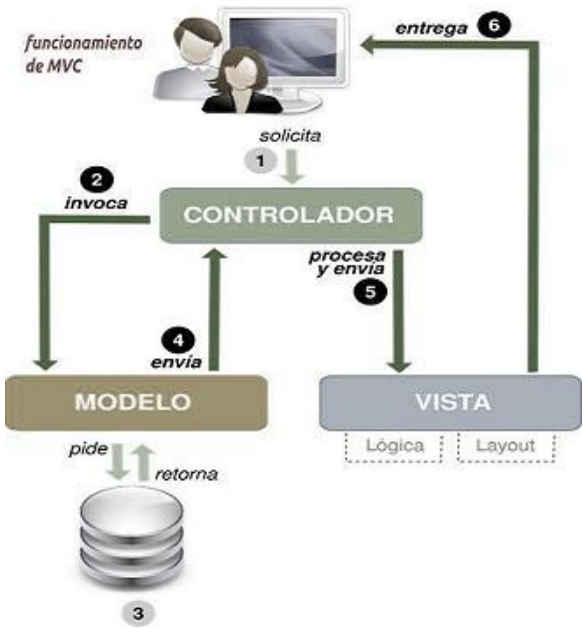

Figura 4: El patrón Modelo Vista Controlador. Es uno de los patrones más usados en las aplicaciones web actuales, mediante este patrón se logra dividir las aplicaciones en partes que hacen más fácil el desarrollo de las mismas.

Para el uso de este patrón en la aplicación, se estructuró por capas de la siguiente forma:

- La capa de vista o de lógica de presentación, estará conformada por las clases JSP, ya que estas son las que estarán interactuando con el cliente (Especialista) en la representación gráfica del servicio de acoplamiento molecular en el Portal de Servicios Bioinformáticos de la UCI.
- La capa de control, estará estructurada por varias clases Java que tendrán la tarea de recibir diferentes peticiones de los especialistas y hacer las llamadas a las funciones convenientes para dar cumplimientos a estas peticiones.
- La capa de datos puede ser implementada a través de componentes que representan objetos en la base de datos, en nuestro caso Plain Old Java Objetc (POJO, por sus siglas en inglés) trabajando con anotaciones Hibernate. Es donde se almacena de forma persistente toda la información

necesaria para facilitar el servicio de acoplamiento molecular en el Portal de Servicios Bioinformáticos.

3.1.2. Arquitectura de Portlet a Desarrollar (Spring Portlet MVC)

Spring Portlet MVC es la arquitectura de portlet que se pretende utilizar para el desarrollo de la interfaz visual de los servicios de acoplamiento molecular, ya que es un módulo de la arquitectura de Spring Framework y provee una serie de recursos que permiten crear portlets con Spring.

Un controlador en Spring es una clase java encargada de procesar una petición del usuario y generar una respuesta. La petición es asignada a través del controlador frontal de Spring. Todos los controladores en Spring retornan un objeto *"ModelAndView"*, en este objeto se guardan el nombre de la página que se mostrará, un modelo de datos (puede ser un Objeto, una Lista, etc.) y un nombre asociado al modelo de datos (identificador).

3.2. Vista Lógica del Sistema

Esta vista nos brinda una base para soportar los requisitos funcionales del sistema, o sea, los servicios que el sistema debe proporcionar. Dentro de la vista lógica se pueden definir diferentes paquetes lógicos que sirven para dividir el sistema en diferentes subvistas o subsistemas.

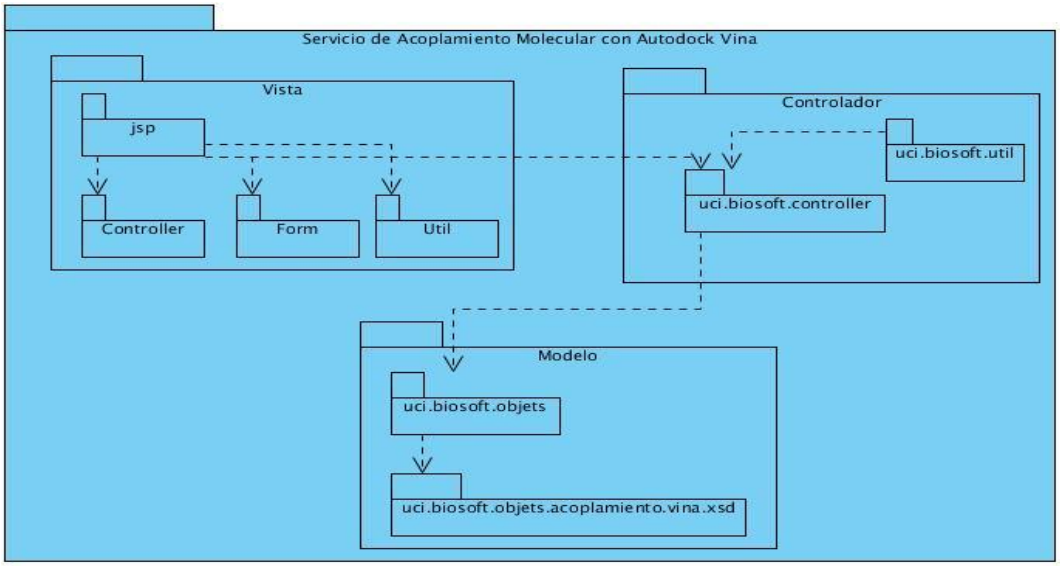

Figura 5: Diagrama de paquetes del diseño. Este diagrama contiene todos los paquetes necesarios del sistema, representando cada capa del patrón arquitectónico Modelo Vista Controlador.

3.3. Vista de Despliegue del Sistema

El diagrama de despliegue nos muestra la disposición física de los distintos nodos que componen el sistema. Representa la disposición de las instancias de los componentes de ejecución en instancias de nodos conectados por enlaces de comunicación.

Sobre el modelo de despliegue deben hacerse las siguientes observaciones:

- Los nodos poseen relaciones que son medios de comunicación entre ellos, tales como TCP/IP, HTTP, USB.
- Cada nodo representa un recurso de cómputo, puede ser un procesador o un dispositivo de hardware.

Seguidamente se muestra el modelo de despliegue del sistema a desarrollar:

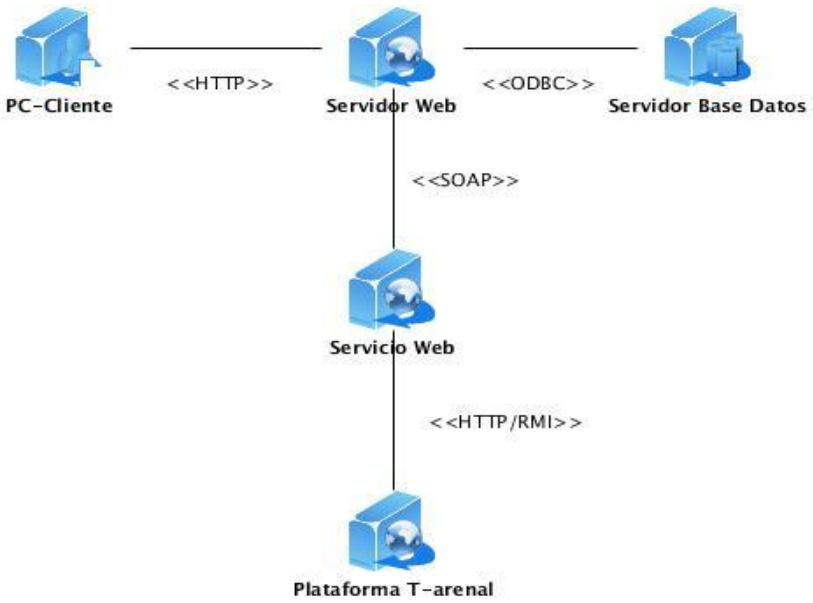

Figura 6: Diagrama de Despliegue del Sistema. Esta vista nos muestra la configuración de hardware del sistema y los nodos físicos que lo componen.

3.4. Patrones GRASP Aplicados:

Los sistemas orientados a objetos se componen de objetos que envían mensajes a otros objetos para que lleven a cabo las operaciones. Una implementación hábil de ellos se funda en los principios cardinales que rigen un buen diseño orientado a objetos. En los patrones GRASP (del inglés, *General Responsibility*

Assignment Software Patterns) se codifican algunos de ellos, que se aplican al diseñar los diagramas de interacción, cuando se asignan las responsabilidades o durante ambas actividades.

Creador: El patrón creador ayuda a identificar quien debe ser el responsable de la creación (o instanciación) de nuevos objetos o clases.

La nueva instancia deberá ser creada por la clase que:

- Tiene la información necesaria para realizar la creación del objeto.
- Usa directamente las instancias creadas del objeto.
- Almacena o maneja varias instancias de la clase.

En la clase *"CreateProcessController.java"* está identificado este patrón arquitectónico ya que es la encargada de crear todos los procesos en el portlet de acoplamiento molecular.

Controlador: El patrón controlador es un patrón que sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es la que recibe los datos del usuario y las que los envía a las distintas clases según el método llamado.

Este patrón sugiere que la lógica de negocios debe estar separada de la capa de presentación, esto para aumentar la reutilización de código y a la vez tener un mayor control.

Se recomienda dividir los eventos del sistema en el mayor número de controladores para poder aumentar la cohesión y disminuir el acoplamiento. Para el desarrollo del portlet se creó un paquete *"controller"* que contiene una serie de clases Java, con el objetivo de controlar con cada una de ellas, cada operación realizada por el especialista en el portlet de acoplamiento molecular, siguiendo los principios del patrón GRASP.

Experto: Este patrón se refleja con claridad en Spring, principalmente en la clase *"Dispatcher.xml"* del portlet donde todas las peticiones son manejadas por un controlador frontal, esta manera de manejar las peticiones es una implementación del patrón con el mismo nombre *"controlador frontal"*. Una vez que el usuario realiza una petición esta llega al controlador frontal y esta a su vez le pasa al controlador encargado de procesarla, cuando el último controlador procesa la petición y retorna un *"ModelAndView"*, el controlador frontal busca la página que se desea mostrar y la envía al usuario.

Alta Cohesión: Una de las características principales de Spring Portlet MVC es la organización de trabajo en cuanto a la estructura del proyecto, lo cual permite crear y trabajar con clases con una alta cohesión. Por ejemplo la clase *"CreateProcessController.java"* es la encargada de crear los procesos del

sistema para una posterior ejecución, la clase *"AdminastratorProcessController.java*" tiene la tarea de ejecutar estos procesos, por otra parte, la *"DownloadProcessController.java*" nos proporciona la función de poder descargar los resultados de cada una de estas ejecuciones; estas clases son las encargadas de controlar alguna de las acciones fundamentales del portlet y por lo tanto pertenecen a la capa controlador dentro de la arquitectura Modelo-Vista-Controlador. Esto hace posible que el software tenga seguridad y flexible a cambios sustanciales con efecto mínimo.

Bajo Acoplamiento: Este patrón esta evidenciado en el framework ya que es mínima la dependencia entre las clases, en otras palabras, comunicándose con el menor número de clases entre ellas, sin que la implementación de cada una de ellas afecte el correcto funcionamiento de la otra.

3.5. Modelo de Diseño

Se puede señalar que el diseño es un moderador del sistema, el cual se centra con más énfasis en los requisitos no funcionales, es decir está enfocado a la manera en que el sistema cumple sus objetivos. Aplicando algunas técnicas y principios con el fin de concretar ya sea un dispositivo, un proceso o un sistema con suficientes detalles para facilitar su interpretación y realización física.

En los diagramas de clases del diseño se observan un conjunto de clases, interfaces y colaboraciones, además de las relaciones que existen entre ellos, siendo estos, la base para los diagramas de componentes y los diagramas de despliegue. Como el sistema que se está modelando es una aplicación web, hay que tener en cuenta que el diagrama de clases en las aplicaciones web es modelado de una manera diferente al de las aplicaciones que se desarrollaban inicialmente, esto se debe a la necesidad de representar las páginas, los enlaces que se establecen entre estas y el contenido dinámico que poseen. Para un mejor entendimiento se hará uso de los estereotipos web; las páginas servidoras que van a ser aquellas que contienen el código que se ejecuta del lado del servidor, las páginas clientes que representan un página web con formato JSP, y los formularios que van a contener los elementos de entrada que son partes de una página cliente. En el diagrama de clases de diseño para el CU *"Administrar Proceso de Acoplamiento Molecular con Autodock Vina"* (Fig. 7) se identifica el patrón de diseño MVC aplicado en el sistema.

