Universidad de las Ciencias Informáticas

Facultad 6

Título: 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:

Osvel Chávez Hernández

Tutores:

MSc. Longendri Aguilera Mendoza

MSc. Taymara Hernández Ortega

La Habana, junio de 2012

"Año 54 de la Revolución"

"*Si he hecho descubrimientos invaluables ha sido más por tener paciencia que cualquier otro talento.*"

Albert Einstein

Declaración de Autoría

Declaro ser el 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 _____ del año _____.

Osvel Chávez Hernández

Firma del autor

MSc. Longendri Aguilera Mendoza

Firma del Tutor

MSc. Taymara Hernández Ortega

Firma del Tutor

Datos de Contacto

Autor:

Osvel Chávez Hernández
Universidad de las Ciencias Informáticas
e-mail: ochavez@estudiantes.uci.cu

Tutores:

MSc. Longendri Aguilera Mendoza
Licenciado en Ciencias de la Computación
Máster en Bioinformática.
e-mail: longe@uci.cu

MSc. Taymara Hernández Ortega
Licenciada en Química
Máster en Bioinformática
e-mail: tay@uci.cu

Agradecimientos

Ante todo agradecer a la Revolución y a la Universidad por permitirme la superación profesional en este centro de altos estudios.

A mis compañeros de los apartamentos 55205, 74101 y 106102, los cuales muchos están aún aquí en la universidad casi al graduarse, otros están en 4to año y lamentablemente, otros ya no están entre nosotros, que me enseñaron que no siempre se respeta el espacio ajeno pero siempre se puede contar con ellos para arreglar el destrozo de todos.

A mis compañeros de los grupos 6105, 6201, 6301, 6304, 6409 y 6509 que les gustaba tirarme fotos mientras me dormía en los turnos de clases o me regañaban cuando decía o hacia algo que ellos consideraban que iba en contra del "bienestar estudiantil" pero que yo consideraba que tenía que decirlo porque era mi punto de vista, a los que comprendían mi forma de actuar, pero que al final me daban por loco y así seguíamos llevándonos bien.

A mis compañeros de producción de los laboratorios 403 y 402 del docente 4 pertenecientes a Bioinformática que permitieron un entorno de superación constante día a día para no quedar atrás en el camino del auto aprendizaje personal.

A los dirigentes de la FEU que durante un periodo muy corto de tiempo me enseñaron que el dirigente es la persona más incomprendida pero que es necesario dar el paso al frente para hacer las tareas que nadie quiere hacer y que es necesario hacer.

A todos los profesores que durante estos 5 años han puesto su granito de arena para que yo lograra convertirme en un profesional de las ciencias informáticas. En especial a los profesores de bioinformática por enseñarme ese maravilloso mundo de unir moléculas y computadoras para contribuir al desarrollo de la ciencia de la vida.

A mis tutores Longendri y Taymara por permitirme hacer esta tesis de grado, servirme de guía en los intrincados caminos que conllevan a hacer ciencia y por su paciencia a la hora de explicarme muchos de los ¿por qué? que surgen con un trabajo como este.

A mis abuelos y a mis padres que supieron educarme, me educan y me educaran mientras aún quede en ellos un soplo de vida. A mi abuela Isora que desde que era un niño me enseño el camino de las ciencias exactas. A mi papá Alejandro que siempre veló porque mantuviera una conducta intachable en los estudios y me enseñó que: "Nunca digas que no puedes hacer hasta no haberlo intentado con todo el esfuerzo". A mi madre que nunca me permitió flaquear ante los retos que me impuso la vida y se que lo seguirá haciendo;

y además por la máxima de que: "Si quieres llegar a ser algo en la vida trázate metas por encima de lo que crees que puedes hacer, una vez cumplida, trázate otras más exigentes que las anteriores". A mi padre Orestes que me enseño que siempre hay que decir la verdad aunque fuera a un niño pequeño y siempre ratificó la máxima de mi madre.

En fin a todos los que han hecho algo para que yo pueda dar lo mejor de mi.

El autor

Dedicatoria

Dedicado a mi familia que han dado todo de si porque yo llegue hasta donde estoy y en especial a mis padres que ya estoy un escalón más cerca de ser un poco más como ellos.

El autor

Resumen

El proceso de identificación de un fármaco eficaz para el tratamiento de una determinada enfermedad tiene un alto costo en tiempo y capital humano, siendo necesario para ello un importante presupuesto monetario. En la actualidad podemos contar con procedimientos computacionales que aminoran en gran medida estos altos costos, pero aún no son aprovechados por un gran número de especialistas debido a una limitada capacidad de cómputo que presentan algunos centros de investigación. El principal objetivo en el diseño de fármacos es encontrar aquellas moléculas candidatos que puedan interactuar de manera efectiva en un determinado blanco terapéutico. Para encontrar las conformaciones moleculares que tributen a la obtención de un complejo molecular estable se emplean los llamados métodos de Acoplamiento Molecular. En este trabajo se presenta un Servicio de Acoplamiento Molecular, el cual utiliza una Plataforma de Tareas Distribuidas desarrollada y desplegada en los laboratorios docentes de la Universidad de las Ciencias Informáticas (UCI), lográndose una disminución significativa del tiempo de respuesta de los métodos computacionales de Acoplamiento Molecular.

Palabras Claves: acoplamiento molecular, sistema distribuido, servicios bioinformáticos.

Índice general

Introducción **1**

1. Fundamento teórico **5**

1.1. Herramientas que permiten realizar Acoplamiento Molecular 5

1.1.1. Dock 5

1.1.2. AutoDock 6

1.2. Tecnologías Web 7

1.2.1. Apache Tomcat 7

1.2.2. Portlet 7

1.2.3. Liferay Portal 7

1.2.4. Portal de Servicios Bioinformáticos de la UCI 7

1.3. Programación distribuida 8

1.3.1. T-Arenal 8

1.4. Marcos de trabajo 8

1.4.1. Spring 8

1.4.2. Hibernate 8

1.4.3. Axis2 9

1.5. Lenguajes de programación 9

1.5.1. Java 9

1.6. Entornos de Desarrollo Integrado 9

1.6.1. Eclipse 10

1.7. Metodología a utilizar 10

1.7.1. OpenUP 10

1.8. Lenguaje de Modelado 10
1.8.1. UML 10
1.9. Herramientas CASE de modelado 11
1.9.1. Visual Paradim para UML 11
1.9.2. Rational Rose Enterprise 11
1.10. Sistemas Gestores de Bases de Datos 12
1.10.1. PostgreSQL 12
1.10.2. MySQL 12
1.11. Conclusiones parciales 12

2. Análisis y diseño **14**
2.1. Modelo de dominio 14
2.1.1. Definición de las clases del Modelo de Dominio 15
2.2. Requerimientos del sistema 15
2.2.1. Requisitos funcionales 16
2.2.2. Requerimientos no funcionales 16
2.3. Definición de los Actores del Sistema 19
2.4. Definición de los Casos de Uso del Sistema 19
2.4.1. Patrones de Casos de Uso 19
2.5. Diagrama de Casos de Uso del Sistema 20
2.5.1. Descripción de los Casos de Uso del Sistema 21
2.6. Arquitectura del Sistema 24
2.6.1. Patrones de la arquitectura 24
2.7. Realización de Casos de Uso del Diseño 26
2.7.1. Patrones de Diseño 26
2.7.2. Diagrama de Clases del Diseño con Estereotipos Web 28
2.7.3. Diagramas de Interacción 29
2.8. Estructura de la base de datos 29
2.9. Conclusiones parciales 30

3. Implementación y Prueba **32**
3.1. Ejecución del Dock 6.5 32

3.2. Servicio Web de Acoplamiento Molecular 33
3.2.1. Métodos públicos 33
3.2.2. Consumo del servicio 33
3.2.3. Código fuente 34
3.2.4. Experimento y resultados 36
3.3. Porlet Cliente 38
3.3.1. Diagramas de componentes 38
3.3.2. Pantallas de la aplicación 39
3.3.3. Casos de prueba 42
3.4. Guía de despliegue 45
3.4.1. Servicio Web. 45
3.4.2. Base de Datos PostgreSQL 45
3.4.3. Porlet Cliente. 45
3.5. Conclusiones parciales 46

Conclusiones **47**

Recomendaciones **48**

Referencias bibliográficas **49**

Índice de figuras

1. Complementariedad de forma ligando – receptor. 1

2.1. Diagrama de Modelo de Dominio. 15
2.2. Diagrama de Casos de Uso del Sistema. 21
2.3. Modelo Vista Controlador. 25
2.4. Diagrama de paquetes del diseño. 26
2.5. Diagrama de Despliegue. 26
2.6. Diagrama de Clases del Diseño del escenario Crear Corrida 28
2.7. Diagrama de Secuencia del escenario Crear Corrida. 29
2.8. Modelo de datos 30

3.1. Tiempo en obtener el resultado de los acoplamientos moleculares para los set de ligando empleados. 37
3.2. SpeedUp del Servicio de Acoplamiento Molecular. 38
3.3. Diagráma de componentes Crear Corridas. 38
3.4. Interfaz de usuario Crear Corrida. 39

Índice de algoritmos

3.1. Algoritmo para ejecutar una corrida en el Servicio Web. 34
3.2. Algoritmo para ejecutar las corridas de los especialistas. 40

Introducción

El descubrimiento de un nuevo medicamento y desarrollo posterior del mismo son dos fases, que condicionan lograr un nuevo producto que sea útil en la terapéutica. La fase de descubrimiento se estima que dure aproximadamente unos 4 - 5 años y la fase de desarrollo se estima que dure aproximadamente unos 5 - 6 años más [1].

En la actualidad, gracias a los avances científicos y tecnológicos en el campo de la biología molecular, la ciencia ha podido explicar el funcionamiento de los medicamentos. Algunos han sido descubiertos al azar, como en el caso de la penicilina, o bien mediante el estudio de remedios tradicionales y productos naturales. Lo cierto es que numerosas macromoléculas han sido identificadas que desempeñan funciones vitales en el funcionamiento de la célula y cuya alteración está relacionada con ciertas enfermedades.

Puesto que las proteínas son los componentes macromoleculares más abundantes, no resulta sorprendente que éstas sean las dianas más frecuentes de los fármacos. Razón por lo cual, cuando se conoce la base biológica de una enfermedad o de un desarreglo metabólico y se pueden definir las proteínas que causan tal desajuste, se diseña un fármaco que es una pequeña molécula (llamada ligando) con las propiedades químicas y biológicas necesarias para unirse al sitio activo de esa proteína y moldear su funcionamiento a condiciones favorables.(ver Figura 1)

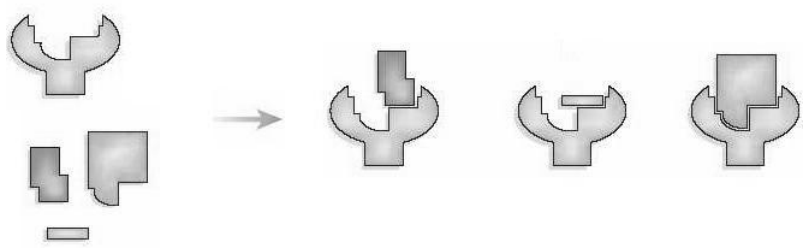

Figura 1: Complementariedad de forma ligando – receptor.

La importancia del diseño de fármaco basado en la estructura del receptor es que pretende mejorar

las interacciones que ocurren entre las moléculas. Lo que se traduce en un fármaco más efectivo. Con este propósito, se recurre a uno de los métodos computacionales más usados en este campo: el "Acoplamiento Molecular" (Molecular Docking en inglés). Este método consiste en calcular computacionalmente cuál es la posición más favorable que tendría una molécula en el blanco molecular trabajando con sus representaciones tridimensionales. Es importante destacar el gran número de grados de libertad que pueden tener dos moléculas para unirse, por lo que este método demanda de gran potencia de cómputo.

