Universidad de las Ciencias Informáticas
Facultad 6

Trabajo de Diploma para optar por el título de
Ingeniero en Ciencias Informáticas

Título: Portlet para la incorporación de las funcionalidades de Alas-SiRNA_Design a la Plataforma de Servicios Bioinformáticos.

Autor: Israel Jesús Bazán Espinosa
Tutores: MSc. Tonysé de la Rosa Martín
Ing. Andry Daniel Díaz León

Junio de 2013
"Año de 55 de la Revolución"

DECLARACIÓN DE AUTORÍA

Declaro ser el único 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 _____.

Israel Jesús Bazán Espinosa

_____
Firma del autor

Tonysé de la Rosa Martín

_____
Firma del tutor

Andry Daniel Díaz León

_____
Firma del tutor

DATOS DE CONTACTO

Tutores:

MSc. Tonysé de la Rosa Martín
Universidad de las Ciencias Informáticas, La Habana, Cuba
Máster en Bioinformática y Biología Computacional, UCI 2013.
E-mail: tdelarosa@uci.cu

Ing. Andry Daniel Díaz León
Universidad de las Ciencias Informáticas, La Habana, Cuba.
Ingeniero en Ciencias Informáticas
Líder del proyecto Plataforma de Servicios Bioinformáticos
E-mail: addiaz@uci.cu

AGRADECIMIENTOS

Mis palabras no me alcanzan para agradecer a todas esas personas que han compartido mis alegrías, mis problemas y sobre todas mis maldades que no son más que otra forma de expresarle que siempre los tengo presente y los llevaré en mi corazón.

A mi pequeña familia por estar ahí para mí, con sus problemas y sus defectos pero siempre listo para mí. A mi padre Pedro Bazán que aunque siempre estoy diciéndole jaranas admiro su bondad, humanismo y solidaridad hacia las demás personas. A mi madrasta Mireya que siempre ha estado para mí y por darme ese cariño de madre. A mis otros padres Sinecio y Lianis por cuidarme, mimarme y sobre todo por encaminarme por el buen camino. A mi madre pinareña, esa personita quisquillosa y cariñosa que se preocupó todo este tiempo por mí, a Cristi por dedicarme parte de sus dolores de cabeza. A Vladimir por recibirme en su familia con tanto cariño.

A mi familión de hermanos a los de sangre y a los de corazón gracias por estar ahí para mí. A mi hermano Idael que aunque tengamos nuestras diferencias no ha dejado de estar a mi lado. A Ismael por ser parte de mis bromas y por todas esas alegrías que me ha brindado. Al mejor de mis amigos mi herma Yuri, mi vida sería corta para agradecerte todo lo que has hecho por mí y por mi familia. A mis tres chicas, las hermanas que siempre quise tener, por soportar mis locuras mis ocurrencias y sobre todo por llorar conmigo en mis momentos difíciles, mi vida no sería la misma sin ustedes Yexi, Diana y mi pinareña Niuvys. A Jorgito por ser el mejor amigo de la universidad, por su apoyo y consejo en todo momento.

A todos esos amigos que he hecho en el transcurso de estos 5 años. A los que ya no están Angel, Ernesto Carrasco y Yunior. A los que ya terminaron Alexei, Reynaldo, Ariel, Wendy y Ochoa. Con los que compartí desde los primeros días de esta universidad Danelis, Yudelis, Sheila, Maricel, Jisel, Lisandra, Yosbel, Rolando, Arian, Yasmany el Fuki, Yasmany el flaco, Quintana, Viltre, Rodolfo y Omar. A Katiuska por haber sido una parte importante de mi vida universitaria. A las demás amistades Ernesto el loco, Arelis, Claudia, la insoportable y su mamá, las tías del comedor Katia y Grisel, Ailsa, Diana Malangón, Diógenes, Raidel. A los grupos 5 y 9 por los cuales tuve el placer de pertenecer. A los amigos de la zona Gleidis, Víctor, Jorgito, Reynier, Yaisel, Yordan, Chávez y Yanela. A todos aquellos que de una forma u otra compartieron conmigo durante la estancia en esta universidad.

DEDICATORIA

En la vida somos guiados por varias personas pero solo pocas hacen que se conviertan en un verdadero ejemplo a seguir. A esas personas especiales en mi vida va dedicado este trabajo.

A los dos seres que se encargaron de formarme y hacer de mí la persona que soy. Las cuales ya no las tengo entre mis seres queridos pero siempre las llevo en mi corazón. A mi abuelo Israel el padre sabio y tierno que desee tener y mi madre Ana María la cual me enseño que la vida está hecha de dificultades pero siempre se puede seguir adelante con una pequeña sonrisa. Hoy estoy orgulloso de ser parte de su familia y le doy las gracias donde quiera que se encuentren.

A la familia Yero por dejarme entrar a su casa y corazón. A Lianis la que supo llenar el vacío que me quedó y por darme todo ese cariño desbordado. A mi hermano Yuri que sin ser mi familia ha estado siempre presente en todos mis momentos difíciles. A Sinecio y Yamilet por soportar mis locuras y hacerme sentir parte de la familia.

A todos aquellos que confiaron en mí, aun cuando ya había perdido la confianza.

RESUMEN

Con el desarrollo científico-tecnológico que ha alcanzado Cuba en los últimos años, sitúa en un lugar muy privilegiado el estudio de la Bioinformática. Por esta razón se han desarrollado varios proyectos entre diferentes centros científicos del país y la Universidad de las Ciencias Informáticas (UCI) para fomentar el auge de esta disciplina. Uno de estos trabajos es la aplicación Alas-SiRNA_Design, la misma fue desarrollada con el lenguaje de programación Java para el diseño de pequeños ARN de interferencia (siRNA, por sus siglas en inglés) a partir de la información del genoma humano. La presente investigación muestra la incorporación de las funcionalidades de Alas-SiRNA_Design a la Plataforma de Servicios Bioinformáticos a través de portlets como componentes de interfaz de usuario, que permiten a los especialistas de todo el país acceder y utilizar las funcionalidades del mismo mediante un Portal de Servicios Bioinformáticos.

PALABRAS CLAVE

Alas-SiRNA_Design, Plataforma de Servicios Bioinformáticos, Portlet, servicio web.

TABLA DE CONTENIDOS

INTRODUCCIÓN

El conocimiento es uno de los recursos estratégicos de la sociedad en la que estamos inmersos y la base fundamental de la futura sociedad que queremos alcanzar. Para poder llegar a ella se debe ser capaz de adquirir y utilizar el capital intelectual del que se dispone, pero también es necesario, y de forma primordial, transferirlo en beneficio de la propia sociedad para que, además de poder ser utilizado por ella, sea un medio que ayude al desarrollo, generando a su vez más conocimiento.

Este conocimiento ha alcanzado gran nivel apoyándose en los avances tecnológicos que se evidencian en la sociedad. Este desarrollo acelerado ha propiciado el surgimiento de las Tecnologías de la Información y las Comunicaciones (TICs).

Las TICs son aplicables en sectores como la economía, la cultura, la sociedad, el deporte, la salud y la educación. Principalmente ciencias como la Biología, la Física, la Química o la Matemática encuentran en las TICs la vía para un incremento de los conocimientos, y por ende contribuir al desarrollo de las mismas. Entre las TICs resaltan las Tecnologías Web. Las cuales sirven para acceder a los recursos de conocimiento disponibles en la red de redes, más conocida como Internet.

Dentro de las Tecnologías Web se destacan los Portales Web, que se definen como un punto de entrada común a una colección de recursos electrónicos integrados, donde se ofrecen una serie de servicios complementarios, tales como: búsqueda interna, personalización, herramientas de comunicación, servicios de información y alerta y otros servicios específicos asociados a la tipología del portal. (1)

Los portales bioinformáticos poseen una colección de herramientas destinadas al trabajo con sistemas biológicos. Entre los existentes para las consultas bioinformáticas se encuentran: el Banco de Datos de ADN de Japón (DDBJ por sus siglas en inglés), el Centro Nacional para la Información Biotecnológica (NCBI, por sus siglas en inglés), el Expasy (acrónimo de *Expert Protein Analysis System*) y el Instituto Europeo de Bioinformática (EBI, por sus siglas en inglés).

Actualmente en Cuba no existen portales dedicados a la Bioinformática, se cuenta con grupos de Bioinformática en algunos polos científicos del país, como el Centro de Ingeniería Genética y Biotecnología (CIGB), el Centro de Inmunología Molecular (CIM) y otros grupos en la Universidad de La Habana y la Universidad Central de Las Villas.