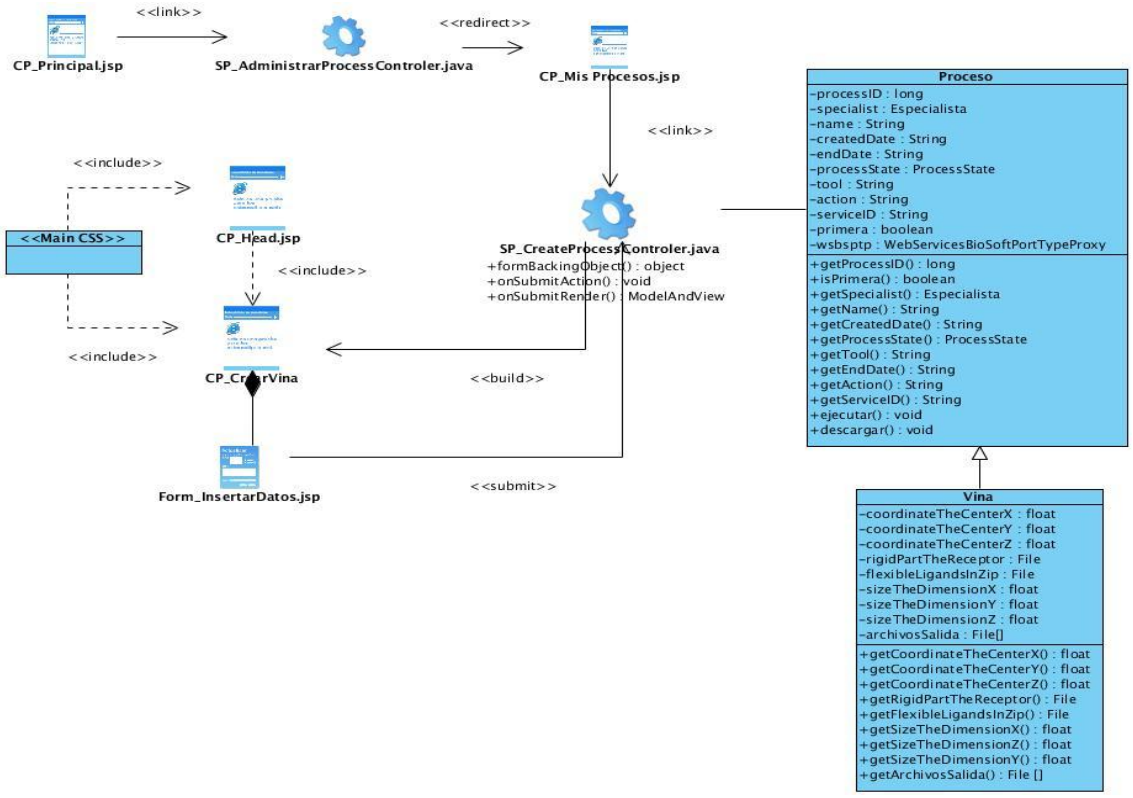

Figura 7: Diagrama de clases de diseño para el CU “Administrar Proceso de Acoplamiento Molecular con Autodock Vina”.

3.5.1. Diagramas de Interacción del Diseño

Los diagramas de interacción se utilizan para modelar los aspectos dinámicos de un sistema lo que conlleva modelar instancias concretas o prototípicas de clases interfaces, componentes y nodos, junto con los mensajes enviados entre ellos, todo en el contexto de un escenario que ilustra un comportamiento. En el presente trabajo se hace el uso de diagramas de secuencia.

Diagramas de secuencia y colaboración

El diagrama de secuencia muestra la interacción entre usuarios, sistemas y subsistemas y hace énfasis en el orden temporal de los mensajes. La figura 8 muestra el diagrama de secuencia para el escenario *“Crear Proceso de Acoplamiento Molecular con Autodock Vina”* donde se define una clase controladora “*AdministratorProcessController.java”* que es la encargada de gestionar las peticiones del usuario y determinar que controlador le dará respuesta a esta petición, en este ejemplo, a “*CrearProcessController.java”.*

Diagramas de colaboración y secuencia para el caso de uso "*Administrar Proceso de Acoplamiento Molecular con Autodock Vina*":

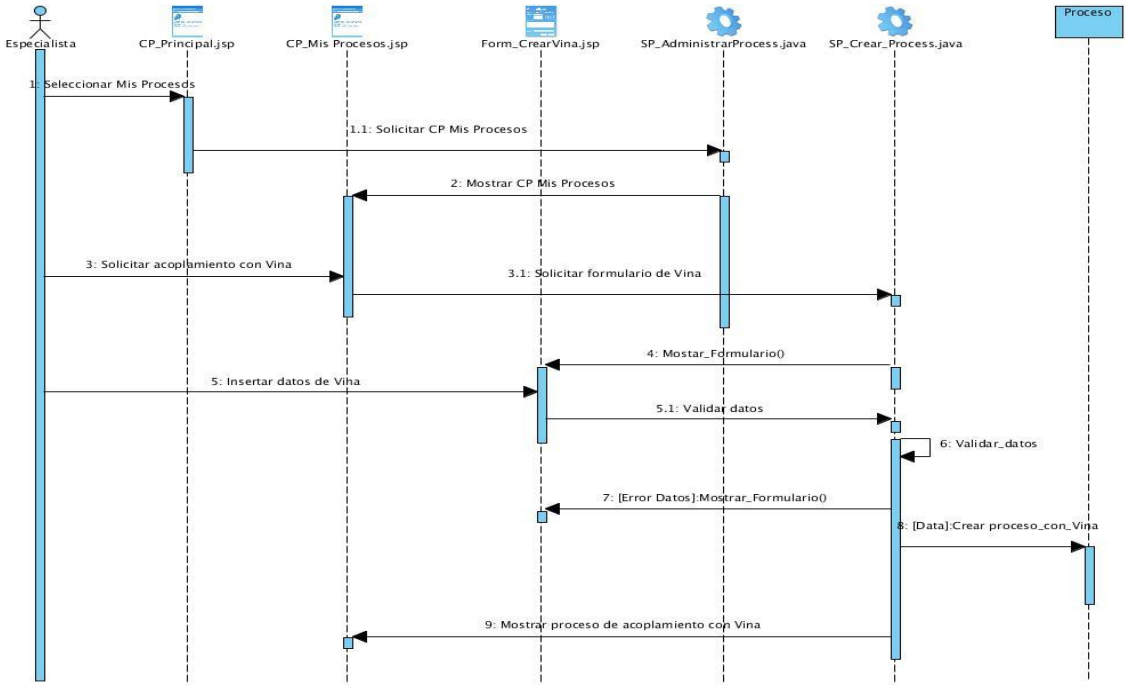

Figura 8: Diagrama de secuencia para escenario "Crear Proceso de Acoplamiento Molecular con Autodock Vina".

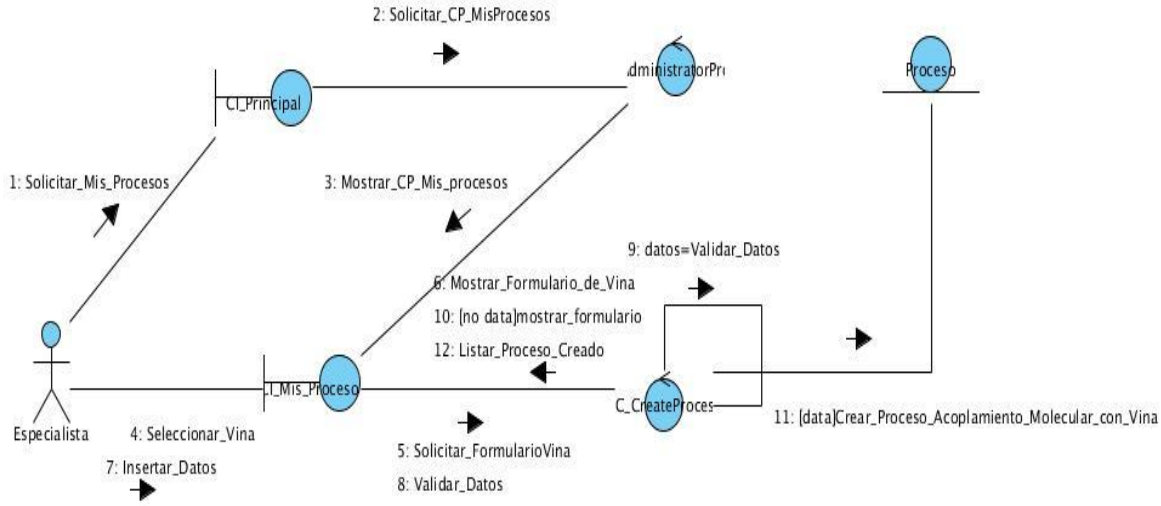

Figura 9: Diagrama de colaboración para el escenario "Crear Proceso de Acoplamiento Molecular con Autodock Vina".

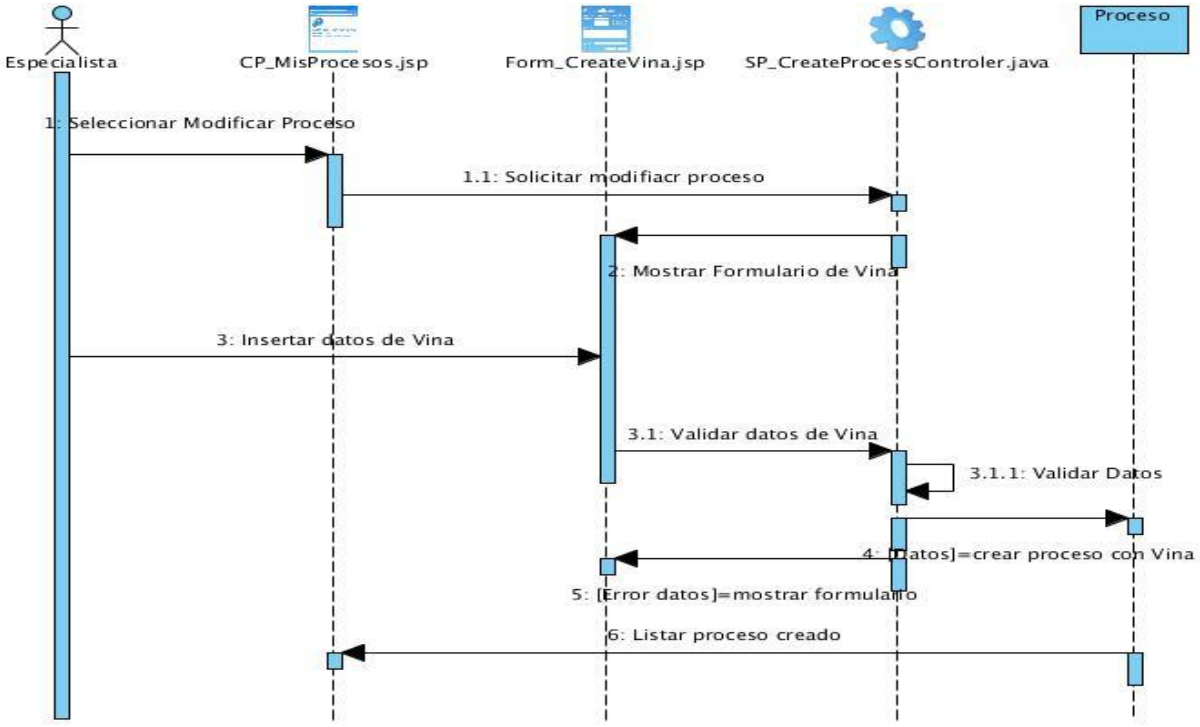

Figura 10: Diagrama de secuencia para el escenario "Modificar Proceso de Acoplamiento Molecular".