En la industria farmacéutica es común trabajar con las llamadas "bibliotecas virtuales": colecciones de miles o millones de estructuras moleculares que se pueden sintetizar. Estas bibliotecas se usan en un proceso conocido como "Tamizaje Virtual" (Virtual Screening en inglés), a fin de seleccionar los mejores compuestos capaces de unirse a una macromolécula diana utilizando métodos computacionales. El tiempo en obtener la respuesta de este proceso se puede ver afectado por el tamaño de la biblioteca virtual y por los grandes cálculos que implica realizar simulaciones de acoplamiento molecular. Después de hacer los cálculos, las moléculas virtuales seleccionadas como posibles candidatos a fármacos son sintetizadas y evaluadas de forma experimental. De tal manera que las computadoras y los métodos computacionales ayudan a filtrar y reducir los costos experimentales.

En Internet podemos encontrar Servidores Profesionales que realizan el Acoplamiento Molecular y presentan el respaldo computacional para realizar los costosos cálculos que se derivan de este proceso, entre ellos podemos encontrar el DockingServer [2] y el RosettaDock [3]. Si se realiza un análisis de los costos de las ejecuciones en estos servidores encontramos que aquellos servicios con mayor número de prestaciones tienen costos elevados, lo que constituye una limitante para los especialistas de países en vías de desarrollo. Por otro lado estos servidores no le brindan a los especialista la información referente a las políticas de protección de sus datos cuando consumen de sus servicios.

En Cuba existen instituciones que presentan recursos computacionales para realizar Acoplamiento Molecular, como son el Centro de Ingeniería Genética y Biotecnología (CIGB) y el Centro de Inmunología Molecular (CIM). Sin embargo no todos los especialistas cubanos cuentan con los recursos computacionales o algún servidor en la red nacional que brinde el servicio de Acoplamiento Molecular. A pesar de existir cooperaciones entre los especialistas de los diferentes centros, la demanda de recursos computacionales de altas prestaciones es superior a la existente en la red de investigación nacional.

La UCI cuenta con varios centros de desarrollo de software, donde se encuentra el Centro de Tecnología y Gestión de Datos (DATEC). En el departamento de Bioinformática de este centro se desarrolla una Plataforma de Servicios Bioinformáticos la cual tiene dentro de sus principales objetivos brindar un con-

junto de servicios a los especialistas del país asociados al área de la bioinformática, siendo el servicio de Acoplamiento Molecular uno de los que se quiere implementar en esta plataforma.

Teniendo en cuenta que desarrollar una infraestructura computacional para realizar el Acoplamiento Molecular es importante para garantizar mejores resultados a menor plazo en la Biotecnología Cubana, contribuyendo a un aumento de las posibilidades de éxitos y a un decrecimiento de los costos, se plantea el siguiente **problema a resolver:**

¿Cómo aumentar el poder de cómputo de la Plataforma de Servicios Bioinformáticos de la UCI para lograr reducir el tiempo en obtener la respuesta de un proceso de Acoplamiento Molecular?

Objetivo General: Desarrollar un servicio, en la Plataforma de Servicios Bioinformáticos de la UCI, que logre aprovechar los recursos computacionales existentes en la institución para reducir el tiempo en obtener la respuesta de un proceso de Acoplamiento Molecular.

Objetivos Específicos:

- Utilizar un conglomerado de computadoras para satisfacer la demanda computacional del Acoplamiento Molecular.
- Realizar el análisis y el diseño del servicio de Acoplamiento Molecular que haga uso de un conglomerado de computadoras.
- Implementar el servicio de Acoplamiento Molecular, previamente analizado y diseñado, en la Plataforma de Servicios Bioinformáticos.
- Validar el servicio de Acoplamiento Molecular implementado en la Plataforma de Servicios Bioinformáticos.

Tareas Investigativas:

1. Estudio y selección del conglomerado de computadoras a utilizar.
2. Aplicación de la Ingeniería de Software con el fin de generar los artefactos correspondientes a las fases de análisis y diseño del servicio de Acoplamiento Molecular.
3. Diseño de la Interfaz Web que se va a incluir en el Portal de la Plataforma de Servicios Bioinformáticos para que los usuarios interactúen con el servicio de Acoplamiento Molecular.

4. Implementación de la Interfaz Web y los componentes de software necesarios para brindar el servicio de Acoplamiento Molecular.

5. Realización de los casos de prueba para el servicio de Acoplamiento Molecular.

6. Ejecución de los casos de prueba para validar el funcionamiento del servicio de Acoplamiento Molecular.

Resultados Esperados:

Un servicio de Acoplamiento Molecular en la Plataforma de Servicios Bioinformáticos de la UCI, que utilice los recursos computacionales que existen y están disponibles en nuestra universidad, para garantizar el soporte tecnológico que permita la realización de cálculos intensos en esta rama de la Ciencia. Lográndose de esta forma, reducir el tiempo en obtener la respuesta de tan importante proceso para la producción de nuevos fármacos y contribuir al desarrollo de la Biotecnología Cubana.

Capítulo 1

Fundamento teórico

Introducción

En este capítulo se hará una búsqueda y comparación de las herramientas que permiten el Acoplamiento Molecular, las herramientas necesarias para el montaje y puesta en marcha de un Servicio de Acoplamiento Molecular y las herramientas, metodologías y lenguajes para la construcción del software.

1.1. Herramientas que permiten realizar Acoplamiento Molecular

Para la búsqueda de los mejores acoplamientos entre un receptor y un ligando candidato, los métodos de Acoplamiento Molecular emplean funciones de puntuación basadas en distintos fundamentos teóricos. Estás funciones de puntuación se encuentran implementadas en varias herramientas o programas informáticos que permiten el acoplamiento de una molécula receptora u objetivo con otra molécula que puede ser un ligando o una proteína [4].

A continuación se describen dos de las herramientas de mayor uso por los especialistas que realizan el acoplamiento molecular [5].

1.1.1. Dock

Es una herramienta que realiza el acoplamiento molecular, fue escrita en lenguaje C++ y es de código abierto [6]. 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 ella y el ligando [7]. Al ser compilado el código fuente para una arquitectura de grid, hace sus corridas de forma paralela sobre el cluster donde este instalado.

Esta herramienta trae incorporado los siguientes algoritmos:

- 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.
- Puntuación de solvatación de PB/SA e incluida la puntuación de flexibilidad del receptor de AMBER.
- La función del marcador con solvente implícito de minimización del total de la mecánica molecular de AMBER.
- Gradiente conjugado.
- Simulación de las capacidades de la dinámica molecular [6].

1.1.2. AutoDock

Es un conjunto de programas escritos en C utilizados para predecir las conformaciones entre un ligando flexible y una estructura macro - molecular conocida. Su técnica combina el recorrido simulado con una rápida red basada en el método de evaluación de la energía [8]. Tiene incorporado un conjunto de herramientas para ser puesto en marcha sobre una red heterogénea de estaciones de trabajo basadas en UNIX para así poder lograr acoplamientos independientes en paralelo [9]. Es gratuito y está disponible bajo la Licencia Pública General de GNU [10].

AutoDock tiene aplicaciones en:

- Cristalografía de rayos X.
- Estructura basada en el diseño de fármacos.
- Biblioteca combinatoria de diseño.
- Acoplamiento proteína-proteína.
- Mecanismo estudios químicos.

1.2. Tecnologías Web

1.2.1. Apache Tomcat

Es un servidor web, de código abierto que une la tecnología Java Servlet y Java Server Pages. Es desarrollado en un entorno abierto y participativo y publicado bajo la licencia Apache 2.0. Está destinado a ser una colaboración de los desarrolladores de los mejores de su clase de todo el mundo. En él se pueden montar aplicaciones de misión crítica a través de la Web [11]. Es un servidor HTTP y un contenedor de servlets (clases de Java), cargándolos y ejecutándolos de forma dinámica. Permite montar servicios web mediante Axis [12] [13].

1.2.2. Portlet

Es un componente Web basado en la tecnología Java que procesa los pedidos del usuario y genera un contenido dinámico final [14], el contenido generado por un Portlet es llamado fragmento y su ciclo de vida es manejado por el contenedor de Portlets. Puede declarar los eventos que quiere emitir y los que desea recibir. El contenedor de Portlet actuará como intermediario y distribuirá los eventos en consecuencia.

1.2.3. Liferay Portal

Es un portal de gestión de porlets, de código abierto e implementado en Java [15]. Se puede desplegar en servidores como Apache Tomcat, es multiplataforma lo que le permite ejecutarse 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 extención .WAR.

1.2.4. Portal de Servicios Bioinformáticos de la UCI

Es un portal desarrollado sobre el contenedor de Porlets, Liferay Portal [15], que provee un ambiente amigable para la definición y la ejecución de análisis en la esfera de la bioinformática. Su objetivo es proporcionarle a los especialistas bioinformáticos el acceso a recursos computacionales, los cuales muchas veces carecen en sus centros de trabajo o de estudios [16].

1.3. Programación distribuida

Es una expresión para referirse al cálculo de grandes problemas dividiéndolos en partes más pequeñas llamadas unidades de trabajo y utilizando, para la ejecución de estas unidades de trabajo, las prestaciones de computo presente en la subred del sistema. La programación distribuida se basa en la arquitectura cliente - servidor [17], donde el cliente es el encargado de realizar la tarea que le asigne el servidor y el servidor es el encargado de almacenar el problema, dividirlo, generar las unidades de trabajo y por último generar una sola respuesta a partir de las respuestas dadas por los clientes.

1.3.1. T-Arenal

Plataforma de Cálculo Distribuido desarrollada en la UCI sobre la arquitectura cliente - servidor [17] y desplegada en los laboratorios docentes de la misma universidad. Basado en la computación voluntaria [18], la cual plantea que: los clientes que son los encargados de realizar el procesamiento de cada uno de los subproblemas, son los que le piden al servidor la unidad de trabajo para su procesamiento. La forma en que el servidor debe generar las unidades de trabajo y la forma en que tiene que unir las respuestas dada por el sistema se especifican en la clase DataManager.java y la forma de ejecución del problema en la PC cliente se especifica en la clase Argorithm.java.

1.4. Marcos de trabajo

1.4.1. Spring

Spring Framework proporciona una amplia configuración de la programación y el modelo de las modernas aplicaciones empresariales, basadas en Java. Un elemento clave de Spring es el apoyo de infraestructura a nivel de aplicación. Se centra en la "fontanería" de las aplicaciones de forma que los equipos pueden centrarse en la lógica del negocio a nivel de aplicación, sin ataduras innecesarias a los entornos de despliegue específicos [19].

1.4.2. Hibernate

Es un conjunto de proyectos relacionados que permiten a los desarrolladores utilizar estilos de modelos de dominio en sus aplicaciones de manera que se extienden mucho más allá del mapeo objeto-relacional [20]. Permite al desarrollador detallar cómo es su modelo de datos, qué relaciones existen y qué forma tienen.

Genera las sentencias SQL y libera al desarrollador del manejo manual de los datos que resultan de la ejecución de dichas sentencias, manteniendo la portabilidad entre todos los motores de bases de datos con un ligero incremento en el tiempo de ejecución.

1.4.3. Axis2

Es un middleware, que permite la comunicación con otras aplicaciones, que permite montar servicios sobre la Web de Apache. Su arquitectura se basa en desarrollar sobre un núcleo simple y extensible que proporciona las abstracciones básicas para el resto del sistema [21].

1.5. Lenguajes de programación

Los lenguajes de programación son herramientas que facilitan la tarea de crear programas y software ya que disponen de sintaxis adecuadas que permiten ser leídas y escritas por las personas [22] y a su vez le permiten la interacción con el hardware y el software presente en la máquina [23].