La Bioinformática, conocida como Biología Molecular Computacional, corresponde como tal a una disciplina científica que utiliza las tecnologías de la información para organizar, analizar y distribuir información de biomoléculas con la finalidad de responder preguntas complejas. (2)

En ocasiones se restringe el estudio de la Bioinformática al manejo y análisis de bases de datos biológicas principalmente de secuencias, podría atribuírsele un sentido más amplio, como la fusión de las técnicas computacionales con el entendimiento y apreciación de datos biológicos, el almacenamiento, recuperación, manipulación y correlación de datos procedentes de distintas fuentes. (3)

Cuba se encuentra inmersa en un proceso de informatización de la sociedad apostando por los avances de la informática a nivel mundial. En el año 2002 fue creada la Universidad de las Ciencias Informáticas (UCI) con el objetivo de formar profesionales capacitados en el desarrollo de *software*. La UCI es un centro que posee amplios recursos tecnológicos y computacionales para el procesamiento de grandes cálculos, desarrollo de aplicaciones y con acceso a Internet para la descarga de programas y artículos científicos.

Debido a la importancia estratégica que posee actualmente el desarrollo de áreas científicas como la Bioinformática, se crea el Departamento de Bioinformática, perteneciente al centro DATEC de la Facultad 6. El departamento se propuso realizar un portal que brinde acceso a programas y servicios para los investigadores de universidades e instituciones del país, en áreas de la Bioinformática. Este portal está concebido para brindar acceso a la Plataforma de Servicios Bioinformáticos.

Con el fin de brindar la mayor cantidad de servicios para procesar información biológica en el Portal Web de Servicios Bioinformáticos surge la necesidad de incorporar a dicho portal las funcionalidades que brinda la aplicación web Alas-SiRNA_Design, la cual permite a los investigadores hacer uso de algoritmos y técnicas para el análisis de la influencia de las mutaciones puntuales en los micro RNA de interferencias (miRNA). Su integración a la Plataforma de Servicios Bioinformáticos permitiría la utilización de los servicios que brinda Alas-SiRNA_Design y que puedan ser utilizadas en conjunto con otras funcionalidades para el desarrollo bioinformático en Cuba.

Por lo anteriormente planteado, el **problema a resolver** de la presente investigación es: ¿Cómo integrar los servicios web de Alas-SiRNA_Design a la Plataforma de Servicios Bioinformáticos?

Se plantea como **objeto de estudio** las tecnologías para la integración de servicios web y se delimita como **campo de acción** integración de los servicios web de Alas-SiRNA_Design a la Plataforma de Servicios Bioinformáticos.

Para dar solución al problema anterior se define como **objetivo general**: Desarrollar un Portlet que incorpore las funcionalidades de Alas-SiRNA_Design a la Plataforma de Servicios Bioinformáticos, del cual se derivan los siguientes **objetivos específicos**:

- Definir los requisitos funcionales del sistema.
- Diseñar los requisitos definidos.
- Implementar las funcionalidades del sistema.
- Probar las funcionalidades implementadas.

Para dar respuesta a estos objetivos se trazaron las siguientes **tareas de la investigación**:

- Análisis del estado del arte de las principales metodologías de desarrollo de software, herramientas y tecnologías para la implementación del sistema.
- Análisis de la arquitectura de la Plataforma de Servicios Bioinformáticos.
- Elaboración de los requisitos funcionales del sistema.
- Selección de los patrones arquitectónicos y de diseño a utilizar.
- Construcción del modelo del diseño del sistema
- Implementación de los requisitos funcionales.
- Realización de pruebas para la evaluación de las funcionalidades.

El presente trabajo está estructurado en cuatro capítulos los cuales se describen a continuación:

Capítulo 1. Fundamento teórico: este capítulo abarca el estudio del arte del tema, así como la elaboración del marco teórico. Se presentan las metodologías de desarrollo y herramientas a utilizar. Se enuncian las tecnologías y el lenguaje de modelado.

Capítulo 2. Características del sistema: profundiza en las funcionalidades que debe tener el sistema. Se presentan los requisitos funcionales y no funcionales. Además se muestran los casos de usos y su descripción, así como los actores que interactúan con el sistema.

Capítulo 3. Diseño de la aplicación: se desarrollan los diagramas de clases del diseño, para ello se analizan los diferentes patrones arquitectónicos y de diseño y se seleccionan los más convencionales, teniendo en cuenta los patrones escogidos se confeccionan los diagramas de interacción.

Capítulo 4. Implementación y Prueba: en este capítulo se exponen los diagramas de componentes, se describen las principales implementaciones realizadas en el sistema. Además se presentan las pruebas realizadas a la aplicación para validar la solución propuesta.

CAPÍTULO 1. FUNDAMENTO TEÓRICO

En las últimas décadas el mundo ha enfrentado continuos cambios en las TICs, lo cual ha propiciado el desarrollo de ciencias emergentes como la Bioinformática. Este capítulo engloba conceptos básicos relacionados con tecnologías web y aplicaciones vinculadas a este sector. Además se analizan las tecnologías y principales herramientas, fundamentando la selección de estas para el desarrollo de la solución propuesta.

1.1. Tecnologías Web e Internet

Los avances en las comunicaciones y las nuevas tecnologías están acercando la información al usuario final, así como facilitando su procesamiento. Uno de los cambios más importantes, tiene que ver con el soporte y canal de transmisión de la información. Internet y las tecnologías web, han conseguido que el usuario esté familiarizado con información hipermedia, incluyendo texto, imágenes, audio y vídeo. Cualquier ordenador conectado a la red constituye una fuente fácil de entrada de información y de servicios. Este hecho hace que cada vez cobre más fuerza la idea de que nos encontramos inmersos en una “sociedad de la información“. (4)

Las tecnologías web se utilizan para acceder a los recursos de conocimiento disponibles en Internet o en las intranets a través de un navegador. Facilitando el desarrollo de sistemas de gestión del conocimiento, su flexibilidad en términos de expandir el sistema; su sencillez de uso y la posibilidad de brindar conocimiento a todos los demás, por encima de jerarquías, barreras formales u otras cuestiones. Estas tecnologías pueden llegar a proporcionar recursos estratégicos debido a lo fácil que es personalizarla y construir con ella sistemas de gestión del conocimiento. (5)

Las tecnologías web permiten el desarrollo de aplicaciones distribuidas basadas en el modelo Cliente/Servidor. Las aplicaciones web suponen un importante cambio de enfoque con respecto al desarrollo de aplicaciones tradicionales. Su principal característica consiste en que la comunicación con el usuario se establece utilizando páginas web, que se pueden visualizar desde un navegador que se esté ejecutando en cualquier ordenador conectado a la red. Otra característica importante, consiste en que el código de la aplicación se puede ejecutar en el cliente, en el servidor o distribuirse entre ambos. (4)

Tecnologías web para integración de servicios web

Bajo el concepto de tecnología web se agrupan una serie de nuevas tecnologías y estándares que hacen posible Internet:

- **HTML (*Hypertext Markup Language*)** es el lenguaje básico para construir archivos de texto con hipervínculos, publicarlos en un servidor web para ser usado por diversos usuarios, que pueden leerlos gracias a los navegadores web.

- **TCP/IP (*Transport Control Protocol/Internet Protocol*)** es el protocolo de comunicación más utilizado de Internet e Intranets, admite la comunicación punto a punto entre diferentes computadoras de una red.

- **HTTP (*Hypertext Transfer Protocol*)** es el protocolo web que gestiona las peticiones y servicios de documentos HTML y está basado en TCP/IP.

1.2. Portales Bioinformáticos

En la actualidad se cuenta con numerosos portales dedicados al estudio bioinformático que brindan varias operaciones y servicios. Entre los más visitados por los investigadores se encuentran EMBL-Bank en el EBI, el DDBJ en el CIB/NIG y el GenBank en el NCBI. Estos portales se encargan del almacenamiento y análisis de las secuencias de nucleótidos. Su principal objetivo es brindar acceso a la información almacenadas en sus bases de datos por lo cual estas bases de datos intentan alojar todas las secuencias de nucleótidos que son de dominio público. Los portales más visitados serán nombrados a continuación.