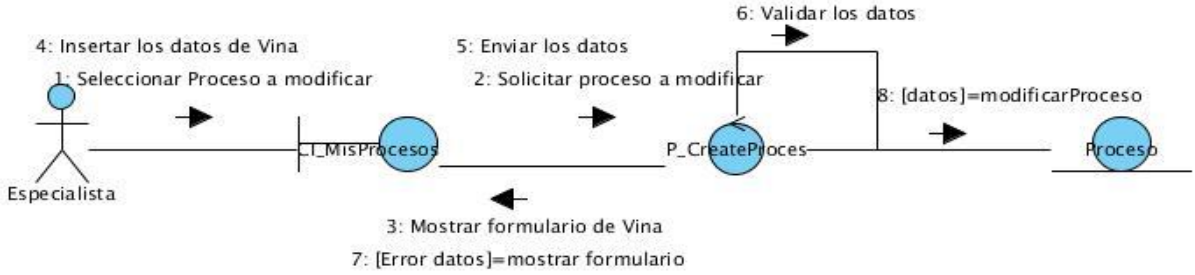

Figura 11: Diagrama de colaboración para el escenario "Modificar Proceso de Acoplamiento Molecular".

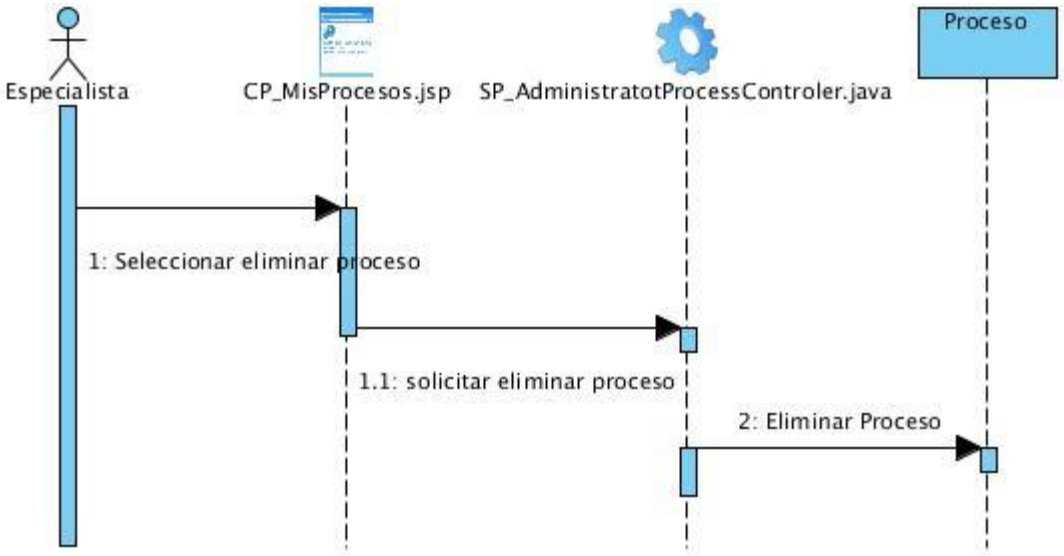

Figura 12: Diagrama de secuencia para el escenario "Eliminar Proceso de Acoplamiento Molecular".

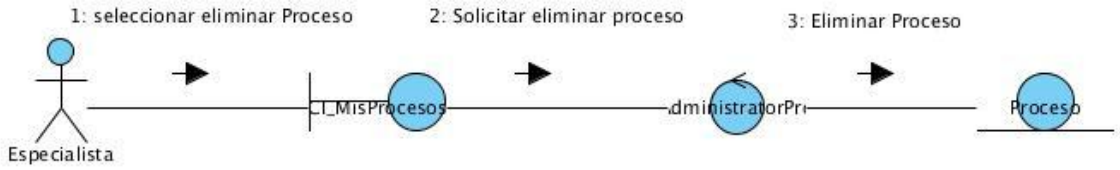

Figura 13: Diagrama de colaboración para el escenario "Eliminar Proceso de Acoplamiento Molecular".

El resto de los diagramas se encuentran en los anexos del trabajo.

3.6. Diseño de Base de Datos

La mayoría de los productos informáticos que se desarrollan actualmente en el mundo, requieren de almacenamiento y gestión de grandes volúmenes de datos. Para ello, se hace necesario el uso de estructuras de datos y con esto, el empleo de una base de datos para almacenar la información.

3.6.1 Modelo de Datos

El modelo de datos describe la representación lógica de las clases persistentes, previendo que la información del sistema sea soportada por una base de datos.

Para el desarrollo de este sistema se hace necesaria la creación de una base de datos para el almacenamiento de los datos de entrada para una correcta ejecución del servicio de acoplamiento molecular con Autodock Vina, así como también el posterior almacenamiento de los ficheros que devuelvan como resultado de este proceso.

En la siguiente figura se muestra cómo están estructuradas las entidades de la base de datos utilizada en el sistema y su relación:

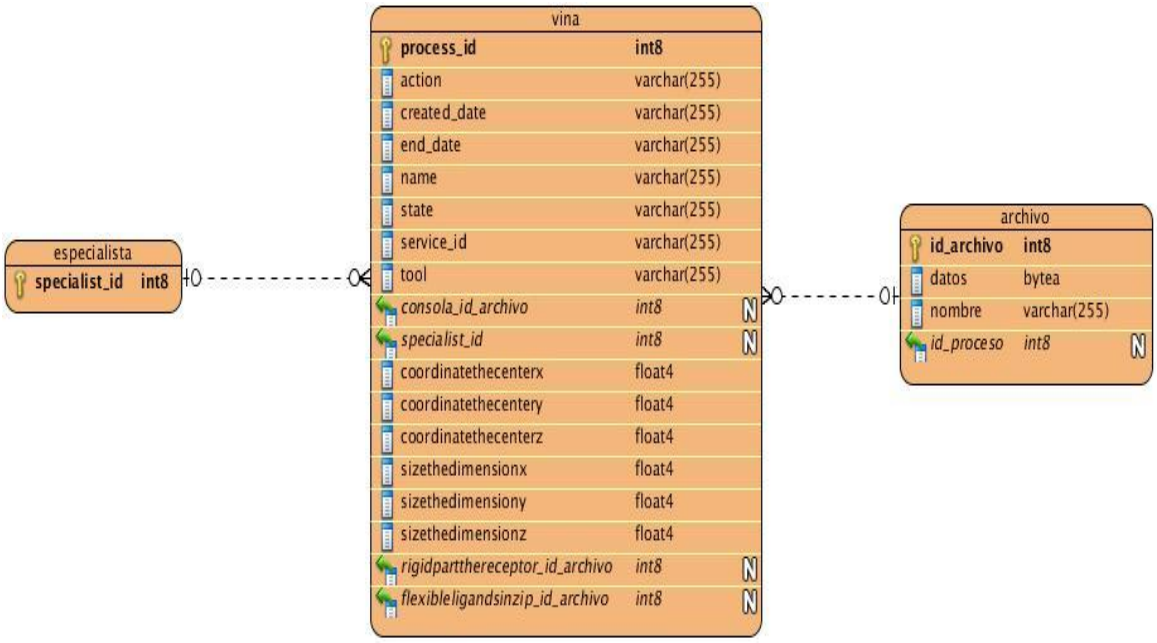

Figura 14: Modelo de Entidad Relación para el almacenamiento de los datos del servicio de acoplamiento molecular con Autodock Vina.

3.6.2. Descripción de las Entidades

Tabla 5: Descripción de la entidad "Vina" del modelo de datos del sistema.

Nombre: Vina		
Descripción: En esta tabla se almacenan los datos del proceso con la herramienta Autodock Vina.		
Atributo	**Tipo**	**Descripción**
process_id	Int8	Almacena el id del proceso.
action	Varchar(255)	Almacena la acción del proceso.
create_date	Varchar(255)	Almacena la fecha de creación del proceso.
end_date	Varchar(255)	Almacena la fecha fin del proceso.
name	Varchar(255)	Almacena el nombre del proceso.
state	Varchar(255)	Almacena el estado del proceso.
service_id	Varchar(255)	Almacena el id del servicio web.

tool	Varchar(255)	Almacena la herramienta del proceso.
consola_id_archivo	Int8	Almacena el id de la consola.
specialist_id	Int8	Almacena el id del especialista.
coordinatethecenterX	Float4	Almacena la coordenada de centro X.
coordinatethecenterY	Float4	Almacena la coordenada de centro Y.
coordinatethecenterZ	Float4	Almacena la coordenada de centro Z.
sizethedimensionX	Float4	Almacena el espacio de dimensión en X.
sizethedimensionY	Float4	Almacena el espacio de dimensión en Y.
sizethedimensionZ	Float4	Almacena el espacio de dimensión en Y.
rigidpartthereceptor_ id_archivo	Int8	Almacena el id del archivo receptor.
flexibleligandinzip_id_ archivo	Int8	Almacena el id del archivo ligando.

Tabla 6: Descripción de la entidad “Archivo” del modelo de datos del sistema.

Nombre: Archivo		
Descripción: En esta tabla se almacenan los datos de la entidad Archivo.		
Atributo	**Tipo**	**Descripción**
id_archivo	Int8	Almacena el id del archivo.
datos	bytea	Almacena el contenido del archivo.
nombre	Varchar(255)	Almacena el nombre del archivo.
id_proceso	Int8	Almacena el id del proceso.

Tabla 7: Descripción de la entidad “Especialista” del modelo de datos del sistema.

Nombre: Especialista		
Descripción: En esta tabla se almacena el dato de la entidad Especialista.		
Atributo	**Tipo**	**Descripción**
specialist_id	Int8	Almacena el id del especialista.

3.7. Conclusiones del Capítulo

En el capítulo que concluye hemos tratado temas relacionados con la arquitectura seleccionada para un correcto diseño de este sistema, se definieron los patrones GRASP aplicados, el modelo de despliegue, donde se muestra la configuración del hardware para este servicio. También se realizó la descripción del modelo de diseño para el caso de uso con prioridad "*Crítico*" del sistema, así como sus diagramas de secuencia y colaboración correspondientes por cada sección. El Modelo Entidad Relación nos permitió obtener un mayor conocimiento de las relaciones entre entidades, así como también se desarrolló la descripción de cada entidad de este modelo mediante tablas, detallando por cada entidad sus atributos, tipo y descripción.

Capítulo 4: Implementación y Prueba

Introducción

Durante el diseño se refinan las clases y sus relaciones con el fin de dar paso a la implantación del sistema. También en esta etapa, se toman las medidas necesarias para lograr un diseño consistente con el entorno en el que se implementará.

De una implementación correcta, que dé la solución que responda a los requisitos planteados, depende en gran medida de las pruebas que se realizarán una vez implementado el sistema, para que se obtengan los resultados esperados por los desarrolladores.

4.1. Implementación para la Ejecución de Autodock Vina en T-arenal

La primera fase de desarrollo de esta aplicación está basada en la integración de la herramienta Autodock Vina en la Plataforma de Tareas Distribuidas T-arenal, con el objetivo de realizar los cálculos de Acoplamiento Molecular sobre varios clientes (PC). Con el uso de esta magnífica herramienta de trabajo desarrollada en la Universidad de las Ciencias Informáticas logramos reducir el tiempo de ejecución de un determinado proceso independientemente de la cantidad de ficheros a procesar a un promedio de 25% con respecto al tiempo total de ejecuciones en una sola estación de trabajo. A continuación se hace una descripción de los algoritmos utilizados en las clases *"DataManeger.java"* y *"Task.java"* para la ejecución de este proceso.

En la clase *"VinaDataManeger.java"* que extiende de la clase *"DataManeger.java"* tuvo la tarea principal de dividir el problema general en subproblemas para posteriormente asignar a cada cliente cada una de estas tareas; cada subproblemas enviado, contiene para su ejecución un fichero que se comportara como el receptor o proteína en el proceso de acoplamiento molecular, un fichero ligando el cual actuará sobre el receptor en este proceso buscando la conformación óptima que hay entre ambos y un fichero configuración donde estarán las coordenadas de posición para ambas moléculas.