1.5.1. Java

Es un lenguaje orientado a objetos desarrollado bajo una licencia libre. Fue desarrollado por la Sun Microsystems con la idea de utilizarse para el desarrollo de aplicaciones web, aunque luego se percataron de la amplia gama de posibilidades que tiene sobre las aplicaciones de escritorio. Los programas desarrollados con Java son independientes de la arquitectura de la computadora donde se ejecutan, así como que también son independientes del sistema operativo que los ejecuta ya que el código Java no se ejecuta sobre el sistema operativo sino sobre una maquina virtual (JDK). Permite aplicaciones bajo el esquema de Cliente – Servidor [22].

1.6. Entornos de Desarrollo Integrado

Los Entornos de Desarrollo Integrado (IDE por sus siglas en inglés) son un conjunto de programas que se ejecutan a partir de una única interfaz de usuario. En el caso de los IDE's para programadores incluyen a menudo un editor de texto, un compilador y un depurador, los cuales se activan y funcionan a partir de un menú común [24].

1.6.1. Eclipse

Desarrollado por IBM y entregado a la Fundación Eclipse sin fines lucrativos para ser utilizado como plataforma de código abierto y disponible para todos los sistemas operativos. Además de proporcionar un entorno de desarrollo amigable, automatiza muchas funciones a la hora del completado del código que de otra manera el desarrollador tendría que poner a mano. Cuenta con el apoyo de muchos observadores que dicen que es la herramienta clave para el desarrollo de la industria del software. Es fácil de usar y mucho más fácil de integrar sus distintos componentes y herramientas [25].

Como es una plataforma IDE, existen plugins para la mayoría de los lenguajes conocidos como por ejemplo: Java, C/C++, PHP, Cobol, etc. [26].

1.7. Metodología a utilizar

1.7.1. OpenUP

Su objetivo es asegurar la producción de software de calidad dentro de los plazos propuestos. Dirigido por casos de usos, centrado en la arquitectura, iterativo e incremental. Desarrollado y mantenido por la comunidad libre bajo la licencia libre [27]. Es un proceso mínimo, está centrado en la arquitectura y solamente es incluido el contenido fundamental para reducir al mínimo los riesgos. Es completo, puede ser manifestado como todo el proceso para construir un sistema. Es extensible, se puede agregar o se puede adaptar según lo vayan requiriendo los sistemas, en otras palabras, es ágil. Proporciona una comprensión detallada del proyecto, beneficiando tanto a clientes como a desarrolladores sobre los productos a entregar. Es la metodología más utilizada por desarrolladores de alto nivel en casi todo el mundo por sus cualidades administrativas [28].

1.8. Lenguaje de Modelado

1.8.1. UML

Lenguaje de Modelado Unificado (UML por sus siglas en Inglés) es la sucesión de una serie de métodos de análisis y diseño orientadas a objetos que aparecen a fines de los 80's y principios de los 90's. Una de las metas principales de UML es avanzar en el estado de la integración institucional proporcionando herramientas de interoperabilidad para el modelado visual de objetos [29]. Se utiliza para definir un sistema, para detallar

los artefactos en el sistema, para documentar y construir. Es el lenguaje en el que está descrito el modelo.

1.9. Herramientas CASE de modelado

Ingeniería de Software Asistida por Computadoras (CASE por sus siglas en Inglés) es la aplicación de métodos y técnicas [30] enfocadas a ayudar a los analistas de software desde el inicio del ciclo de vida de un software hasta que termina con la última iteración del mismo proporcionándoles diagramas de casos de usos y de interfaces y auto - generando código de programación a partir de estos diagramas [31].

1.9.1. Visual Paradim para UML

Es una herramienta UML profesional, libre, que soporta el ciclo de vida completo del desarrollo del software: análisis y diseño orientado a objetos, construcción, pruebas y despliegue. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación que sirven de puente entre los arquitectos, analistas y diseñadores del sistema haciendo el trabajo más fácil y dinámico. También automatiza tareas tediosas que pueden distraer a los diseñadores. La navegación es intuitiva entre el código y el modelo visual. Es un poderoso generador de informes. Mantiene la demanda en tiempo real. Contiene un ambiente modelador visual superior y un sofisticado diseño de diagramas [32].

1.9.2. Rational Rose Enterprise

Herramienta de modelado desarrollada por IBM. Proporciona un lenguaje de modelado común, utilizando el UML, lo que agiliza la creación de un software de calidad. Soporte para el análisis de los patrones ANSI C++, Rose J y Visual C++. Control por separado de componentes de modelado que permite una administración más granular y el uso de modelos. Soporta ingeniería directa y/o inversa para algunos de los conceptos más comunes de Java 1.5. Generación de código Ada, ANSI C ++, C++, CORBA, Java y Visual Basic, con capacidad de configurar la sincronización entre el modelo y el código. Modelado UML para diseñar bases de datos, con la capacidad de representar la integración de los datos y los requerimientos de la aplicación a través de diseños lógicos y físicos. Disponible solo para sistemas operativos de la familia de Windows [33].

1.10. Sistemas Gestores de Bases de Datos

Son una capa intermedia entre los programas del usuario y la base de datos. Permiten al usuario definir, crear y mantener la base de datos y proporciona un acceso controlado a la misma mediante los mecanismos de seguridad de acceso [34]. Proporcionan una visión abstracta de los datos. Utiliza una variedad de técnicas eficientes para almacenar y recuperar los datos así como su seguridad en el almacenamiento y transporte [35].

1.10.1. PostgreSQL

Sistema de gestión de bases de datos objetos - relacional, de propósito general, multiplataforma, multiusuario, de código abierto y diseñado para ambientes de misión crítica y mantenimiento de grandes bases de datos. Soporta el almacenamiento de objetos binarios grandes. Contiene columnas auto incrementales. Implementa la herencia de tablas y los disparadores comunes, por columna, condicionales. Permite las consultas recursivas [36].

1.10.2. MySQL

Es uno de los sistemas de gestión de bases de datos más populares, desarrollado y distribuido por Oracle. Es programado en C/C++, es multiplataforma y de código abierto bajo una licencia GPL o propietaria. Implementa un sistema de privilegios y contraseñas flexibles y seguro, verificable por un host, así como contraseña de seguridad para cifrar el tráfico de contraseñas con el servidor. Soporta bases de datos grandes con 50 millones de archivos, 200 mil tablas y 5 mil millones de filas [37].

1.11. Conclusiones parciales

Según lo antes expuesto para dar respuesta al problema planteado, se realizará un servicio web sobre el middleware Axis2 el cual se desplegará sobre un servidor de aplicaciones web Apache Tomcat 6.0 y utilizará el Dock 6.5 como herramienta que realiza el Acoplamiento Molecular por tener precedentes de corridas con él sobre la plataforma T-Arenal, permite hacer acoplamiento molecular no solo proteína - ligando, sino también proteína - proteína, además de poder compilarse para cualquier arquitectura y ser de licencia libre. Para el desarrollo del cliente que consumirá el servicio web se utilizará la metodología OpenUP por ser una arquitectura ágil y la utilizada en el proyecto Plataforma de Servicios Bioinformáticos. Como herramienta de modelado se hará uso del Visual Paradim para UML 6.4 que es una herramienta que cubre todo el ciclo

de desarrollo del software y además es de licencia libre. El lenguaje de programación a utilizar será el Java por ser un lenguaje multiplataforma, orientado a objetos y de licencia libre. Como IDE de desarrollo será utilizado el Eclipse Helios 3.6, el cual permite trabajar con el lenguaje de programación propuesto, es fácil de integrar con Axis2 y es un software de licencia libre. Como sistema gestor de base de datos se utilizará el PosgreSQL 8.4, ya que permite el almacenado de grandes extensiones de datos y presenta licencia libre. Como marco de trabajo se utilizarán el Spring 2.6 para la implementación de la Interfaz Web por ser de software libre y de código abierto, y el Hibernate 3.3.1 como herramienta de mapeo objeto - relacional, por ser de software libre y distribuido bajo los términos de la licencia GNU/LGPL.

Capítulo 2

Análisis y diseño

Introducción

En el presente capítulo se hace el análisis y el diseño a la propuesta del sistema. En el análisis se especifican los requisitos funcionales y no funcionales; los casos de uso y los actores, sus descripciones textuales y se relacionan los casos de uso con los actores mediante el diagrama de casos de uso del sistema. En el diseño se modelan los artefactos correspondientes a esta fase, se describe la arquitectura del sistema, se definen los patrones del diseño, se describe el modelo de diseño mediante diagramas de clases del diseño con estereotipos web y el modelo de despliegue.

2.1. Modelo de dominio

Un Modelo de Dominio (MD) se aplica cuando no es posible realizar el modelo del negocio. Este modelo debe ser muy delgado, capturando las entidades principales de negocio y las relaciones entre ellos. El mismo representa un aparte visual de las clases y sus interacciones con el entorno del proyecto [38]. El diagrama (ver figura 2.1) representa los objetos relacionados con los principales conceptos que se emplearán en este trabajo utilizando notación UML de modelado de datos.

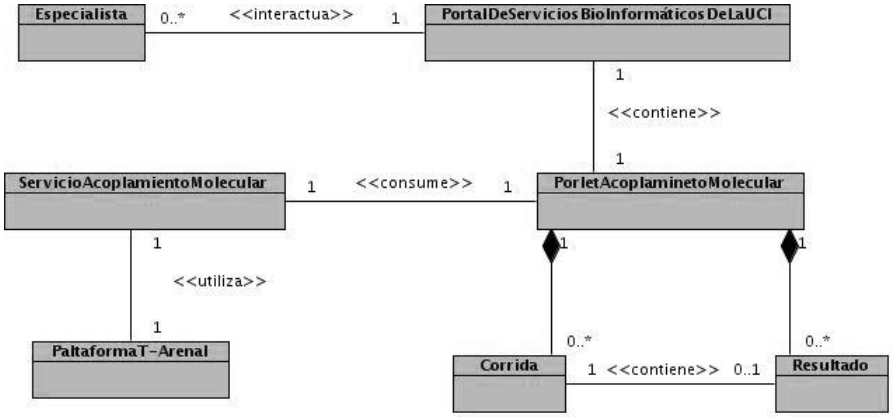

Figura 2.1: Diagrama de Modelo de Dominio.

2.1.1. Definición de las clases del Modelo de Dominio

Especialista: Usuario que posee privilegios para interactuar con el Portal de Servicios Bioinformáticos de la UCI.

PortalDeServiciosBioInformáticosDeLaUCI: Portal con interfaces gráficas donde los especialistas pueden consumir los servicios que brinda la Plataforma de Servicios Bioinformáticos de la UCI.

PorletAcoplamientoMolecular: Porlet que le permite al especialista la gestión de sus corridas y la administración de sus ejecuciones.

ServicioAcoplamientoMolecular: Servicio que brinda la posibilidad de realizar Acoplamiento Molecular.

PlataformaT-Arenal: Plataforma de Cálculo Distribuido implementada y desplegada en la UCI.

Corrida: Cada una de las ejecuciones del especialista sobre el Servicio de Acoplamiento Molecular.

Resultado: Respuesta dada por el Servicio de Acoplamiento a una corrida.

2.2. Requerimientos del sistema

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

2.2.1. Requisitos funcionales

Los requerimientos funcionales surgen de la entrevista con el cliente, el cual plantea de forma clara las necesidades que considera debe satisfacer el sistema. A continuación se enumeran.

RF1 Realizar Acoplamiento Molecular: Este software debe ser capaz de realizar Acoplamiento Molecular.

RF2 Crear corrida: Este software debe brindar la posibilidad de crear una nueva corrida.

RF3 Buscar corrida: Este software debe brindar la posibilidad de buscar una corrida.

RF4 Listar corridas: Este software debe listar las corridas.

RF5 Modificar corrida: Este software debe brindar la posibilidad de modificar una corrida.

RF6 Eliminar corrida: Este software debe brindar la posibilidad de eliminar la/s corrida/s marcada/s.

RF7 Buscar resultado: Este software debe brindar la posibilidad de buscar un resultado.

RF8 Listar resultados: Este software debe listar todos los resultados.