El *Banco de Datos de ADN de Japón* (DDBJ) es el único banco de datos de secuencia de nucleótidos en Asia, que tiene la certificación oficial para recoger las secuencias de nucleótidos de los investigadores y para emitir el número de acceso de reconocimiento internacional a los que envían datos. Intercambia los datos recogidos con EMBL-Bank/EBI, el Instituto Europeo de Bioinformática y el GenBank / NCBI. (6)

El *Centro Nacional para la Información Biotecnológica* (NCBI) cuenta con sistemas automatizados de almacenamiento y análisis de los conocimientos sobre la Biología Molecular, la Bioquímica y la Genética, facilitando el uso de dichas bases de datos y el *software* de la investigación entre la comunidad médica, coordinando los esfuerzos por reunir la información biotecnológica tanto a nivel nacional como internacional. (7)

El *Expasy* es el Portal de Recursos Bioinformáticos del *Swiss Institute of Bioinformatics* (SIB, por sus siglas en inglés), que proporciona acceso a bases de datos científicas y herramientas de software, en diferentes áreas de ciencias de la vida, incluida la Proteómica, la Genómica, la Filogenia, la Biología de sistemas, la Genética de poblaciones y la Transcriptómica. (8)

El *Instituto Europeo de Bioinformática* está dedicado a la investigación académica, se encuentra en el *Welcome Trust Genome Campus* en *Hinxton*, cerca de *Cambridge* (Reino Unido), que forma parte de la *European Molecular Biology Laboratory* (EMBL). Entre sus objetivos principales se encuentran:

- Proporcionar datos de libre acceso y servicios de Bioinformática para la comunidad es científicas que promueven el progreso científico.
- Contribuir al avance de la Biología básica a través de la investigación impulsada por la Bioinformática.
- Proporcionar formación avanzada de la Bioinformática para científicos de todos los niveles, desde estudiantes de doctorado hasta los investigadores independientes. (9)

1.3. Alas-SiRNA_Design

La plataforma Alas-SiRNA_Design permite el diseño de pequeños ARN de interferencia (siRNAs, por sus siglas en inglés) y la búsqueda, representación y visualización de datos biológicos para la toma de decisiones. Además ofrece a los especialistas una herramienta propia donde pueden incorporar nuevas ideas y algoritmos como la secuencia o los identificadores de los genes que se desean silenciar. Esta plataforma está compuesta por dos aplicaciones informáticas: el servidor SiRNA Web Services y el cliente Alas-SiRNA_Design. SiRNA Web Services es un servidor web que brinda una serie de servicios web relacionados con la información que se necesita de la base de datos y la implementación de los principales algoritmos. El cliente web Alas-SiRNA_Design utiliza los servicios del servidor para consultar y mostrar la información solicitada por los usuarios. (10)

Entre las funcionalidades que brinda Alas-SiRNA_Design se encuentra: búsqueda de datos de genes y transcritos, análisis de secuencias de mRNA, ensamblaje de secuencias comunes a un gen o específicas de un transcrito de un gen, muestra los polimorfismos de un solo nucleótido (SNPs, por sus siglas en inglés) que estén en determinada región de un cromosoma o que coincidan con los micros ARN de interferencia (miRNAs, por sus siglas en inglés), así como el cálculo de energía libre. Tiene

como ventaja que incorpora al diseño de siRNAs información sobre la variabilidad genética de los individuos. (11)

1.4. Plataforma de Servicios Bioinformáticos

Esta plataforma está concebida y pensada en la Universidad de las Ciencias Informáticas. Tiene como nombre UCIBioSoft. El objetivo de este sistema es permitir a los investigadores acceder a características avanzadas, como son los flujos de trabajo y el uso de ontologías para la composición y ejecución de tareas, sin tener que manejar conceptos complejos, ni instalar programas sofisticados. Además de brindar servicios y aplicaciones que puedan facilitar el estudio y desarrollo de la Bioinformática en Cuba.

1.5. Servicios Web y Arquitectura Orientada a Servicios

El Consorcio de la Red de Redes (W3C) define a los servicios web como un sistema *software* reconocido por un Identificador Uniforme de Recursos (URI, siglas en ingles de *Uniform Resource Identifier*), cuyas interfaces públicas y enlaces se definen y describen usando Lenguaje de Marcado Extensible (XML, siglas en inglés de *eXtensible Markup Language*). Su definición puede ser descubierta por otros sistemas. Estos pueden interactuar con el servicio Web de la forma prescrita por su definición, usando mensajes basados en XML a través de protocolos estándares de Internet como SOAP (siglas del inglés *Simple Object Access Protocol*). (12)

Características de los servicios web

El esquema de funcionamiento de los servicios web, requiere de tres elementos fundamentales:

1. Un proveedor del servicio web, donde se diseña, desarrolla e implementa y se pone disponible para su uso, ya sea dentro de la misma organización o público.

2. Un consumidor del servicio, que accede al proveedor de servicios para utilizar las funcionalidades que éste brinda.

3. Un agente de servicio, que sirve como enlace entre el proveedor y el consumidor para efectos de publicación, búsqueda y localización del servicio. (13)

La arquitectura orientada a servicios (SOA, por sus siglas en inglés) consiste en un método de diseño de *software* donde las aplicaciones de negocio se descomponen en "servicios" individuales que pueden ser utilizados independientemente de las aplicaciones de las que forman parte y de las plataformas informáticas sobre las que se ejecutan. (14)

1.6. Tecnologías y herramientas

En el proceso desarrollo de la aplicación es fundamental la correcta elección de las tecnologías y herramientas que mejor se adapten, teniéndose en cuenta las tendencias actuales y las novedades de cada una de estas. Para esto se tuvo en cuenta además el uso de estándares y tecnologías que utiliza el Portal de Servicios Bioinformáticos, así como de la implicación directa del cliente. Basado en ello, se seleccionaron las que a continuación se enuncian.

1.6.1. Java 2 Enterprise Edition

Java es uno de los lenguajes de programación más utilizado para la creación de aplicaciones. Ha desarrollado tres ediciones de plataformas diferentes, cada una de ellas destinada a cubrir un conjunto diferente de necesidades de programación. La plataforma Java 2 Edición Estándar (J2SE, por sus siglas en inglés) enfocada para crear una amplia variedad de aplicaciones y *applets*, la plataforma Java 2 Edición Micro (J2ME, por sus siglas en inglés) permite la creación de aplicaciones para micro-dispositivos y la plataforma Java 2 Edición Empresarial (J2EE, por sus siglas en inglés) destinada a crear aplicaciones de servidor. (15)

Las aplicaciones desarrolladas bajo la plataforma Java pueden ser desplegadas en cualquier sistema operativo. J2EE utiliza el lenguaje de programación Java, el cual está patentado bajo la Licencia Pública General de GNU (GNU/GPL, por sus siglas en ingles), no siendo así con las librerías necesarias para ejecutar el código, conocido como Paquete de Desarrollo de Java (JDK siglas en inglés de *Java Development Kit*), sin embargo se dispone de una versión completamente libre conocida como OpenJDK. Entre las características más relevantes de esta plataforma se encuentran: el alto rendimiento, aplicaciones distribuidas y multiusuario, escalabilidad, gestión del estado, persistencia, transacciones, seguridad e interceptores.

J2EE está regulada por la Solicitud de Especificación de Java 58 (JSR siglas en inglés de *Java Specification Requests*) y engloba dentro de sí un conjunto de especificaciones de Interfaz de

Programación de Aplicación (APIs, por sus siglas en inglés), tales como Conectividad de Base de Datos de Java (JDBC siglas en inglés de *Java DataBase Connectivity*), Invocación de método remoto (RMI siglas en inglés de *Remote Method Invocation*), Servicio de mensajes de Java (JMS siglas en inglés de *Java Message Service*), servicios web, XML, además configura algunas especificaciones únicas como *Enterprise JavaBeans*, servlets, *Java Server Pages*, portlets y varias tecnologías que le permiten al desarrollador crear una aplicación empresarial portable entre plataformas y a la vez integrable con otras tecnologías.

1.6.2. Portlets

Los portlets son componentes web gestionados por un contenedor que tras recibir la petición de un usuario de portal generan y presentan contenidos dinámicos. Permite la personalización, presentación y gestión de la seguridad (16). La especificación Java Portlet (JSR 168) define los portlets como componentes web basados en Java, administrados por un contenedor de portlets, que procesa solicitudes y genera contenido dinámico.

El Portlet tiene un ciclo de vida y según la especificación Java Portlet (JSR168) está basado en tres fases:

- **Inicio (*Init*):** El usuario, al interactuar con el portal arranca el Portlet activando el servicio.
- **Gestión de peticiones (*Handle requests*):** Procesa la petición mostrando diferentes informaciones y contenido según el tipo de petición. Los datos pueden redimir en sistemas diferentes. Dentro de esta fase se encuentra la Presentación, en la que el Portlet da salida a la información en código de hipertexto para su visualización en el navegador.
- **Destrucción (*Destroy*):** Elimina el Portlet cuyo servicio deja de estar disponible. (17)

1.6.3. Contenedor de portlets. Liferay

Liferay Portal es un Sistema de Gestión de Contenidos basado en Java que permite la creación de portales Web de una manera sencilla y rápida. Cuenta con más de 60 portlets listos para su utilización. Liferay es una plataforma para el desarrollo, la integración y la colaboración. Ofrece todas las características necesarias para la creación de portales y aplicaciones web, todo ello desarrollado sobre una plataforma de código abierto completamente equipada. (18)

Liferay Portal basa su contenido en portlets. Incluye muchas aplicaciones portlets tales como mensajería instantánea, foros y biblioteca de documentos. Soporta múltiples base de datos como: PostgreSQL, MySQL, Oracle, SQL Server, Sybase e InterBase, se puede desplegar con muchos servidores como Jetty, JBoss, Sun GlassFish, Oracle AS y Apache Tomcat, es multiplataforma ya que puede ejecutarse en cualquier sistema operativo como Windows, Linux y MacOS (19). Por lo antes expuesto, se empleará para la gestión del Portlet del sistema la versión 6.0.5.