En general, el objetivo principal de la implementación de esta clase, es formar por cada ligando a interactuar con un mismo receptor, una tarea a ejecutarse en la clase *"Task.java"*, en la figura 15 se evidencia lo que describimos anteriormente para una mejor comprensión del funcionamiento del algoritmo implementado.

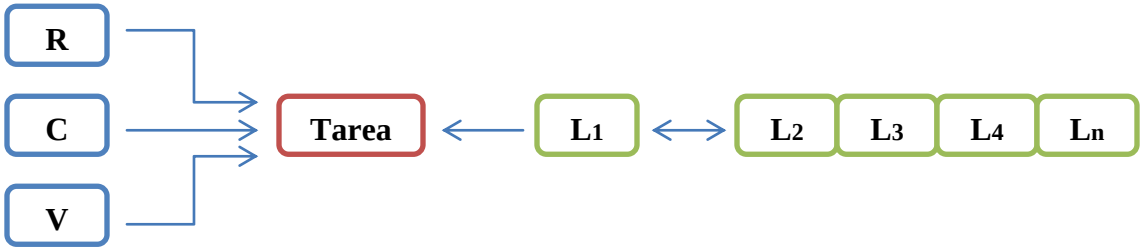

Figura 15: Etapa de dividir el problema general en varios subproblemas en la clase "VinaDataManeger.java" de T-arenal.

Descripción de la figura:

Tarea: Contiene los ficheros necesarios para una ejecución del programa Autodock Vina.

R: Fichero receptor destinado para una ejecución de acoplamiento molecular con Autodock Vina, este fichero no varía para cada tarea creada.

C: Fichero de configuración donde se guardan las coordenadas de posición para una ejecución de acoplamiento molecular con Autodock Vina, este fichero no varía para cada tarea creada.

V: Programa Autodock Vina.

L: Fichero ligando que actuará sobre una proteína en el acoplamiento molecular utilizando el programa Autodock Vina, este fichero varía para cada tarea creada. Por cada ligando existente en el problema se crea una tarea a ejecutar.

Los problemas quedarían formados como se muestran en la figura 16:

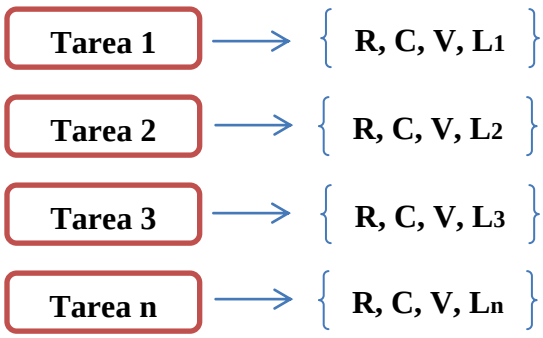

Figura 16: Representación de cada tarea creada y sus ficheros correspondientes para una ejecución. Cada tarea es ejecutada por cada cliente de T-arenal reportado ocioso a su servidor de peticiones.

Como habíamos descrito anteriormente éstas tareas creadas se distribuyen por cada cliente de T-arenal reportado disponible, en la figura que se muestra a continuación se representa lo antes descrito.

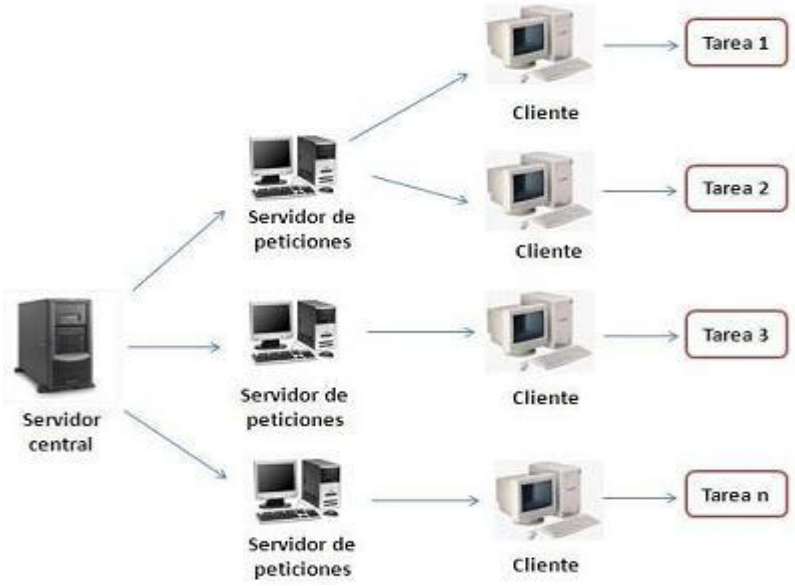

Figura 17: Representación de las tareas distribuidas por los clientes de T-arenal. Cada cliente es encargado de ejecutar una o varias tareas asignadas por su servidor de peticiones.

La clase *"VinaTask.java"* que extiende de la clase *"Task.java"*, es la encargada de recepcionar los ficheros necesarios para una ejecución de Autodock Vina, ejecutar esta tarea y devolver los resultados del acoplamiento molecular; a continuación en la (Fig. 18), se ilustra el algoritmo que hace uso de un comando para ejecutar el programa Autodock Vina en cada cliente de T-arenal.

```
String comand = vina + " --config " + config + " --ligand " + ligand + " --out " + out + " --log " + log;
Process proces = Runtime.getRuntime().exec(comand, null, PROBLEMDIRECTORY);
proces.waitFor();
```

Figura 18: Ejecución del programa Autodock Vina mediante un comando utilizando Java. Este algoritmo es ejecutado en cada cliente de T-arenal destinado al procesamiento de una tarea de acoplamiento molecular con Autodock Vina.

Cuando se ejecuta este método hace llamada al algoritmo *"downloadRequiredData"* que es el encargado de descargar los ficheros correspondientes del *PROBLEMDIRECTORY* del servidor de peticiones hacia el cliente para la ejecución de la misma tarea.

Finalmente, se reciben las respuestas de cada ejecución mediante el método *"processResult"* de la clase "*VinaDataManager.java"* que obtiene el contenido de los ficheros de salida de cada cliente por un vector y los construye posteriormente.

4.2. Diagrama de Componentes

Los diagramas de componentes describen los elementos físicos del sistema y sus relaciones. Muestran las opciones de realización incluyendo código fuente, binario y ejecutable. Los componentes representan todos los tipos de elementos software que entran en la fabricación de aplicaciones informáticas. Pueden ser simples archivos, paquetes o bibliotecas cargadas dinámicamente. El diagrama de componentes es el empaquetamiento físico de los elementos de un modelo, como son las clases en el modelo de diseño.

El diagrama de componentes describe la descomposición física del sistema de software (y eventualmente, de su entorno organizativo) en componentes, a efectos de construcción y funcionamiento. En la (Fig.19) se muestra el diagrama de componente del sistema para el flujo *"Crear Proceso de Acoplamiento Molecular con Autodock Vina"* del CU: *"Administrar Proceso de Acoplamiento Molecular con Autodock Vina"*:

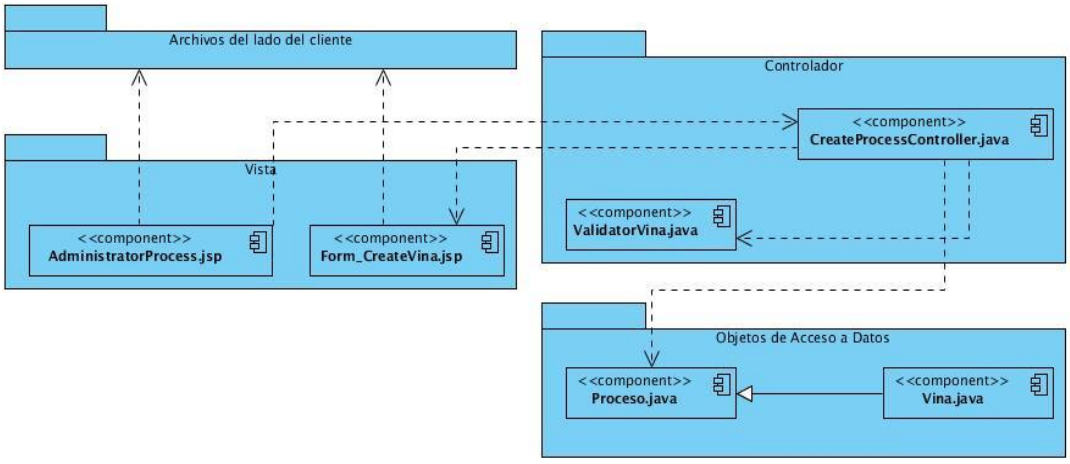

Figura 19: Diagrama de componentes para el CU "Administrar Proceso de Acoplamiento Molecular con Autodock Vina" específicamente para el flujo "Crear Proceso de Acoplamiento Molecular con Autodock Vina".

4.2.1. Descripción de los Componentes

Tabla 8: Descripción de los componentes del sistema.

Componente	Propósito
Archivos del lado del cliente	Archivos necesarios proporcionados por el especialista para crear un proceso de acoplamiento molecular con Autodock Vina.
Form_CreateVina.jsp	Página JSP que mostrará al usuario el formulario que permite

	insertar los datos para realizar el acoplamiento molecular con Autodock Vina.
AdministratorProcess.jsp	Interface principal para gestionar los procesos de acoplamiento molecular.
ValidatorVina.java	Clase implementada en java para la validación de los datos necesarios para la creación de un proceso con Autodock Vina.
Proceso.java	Clase implementada en java, comportándose como clase padre del objeto tipo “Proceso” de donde heredan otras clases procesos, de acoplamiento molecular.
Vina.java	Clase modelo implementada en java donde se construye el proceso tipo “Vina” que realiza acoplamiento molecular utilizando esta herramienta.
createProcessController.java	Clase implementada en java encargada de controlar la creación de un proceso tipo T de acoplamiento molecular.

4.3. Mapa de Navegación

Los mapas de navegación proporcionan una representación esquemática de la estructura del hipertexto, indicando los principales conceptos incluidos en el espacio de la información y las interrelaciones que existen entre ellos. En el mapa que mostraremos en la figura 20, se hará una representación de la estructura del sistema para el funcionamiento de la nueva herramienta de acoplamiento molecular Autodock Vina, con el objetivo de orientar al usuario durante la navegación en el portlet y facilitarle un acceso directo al lugar de su interés. Se reflejará en la imagen, la estructura web por medios de enlaces a los nodos principales, representándolos en varios colores según sus características o funciones.

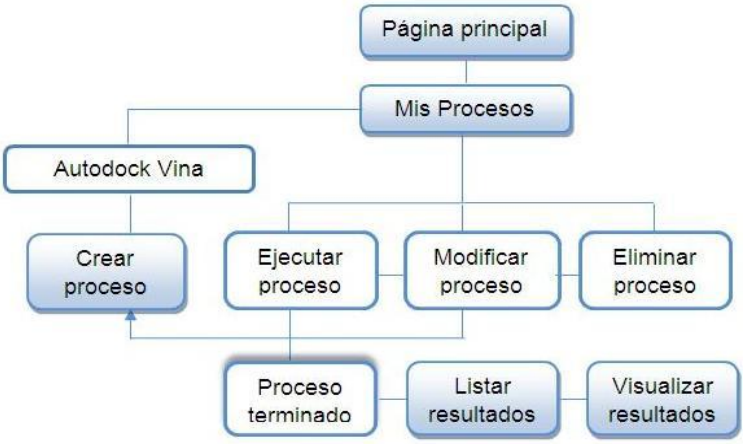

Figura 20: Mapa de navegación del sistema.

Descripción del mapa

Nodos coloreados: Representación de interfaces en el portlet de acoplamiento molecular.

Nodos sin colorear: Representación de tareas en el portlet de acoplamiento molecular.

Nodo sombreado: Representación del estado del proceso.

4.3.1. Interfaz de la Aplicación

Interfaz para crear proceso de acoplamiento molecular con Autodock Vina.