RF9 Mostrar contenido de un resultado: Este software debe brindar la posibilidad de mostrar el contenido de un resultado.

RF10 Eliminar resultado: Este software debe brindar la posibilidad de eliminar el/los resultado/s marcado/s.

RF11 Descargar resultados: Este software debe brindar la posibilidad de descargar los resultados obtenidos.

2.2.2. Requerimientos no funcionales

Los requerimientos no funcionales son propiedades o cualidades que el sistema debe tener. Son importantes para que los usuarios valoren las características no funcionales del sistema. A continuación se muestran.

Seguridad

Los especialistas tienen que estar autenticados en el Portal de Servicios Bioinformáticos de la UCI para acceder a todas las funcionalidades del sistema. La aplicación contará con protección contra acciones no autorizadas y que puedan afectar la integridad de los datos por lo que cada especialista solo tendrá acceso a su propia información, garantizando así la confidencialidad.

Rendimiento

El sistema está concebido para un ambiente cliente - servidor por lo que la rapidez del servicio dependerá del volumen de la información que necesite procesar y de la cantidad de corridas en la cola de ejecución, es por ello que se propone su integración a la Plataforma de Cálculos Distribuidos: T-Arenal; haciendo el trabajo más eficaz.

Usabilidad

El sistema se desplegará en el Protal de Servicios Bioinformáticos de la UCI. Tendrá un ambiente sencillo y será fácil de manejar por cualquier investigador aunque no tenga mucha experiencia en el trabajo con las computadoras. Contendrá un manual de usuario con las descripciones básicas de cómo trabajar con el sistema. Deberá estar disponible las 24 horas del día los 7 días de la semana garantizando un acceso de forma fácil y rápida para los especialistas.

Apariencia o interfaz externa

El sistema debe contar con una interfaz amigable que permita al especialista interactuar de forma cómoda, le agilice y facilite el trabajo con el sistema. La página principal mostrará la guía de uso que servirá de ayuda al especialista.

Portabilidad

El sistema podrá ser ejecutado en cualquier sistema operativo, por su característica de ser multiplataforma, permitiendo una fácil migración haciendo uso de estándares y tecnologías de código abierto.

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 su uso. Se actualizará a medida que se diseñen las mejoras.

Software

En el cliente: Se debe disponer para poder ejecutar la aplicación:

- Navegador Web estándar con capacidad de interpretación de Java Script, CSS compatible con la W3C ejemplo Netscape, optimizado para Mozilla 1.7 o superior o FireFox 0.9.3 o superior.

En el servidor: Se debe disponer para poder instalar la aplicación:

- Servidor Web Apache Tomcat 6.0.14 o superior.
- JDK6 o superior.
- PostgreSQL 8.4 o superior.

Hardware

En el cliente: Se debe disponer para poder ejecutar la aplicación:

Las estaciones de trabajo de los especialistas se necesita una PC con microprocesador Intel Pentium II o superior, con un mínimo de 128MB de memoria RAM y tarjeta de red para la conexión de red con el Portal de Servicios Bioinformáticos de la UCI.

En el servidor: Se debe disponer para poder instalar la aplicación:

Los servidores deberán contar con un microprocesador Intel Pentium IV o superior, a 450 MHz o superior, con un mínimo de 512MB de memoria RAM, un disco duro con capacidad disponible de almacenamiento de 40GB y una tarjeta de red para poder brindar el servicio utilizando la red.

Legales

Se usarán herramientas de software libre y código abierto, o que funcionen bajo las licencias GNU/GPL, por lo que el servicio que se brindará esta basado también en la licencia GNU/GPL.

Restricciones en el diseño y la implementación

El análisis y el diseño de la aplicación será basado en la Metodología OpenUp con el uso del lenguaje de modelado UML. Como herramienta CASE será usada Visual Paradigm para el modelado y como lenguaje de programación Java (JSP).

2.3. Definición de los Actores del Sistema

Los actores representan terceros fuera del sistema que interactúan con él. Cada actor juega un rol dentro del sistema al interactuar con él, diferentes especialistas pueden asumir el mismo rol de actor.

Actor	**Descripción**
Especialista	Es el rol que podrá realizar un conjunto de tareas para hacer una corrida sobre el Servicio de Acoplamiento Molecular y acceder a la respuesta dada por el mismo.
Autómata	Es el subsistema encargado de acceder a la Base de Datos, obtener una corrida, ejecutarla en el Servicio de Acoplamiento Molecular, recepcionar el resultado dado por el mismo y guardarlo en la Base de Datos.
T-Arenal	Es la plataforma encargada de ejecutar el algoritmo de Acoplamiento Molecular.

2.4. Definición de los Casos de Uso del Sistema

Los casos de uso representan uno o varios requisitos funcionales, por lo que representa funcionalidades que el sistema ofrece para que los actores puedan interactuar con ellos.

2.4.1. Patrones de Casos de Uso

Para hacer más fácil el trabajo con los sistemas y mucho más simple su mantenimiento se aplican patrones en el diseño de los casos de uso que permiten reflejar con más precisión los requisitos reales. Se presentan a modo de herramientas que permiten resolver los problemas que se les planteen a los desarrolladores de una forma ágil y sistemática. Estos patrones se enfocan hacia el diseño y las técnicas utilizadas en modelos de alta calidad, y no en cómo modelar casos de uso específicos.

Dentro de los patrones utilizados está el:

CRUD, el cual modela todas las operaciones que se pueden realizar sobre una parte de la información, como son: creación, lectura, actualización y eliminación.

Listado de casos de uso

CU1 Realizar Acoplamiento Molecular.

RF1 Realizar Acoplamiento Molecular.

CU2 Gestionar corrida.

RF2 Crear corrida.

RF3 Buscar corrida.

RF4 Listar corridas.

RF5 Modificar corrida.

RF6 Eliminar corrida.

CU3 Administrar resultado.

RF7 Buscar resultado.

RF8 Listar resultados.

RF9 Mostrar contenido de un resultado.

RF10 Eliminar resultado.

CU4 Descargar resultado.

RF11 Descargar resultados.

2.5. Diagrama de Casos de Uso del Sistema

Un diagrama de Casos de Uso del sistema es la representación gráfica de los actores con los procesos. Este Diagrama está formado por los Casos de Uso del sistema y los actores que interactúan con ellos.

El diagrama de Casos de Uso del Sistema, (ver figura 2.2), muestra los Casos de Uso y los actores definidos, con sus respectivas asociaciones.

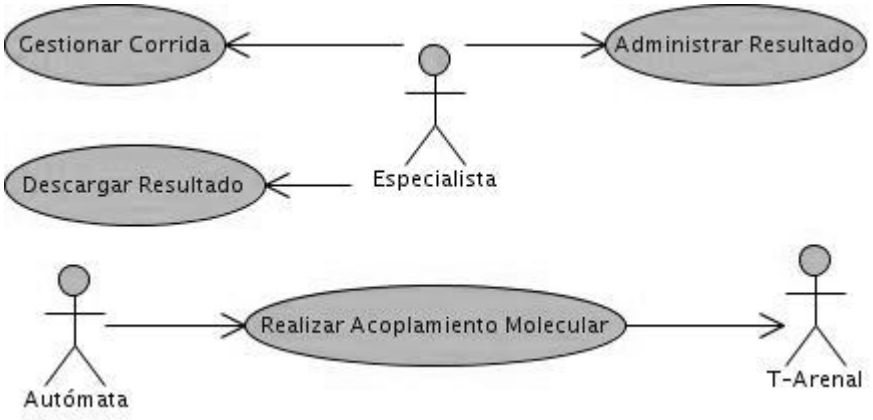

Figura 2.2: Diagrama de Casos de Uso del Sistema.

2.5.1. Descripción de los Casos de Uso del Sistema

La descripción de los casos de uso del sistema detalla la secuencia de eventos que los actores llevan a cabo para completar un proceso a través de la aplicación.

A continuación se describe detalladamente los Casos de Uso presentes en el Diagrama de Casos del Sistema ya que no es suficiente con su representación gráfica.

Descripción textual del CU Gestionar Corrida

Nombre del CU	**Gestionar Corrida**
Actor(es)	Especialista (inicia)
Propósito	Poder listar, crear, modificar, buscar o eliminar las corridas con las que vaya a trabajar el especialista.
Resumen:	El caso de uso se inicia cuando el especialista selecciona la opción de Mis Corridas en la interfaz principal.
	El sistema muestra la interfaz correspondiente según la solicitud y ejecuta la acción de listar corridas. El caso de uso finaliza cuando se obtiene un resultado de la operación efectuada.
Referencia	RF 2, RF 3, RF 4, RF 5, RF 6

Precondiciones	
Poscondiciones	Actualizar la lista de corridas del especialista.
Curso Normal de eventos	
Acciones del Actor	**Respuesta del sistema**
1. El especialista inicia el caso de uso seleccionando la opción de Mis corridas en la interfaz principal.	2. El sistema muestra la interfaz correspondiente y lista las corridas del especialista.
3. El especialista escoje una de las siguientes opciones: Crear corrida. Modificar corrida. Buscar corrida. Eliminar corrida.	4. El sistema en dependencia de la opción seleccionada por el especialista, hace lo siguiente: Si el especialista decide crear corrida ir a la sección Crear corrida. Si el especialista decide modificar corrida ir a sección Modificar corrida. Si el especialista decide buscar corrida ir a sección Buscar corrida. Si el especialista decide eliminar corrida ir a sección Eliminar corrida.
Sección Crear corrida	
Acciones del Actor	**Respuesta del sistema**
1. El especialista selecciona mediante la barra de menú la opción de crear nueva corrida, Crear.	2. El sistema muestra la interfaz para que el especialista introduzca los datos necesarios para crear la nueva corrida.
3. El especialista entra los datos (nombre de la corrida, archivo configuración.in y archivos necesarios) de la corrida que desea crear y presiona el botón Aceptar.	4. El sistema valida los datos entrados por el especialista y si son correctos crea la corrida en la base de datos.
Sección Flujo alterno de la sección Crear corrida	
	4.1. Si existen errores en los datos entrados por el especialista, el sistema muestra en la interfaz un diálogo mostrando el error.

Sección Modificar corrida	
Acciones del Actor	**Respuesta del sistema**
1. El especialista selecciona mediante la barra de menú la opción de modificar corrida, Modificar.	2. El sistema valida que se haya seleccionado solo una corrida y muestra la interfaz con los datos de la corrida para que el especialista los modifique.
3. El especialista modifica los datos de la corrida y presiona el botón Aceptar.	4. El sistema valida los datos entrados por el especialista y si son correctos modifica los datos de la corrida en la base de datos.
Sección Flujo alterno de la sección Modificar corrida	
	2.1. Si no hay corridas seleccionadas o hay más de una corrida seleccionada, el sistema muestra en la interfaz un diálogo mostrando el error. 2.2. Si la corrida ya está en ejecución, el sistema muestra en la interfaz un diálogo mostrando la información.
	4.1. Si existen errores en los datos entrados por el especialista, el sistema muestra en la interfaz un diálogo mostrando el error.
Sección Buscar corrida	
Acciones del Actor	**Respuesta del sistema**
1. El especialista introduce el criterio de búsqueda en el campo de texto y selecciona en la barra de menú la opción de buscar corrida, Buscar.	2. El sistema valida el criterio de búsqueda y si es correcto, lista en la interfaz las corridas que cumplan con el criterio.
Sección Flujo alterno de la sección Buscar corrida	
	2.1. Si existen errores en en el criterio de búsqueda entrado por el especialista, el sistema muestra en la interfaz un diálogo mostrando el error.

Sección Eliminar corrida	
Acciones del Actor	**Respuesta del sistema**
1. El especialista selecciona una o varias corridas a eliminar y luego selecciona en la barra de menú la opción de eliminar corrida, Eliminar.	2. El sistema valida que se haya seleccionado al menos una corrida y elimina la(s) corrida(s) de la base de datos.
Sección Flujo alterno de la sección Eliminar corrida	
	2.1. Si no hay corridas seleccionadas, el sistema muestra en la interfaz un diálogo mostrando el error. 2.2. Si la corrida está en ejecución, el sistema muestra en la interfaz un diálogo mostrando la información.
Prototipo de Interfaz 	
Prioridad	Crítico

El resto de los CU identificados se localizan en el anexo del trabajo.