1.6.4. Sistema Gestores de Base de Datos. PostgreSQL

Un sistema gestor de bases de datos o DBMS (siglas del inglés *Data Base Management System*), es una colección de programas que permiten crear y mantener bases de datos para responder a las necesidades de una institución (20). Existen varios DBMS entre los que se encuentran: MySQL, MAGIC, WindowBase y PostgreSQL.

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD (siglas del inglés *Berkeley Software Distribution*) y con su código fuente disponible libremente. Es el sistema de gestión de bases de datos de código abierto más potente y robusto del mercado (21), funciona muy bien con grandes cantidades de datos y con una alta concurrencia de usuarios accediendo al sistema. PostgreSQL utiliza un modelo cliente/servidor y usa multiprocesos para garantizar la estabilidad del sistema. Está implementado para que se ejecute en varios sistemas operativos y permite utilizar numerosos tipos de datos así como definir los propios. Se caracteriza por tener una buena escalabilidad, ya que es capaz de ajustarse al número de ordenadores y a la cantidad de memoria disponible, además cuenta con un robusto sistema de seguridad mediante la gestión de usuarios, grupos de usuarios y contraseñas (22).

1.6.5. Herramientas CASE. Visual Paradigm

Las herramientas de Ingeniería del Software Asistida por Computadora (CASE, por sus siglas en inglés) proporcionan la posibilidad de automatizar actividades manuales y mejora la visión general de la ingeniería ayudando a garantizar la calidad del producto antes de llegar a construirlo (23). Tienen como objetivo incrementar la productividad y la calidad del ciclo de vida de un proyecto, mejorar la planificación del mismo, así como reducir el tiempo y coste de su desarrollo. Existen numerosas herramientas CASE, algunas de ellas son: Umbrello, Rational Rose y Visual Paradigm.

Visual Paradigm es una herramienta profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Permite representación de todo tipo de diagramas, generando código a partir de ellos. (24)

Su objetivo fundamental es reducir el tiempo de desarrollo de analistas, arquitectos, diseñadores y desarrolladores. Es una herramienta multiplataforma que proporciona soporte a varios lenguajes en generación de código e ingeniería inversa a través de plataformas Java. (25)

1.6.6. Entorno de Desarrollo Integrado. Eclipse Helios 3.6

Un entorno de desarrollo integrado es una herramienta de *software* dedicada exclusivamente al desarrollo de programas informáticos, brindando una serie de complementos que facilitan el desarrollo ágil de software. Para el desarrollo de aplicaciones en el lenguaje de programación Java se utilizan fundamentalmente dos IDEs: Eclipse y NetBeans, sin descartar a IntelliJ, pero este por ser comercial y de elevado costo no posee un buen respaldo en la comunidad. (26)

Eclipse Helios es una plataforma de programación potente y completa para el desarrollo y compilación de elementos tan variados como sitios web, programas en C++ o aplicaciones Java. No es más que un entorno de desarrollo integrado (IDE) en el que se encuentran todas las herramientas y funciones necesarias para el trabajo, recogidas además en una atractiva interfaz que lo hace fácil y agradable de usar. Emplea módulos (en inglés plug-in) como Tomcat, Axis2, Spring, JQuery para proporcionar toda su funcionalidad, a diferencia de otros entornos monolíticos donde las funcionalidades están todas incluidas, las necesite el usuario o no. La arquitectura basada en componentes permite escribir cualquier extensión deseada en el ambiente. (27)

1.6.7. Apache Tomcat

Un servidor web no es más que un programa que ejecuta de forma continua en un ordenador manteniéndose a la espera de peticiones por parte de un cliente. *Apache Software Foundation* es la organización que más prestigio tiene debido a que sus proyectos contienen gran estabilidad y confiabilidad. Dentro de los servidores web que tiene esta organización se encuentra el Apache Tomcat.

Un servidor web no es más que un programa que ejecuta de forma continua en un ordenador

Apache Tomcat es un servidor web, de código abierto que une la tecnología Java Servlet y *Java Server Pages* (28). Es mantenido y desarrollado por miembros de la *Apache Software Foundation* y voluntarios independientes. Tomcat fue escrito en Java, por lo que funciona en cualquier sistema

operativo que disponga de la máquina virtual. Es un producto muy robusto, altamente eficiente y uno de los más potentes contenedores de servlets existentes. Su único punto débil reside en la complejidad de su configuración, dado el gran número de opciones existentes (29). Es desarrollado y publicado bajo la licencia Apache 2.0. Funciona como un contenedor de servlets y servidor HTTP, implementa las especificaciones de los servlets y de *Java Server Pages* (JSP) de Sun Microsystems. Permite montar servicios web mediante Axis2. Se define utilizará la versión 6.0.26 para el despliegue de los servicios web.

1.6.8. Apache Axis 2

La gran ventaja de los servicios web es que independizan a las aplicaciones de forma tecnológica, ya que las funciones que realizan pueden ser llamadas desde cualquier lenguaje como PHP, Python o Perl. Existen varias bibliotecas para la implementación de servicios web como Metro, Apache CXF y Apache Axis 2.

Apache Axis 2 es un motor de servicios web basado en una arquitectura extensible y flexible. Apache Axis 2 se puede utilizar para proporcionar y consumir servicios web. Apache Axis2 es ampliamente utilizado como uno de los motores principales en el desarrollo de servicios web. Se trata de un rediseño completo y re-escritura construido sobre las lecciones aprendidas de Apache Axis. Apache Axis2 es más eficiente y más modular y más XML orientado a que la versión anterior.

1.6.9. Spring

Uno de los marcos de trabajo más populares en la comunidad de Java es Spring Framework, este incluye varias librerías que ayudan a solventar problemas comunes en el desarrollo de aplicaciones. Spring soporta la Programación Orientada a Aspecto e implementa el patrón Modelo-Vista-Controlador, separando la interfaz de usuario, los controladores y la capa de acceso a datos. Emplea el Objeto de Acceso a Datos (DAO, siglas en inglés de *Data Access Object*), que es un patrón de diseño para gestionar las bases de datos (30).

Spring es un marco de código abierto, creado por Rod Johnson y descrito en su libro *Expert One-on-One: J2EE Design and Development*. Fue creado para hacer frente a la complejidad del desarrollo de aplicaciones empresariales. Sin embargo, la utilidad de Spring no se limita al desarrollo del lado del servidor. Cualquier aplicación Java puede beneficiarse de Spring en términos de simplicidad, capacidad de prueba y el acoplamiento débil. (31)