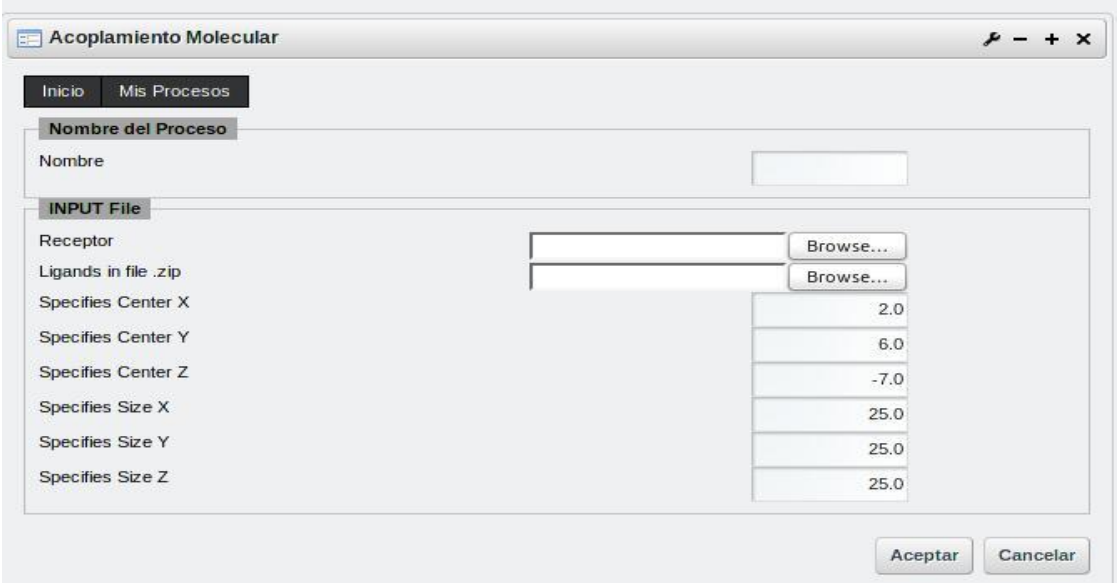

Figura 21: Interfaz de la sección “Crear Proceso de Acoplamiento Molecular con Autodock Vina” del portlet de acoplamiento molecular, perteneciente al Portal de Servicios Bioinformáticos de la UCI.

El resto de las interfaces de usuario se encuentran en el anexo del trabajo.

4.5. Experimentos y Resultados

El experimento constituye uno de los elementos claves de la investigación científica y es fundamental para ofrecer explicaciones causales. El siguiente consiste en pruebas realizadas al sistema sobre su funcionamiento y principalmente con respecto al tiempo de ejecución y obtención de la respuesta del proceso.

Autodock Vina es un programa de acoplamiento molecular el cual está implementado para ejecutarse sobre una PC por un usuario haciendo una ejecución a la vez, por cada ligando interactuando con un receptor. Si tomamos en cuenta que cada ejecución en una máquina con hardware: Pentium IV de 2.50 GHz de velocidad del microprocesador y 1 GB de memoria RAM; demora aproximadamente 25 segundos, con el despliegue de este programa sobre una plataforma de tareas distribuidas la reducción de este tiempo es considerable si el número de ligandos es excesivo.

En este experimento fue utilizada una base de datos de 1, 5 10, 20, 50, 100, 200 y 500 ligandos sobre una proteína, fueron utilizadas las máquinas de los laboratorios de la UCI donde se encuentra desplegado el sistema T-arenal, todas con sistemas operativos basados en GNU/Linux.

Tabla 9: Experimento donde se comparan los tiempos de ejecución del programa Autodock Vina con respecto a ejecutarse sobre una PC cliente y sobre el servicio de acoplamiento molecular.

Cantidad de Ligandos	Una PC(TA)	Servicio de Acoplamiento Molecular	
		Clientes	Tiempo
1	0:00:25	1	0:00:40
5	0:02:08	5	0:01:38
10	0:04:16	7	0:01:45
20	0:08:34	13	0:02:13
50	0:21:10	21	0:03:45
100	0:42:00	27	0:06:10
200	1:23:34	32	0:09:00
500	3:47:00	28	0:20:10

Descripción de las variables:

Cantidad de Ligandos: Número de ligandos a ser procesados utilizando el programa Autodock Vina.

Una PC (TA): Tiempo aproximado en el que se ejecuta una cantidad *N* de ligandos sobre una PC.

Servicio de Acoplamiento Molecular: Servicio web que brinda la posibilidad de realizar acoplamiento molecular con el programa Autodock Vina utilizando la Plataforma de Tareas Distribuidas T-arenal.

Clientes: Cantidad de estaciones de trabajo (PC) utilizadas por T-arenal que se reportaron ociosos, para ejecutar cada tarea de acoplamiento molecular utilizando el programa Autodock Vina.

Tiempo: Tiempo total de procesamiento del servicio de acoplamiento molecular para una cantidad *N* de ligandos sobre otra cantidad *X* de clientes.

4.5.1. Representación Gráfica del Experimento

Una gráfica es una denominación de la representación de datos generalmente numéricos, mediante recursos gráficos (líneas, vectores, superficies o símbolos) donde se manifiesta visualmente la relación matemática o estadística que guardan entre sí, en la (Fig. 22) se hace una representación graficada donde se comparan los resultados obtenidos del experimento y donde se evidencia la solución al problema planteado en la introducción de este trabajo, ya que logró reducir el 75% del tiempo de respuesta de una simulación molecular excesiva representado en la tabla 9.

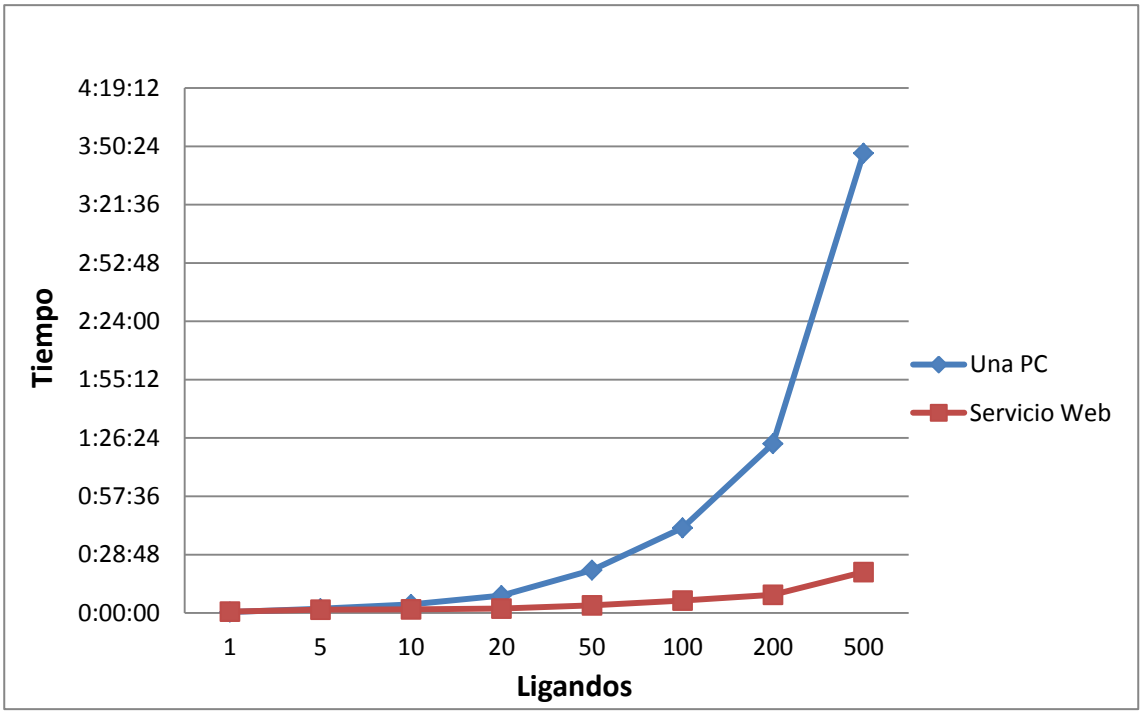

Figura 22: Tiempo en obtener el resultado de los acoplamientos moleculares para los set de ligando empleados a ejecutarse sobre una máquina y el servicio web.

Representación Horizontal: Cantidad de ligandos que fueron procesados.

Representación Vertical: Tiempo en horas, minutos y segundos que demora en ejecutarse un proceso de acoplamiento molecular con Autodock Vina con una *N* cantidad de ligandos.

4.5.2. Prueba de Speed-up

Para obtener ganancia de velocidad del algoritmo en función del tamaño del problema y el número de procesadores utilizados para su ejecución, se aplicó la prueba de *Speed-Up*, la cual se calcula mediante la fórmula:

Speed-up = Ts/TnCpus

Donde**Ts** (Tiempo Secuencial) = Tiempo de procesamiento en una PC y **TnCpus** = Tiempo de procesamiento en las *N* Pc utilizadas.

Para esta prueba se utilizaron una cantidad de 300 ligandos para ser procesados sobre otra cantidad de 1, 10, 20, 50, 80 y 100 de clientes de T-arenal, para la ejecución de cada una de ellas, todas con la misma cantidad de ficheros de entrada. La tabla que se muestra a continuación contiene los resultados de esta prueba.

Tabla 10: Speed-up de acoplamiento molecular en el Servicio de Acoplamiento Molecular con Autodock Vina.

Cantidad de PCs	Tiempo de Ejecución	Speed-Up
1	7:53:20	1
10	0:37:33	11.21
20	0:16:35	25.38
50	0:10:07	42.01
80	0:10:34	39.83
100	0:11:56	35.26

Si representamos en una gráfica la tabla anterior podremos apreciar que el *Speed-up* del algoritmo logra su valor óptimo a medida que la cantidad de PCs fueron incrementándose mientras se mantuvieron por debajo de 50, a partir de este valor el *Speed-up* del algoritmo tendió a disminuir la velocidad de procesamiento. En la figura 23 se ilustra con más claridad.

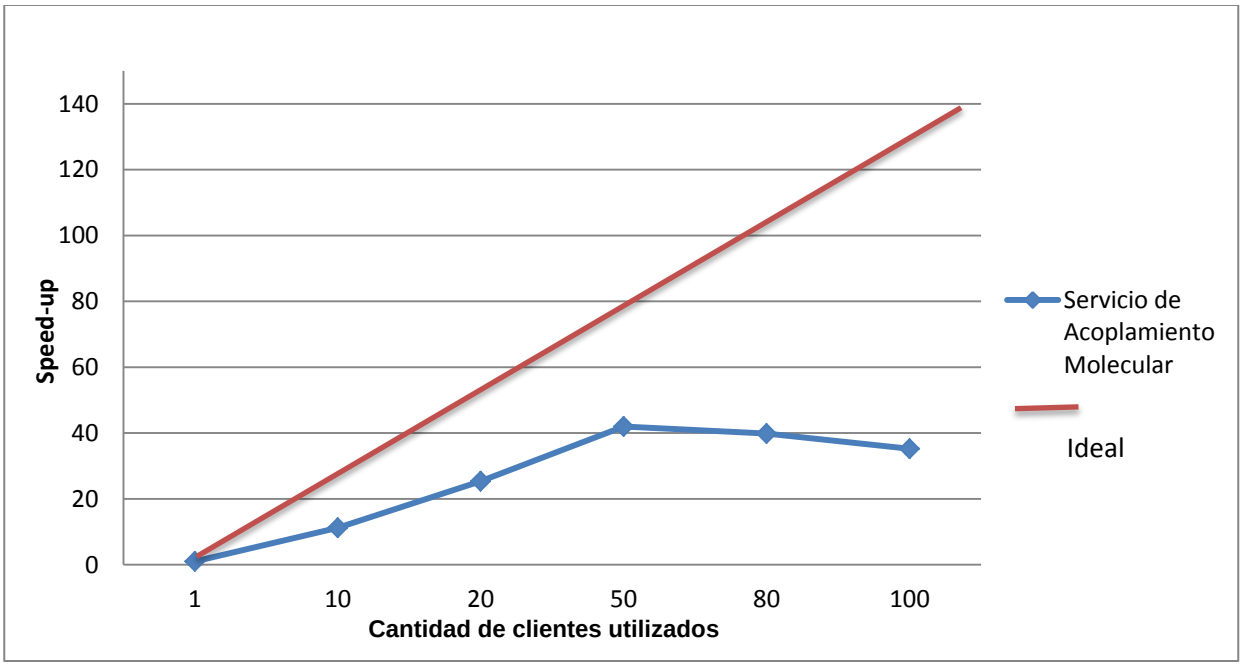

Figura 23: Speed-up del Servicio de Acoplamiento Molecular con Autodock Vina.