2.6. Arquitectura del Sistema

2.6.1. Patrones de la arquitectura

Modelo Vista Controlador(MVC)

- La capa de presentación o vista está formada por las clases JSPs, que son las clases con las que interactúa directamente el especialista desde el Portal de Servicios Bioinformático de la UCI.
- La capa de control se implementa como uno o varios Servlet que son las clases encargadas de recibir las diferentes peticiones de los especialista y hacer las llamadas a las funciones necesarias 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 Object (POJO) con 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 de la UCI.

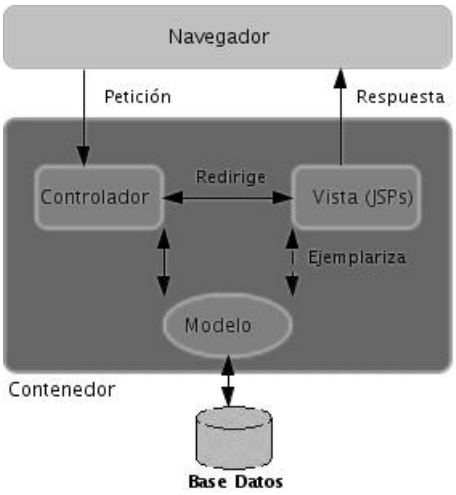

Figura 2.3: Modelo Vista Controlador.

Arquitectura del portlet a implementar

En el sistema se desarrolla bajo el patrón de arquitectura MVC para el desarrollo del portlet. Contiene un objeto para el control, parametrizable desde un archivo XML, que es el responsable de controlar el flujo de la aplicación, instanciar los beans y servir las vistas activas en cada instancia cliente. Controla los estados del sistema desde que el especialista genera un evento, hasta que el sistema retorna el contenedor correspondiente.

Vista Lógica

En la figura 2.4 se ponen de manifiesto los cuatro casos de uso que son programados en el sistema, por lo que son casos de uso arquitectónicamente significativos. En el epígrafe 2.2.1 se hace la descripción textual de estos casos de uso.

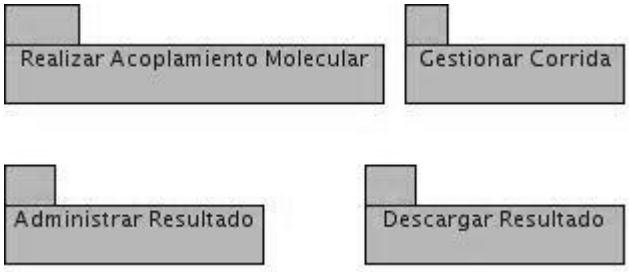

Figura 2.4: Diagrama de paquetes del diseño.

Vista de Despliegue

Diagrama de Despliegue

La figura 2.5 muestra el diagrama de la configuración del hardware del sistema y los nodos físicos que lo componen. El sistema necesita para su ejecución un servidor web, Apache Tomcat, un servidor de bases de datos PostgreSQL, la Plataforma de Cálculo Distribuido T-Arenal y las terminales web donde cada especialista tendrá acceso al sistema.

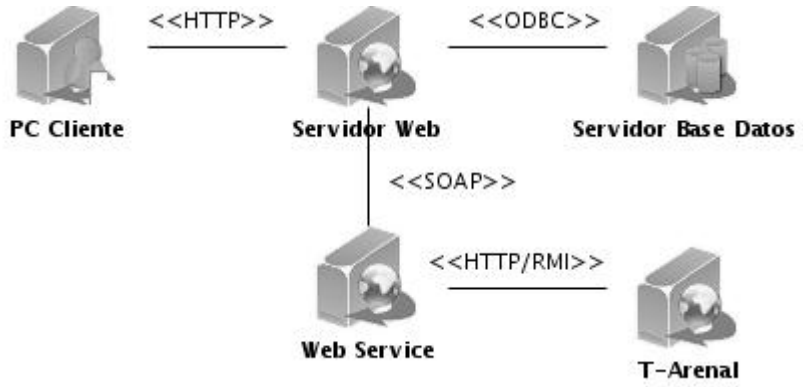

Figura 2.5: Diagrama de Despliegue.

2.7. Realización de Casos de Uso del Diseño

2.7.1. Patrones de Diseño

Los Patrones de Diseño para la Asignación de Responsabilidades (GRASP, por sus siglas en inglés), son patrones generales de software para asignación de responsabilidades. Aunque se considera que más que patrones propiamente dichos, son una serie de buenas prácticas de recomendable aplicación en el diseño de software.

El GRASP de experto en información es el principio básico de asignación de responsabilidades. Nos indica que la responsabilidad de la creación de un objeto o la implementación de un método, debe recaer sobre la clase que conoce toda la información necesaria para crearlo. De este modo obtendremos un diseño con mayor cohesión y así la información se mantiene encapsulada.

Descripción de los patrones GRASP

Patrón	**Descripción**
Bajo acoplamiento	Es la idea de tener las clases lo menos ligadas entre sí que se pueda. De tal forma que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de clases, potenciando la reutilización, y disminuyendo la dependencia entre las clases. Para determinar el nivel de acoplamiento de clases, son muy buenos los diagramas de colaboración de UML.
Alta cohesión	Nos dice que la información que almacena una clase debe de ser coherente y debe estar relacionada con la clase. Cada elemento de nuestro diseño debe realizar una labor única dentro del sistema, no desempeñada por el resto de los elementos y auto - identificable.

Problema: Que debe haber poca dependencia entre las clases.

Solución: Para solucionar esto se aplica el patrón Bajo Acoplamiento de los GRASP.

Aplicación: Existe mínima dependencia entre las clases, puesto que cada controladora hace uso de su correspondiente clase administrar de la base de datos para acceder a los datos almacenados en la misma, o sea que la implementación de cada una de ellas no afecta al correcto funcionamiento de la otra, solo tienen que estar bien implementadas.

Problema: Que se le asignen responsabilidades a las clases de manera que todos sus métodos tengan un comportamiento bien definido.

Solución: Para solucionar esto se aplica el patrón Alta Cohesión de los GRASP.

Aplicación: En el sistema cada clase realiza únicamente la tarea que le corresponde por ejemplo la clase AdministrarCorrida se encarga de acceder a la base de datos y realizar las operaciones de agregar,

eliminar, obtener y modificar las corridas; la clase Autómata se encarga de realizar las operaciones de obtención de la corrida de más prioridad, mandar a ejecutarla, obtener la respuesta y almacenarla en la base de datos.

2.7.2. Diagrama de Clases del Diseño con Estereotipos Web

Los Diagramas de Clases del Diseño describen gráficamente las especificaciones de las clases de software y de las interfaces. Este tipo de diagrama contiene la siguiente información:

- Clases, asociaciones y atributos.
- Interfaces, con sus operaciones y constantes.
- Métodos.
- Información sobre los tipos de los atributos.
- Navegabilidad.
- Dependencias.

Diagrama de Clases del Diseño del escenario Crear Corrida

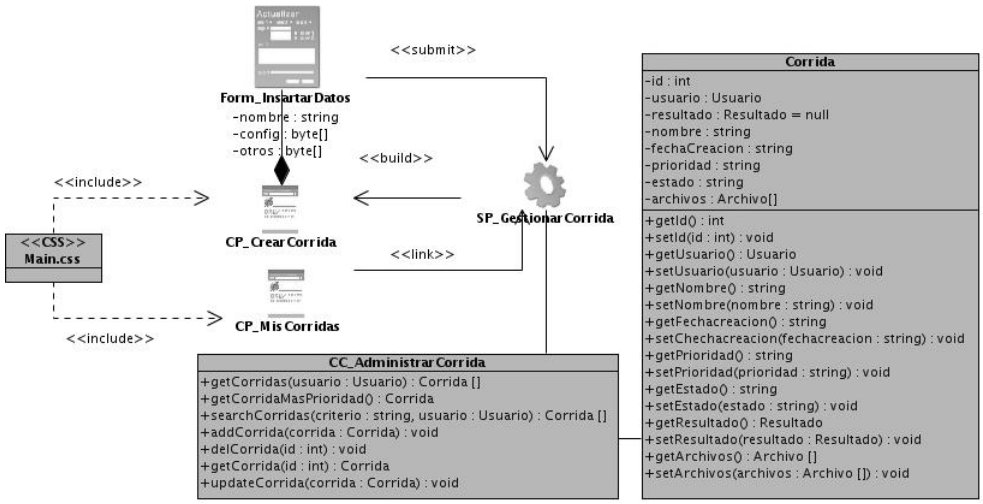

Figura 2.6: Diagrama de Clases del Diseño del escenario Crear Corrida

El resto de los Diagramas de Clases del Diseño se encuentran en el anexo del trabajo.

2.7.3. Diagramas de Interacción

Se trata de un término genérico que se aplica a varios tipos de diagramas que hacen hincapié en las interacciones entre objetos. El patrón de interacción entre objetos se muestra en un Diagrama de Interacción. Estos tienen dos formas de representarlos, basadas en una misma información subyacente pero, resaltando cada una un punto de vista de la misma: Diagramas de Secuencia, Diagramas de Colaboración. Nos centraremos en la realización de Diagramas de Secuencia. Figura 2.7.

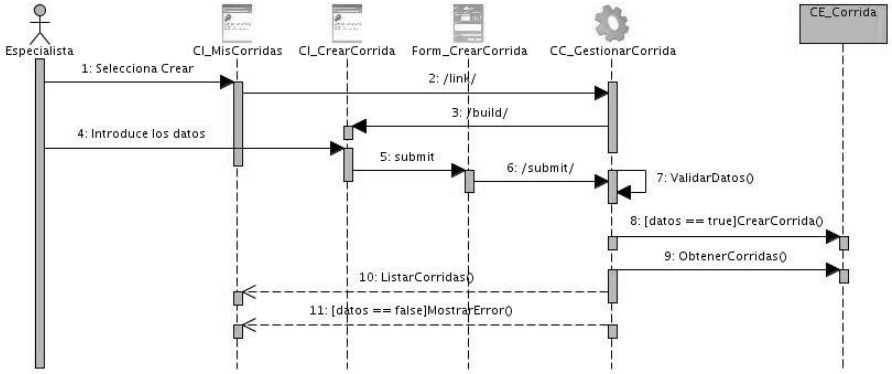

Figura 2.7: Diagrama de Secuencia del escenario Crear Corrida.

El resto de los Diagramas de Secuencia se encuentran en los anexos del trabajo.

2.8. Estructura de la base de datos

Para el desarrollo del Servicio de Acoplamiento Molecular es necesario la creación de una base de datos para el almacenamiento de los datos de entrada y de los datos de salida del servicio. Las tablas contienen instancias de cada uno de los módulos del sistema por lo que contienen un identificador propio, lo que permite un correcto funcionamiento del sistema.

Con la implementación del Caso de Uso Gestionar Corrida se incorpora la tabla: corrida; en esta nueva tabla se insertan los datos referentes a la corrida que vaya a crear el especialista en ese momento.

Descripción de la tabla corrida

Nombre: corrida		
Descripción: Registra la información referente a los datos necesarios para crear una corrida en el sistema.		
Atributo	**Tipo**	**Descripción**
id	integer(11)	identificador de la tabla.
usuarioid	varchar(255)	identificador de la tabla usuario.
nombre	varchar(255)	nombre dado por el especialista a la corrida.
fechacreacion	varchar(255)	fecha en que fue creada la corrida.
prioridad	varchar(255)	prioridad de la corrida en el sistema.
estado	varchar(255)	estado en que se encuentra la corrida en el sistema.