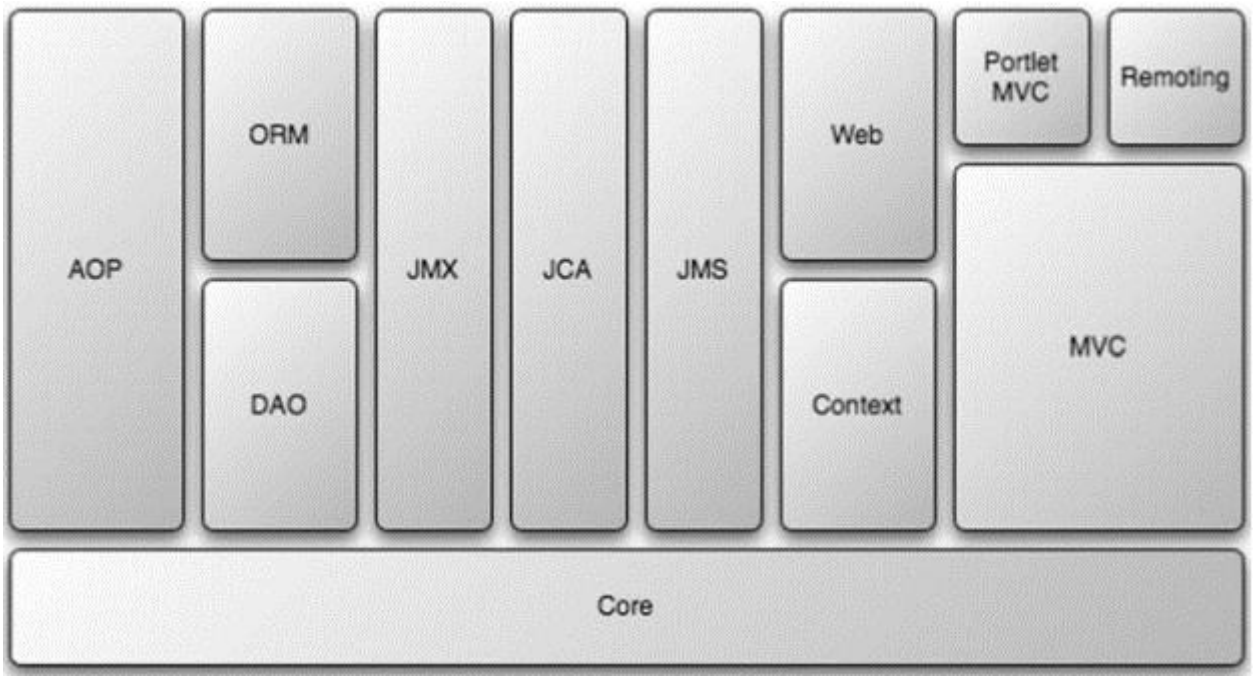

Figura. 1 Módulos que componen el Framework Spring

1.7. Metologías de Desarrollo. OpenUP

Proceso Unificado Abierto (OpenUP, por sus siglas en inglés) es un proceso ágil y unificado que contiene el conjunto mínimo de prácticas que ayudan a los equipos a ser más eficaces en el desarrollo de software. Abarca una filosofía pragmática y ágil que se centra en la naturaleza colaborativa del desarrollo (32). Está basado en casos de uso y escenarios, gestión del riesgo y un enfoque centrado en la arquitectura para impulsar el desarrollo. OpenUP se organiza en dos dimensiones distintas, correlacionadas: contenido de método y contenido del proceso. OpenUP estructura el ciclo de vida de un proyecto en cuatro fases: inicio, elaboración, construcción y transición. (33)

Beneficios en el uso del OpenUp

- Es apropiado para proyectos pequeños y de bajos recursos. Esto permite disminuir las probabilidades de fracaso en los proyectos pequeños e incrementar las probabilidades de éxito.
- Permite detectar errores en etapas tempranas de un ciclo iterativo.

- Evita la elaboración de documentación, diagramas e iteraciones innecesarios requeridos en la metodología RUP[1].
- Tiene un enfoque centrado al cliente y con iteraciones cortas. (34)

Conclusiones

En el presente capítulo se describieron los principales conceptos relacionados con las tecnologías para la integración de los servicios web de Alas-SiRNA_Design a la Plataforma de Servicios Bioinformáticos, para ello se analizaron las características más representativas de los portales bioinformáticos más utilizados en el mundo.

Se determinó el uso de servicios web para la implementaran de las funcionalidades del sistema Alas-SiRNA_Design bajo las directrices de una Arquitectura Orientada a Servicios la que se desarrollará bajo el lenguaje de programación Java, para así aprovechar las características de la plataforma J2EE a través de componentes como JSP y portlets. Para el desarrollo de los algoritmos se definió como entorno de desarrollo el Eclipse Helios. Por otra parte se fundamentó la selección de Liferay Portal como portal de aplicaciones y contenedor de Portlet, Apache Axis 2 para la implementación de los servicios web y como marco de trabajo se optó por Spring Framework, garantizando de esta forma una arquitectura robusta basada en patrones como el Modelo Vista Controlador. Además se estableció PostgreSQL como gestor de base de datos, el cual se encargará de almacenar la información persistente. Finalmente el proceso de desarrollo estará guiado por la metodología de desarrollo OpenUP, la cual utiliza UML como lenguaje de modelado y se eligió como herramienta CASE al Visual Paradigm.

[1] Proceso Unificado de Rational

CAPÍTULO 2. CARACTERÍSTICAS DEL SISTEMA

En este capítulo se confeccionan los artefactos de ingeniería del *software* que permitirán desarrollar el sistema de forma más fácil. Se realiza el modelo conceptual o de dominio, se hace un estudio del negocio permitiendo identificar los requisitos funcionales y no funcionales. Por último se realiza una descripción de los actores que interactúan con el sistema, así como de los casos de usos identificados para alcanzar una mayor comprensión de los mismos.

2.1. Modelo de dominio

El modelo de dominio es una representación gráfica del problema mediante el enlace de clases, que no identifican componentes del software, sino objetos del mundo real. El análisis orientado a objetos tiene por finalidad estipular una especificación del dominio del problema y los requerimientos desde la perspectiva de la clasificación por objetos y desde el punto de vista de entender los términos empleados en el dominio. Para descomponer el dominio del problema hay que identificar los conceptos, los atributos y las asociaciones del dominio que se juzgan importantes. (35)

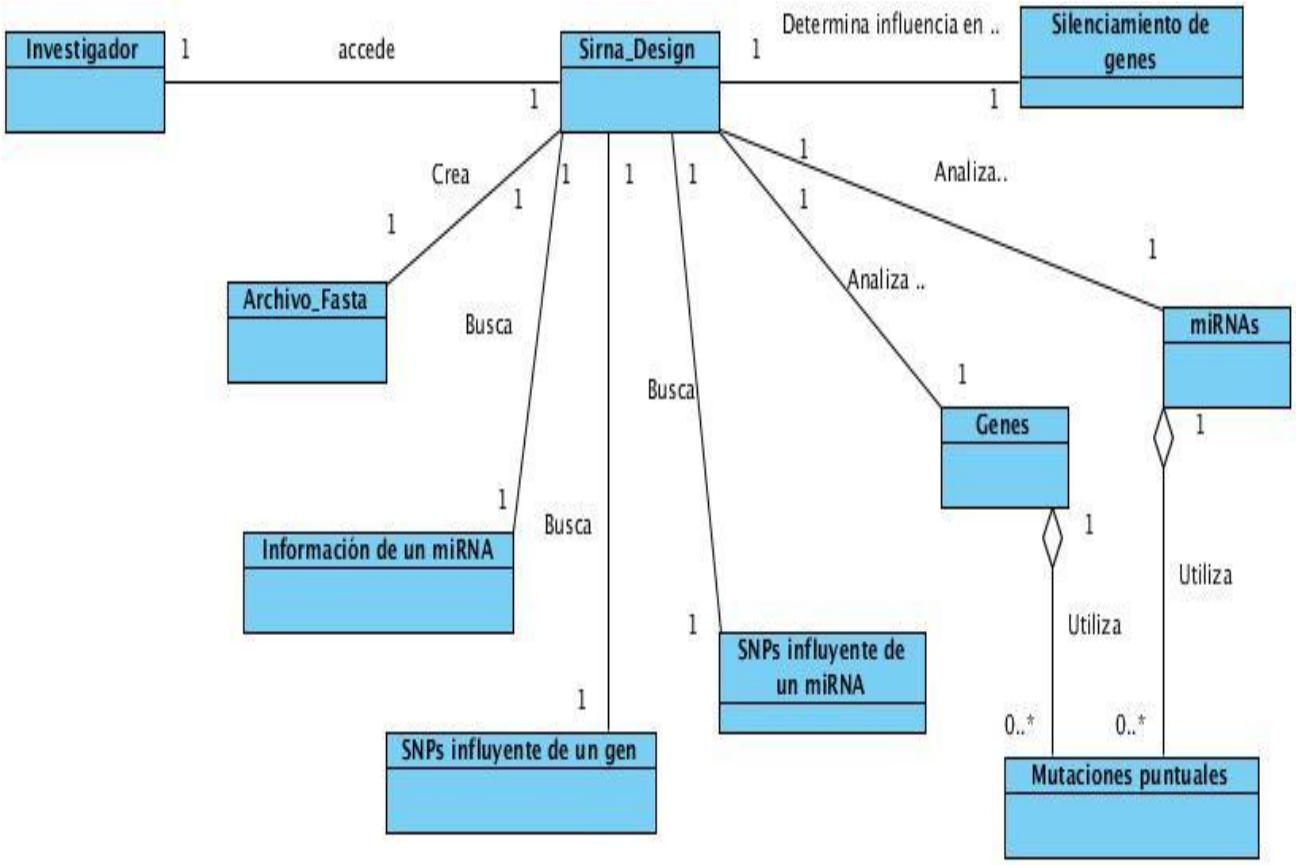

Figura. 2 Diagrama de Clases del Modelo de Dominio de la aplicación. Representa los conceptos del mundo real y nunca los relacionados con objetos de software.