La disminución de la aceleración de procesamiento del algoritmo se debe a que independientemente de la cantidad de procesadores que se utilicen, cuando se obtienen los recursos necesarios para ejecutar esta tarea, que en este caso se logró una maseta de 50 máquinas, el incremento de otros clientes provoca que disminuya la velocidad de procesamiento por el intercambio de datos mediante la red.

4.6. Pruebas de Caja Negra

Las Pruebas de Caja Negra son pruebas que se desarrollan en aplicaciones con interfaces gráficas. Las mismas se encargan de verificar que las funciones que debe desempeñar el sistema son operativas. Se centran en los requisitos funcionales del software, sin interesarse por el funcionamiento interno del mismo. La realización de estas pruebas permite encontrar:

- Funciones incorrectas.
- Errores de interfaz.
- Errores en las salidas.
- Errores en el acceso a datos.

4.6.1. Casos de Pruebas de Caja Negra

Los casos de pruebas se realizan para validar el correcto funcionamiento de la aplicación sobre un conjunto de entradas, condiciones de ejecución y los resultados esperados, con el objetivo de verificar el correcto funcionamiento de un objetivo específico del sistema.

Antes de ejecutar un caso de prueba es necesario definir una serie de variables que serán utilizadas. En el ejemplo que se demuestra a continuación para la sección *"Crear Proceso de Acoplamiento Molecular con Autodock Vina"* correspondiente al CU: *"Administrar Proceso de Acoplamiento Molecular con Autodock Vina"* se definieron nueve variables que se describen en la siguiente tabla:

Tabla 11: Variables para el caso de prueba "Crear Proceso de Acoplamiento Molecular con Autodock Vina".

No.	Nombre del Campo	Clasificación	Valor nulo	Descripción
1	Nombre	Campo de texto	No	Secuencia de caracteres a-zA-Z0-9.
2	Receptor	Campo de entrada de Fichero	No	Fichero de bytes con extensión (.pdbqt).
3	Ligands	Campo de entrada de Fichero	No	Fichero de bytes, comprimido en .zip.
4	Center Y	Campo numérico	No	Número distinto de 0.
5	Center X	Campo numérico	No	Número distinto de 0.
6	Center Z	Campo numérico	No	Número distinto de 0.
7	Size X	Campo numérico	No	Número mayor que 0.
8	Size Y	Campo numérico	No	Número mayor que 0.
9	Size Z	Campo numérico	No	Número mayor que 0.

A continuación en la tabla 11 se muestra el comportamiento del sistema a medida que vamos cambiando los datos de entrada para el sistema. Como las validaciones para las coordenadas "Center X, Y, Z" son iguales y no hacer de esta tarea algo tedioso, estas variables las trataremos en conjunto como "Center" para el caso de prueba que se demuestra a continuación, esto sucede también con las coordenadas "Size X, Y, Z" donde las validaciones son totalmente iguales y lo tomaremos en conjunto como "Size":

Tabla 12: Caso de prueba: “Crear Proceso de Acoplamiento Molecular con Autodock Vina”

Escenario	Descripción	Nombre	Ligands	Receptor	Center	Size	Respuesta del sistema	Flujo central
EC 1.1 Introducir los datos para un proceso de acoplamiento con Autodock Vina.	Se llenan los campos correctamente y se oprime el botón “Aceptar”.	V: P1	V: A1.zip Anexo A. Fig. 24	V: 1nhz_protein.pdbqt Anexo A. Fig. 25	V: 2; 6; -7	V: 25; 25; 25	Se crea el proceso y se lista en la interfaz “Mis Procesos”.	1- Acoplar con Autodock Vina. 2- Se llenan los datos. 3- Se crea.
EC 1.2 Introducir valores en blanco.	Se desea crear el proceso dejando campos en blanco.	I: ( )	V: A1.zip Anexo A. Fig. 24	V: 1nhz_protein.pdbqt Anexo A. Fig. 25	V: 2; 6; -7	V: 25; 25; 25	Se muestra un mensaje de error en el campo vacío.	1- Acoplar con Autodock Vina. 2- Se llenan los datos. 3- Se crea.
EC 2.1 Introducir fichero con extensión errónea para el campo Receptor.	Se desea crear el proceso introduciendo un fichero receptor inadecuado.	V: P1	V: A1.zip Anexo A. Fig. 24	I: Protein a.txt	V: 2; 6; -7	V: 25; 25; 25	Se muestra un mensaje de error en el campo que se introdujo un fichero erróneo.	1- Acoplar con Autodock Vina. 2- Se llenan los datos. 3- Se crea.
EC 3.1 Introducir datos menores que 0 para el campo Size.	Se desea crear el proceso introduciendo datos numéricos menores que 0 para el campo Size.	V: P1	V: A1.zip Anexo A. Fig. 24	V: 1nhz_protein.pdbqt Anexo A. Fig. 25	V: 2; 6; -7	I: 4; 25; -8	Se muestra un mensaje de error en el campo que se introdujo los datos incorrectos.	1- Acoplar con Autodock Vina. 2- Se llenan los datos. 3- Se crea.

4.7. Conclusiones del Capítulo

En este capítulo se realizó un breve análisis sobre la implementación de los algoritmos desarrollados en las clases *"VinaDataManeger"* y *"VinaTask"* de T-arenal que tienen como objetivo general la división y ejecución de varias tareas de acoplamiento molecular con Autodock Vina, se realizó el diagrama de componentes del CU: *"Administrar Proceso de Acoplamiento Molecular con Autodock Vina"*, específicamente para la sección *"Crear Proceso de Acoplamiento Molecular con Autodock Vina"*, en el que se definieron los componentes del sistema necesario para el correcto funcionamiento de este CU y fueron descritos cada uno de ellos para lograr una mejor comprensión acerca de su función.

Se definieron los casos de prueba de caja negra para el CU: *"Administrar Proceso de Acoplamiento Molecular con Autodock Vina"*, los que se aplicaron a la sección: *"Crear Proceso de Acoplamiento Molecular con Autodock Vina"*. Estas pruebas se realizaron a través de la interfaz del sistema y se detectaron inconformidades como campos no validados, todas ellas solventadas rápidamente mediante la corrección de los objetos correspondientes.

Conclusiones

Una vez concluida la investigación podemos resumir que se integró la herramienta Autodock Vina al Módulo de Acoplamiento Molecular de la Plataforma de Servicios Bioinformáticos de la UCI. Para ello:

- Se implementó un servicio web haciendo el uso de la Plataforma de Tareas Distribuidas T-arenal para realizar el acoplamiento molecular utilizando la herramienta Autodock Vina e integrándola al Portlet de Acoplamiento Molecular del PSBio de la UCI.
- Se logró reducir significativamente el tiempo de respuesta de una ejecución X utilizando un conglomerado de computadoras como clientes de la Plataforma de Tareas Distribuidas T-arenal.
- Se realizó el diseño de la base de datos para almacenar los ficheros de entrada y de respuesta independientemente de la cantidad que sea necesaria guardar.
- Se realizaron y se ejecutaron los casos de prueba para validar el servicio web de Acoplamiento Molecular con la nueva herramienta insertada Autodock Vina.

Recomendaciones

Una vez obtenidos los resultados del acoplamiento molecular con la herramienta Autodock Vina se recomienda insertar en el portlet de este módulo un sistema que muestre gráficamente las mejores poses de salida ligando-receptor.

Bibliografía

1. The scripps research Institute, Autodock Vina. [Online] http://www.autodock.scripps.edu.

2. DockingServer: molecular docking calculations online. [Online] http://www.ncbi.nlm.nih.gov.

3. Docking.org. [Online] http://www.docking.org.

4. Ing. Cesar Raul García Jacas y Ing. Longendri Aguilera Mendoza. Manual del Usuario de la Plataforma de Tareas Distribuidas T-arenal. 2010.

5. NIH National Institutes of Health, Autodock Vina. The Scripps Research Institute. Olson, Oleg Trott and Arthur J. 2010.

6. Simulación del reconocimiento entre proteínas y moléculas orgánicas o Docking. Aplicación al diseño de fármacos. Jaqueline Padilla Zúñiga y Arturo Rojo Domínguez. Mexico D.F : Universidad Autónoma Metropolitana-Iztapalapa, 2002.

7. The scripps research Institute, Autodock. [Online] http://www.autodock.scripps.edu.

8. The scripps research Institute, Autodock. [Online] http://www.autodock.scripps.edu.

9. Hernández, Osvel Chávez. Servicio de Acoplamiento Molecular para la Plataforma de Servicios Bioinformáticos de la UCI. La Habana : s.n, 2012.

10. AutoDock Vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization and multithreading. Olson, Oleg Trott and Arthur J. 2010.

11. Maikel Zúñiga Súarez, Keiler Arnaldo González Torres. Tesis de Postgrado. Integración de la Plataforma T-arenal con el middleware Globus Toolkit 4. La Habana-Cuba : s.n., 2008.

12. CLUSTERS. [Online] http://clusterfie.epn.edu.ec/clusters/.

13. Ing. Cesar Raul Garcia Jacas e Ing. Longendry Aguilera Mendoza. Manual del usuario de la Plataforma de Tareas Distribuidas T-arenal.

14. The Apache Software fundation . Apache tomcat. [Online] http://www.tomcat.apache.org.

15. Apache Portals. [Online] http://www.portals.apache.org.

16. Liferay Portal 4 - Portlet development guide. Joseph Shum, Alexander Chow, Jorge Ferrer, Ed Shin. 2007.

17. 0208_Proyecto Ténico. Plataforma de Servicios Bioinformáticos. La Habana.

18. Roughley, I.. Starting Struts 2. [Online] http://infoq.com/minibooks/starting-struts2.

19. Hibernate Quickly. Quickly. Greenwich, Londres, U.K.. Manning Publications. Peak, P. and Heudecker. 2006.

20. Axis2, Middleware for Next Generation Web Services. IEEXplore. [Online] http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=4032100&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D4032100.

21. Aranda, Vicente Trigo. Historia y evolución de los lenguajes de programación. Entorno Virtual de Aprendizaje. [Online] http://eva.uci.cu/file.php/136/IP/Bibliografia/Articulos/História_y_evolución_de_los_lenguajes_de_programación.pdf.

22. Alvarez, M. A. ¿Qué es el PHP? [Online] 2001. http://www.desarrolloweb.com/articulos/831.php.

23. Entorno de Desarrollo Integrado. [Online] http://petra.euitio.uniovi.es/~i1667065/HD/documentos/Entornos%20de%20Desarrollo%20Integrado.pdf.

24. Eclipse. [Online] http://www.eclipse.org/projects/.

25. Roberth G. Figueroa, Camilo J. Solis, Armando A.Cabrera. Metodologías Tradicionales Vs Metodologías Ágiles. [Online] http://eva.uci.cu/mod/resource/view.php?id=9303&subdir=/Metodologías.

26. Introduction to OpenUP. OpenUp. [Online] http://www.epf.eclipse.org/wikis/openup/.

27. Introduction to UML. Unified Modeling Language. [Online] http://www.uml.org.

28. Pascual González López, Ana Amelia González López, josé Antonio Gallud Lázaro. Herramientas CASE. ¿Cómo incorporarlas con éxito en nuestra organización? [Online] http://www.uclm.es/ab/educacion/ensayos/pdf/revista10/10_17.pdf.

29. Visual Paradigm. Boost Productivity with Innovative and Intuitive technologies. [Online] http://www.visual-paradigm.com/aboutus/.