Las otras tablas creadas son: usuario, que guarda la información necesaria de los especialistas que acceden al sistema; respuesta, que almacena la información dada por el sistema a la corrida del especialista; y archivo que almacena la los archivos necesarios para generar una la corrida del especialista, así como los archivos de los resultados del especialista.

Las descripciones los atributos de las otras tablas de la base de datos se encuentran en los anexos del trabajo.

A partir de la relación de cada una de las entidades utilizadas, se genera el siguiente Modelo de Datos:

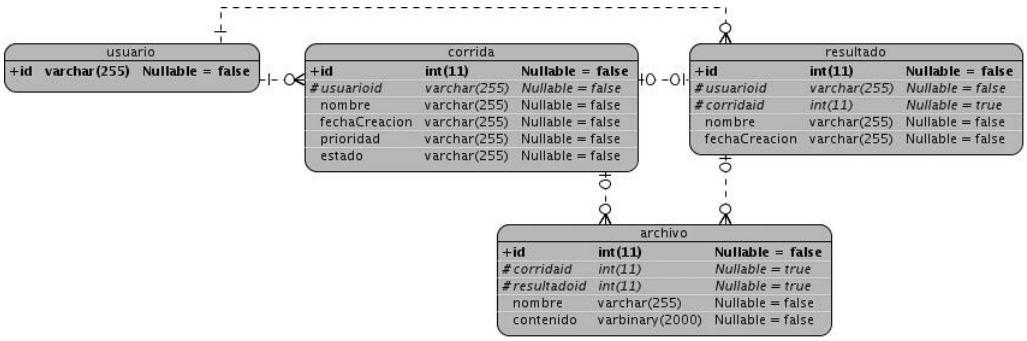

Figura 2.8: Modelo de datos

2.9. Conclusiones parciales

El Modelo de Dominio permitió ofrecer una visión de los tipos más importantes de objetos que existen o los eventos que suceden en el entorno donde se desarrollará el servicio de Acoplamiento Molecular. Con-

stituyó además, la base para identificar los requisitos funcionales con los cuales debe cumplir el sistema, dichos requisitos fueron las especificaciones del cliente.

Se determinaron aquellas restricciones con las cuáles deben cumplir el servicio de Acoplamiento Molecular mediante la definición los requisitos no funcionales.

Los artefactos modelados durante el Análisis y el Diseño constituyen la base de la implementación exitosa del producto.

El Modelo de datos, permitió obtener un mayor conocimiento de las relaciones entre las clases persistentes que posee la Plataforma de Servicios Bioinformáticos de la UCI para el desarrollo del servicio de Acoplamiento Molecular.

Capítulo 3

Implementación y Prueba

Introducción

Este capítulo tiene como objetivo convertir los elementos del diseño en elementos de implementación. Describe los elementos del modelo del diseño y se implementa un servicio web y una interfaz cliente en términos de componentes. También presenta el código fuente de las principales clases y pantallas de la aplicación como los prototipos de interfaz de usuario y el mapa de navegación.

3.1. Ejecución del Dock 6.5

Para realizar los cálculos de acoplamiento molecular mediante el Dock 6.5 se utilizó la Plataforma de Cálculo Distribuido T-Arenal.

En la clase DataManager.java se realizó la división del problema en subproblemas más pequeños con una granulalidad de uno: a cada PC cliente se le entrega la molecula objetivo y un ligando; se recepcionan las respuestas dadas por las PCs clientes y se contruyen los ficheros de salida. No se le puso restricciones en la cantidad de PCs a utilizar.

En la clase Algorithm.java se recepciona, ejecuta y se retorna una solución a la ejecución del Dock 6.5.

3.2. Servicio Web de Acoplamiento Molecular

3.2.1. Métodos públicos

El servicio web de Acoplamiento Molecular desarrollado presenta los siguientes métodos públicos para la realización de una corrida:

- public void closeDockExecution(String arg0)
- public String createDockExecution(String arg0)
- public byte[] downLoadFile(String arg0, String arg1)
- public void execute(String arg0)
- public boolean isFinish(String arg0)
- public String[] getFilesNames(String arg0)
- public boolean upLoadFiles(String arg0, String arg1, byte[] arg2)

3.2.2. Consumo del servicio

Un correcto consumo del servicio de Acoplamiento Molecular viene dado por la siguiente sucesión de pasos:

1. createDockExecution donde se le pasa como parámetros un nombre a dicha ejecución y se obtiene un identificador propio de la ejecución o un mensaje de error si no se pudo crear la ejecución en el servidor; el método crea la ejecución en el servidor.

2. upLoadFiles donde se le pasan como parámetros el identificador definido en el paso 1, el nombre del i-ésimo archivo a subir y el contenido del i-ésimo archivo a subir en forma de arreglo de byts y se obtiene un afirmativo si se logró subir el archivo, negativo en cualquier otro caso; el método crea el archivo i-ésimo en el servidor.

3. execute donde se le pasa como parámetros el identificador definido en el paso 1; el método ejecuta la corrida.

4. isFinish donde se le pasa como parámetros el identificador definido en el paso 1 y se obtiene un afirmativo si se terminó la ejecución, negativo en caso contrario.

5. getFilesNames donde se le pasa como parámetros el identificador definido en el paso 1 y se obtiene los nombres de los archivos generados por la ejecución.

6. downLoadFile donde se le pasan como parámetros el identificador definido en el paso 1 y el nombre del archivo i-ésimo obtenido en el paso 5; y se obtiene el contenido en forma de arreglo de byts del archivo i-ésimo obtenido en el paso 5.

7. closeDockExecution donde se le pasa como parámetros el identificador definido en el paso 1; el método cierra y borra todos los archivos correspondiente a la corrida de identificador definido en el paso 1.

3.2.3. Código fuente

execute(String arg0)

Como bien se observa el método execute es uno de los más importantes del Servicio Web de Acoplamiento Molecular, que es el encargado de conectarse con la Plataforma de Cálculo Distribuido T-Arenal, enviarle los archivos necesarios para la ejecución, recepcionar los archivos de respuesta y almacenarlos en el servidor; a continuación se muestra el código fuente del método:

Algoritmo 3.1 Algoritmo para ejecutar una corrida en el Servicio Web.

```
public void execute(String arg0) {
    try {
        // ----- 1 -----
        UserAuthentication dockUser = new UserAuthentication("usuario", "password");
        HostCommunication dockHost = new HostCommunication("ipServerTArenal", port);
        SystemConnection dockSystem = new SystemConnection(Locale.getDefault(), dockUser, dockHost);
        // ----- 2 -----
        Problem dockProblem = new Problem(arg0, arg0, ExecutionPriority.Low);
        Vector<String> users = new Vector<String>();
        File[] lib = LIB.listFiles();
        File problemFiles = new File(USERFILES, arg0);
        File[] files = problemFiles.listFiles();
        File[] filesExecution = new File[files.length + 1];
        filesExecution[0] = dock;
        System.arraycopy(files, 0, filesExecution, 1, files.length);
        dockSystem.addProblem(dockProblem, users, algorithm, datamanager, null, lib, filesExecution).start();
        // ----- 3 -----
        while (dockSystem.getProblemManager().getProblem(arg0).getState() == ProblemState.Disabled) {
            Thread.sleep(1000);
        }
        // ----- 4 -----
        ExecutionFilesUploader dockJob = dockSystem.getProblemManager().executeProblem(arg0, new File[0]);
        dockJob.start();
```

```
        // ----- 5 -----
        Vector<Solution> v = null;
        int pos;
        do {
            Thread.sleep(60000);
            v = dockSystem.getSolutionManager().getAllSolutions();
        } while ((pos = search(arg0, v)) == -1);
        // ----- 6 -----
        File solution = new File(USERRESULT, arg0);
        Solution dockSolution = v.get(pos);
        String solutionID = dockSolution.getName();
        long solutionSize = dockSolution.getSize();
        File outputFile = new File(solution, arg0 + ".zip");
        dockSystem.getSolutionManager().downloadSolution(solutionID, outputFile).start();
        // ----- 7 -----
        while (solutionSize != outputFile.length()) {
            Thread.sleep(1000);
        }
        // ----- 8 -----
        File myRes = new File(solution, "userResult");
        myRes.mkdir();
        Process pUnZip = Runtime.getRuntime().exec("unzip " + outputFile + " -d " + myRes);
        pUnZip.waitFor();
        // ----- 9 -----
        dockSystem.getSolutionManager().removeSolution(solutionID);
        dockSystem.getProblemManager().removeProblem(arg0);
        // ----- 10 -----
        compWriter(new File(USERRESULT, arg0), "true");
    } catch (Exception e) {
    }
}
```

Leyenda

1. Crear la conexión a T-Arenal.

2. Crear el problema para T-Arenal.
3. Esperar hasta que termine de crear el problema en T-Arenal.
4. Mandar a ejecutar el problema en T-Arenal.
5. Esperar a que termine la ejecución del problema en T-Arenal.
6. Descargar la solución del problema desde T-Arenal.
7. Esperar a que termine la descarga de la solución del problema desde T-Arenal.
8. Descomprimir la solución del problema.
9. Eliminar la solución y el problema de T-Arenal.
10. Cambiar el estado de la ejecución a terminado.

3.2.4. Experimento y resultados

En estos experimentos fueron utilizados una base de datos de 500 ligandos del ZINC [39] y como proteína objetivo con el código 1VRT del PDB [40], ambos provenientes de los DEMOS del Dock6.5. Fueron utilizadas las máquinas de los laboratorios de la UCI donde se encuentra montado el sistema T-Arenal, todas con sistema operativo basado en GNU/Linux.

Tiempos de respuesta

En el experimento fueron utilizados subconjuntos de 1, 50, 100, 200, 250 y 500 ligandos. La **Tabla** 1 muestra los tiempos de respuesta obtenidos como resultado del uso del servicio de Acoplamiento Molecular.

Tabla 1 Ejecuciones de Acoplamiento molecular en el Servicio de Acoplamiento Molecular.

Cantidad de ligandos	1	50	100	200	250	500
Una PC	0:00:02	0:02:04	0:05:45	0:18:16	0:27:12	1:23:23
Servicio Acoplamiento Molecular	0:01:11	0:04:19	0:03:13	0:03:12	0:03:09	0:07:41

Como se muestra en la **Figura** 3.1, el servicio desarrollado ha dado solución al problema planteado en la introducción de este trabajo, ya que logra reducir hasta en un 75 % el tiempo de respuesta de una gran simulación de Acoplamiento Molecular.

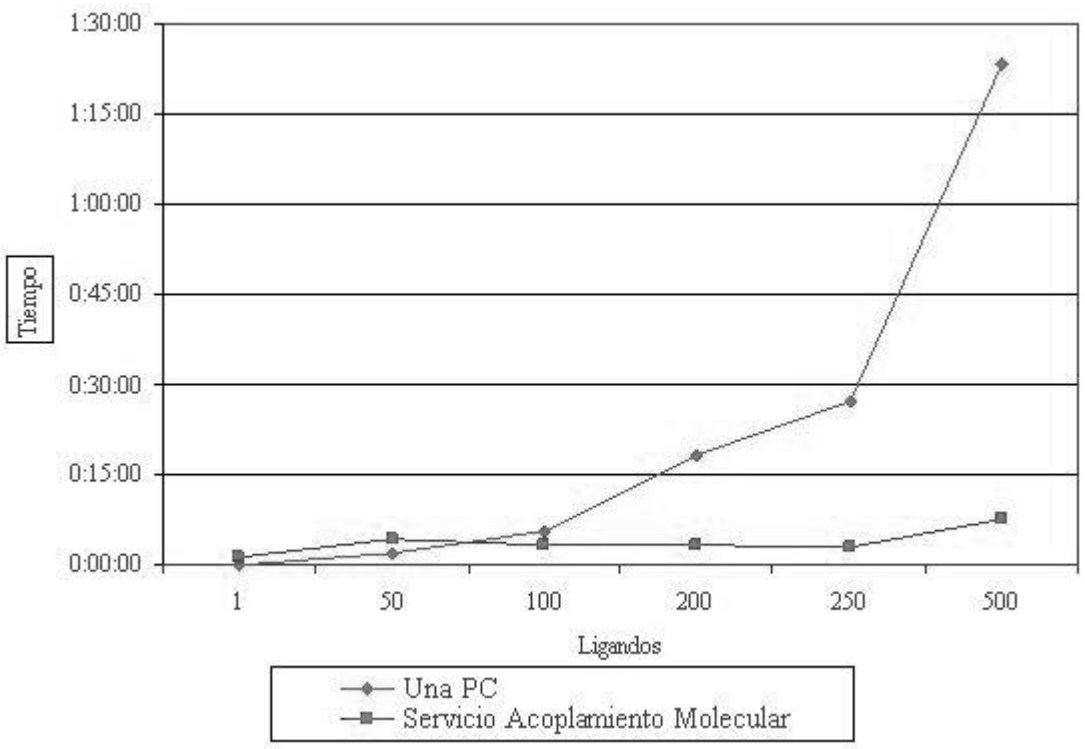

Figura 3.1: Tiempo en obtener el resultado de los acoplamientos moleculares para los set de ligando empleados.