Tabla 1 Descripción de las clases del Modelo de Dominio

Investigador	Persona habilitada para trabajar con el sistema.
Sirna_Design	Clase que representa un sitio web.
Genes	Un gen es una secuencia lineal y organizada de nucleótidos en la molécula de DNA (o RNA en el caso de algunos virus).
miRNAs	Los miRNAs son moléculas de RNA transcritas a partir de genes de DNA, pero no son traducidos a proteínas, su función es regular la acción de otros genes.
Mutaciones puntuales	Cubren una variedad de mutaciones desde cambios genuinos en un par de bases hasta pequeñas delecciones[2] e inserciones.
Silenciamiento de genes	El silenciamiento génico[3] es un proceso llevado a cabo en los organismos eucariotas, con diferentes objetivos, dentro de los

[2] Pérdida de un segmento cromosómico que lleva consigo la desaparición de la información genética contenida en él
[3] Relativo al gen o a los genes

	cuales se destacan la regulación de la expresión y la eliminación y el control de material genético que podría causar un daño a la célula.
Archivo Fasta	Genera un archivo con formato Fasta.
Información de un miRNA	Busca información de todos los miRNAs maduros contenidos en un determinado miRNA.
SNPs influyente en un gen	Busca todos los SNPs influyentes en un gen.
SNPs influyente en un miRNA	Busca todos los SNPs influyentes en un miRNA.

2.2. Requisitos funcionales

Los requisitos o requerimientos funcionales de una aplicación son las capacidades o condiciones que el sistema debe cumplir para satisfacer las necesidades primarias del cliente.

A continuación se definen los requisitos funcionales seleccionados para dar cumplimiento al desarrollo del sistema.

RF 1: Buscar información de los genes en el genoma humano.

RF 2: Guardar archivo Fasta con información de los genes en el genoma humano.

RF 3: Mostrar SNPs presentes en un miRNA.

RF 4: Mostrar genes blancos de un miRNA.

RF 5: Visualizar SNPs influyentes de un gen.

RF 6: Visualizar SNPs influyentes en el silenciamiento de un miRNA.

2.3. Requisitos no funcionales

Los requisitos no funcionales son las propiedades o cualidades que el producto debe presentar. Estos requisitos definirán las características del producto final y muchas veces están implicados en el éxito final que se desea alcanzar.

-De hardware.

Para el servidor de base de datos:

- Procesador Pentium 4 o superior.
- 1 GB de memoria RAM o superior.
- Disco duro con 80 GB de espacio libre o superior

El servidor de servicios web y de aplicaciones:

- Procesador Pentium 4 o superior.
- Memoria RAM 4GB o superior
- Disco duro con 80 GB de espacio libre o superior

Los ordenadores clientes deben constar con:

- 512 MB de RAM o superior.
- Procesador Pentium 3 o superior

-De software.

Para el servidor de aplicaciones:

- Liferay portal 6.5.0.

Para el servidor de servicios web debe constar con:

- Apache Tomcat 6.0.14 o superior.

Para el servidor de base de datos:

- PostgreSQL 8.4 o superior.

Para los ordenadores del cliente debe constar con:

- Un navegador web como Firefox 15 o superior y Google Chrome 18 o superior.

-De usabilidad.

- Debe dar la posibilidad del uso de la herramienta por especialistas, con ciertos conocimientos bioinformáticos.

-De seguridad.

- Confidencialidad
    - La herramienta estará disponible sólo para los especialistas registrados en la Plataforma de Servicios Bioinformáticos.

-Requerimientos legales.

- Los requisitos legales y el derecho de autor serán registrados por la Universidad de la Ciencias Informáticas.

2.4. Definición de los Casos de Usos del Sistema

Un diagrama de casos de uso (CU) de sistema es una representación visual de un conjunto de casos de uso, los actores y la relación entre éstos y los casos de uso conformando así el sistema. (36) Estos diagramas sirven para facilitar la comunicación con los futuros usuarios del sistema, y con el cliente, y resultan especialmente útiles para determinar las características necesarias que tendrá el sistema.

2.4.1. Actor del sistema

Los actores pueden ser sistemas, dispositivos externos o personas que interactúan con el sistema, ya sea para inicializar una funcionalidad o brindarle información al mismo. Son los que se benefician de las funcionalidades dentro de los casos de uso. El actor es una entidad externa del sistema que de alguna manera participa en la historia del caso de uso. Por lo regular estimula el sistema con eventos de entrada o recibe algo de él. Conviene escribir su nombre con mayúscula en la narrativa del caso para facilitar la identificación. Acorde a la situación en la que se desarrollará la herramienta se identificó el siguiente actor:

Tabla 2 Definición del actor del sistema.

Actor	Descripción
Especialista	Cualquier persona capacitada para interactuar con el *software* a través del Portal de Servicios Bioinformáticos.

2.4.2. Casos de Usos del Sistema

1. **CU** Mostrar información de un miRNA. Hace referencia a los RF3 y RF4
2. **CU** Guardar archivo Fasta con información de genoma humano. Hace referencia a los RF1 y RF2
3. **CU** Visualizar SNPs influyentes en silenciamiento de un miRNA. Hace referencia al RF6
4. **CU** Visualizar SNPs influyentes de un gen. Hace referencia al RF5

2.4.3. Diagrama de Casos de Usos del Sistema

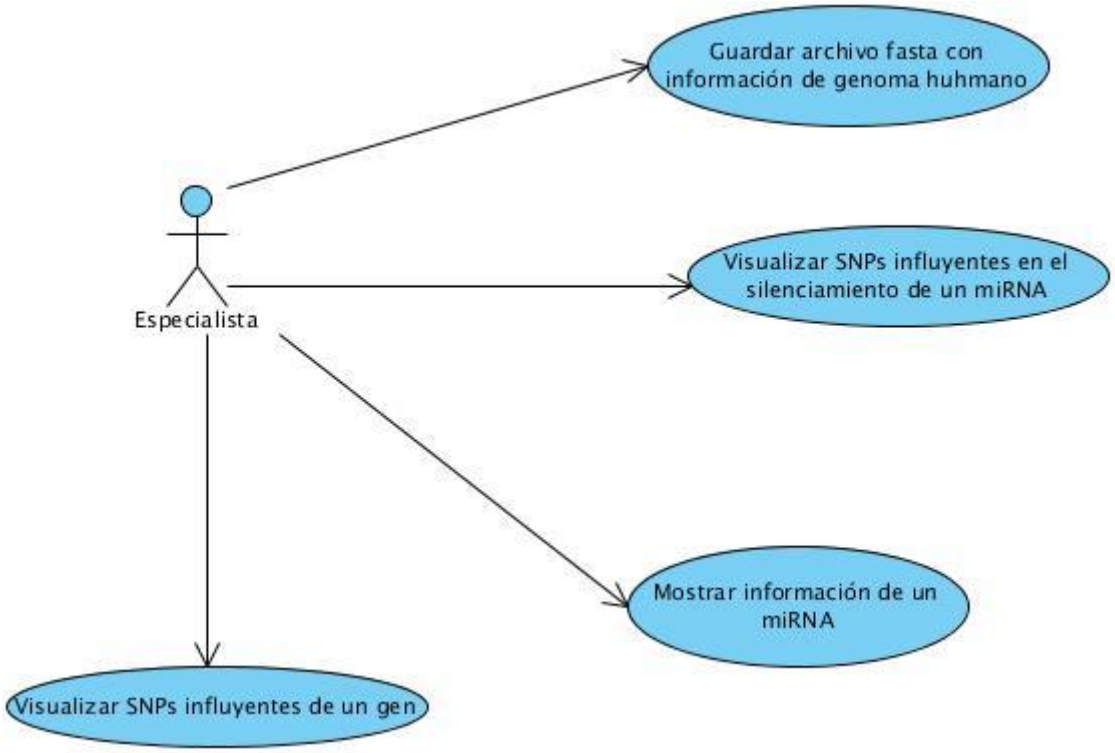

Figura. 3 Diagrama de Casos de Usos de Sistema. Los CU pueden contener en ocasiones varios requisitos funcionales. Los CU serán iniciados en todos los casos por el investigador.

2.5. Descripción de los Casos de Usos del sistema

Tabla 3 Descripción del Caso de Uso Visualizar SNPs influyentes de un gen.

Caso de Uso	Visualizar SNPs influyentes de un gen.
Actores	Especialista
Propósito	Muestra aquellos SNPs que tienen influencia en un determinado gen.
Resumen	El caso de uso se inicia cuando el especialista desea que la aplicación muestre los SNPs influyentes en el transcrito y la posición de estos en la secuencia del transcrito.
Precondiciones	
Referencias	RF 5
Prioridad	Alta.
Flujo normal de eventos	
Acción del actor	**Respuesta del sistema**
1. El especialista escoge la opción Visualizar SNPs influyentes de un gen.	2. El sistema muestra un formulario para que inserte el nombre del transcrito.
3. El especialista introduce el nombre del transcrito.	4. El sistema busca los SNPs influyentes así como toda la información referente a este. 5. El sistema muestra los SNPs influyentes así como toda la información referente a este.
Flujos alternos	
	2.1 El sistema muestra un mensaje de error si el nombre del transcrito introducido por el especialista es incorrecto.
Poscondiciones	

El resto de las descripciones de los casos de usos se encuentran en los anexos del documento.