30. Portal en español sobre postGresql. PostgreSQL.es. [Online] http://www.postgresql.org.es/sobre_postgresql.

31. Guía Documentada para Ubuntu. PgAdmin III. [Online] http://www.guia-ubuntu.org/index.php?title=PgAdmin_III.

Anexos

A. Ficheros utilizados para las pruebas de Caja Negra

```
REMARK  0 active torsions:
REMARK  status: ('A' for Active; 'I' for Inactive)
ROOT
HETATM    1  C             1       0.000   0.000   0.000  0.00  0.00     -0.031 C
HETATM    2  C             2       1.525   0.000   0.000  0.00  0.00      0.018 C
HETATM    3  C             3       1.603   2.643   0.000  0.00  0.00      0.000 C
HETATM    4  C             4       2.236   3.922  -0.623  0.00  0.00      0.003 C
HETATM    5  C             5       3.789   3.835  -0.612  0.00  0.00      0.028 C
HETATM    6  C             6       4.184   2.562  -1.419  0.00  0.00      0.003 C
HETATM    7  C             7       5.702   2.738  -1.668  0.00  0.00      0.003 C
HETATM    8  C             8       5.873   4.273  -1.854  0.00  0.00      0.033 C
HETATM    9  C             9       4.509   4.925  -1.464  0.00  0.00      0.152 C
HETATM   10  C            10       4.328   3.874   0.859  0.00  0.00      0.003 C
HETATM   11  C            11       1.951  -0.135   1.501  0.00  0.00      0.004 C
HETATM   12  C            12      -0.748  -1.096  -0.234  0.00  0.00     -0.031 C
HETATM   13  C            13      -0.180  -2.447  -0.600  0.00  0.00      0.026 C
HETATM   14  C            14       1.358  -2.539  -0.441  0.00  0.00      0.005 C
HETATM   15  C            15       2.009  -1.204  -0.888  0.00  0.00      0.004 C
HETATM   16  C            16       3.557  -1.303  -0.955  0.00  0.00      0.000 C
HETATM   17  C            17       4.150   0.004  -1.540  0.00  0.00      0.000 C
HETATM   18  C            18       3.680   1.256  -0.743  0.00  0.00      0.000 C
HETATM   19  C            19       2.106   1.313  -0.662  0.00  0.00      0.004 C
HETATM   20  O            20       4.672   6.165  -0.760  0.00  0.00     -0.224 C
ENDROOT
TORSDOF 0
```

Figura 24: Fichero ligando utilizado para las pruebas de Caja Negra.

```
REMARK   4 XXXX COMPLIES WITH FORMAT V. 2.0
ATOM      1  N   PRO A 530       1.416  42.867 -14.065  1.00 27.45     -0.038 N
ATOM      2  HN1 PRO A 530       1.659  43.838 -13.867  1.00  0.00      0.280 HD
ATOM      3  HN2 PRO A 530       2.272  42.590 -14.545  1.00  0.00      0.280 HD
ATOM      4  CA  PRO A 530       1.203  42.082 -12.813  1.00 27.02      0.259 C
ATOM      5  C   PRO A 530       2.482  41.507 -12.229  1.00 26.16      0.259 C
ATOM      6  O   PRO A 530       3.293  40.917 -12.935  1.00 27.30     -0.271 OA
ATOM      7  CB  PRO A 530       0.262  40.961 -13.272  1.00 27.27      0.042 C
ATOM      8  CG  PRO A 530      -0.554  41.559 -14.389  1.00 26.96      0.035 C
ATOM      9  CD  PRO A 530       0.236  42.766 -14.946  1.00 28.10      0.210 C
ATOM     10  N   THR A 531       2.666  41.692 -10.930  1.00 25.10     -0.336 N
ATOM     11  HN  THR A 531       2.036  42.308 -10.416  1.00  0.00      0.164 HD
ATOM     12  CA  THR A 531       3.756  41.027 -10.224  1.00 23.96      0.186 C
ATOM     13  C   THR A 531       3.549  39.531 -10.221  1.00 23.04      0.253 C
ATOM     14  O   THR A 531       2.427  39.020 -10.347  1.00 22.49     -0.270 OA
ATOM     15  CB  THR A 531       3.788  41.412  -8.762  1.00 24.05      0.140 C
ATOM     16  OG1 THR A 531       2.556  40.968  -8.164  1.00 21.96     -0.382 OA
ATOM     17  HG1 THR A 531       2.576  41.210  -7.246  1.00  0.00      0.210 HD
ATOM     18  CG2 THR A 531       3.853  42.961  -8.575  1.00 24.17      0.034 C
ATOM     19  N   LEU A 532       4.646  38.853  -9.960  1.00 22.16     -0.337 N
ATOM     20  HN  LEU A 532       5.501  39.363  -9.736  1.00  0.00      0.164 HD
ATOM     21  CA  LEU A 532       4.677  37.424  -9.980  1.00 21.83      0.159 C
ATOM     22  C   LEU A 532       3.972  36.824  -8.775  1.00 21.13      0.251 C
```

Figura 25: Fichero receptor utilizado para las pruebas de Caja Negra.

B. Diagramas de Clases de Diseño

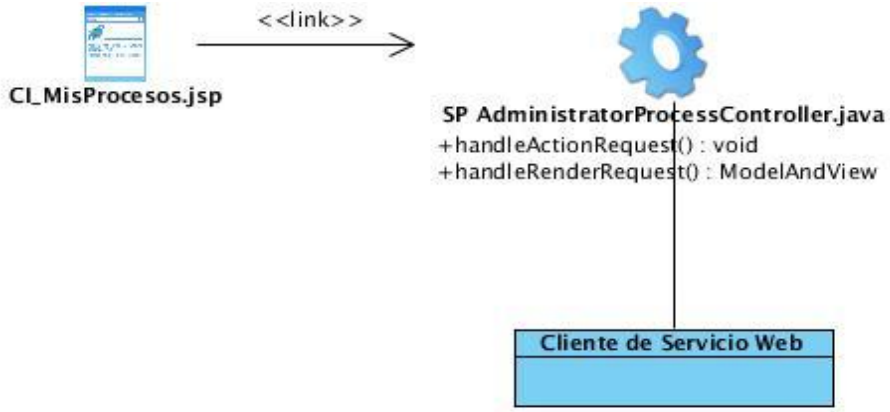

Figura 26: Diagrama de clases de diseño para el CU "Realizar Proceso de Acoplamiento Molecular con Autodock Vina".

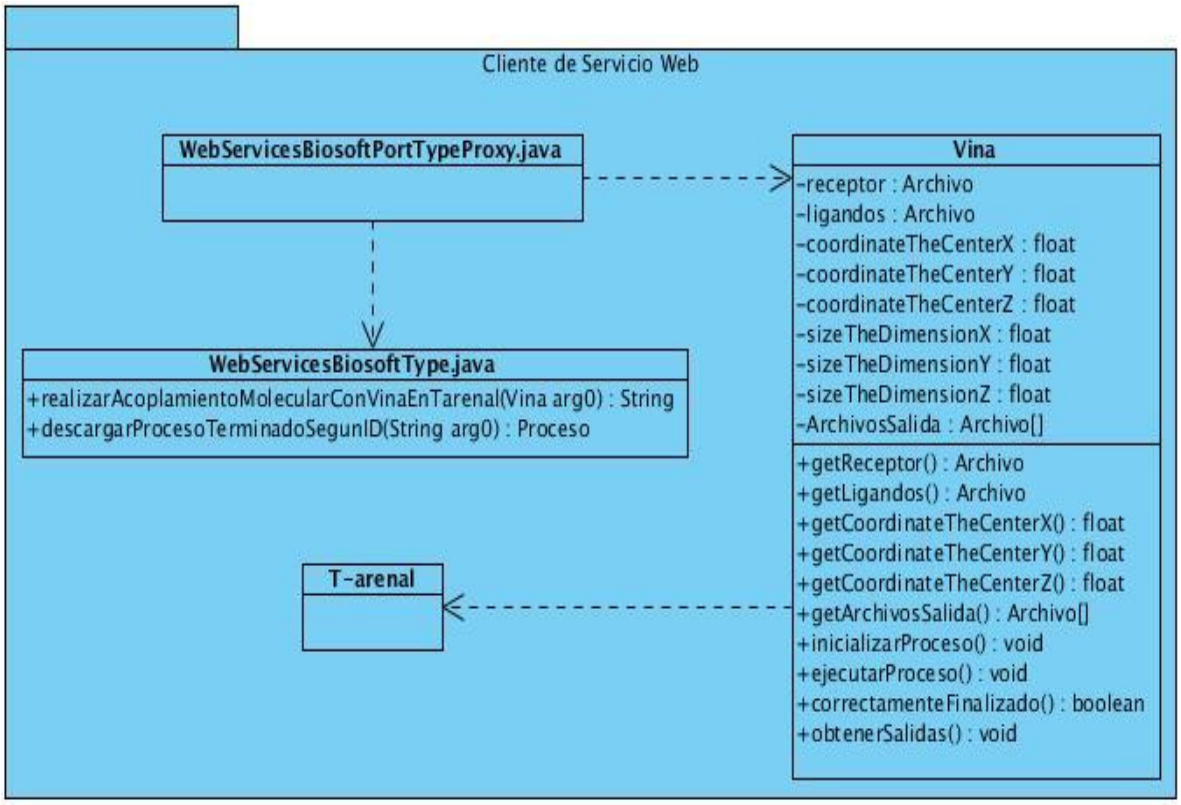

Figura 27: Diagrama de clases del diseño para el paquete Cliente de Servicios Web. Este paquete contiene las clases necesarias para invocar el servicio web que ejecuta la operación del sistema.

C. Diagramas de Interacción

Figura 28: Diagrama de secuencia para el CU "Realizar Acoplamiento Molecular con Autodock Vina".

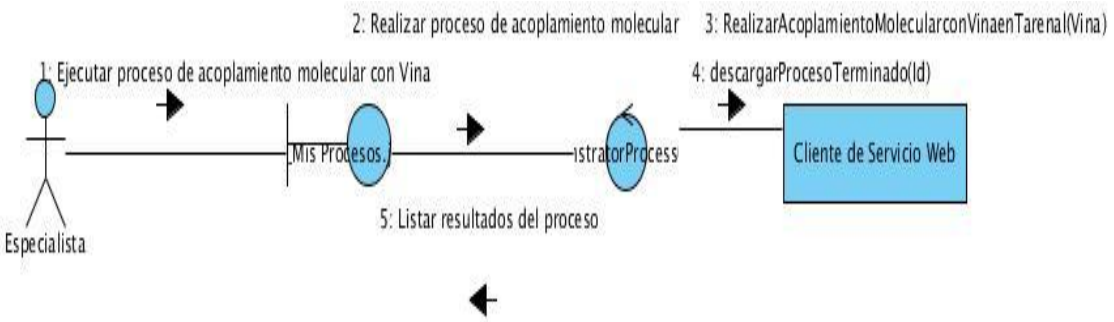

Figura 29: Diagrama de colaboración para el CU "Realizar Acoplamiento Molecular con Autodock Vina".

Figura 30: Diagrama de secuencia para el CU " Listar Ficheros de Salida de Autodock Vina".

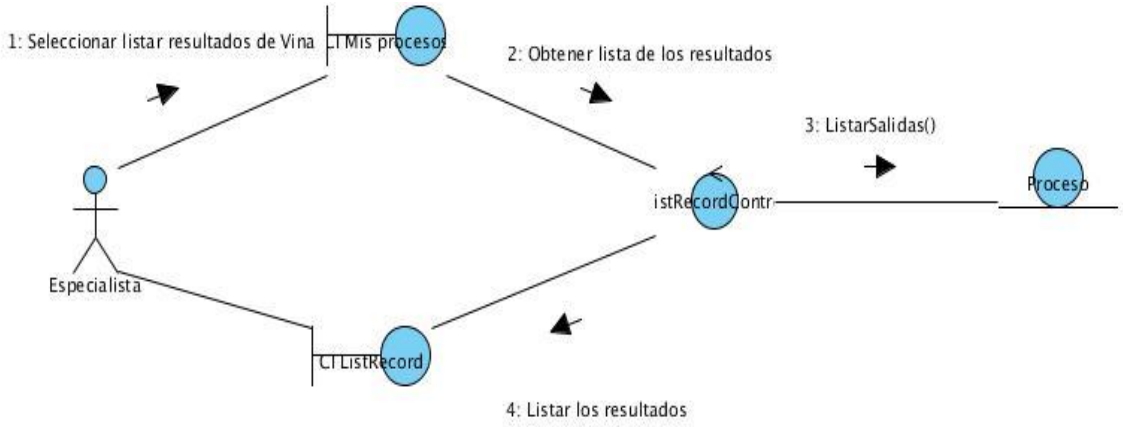

Figura 31: Diagrama de colaboración para el CU "Listar Ficheros de Salida de Autodock Vina".