SpeedUp

Para el cálculo del SpeedUp que es la relación que tienen las ejecuciones del proceso distribuido con la ejecución en una sola estación de trabajo, fueron utilizadas 1, 20, 40, 80 y 100 máquinas, dando como resultado: ver **Tabla** 2.

Tabla 2 SpeedUp de Acoplamiento molecular en el Servicio de Acoplamiento Molecular.

Cantidad de PCs	Tiempo ejecución	SpeedUp
1	1:55:36	1
20	0:15:29	7.47
40	0:07:15	15.94
80	0:07:20	15.76
100	0:09:09	12.63

Como se muestra en la **Figura** 3.2 tenemos un incremento gradual de la velocidad de procesamiento a

medida que se fueron incrementando la cantidad de PCs, hasta llegar a 40 máquinas a partir de la cual se logró una meseta y luego de esto a partir de las 80 PCs tiende a disminuir la velocidad de procesamiento.

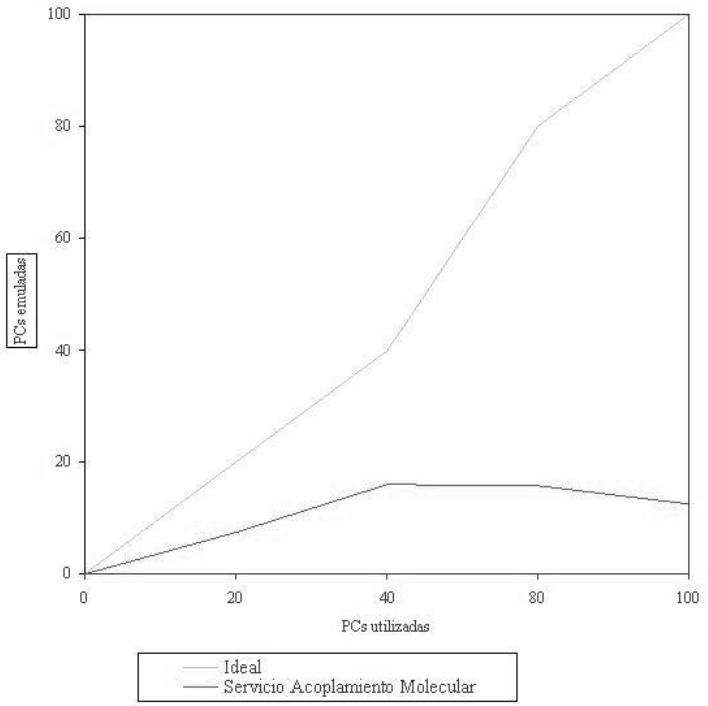

Figura 3.2: SpeedUp del Servicio de Acoplamiento Molecular.

3.3. Porlet Cliente

3.3.1. Diagramas de componentes

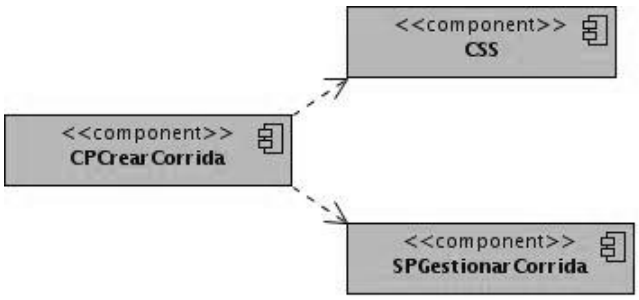

Figura 3.3: Diagráma de componentes Crear Corridas.

El resto de los diagramas de componentes se encuentran en el anexo del trabajo.

3.3.2. Pantallas de la aplicación

Mapa de Navegación

Interfaz de Usuario

Figura 3.4: Interfaz de usuario Crear Corrida.

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

Código Fuente

executarDock()

Este método es el encargado de obtener una corrida de la base de datos, mandarla a ejecutar sobre el Servicio de Acoplamiento Molecular, obtener la solución, guardarla en la base de datos, y repetir el ciclo.

Algoritmo 3.2 Algoritmo para ejecutar las corridas de los especialistas.

```
private void executarDock() {
   while (true) {
      try {
         // ----- 1 -----
         Corrida corrida = AdministrarCorrida.getCorridaMasPrioridad();
         if (corrida != null) {
            // ----- 2 -----
            corrida.setEstado(Estado.Ejecutando.name());
            AdministrarCorrida.updateCorrida(corrida);
            String nombre = corrida.getNombre();
            // ----- 3 -----
            DockingPortTypeProxy servicio = new DockingPortTypeProxy();
            // ----- 4 -----
            String id = servicio.createDockExecution(nombre);
            // ----- 5 -----
            List<Archivo> listaArchivosCorrida = AdministrarArchivo.getArchivos(corrida);
            for (int i = 0; i < listaArchivosCorrida.size(); i++) {
               Archivo archivo = listaArchivosCorrida.get(i);
               boolean up = false;
               do {
                  up = servicio.upLoadFiles(id, archivo.getNombre(), archivo.getContenido());
               } while (!up);
            }
            // ----- 6 -----
            service.execute(id);
            // ----- 7 -----
            while (!service.isFinish(id)) {
               Thread.sleep(60000);
            }
            // ----- 8 -----
            List<Archivo> listaArchivosResultado = new LinkedList<Archivo>();
            String[] files = servicio.getFilesNames(id);
            for (int i = 0; i < files.length; i++) {
               String name = files[i];
               byte[] bufferd = servicio.downLoadFiles(id, name);
               Archivo archivoResultado = new Archivo(0, name, bufferd);
               listaArchivosResultado.add(archivoResultado);
            }
            // ----- 9 -----
            servicio.closeDockExecution(id);
```

```
            // ----- 10 -----
            Date dat = new Date();
            String fechacreacion = "" + dat.getDate() + Character.toChars(47)[0] + Mes.values()[dat.getMonth()]
               + Character.toChars(47)[0] + dat.toString().split("␣")[5];
            // ----- 11 -----
```

```
                Resultado resultado = new Resultado(0, corrida.getUsuario(), corrida.getNombre(), fechacreacion);
                AdministrarResultado.addResultado(resultado);
                // ----- 12 -----
                for (int i = 0; i < listaArchivosResultado.size(); i++) {
                    listaArchivosResultado.get(i).setResultado(resultado);
                    AdministrarArchivo.addArchivo(listaArchivosResultado.get(i));
                }
                // ----- 13 -----
                corrida.setEstado(Estado.Terminado.name());
                AdministrarCorrida.updateCorrida(corrida);
            } else {
                // ----- 14 -----
                Thread.sleep(60000);
            }
        // ----- 15 -----
        } catch (Exception e) {
            try {
                Thread.sleep(60000);
            } catch (Exception e1) {
            }
        }
    }
}
```

Leyenda

1 Se obtiene la corrida de mayor prioridad si es diferente de nulo se continua con la ejecución, sino, ir a **14**.

2 Se le cambia el estado a la corrida a ejecutando.

3 Se realiza la conexión al servicio web.

4 Se crea la ejecución en el servicio web.

5 Se obtienen de la base de datos los archivos de la corrida y se suben al servicio web.

6 Se ejecuta la ejecución en el servicio web.

7 Se pregunta por la respuesta a la ejecución, se espera un minuto antes de volver a preguntar.

8 Descargamos los archivos del resultado.

9 Cerramos la ejecución en el servicio web.

10 Crea la fecha de terminación de la ejecución.

11 Crea y guarda el resultado en la base de datos.

12 Se le asigna la corrida a los archivos del resultado.

13 Se le cambia el estado a la corrida a terminada.

14 Se espera un minuto antes de pedir otra corrida.

15 En caso de error se espera un minuto antes de comenzar la ejecución otra vez.

3.3.3. Casos de prueba

Los casos de prueba se realizan para validar el correcto funcionamiento de la aplicación y así hacer la correcta liberación del producto.

A continuación se realiza el caso de prueba de Caja Negra para la sección Crear Corrida, el cual consiste en comprobar como se comporta el sistema a medida que vamos cambiando el set de datos utilizado.

Nombre de la Sección	**Escenarios de la Sección**	**Descripción de la funcionalidad**	**Flujo central**
SC1: Crear Corrida.	**EC1.1:** Introducir de forma correcta los datos necesarios para la creación de una corrida en el sistema.	El especialista selecciona la opción de Crear. El sistema muestra una interfaz que le permitirá al especialista ponerle nombre a la corrida y subir los archivos necesarios para su ejecución y presiona el botón Aceptar.	Selecciona la opción Crear.

	EC1.2.1: Introducir de forma errónea el nombre necesario para la creación de una corrida en el sistema.	El especialista selecciona la opción de Crear. El sistema muestra una interfaz que le permitirá al especialista ponerle nombre a la corrida y subir los archivos necesarios para su ejecución y presiona el botón Aceptar.	Selecciona la opción Crear.
	EC1.2.2: Introducir de forma errónea el archivo de configuración necesario para la creación de una corrida en el sistema.	El especialista selecciona la opción de Crear. El sistema muestra una interfaz que le permitirá al especialista ponerle nombre a la corrida y subir los archivos necesarios para su ejecución y presiona el botón Aceptar.	Selecciona la opción Crear.
	EC1.1.1: Cancelar la creación de una corrida en el sistema.	El especialista selecciona la opción de Crear. El sistema muestra una interfaz que le permitirá al especialista ponerle nombre a la corrida y subir los archivos necesarios para su ejecución y presiona el botón Aceptar.	Selecciona la opción Cancelar.

Variables

1. Nombre correcto de la corrida: Corrida1
2. Nombre incorrecto de la corrida: Corrida 1 ó _
3. Archivo configuración correcto: dock1.in
4. Archivo configuración incorrecto: dock1.conf
5. Otros archivos: Archivos necesarios definidos en dock1.in

Id del escenario	**Escenario**	**V1**	**V2**	**V3**	**V4**	**V5**	**Respuesta del sistema**	**Resultado de la prueba**
EC1.1	Crear una Corrida en el sistema.	V	F	V	F	V	El sistema muestra una interfaz que le permitirá al especialista crear una Corrida en el sistema.	Se creó la corrida en el sistema, se mostró la interfaz Mis Corridas y se listó las corridas del especialista.
EC1.2.1	Crear una Corrida en el sistema.	F	V	V	F	V	El sistema muestra una interfaz que le permitirá al especialista crear una Corrida en el sistema.	El sistema muestra un mensaje indicando el error.
EC1.2.2	Crear una Corrida en el sistema.	V	F	F	V	V	El sistema muestra una interfaz que le permitirá al especialista crear una Corrida en el sistema.	El sistema muestra un mensaje indicando el error.
EC1.1.1	Cancelar la creación de una Corrida en el sistema.	-	-	-	-	-	El sistema muestra la interfaz Mis Corridas y lista las corridas del especialista.	Se canceló la creación de la corrida en el sistema, se mostró la interfaz Mis Corridas y se listó las corridas del especialista.