Conclusiones

En este capítulo se definieron los artefactos de la fase de requisitos correspondiente a la metodología de desarrollo OpenUP. Aprovechando la extensibilidad de la metodología se definió el modelo conceptual del dominio y se describieron los objetos del dominio del problema tales como Archivo Fasta, Silenciamiento de genes y miRNAs. Se identificaron seis requisitos funcionales y 16 requisitos no funcionales. Los RF identificados se agruparon en el modelo de casos de uso del sistema, destacando el CU:"Visualizar SNPs influyentes de un gen" por tener gran impacto en la arquitectura, el mismo fue descrito para lograr una mayor comprensión de su funcionamiento.

CAPÍTULO 3. DISEÑO DE LA APLICACIÓN

En este capítulo se traducen los requisitos a una especificación que describe cómo implementar el sistema, dando paso al diseño de la aplicación y generando los artefactos necesarios para cada fase. Además se realizan los diagramas de clases y de interacción para el caso de uso arquitectónicamente significativo definido en el capítulo anterior. Se especificará la arquitectura que tendrá la aplicación y se mostrará el modelo datos, la vista de despliegue. Por último se definirán los principales patrones arquitectónicos y de diseño a utilizar.

3.1. Arquitectura de Software

La arquitectura de software consiste en la estructura o sistema de estructuras, que comprenden los elementos de *software*, las propiedades externas visibles de esos elementos y la relación entre ellos. (23) Define una solución para los requisitos técnicos y operacionales del mismo. Este proceso precisa qué componentes forman el *software*, cómo se relacionan entre ellos, y cómo mediante su interacción llevan a cabo la funcionalidad especificada, cumpliendo con los criterios previamente establecidos; como seguridad, disponibilidad, eficiencia o usabilidad. (37)

3.1.1. Estilos y patrones arquitectónicos

En la actualidad no existe un convenio para diferenciar patrones y estilos arquitectónicos. (38) Un estilo arquitectónico expresa componentes y las relaciones entre estos, con las restricciones de su aplicación y la composición asociada, así como también las reglas para su construcción. Un patrón de arquitectura de software describe un problema particular y recurrente del diseño, que surge en un contexto específico, y presenta un esquema genérico y probado de su solución.

Definen la estructura de un sistema *software*, los cuales a su vez se componen de subsistemas con sus responsabilidades. Se define como una regla que consta de tres partes, la cual expresa una relación entre un contexto, un problema y una solución.

- **Contexto**: Es una situación de diseño en la que aparece un problema de diseño
- **Problema**: Es un conjunto de fuerzas que aparecen repetidamente en el contexto
- **Solución**: Es una configuración que equilibra estas fuerzas. Ésta abarca:

- Estructura con componentes y relaciones
- Comportamiento a tiempo de ejecución: aspectos dinámicos de la solución, como la colaboración entre componentes y la comunicación entre ellos. (39)

3.1.2. Patrón Arquitectónico Modelo Vista Controlador (MVC).

Es un patrón de arquitectura de *software* que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes diferentes. El patrón MVC (Figura. 4) se ve frecuentemente en aplicaciones web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página, el modelo es el Sistema de Gestión de Base de Datos y la Lógica de negocio y el controlador es el responsable de recibir los eventos de entrada desde la vista. La finalidad del modelo es mejorar la reusabilidad por medio del bajo acoplamiento entre las vistas y el modelo de datos. (40)

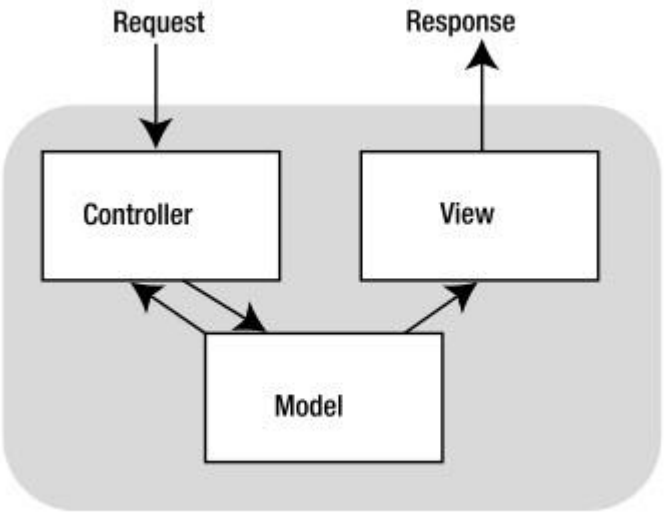

Figura. 4 Patrón Arquitectónico Modelo-Vista-Controlador. La utilización de este patrón evita en gran medida el acoplamiento de los componentes del sistema. Cada capa obedece a la solicitud de otra manteniendo una cohesión entre las mismas.

Modelo: Accede a la capa de almacenamiento de datos. Lo ideal es que el modelo sea independiente del sistema de almacenamiento. Define las reglas de negocio (la funcionalidad del sistema).

Vista: Recibe datos del modelo y los muestra al usuario. Tienen un registro de su controlador asociado.

Controlador: Recibe los eventos de entrada o los datos que infieren la ejecución de una tarea en particular, procesan la información y devuelven la vista correspondiente.

3.2. Patrones de diseño

Un patrón es un par problema/solución con nombre que se puede aplicar en nuevos contextos con consejos acerca de cómo aplicarlo en nuevas situaciones y discusiones sobre sus compromisos. (41)Un patrón de diseño es una solución repetible a un problema recurrente en el diseño de software. Esta solución no es un diseño terminado que puede traducirse directamente a código, sino más bien una descripción sobre cómo resolver el problema. (42)

En resumen los patrones de diseños se pueden definir de la siguiente manera:

- Describen un problema recurrente y una solución.
- Cada patrón nombra, explica, evalúa un diseño recurrente en sistemas orientados a objetos.

En la realización del diseño para la aplicación informática a implementar, se utilizaron los patrones GRASP (acrónimo del inglés *General Responsibility Assignment Software Patterns*). Se considera que más que patrones son una serie de "Buenas Prácticas" de aplicación recomendable en el diseño de software.

3.2.1. Patrón Bajo Acoplamiento

Este patrón asigna la responsabilidad de controlar el flujo de eventos del sistema a clases específicas. Esto facilita la centralización de actividades. La clase controladora no realiza las actividades, las delega en otras clases con las que mantiene un modelo de alta cohesión.

En el diseño del software, se pretende el acoplamiento más bajo posible. Una conectividad sencilla entre los módulos da como resultado una aplicación más fácil de entender y menos propenso a tener un “efecto ola”, causado cuando ocurren errores en un lugar y se propagan por el sistema. (43) Este patrón se refleja en la Figura 5.

Figura. 5 Representación del Patrón Bajo Acoplamiento en el Diseño del Sistema. Explica cómo dar soporte a una dependencia escasa y a un aumento de la reutilización.

3.2.2. Patrón Alta Cohesión

El patrón alta cohesión describe cuan fuertemente los contenidos internos de una rutina están relacionados entre sí. (44) Este patrón propone asignar la responsabilidad de manera que la complejidad se mantenga dentro de límites manejables asumiendo solamente las responsabilidades que deben manejar, evadiendo un trabajo excesivo. En la Figura 8 se evidencia comó la responsabilidad y el manejo del flujo están controlada por la clase ControladorVisualizarInfSNPGen.

3.2.3. Patrón Creador

El Patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. El propósito fundamental de este patrón es encontrar un creador que conecte con el objeto producido en cualquier evento. (45)

La nueva instancia del objeto deberá ser creada por la clase que: tiene la información necesaria para realizar la creación del objeto, usa directamente las instancias creadas del objeto, o almacena o maneja varias instancias de la clase. Este patrón brinda soporte de bajo acoplamiento, lo cual supone menos dependencias entre clases y posibilidades. Se ejemplifica en la Figura 6.