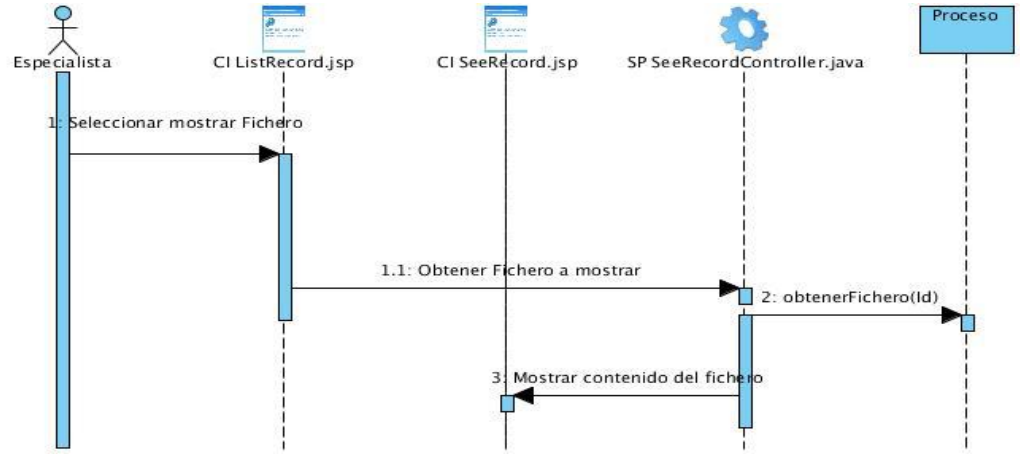

Figura 32: Diagrama de secuencia para el CU "Visualizar Resultados de Autodock Vina".

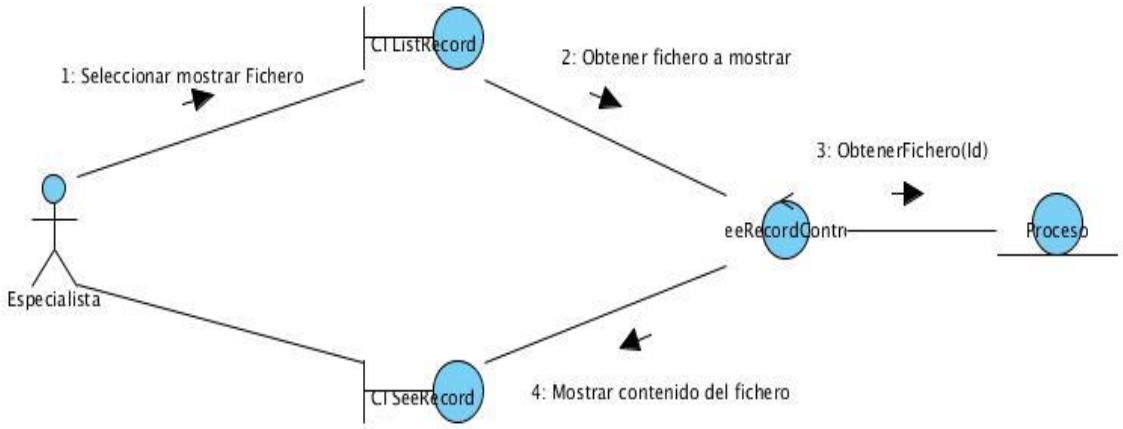

Figura 33: Diagrama de colaboración para el CU "Visualizar Resultados de Autodock Vina".

D. Descripciones de los CU del Sistema

Tabla 13: Descripción del CU "Realizar Acoplamiento Molecular con Autodock Vina".

Nombre del Caso de Uso	Realizar Acoplamiento Molecular con Autodock Vina
Actores:	Especialista
Resumen:	El caso de uso se inicia cuando el especialista ejecuta el proceso en la lista de procesos. El sistema ejecuta el servicio de acoplamiento molecular enviando esta tarea mediante el servicio web a la Plataforma de Tareas Distribuidas T-arenal.
Referencia:	RF1
Precondiciones:	El proceso debe estar creado.
Poscondiciones:	El sistema ejecuta el proceso de acoplamiento molecular y muestra los resultados.
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del sistema**
1 El especialista presiona el botón ejecutar en el proceso creado.	1.1 El sistema hace llamada al servicio web. 1.2 El servicio Web envía los archivos necesarios para la ejecución en la Plataforma de Tareas Distribuidas T-arenal. 1.3 La Plataforma de Tareas Distribuidas T-arenal ejecuta el proceso de acoplamiento molecular con Autodock Vina y devuelve los resultados. 1.4 El servicio web obtiene los datos y muestra en el sistema, el proceso con estado “Terminado”.
Prioridad	Crítico

Tabla 14: Descripción del CU “Listar Ficheros de Salida de Autodock Vina”.

Nombre del Caso de Uso	Listar Ficheros de Salida de Autodock Vina
Actores:	Especialista
Resumen:	El caso de uso se inicia cuando el especialista ejecuta la acción “Listar resultados” en el proceso con estado “Terminado”.
Referencia:	RF5
Precondiciones:	El proceso debe estar en estado “Terminado”.
Poscondiciones:	El sistema lista los ficheros de salida del proceso.
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del sistema**
1 El especialista presiona el botón “Listar resultados” en el proceso con estado “Terminado”.	1.1 El sistema obtiene los ficheros de salida que corresponden al proceso. 1.2 El sistema lista los ficheros.
Prioridad	Secundario

Tabla 15: Descripción del CU "Visualizar Resultados de Autodock Vina".

Nombre del Caso de Uso	Visualizar Resultados de AutodockVina.
Actores:	Especialista
Resumen:	El caso de uso se inicia cuando el especialista ejecuta la acción “Mostrar resultado” en un archivo de la lista de los ficheros de salida del proceso finalizado.
Referencia:	RF6
Precondiciones:	El proceso debe estar en estado “Terminado”.
Poscondiciones:	El sistema visualiza el contenido del fichero seleccionado.
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del sistema**
1 El especialista elige el fichero que desea visualizar y oprime el botón “Mostrar resultados”.	1.1 El sistema muestra en una interfaz el contenido del fichero.
Prioridad	Secundario

E. Diagrama de Componentes

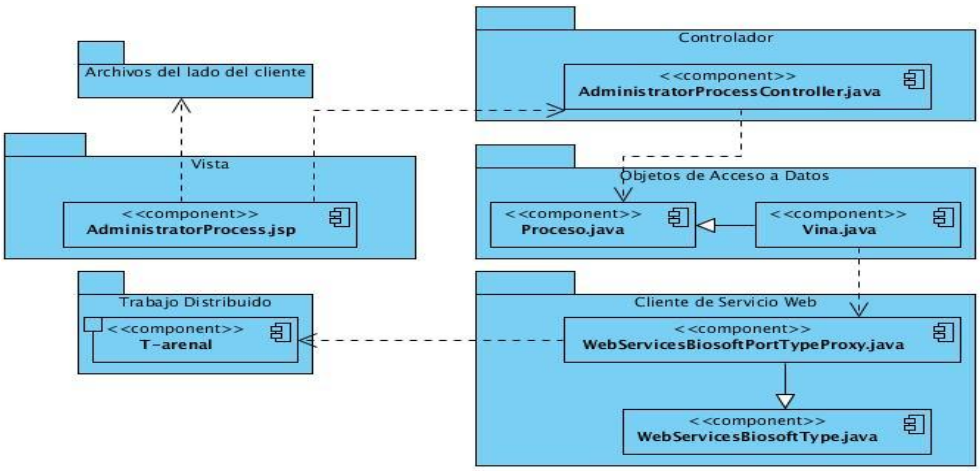

Figura 34: Diagrama de componentes para el CU Realizar Acoplamiento Molecular con Autodock Vina.

F. Interfaces del Sistema

Figura 35: Interfaz para el proceso creado con Autodock Vina.

Figura 36: Interfaz de la aplicación para el RF "Realizar Proceso de Acoplamiento Molecular con Autodock Vina".

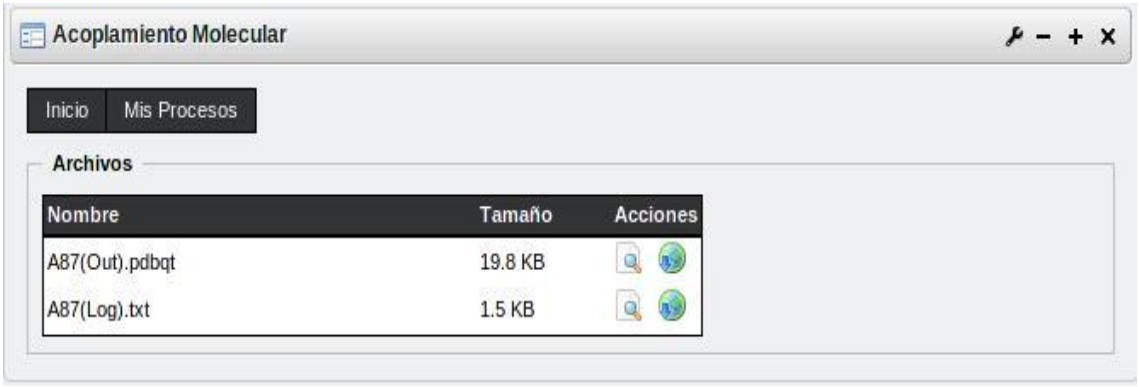

Figura 37: Interfaz de la aplicación para el RF "Listar Ficheros de salida de Autodock Vina".

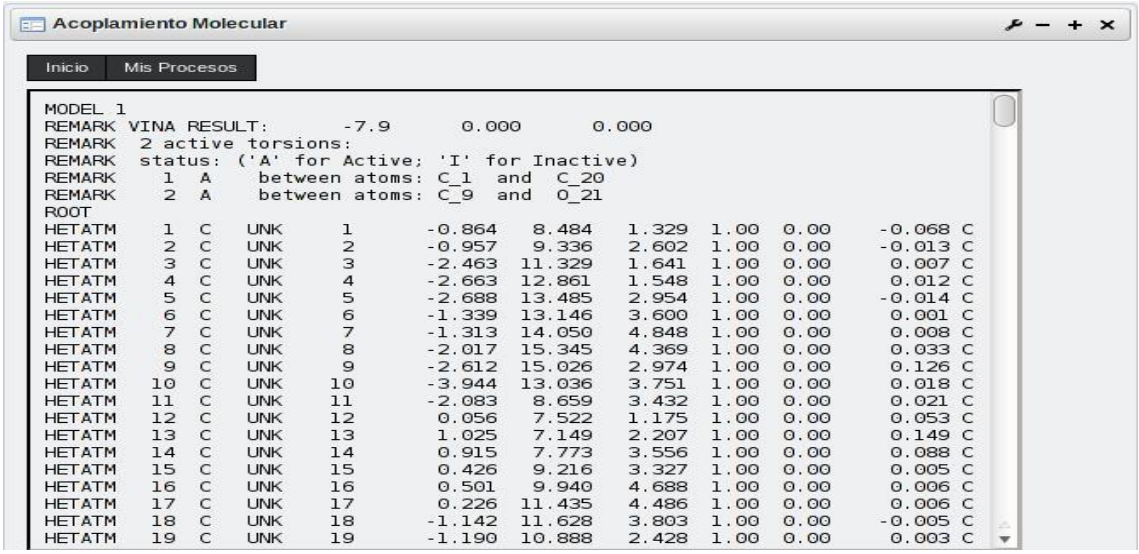

Figura 38: Interfaz de la aplicación para el RF "Visualizar Resultados de salida de Autodock Vina".