El resto de los casos de prueba se encuentran en el anexo del trabajo.

3.4. Guía de despliegue

En esta guía se explica como desplegar cada componente del Servicio de Acoplamiento Molecular para la Plataforma de Servicios Bioinformáticos de la UCI. Las palabras en **negrita** son las que se tienen que modificar para un correcto despliegue.

3.4.1. Servicio Web.

El servcio web: WSDock es el encargado de realizar la ejecución del Dock6.5 sobre T-Arenal. Para su despliegue y ejecución sobre la Plataforma de Cálculos Distribuidos T-Arenal, es necesario descomprimir el archivo DockFiles.zip, presente en /tools, en la raíz del servidor del servicio web, en caso de estar en sistema operativo Linux en: /root.

Luego especificar la ubicación del servidor central de T-Arenal, para ello importar el proyecto WSDock en EclipseEE y abrir el archivo Dockin.java presente en /src/wservices y en la sección execute, llenar los datos necesarios:

UserAuthentication dockUser = new UserAuthentication("**usuario**", "**password**");

HostCommunication dockHost = new HostCommunication("**direcciónIp.servidor.central.T-Arenal**", **puerto**);

Por último compilar como servicio web bajo el nombre WSDock.war y desplegar en Apache Tomcat.

3.4.2. Base de Datos PostgreSQL

La base de datos es la encargada de almacenar los datos persistentes de todo el sistema. Para su despliegue sobre PostgreSQL 8.4 o superior, es necesario el scrip dock.sql presente en /ScripSQL. Se ejecuta el scrip en el gestor y queda montada la base de datos.

3.4.3. Porlet Cliente.

El porlet cliente: WSPDock-portlet es el encargado de gestionar las corridas y administrar los resultados de los especialistas. Para su despliegue y ejecución en el Portal de Servicios Bioinformáticos de la UCI, es necesario especifiacar la ubicación del gestor de base de datos PostgreSQL, para ello importar el proyecto WSPDock en EclipseEE y abrir el archivo hibernate.cfg.xml presente en /src y llenar los datos necesarios:

<property name="hibernate.connection.username">**usuarioBaseDatos**</property>

<property name="hibernate.connection.password">**passwordBaseDatos**</property>

<property name="hibernate.connection.url">jdbc:postgresql://**direcciónIp.servidor.baseDatos.PostgreSQL**:**puerto**/**NombreBaseDatos**</property>

Luego especificar la ubicación del servicio web WSDock, para ello abrir el archivo DockingLocator.java presente en /src/conexion/ws y llenar los datos necesarios:

// Use to get a proxy class for DockingHttpSoap11Endpoint private java.lang.String DockingHttpSoap11Endpoint_address = "http://**direcciónIp.servidor.servicioWeb.WSDock**:**puerto**/WSDock/services/Docking.DockingHttpSoap11Endpoint/";

Por último compilar como ejecutable web bajo el nombre WSPDock-portlet.war y desplegar en el Portal de Servicios Bioinformáticos de la UCI.

3.5. Conclusiones parciales

En este capítulo se logra el resultado del diseño, implementando el sistema en términos de componentes, realizando el diagrama de componentes. Se presenta ejemplos de código fuente de las principales clases, ejemplos de la interfaz de usuario y el mapa de navegación del sistema. Se realizaron experimentos que evidenciaron como a partir de un determinado número de ligandos los tiempos obtenidos mediante el servicio implementado son menores que el tiempo de cálculo en una sola PC. Por último se crea una guía de despliegue para la instalación de todo el sistema.

Conclusiones

- Partiendo del análisis y el diseño se implementó un Servicio Web que hace uso de la Plataforma de Cálculo Distribuido T-Arenal para realizar el Acoplamiento Molecular y se desarrolló un Porlet Cliente que consume el Servicio Web y le permite al especialista Gestionar sus ejecuciones y Administrar los resultados dado por el sistema.
- Se utilizó la Plataforma de Cálculo Distribuido T-Arenal, sobre un conglomerado de estaciones de trabajo lográndose reducir significativamente el tiempo en obtener la respuesta de grandes simulaciones de acoplamiento molecular.
- Se realizó el análisis y el diseño del servicio de Acoplamiento Molecular y del Porlet Cliente, obteniéndose los artefactos correspondientes a cada fase de la metodología propuesta.
- Se realizó el diseño de una Base de Datos relacional para el almacenamiento del gran volumen de datos que se hará necesario guardar.
- Se realizaron y se ejecutaron los casos de prueba para validar el Servicio Web de Acoplamiento Molecular y el Porlet Cliente.

Recomendaciones

- Enriquecer el Servicio con otras herramientas computacionales que realicen acoplamiento molecular para que así los especialistas tengan la posibilidad de escoger las más adecuadas en correspondencia a su investigación.
- Incorporar un mecanismo de seguridad para el Servicio Web de Acoplamiento Molecular implementado que permita hacer frente a posibles ataques o intrusiones malignas al sistema desarrollado.

Referencias bibliográficas

[1] Abbas I. Integración de los modelos de simulación en el diseño de los ensayos clínicos. Tesis Doctorales en Red. 2003;Available from: http://www.tesisenred.net/handle/10803/6506.

[2] E H. DockingServer: molecular docking calculations online. Acta Pharm Hung. 2009;Available from: http://www.ncbi.nlm.nih.gov/pubmed/19526678.

[3] Lyskov S. The RosettaDock server for local protein-protein docking. Oxford Journals. 2008 abril;36(2). Available from: http://nar.oxfordjournals.org/content/36/suppl_2/W233.short.

[4] Kitchen DB. Docking and scoring in virtual screening for drug discovery: methods and applications. Nature Reviews, Drug Discovery. 2004 noviembre;3. Available from: http://www.nature.com/nrd/journal/v3/n11/pdf/nrd1549.pdf.

[5] Click2Drug;Available from: http://www.click2drug.org/directory_Docking.html.

[6] The Official UCSF DOCK Web-site;Available from: http://dock.compbio.ucsf.edu/DOCK_6/index.htm.

[7] Tsui B. PRIME 2011 Osaka University. 2011;.

[8] Goodsell DS. Automated docking of flexible ligands: Applications of autodock. Journal of Molecular Recognition. 1998 diciembre;9(1). Available from: http://onlinelibrary.wiley.com/doi/10.1002/(SICI)1099-1352(199601)9:1%3C1::AID-JMR241%3E3.0.CO;2-6/abstract.

[9] Morris GM. Distributed automated docking of flexible ligands to proteins: Parallel applications of AutoDock 2.4. Journal of Computer-Aided Molecular Design. 1996;10(4):293 – 304. Available from: http://www.springerlink.com/content/u35435xp5j327175/.

[10] Sitio Oficial AutoDock. 1989 - 2012;Available from: http://autodock.scripps.edu/.

[11] Apache Tomcat. Web Official Cite;Available from: http://tomcat.apache.org/.

[12] Mérida D. Adaptación de Contenidos Web Considerando las Características de los Dispositivos de Acceso. Enlace Informático. 2006 Dic;5(1):9. Available from: http://atreides.udg.edu/bcds/images/bcds/papers/pdf/adaptaci_n_de_contenidos_web_considerando_las_caracter_sticas_de_los_dispositivos_de_acceso.pdf.

[13] Introducción a Apache Tomcat 5.5. Universidad de Sevilla;Available from: http://www.lsi.us.es/docencia/get.php?id=1923.

[14] de León Estupiñan J. Portales y Portlets Web. 2011;Available from: http://www.slideshare.net/jossydeleon/portales-y-portlets-web-9377907.

[15] Andry Daniel Díaz León. Módulo básico de la Plataforma de Servicios Bioinformáticos. 2012;.

[16] Orlando Martínez Pérez ADDL Alina Agramonte Delgado. UCIBioSoft: Portal web de servicios bioinformáticos. VIII Congreso Internacional de Informática en la salud. 2011;Available from: http://www.informaticahabana.cu/node/2505.

[17] Longendri Aguilera Mendoza DMMT César Raúl García Jacas. Sistema de Cómputo Distribuido. Telem@tica. 2008;16. Available from: http://www.informaticahabana.cu/node/2505.

[18] Informatica en salud 2009;Available from: http://www.informatica2009.sld.cu/Members/dmmirayes/sistema-de-computo-distribuido/.

[19] SPRING FRAMEWORK;Available from: http://www.springsource.org/spring-framework.

[20] JBoss Community;Available from: http://www.hibernate.org/.

[21] Perera S. Axis2, Middleware for Next Generation Web Services. Web Services, 2006 ICWS '06 International Conference on. 2006 septiembre;p. 833 – 840. Available from: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=4032100&abstractAccess=no&userType=inst.

[22] Lenguajes de Programación;Available from: http://www.lenguajes-de-programacion.com/lenguajes-de-programacion.shtml.

[23] Definición de lenguaje de programación.;Available from: http://www.definicion.org/lenguaje-de-programacion.

[24] IDE. 2006 octubre;Available from: http://tecnologia.glosario.net/terminos-tecnicos-internet/ide-860.html.

[25] Geer D. Eclipse becomes the dominant Java IDE. IEEEXplorer digital library. 2005 julio;38(7):16 – 18. Available from: http://ieeexplore.ieee.org/xpl/tocresult.jsp?isnumber=31455.

[26] PlataformaEclipse.com;Available from: http://plataformaeclipse.com/faq/.

[27] Clase Modelo;Available from: http://www.antares.itmorelia.edu.mx/~jcolivar/courses/pm10a/pm_u3.ppt.

[28] Lizet Torres Flores C. Establecimiento de una Metodología de Desarrollo de Software para la Universidad de Navojoa Usando OpenUP;Available from: http://fit.um.edu.mx/CIDET/publicaciones/COMP-004-2008%20Establecimiento%20de%20una%20Metodolog%C3%ADa%20de%20Desarrollo%20de%20Software%20para%20la%20Universidad%20de%20Navojoa%20Usando%20OpenUP.pdf.

[29] Cornejo JEG. ¿Qué es UML?;Available from: http://www.docirs.cl/uml.htm.

[30] Herramientas CASE;Available from: http://www.cyta.com.ar/biblioteca/bddoc/bdlibros/proyectoinformatico/libro/c5/c5.htm.

[31] Zapata CM. UNC - Diagramador una herramienta upper CASE para la obtención de diagramas UML desde esquemas preconceptuales. Revista Universidad Eafit. 2007;43(147). Available from: http://www.doaj.org/doaj?func=abstract&id=337842.

[32] Callejas Cuervo M. Herramientas libres para modelar software. Revista Facultad de Ingeniería. 2005 diciembre;Available from: http://ervisual.hostoi.com/pdf/er_tecnico.pdf.

[33] Innova GS. Rational Rose Enterprise.;Available from: http://www.rational.com.ar/herramientas/roseenterprise.html.

[34] Zambrano Ramírez R. Sistemas gestores de base de datos;(14). Available from: http://www.csi-csif.es/andalucia/modules/mod_ense/revista/pdf/Numero_14/RAQUEL_ZAMBRANO_2.pdf.

[35] Ramakrishnan R. Database management system;Available from: http://pages.cs.wisc.edu/~dbbook/.

[36] Sitio oficial de PostgreSQL;Available from: http://www.postgresql.org/about/press/presskit91/es/.

[37] Sitio oficial de MySQL;Available from: http://dev.mysql.com/doc/refman/5.0/es/features.html.

[38] Agile Modeling;Available from: http://www.agilemodeling.com/artifacts/.

[39] ZINC a free database of commercially-available compounds for virtual screening.;Available from: http://zinc.docking.org/.

[40] PDB a Portal to Biological Macromolecular Structures.;Available from: http://www.rcsb.org/pdb/home/home.do.