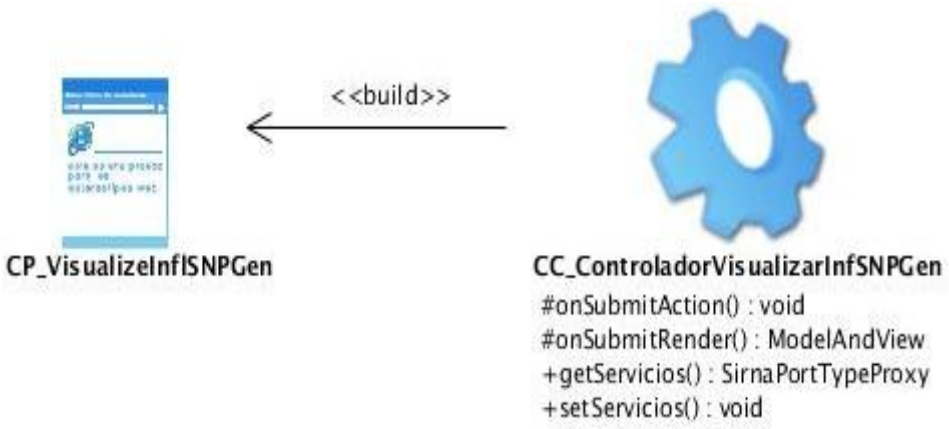

Figura. 6 Representación del Patrón Creador en el Diseño del Sistema. Explica que clase es la encargada de crear objetos, en determinados escenarios de ejecución.

3.2.4. Patrón Controlador

El Patrón Controlador es un objeto de interfaz no destinada al usuario que se encarga de manejar un evento del sistema a una clase que represente un sistema global. Define además el método de su operación. (42) Está representado en la Figura 7.

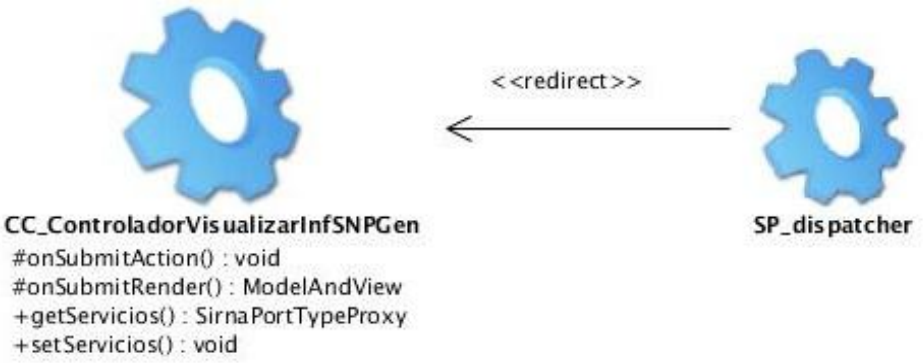

Figura. 7 Representación del Patrón Controlador en el Diseño del Sistema. Asigna la responsabilidad del manejo de los eventos de un sistema a una clase que represente un sistema global.

3.3. Modelo de diseño

El modelo de diseño detalla los modelos de análisis, tomando en cuenta todas las implicaciones y restricciones técnicas. El propósito es especificar una solución que trabaje y pueda ser fácilmente convertida en código fuente y construir una arquitectura simple y extensible. Las clases definidas en el

análisis se detallan y se añaden nuevas clases para manejar áreas técnicas como bases de datos, interfaces de usuario, dispositivos, entre otros. (46)

3.3.1. Diagrama de clases del diseño

Los diagramas de clases son utilizados durante el proceso de análisis y diseño de los sistemas, donde se crea el diseño conceptual de la información que se manejará en el sistema, y los componentes que se encargaran del funcionamiento y la relación entre uno y otro.

En el diagrama de clases del diseño para el CU “Visualizar SNPs influyentes de un gen” (Figura 8) se evidencia el empleo del patrón Modelo Vista Controlador divididos en cuatro paquetes, que simbolizan: la Presentación al usuario, la Lógica con la cual el sistema trabaja, los servicios web con el cual se encuentran las funcionalidades y los Servicios de Acceso a Datos. El resto de los diagramas de clases del diseño de la aplicación se encuentran en los anexos del documento.

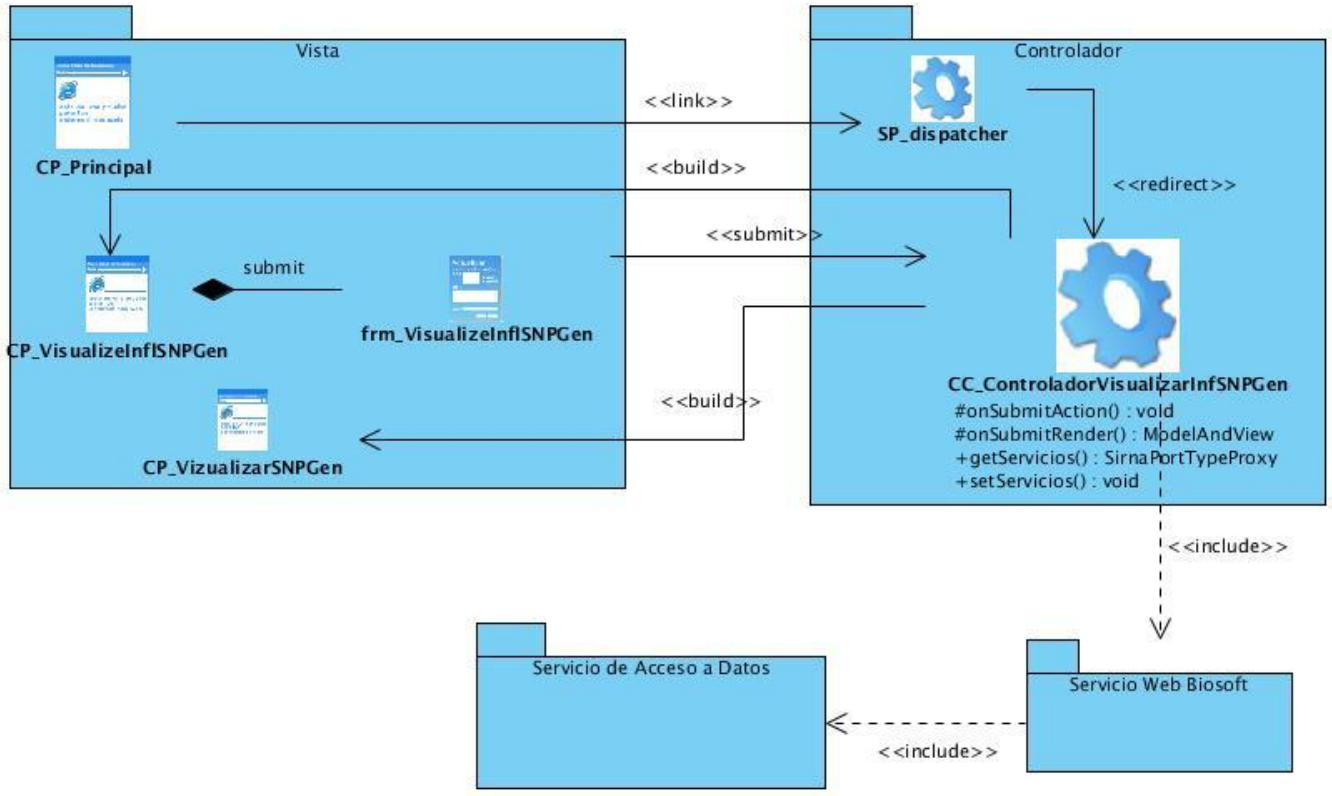

Figura. 8 Diagrama de Clases del Diseño del CU Visualizar SNPs influyentes de un gen. Representación del patrón arquitectónico Modelo Vista Controlador

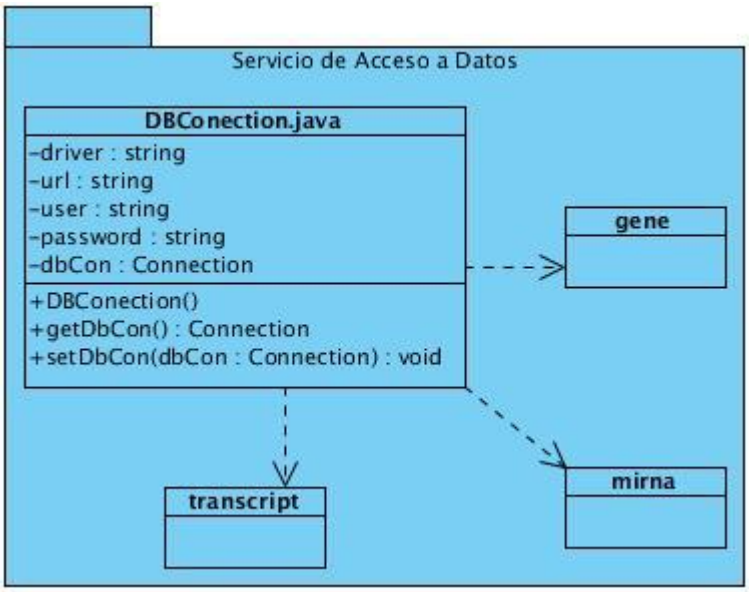

Figura. 9 Paquete destinado al Servicio de Acceso a Datos

Descripción de las clases del diseño del CU Visualizar SNPs influyentes de un gen.

Tabla 4 Descripción de la clases de diseño CU Visualizar SNPs influyentes de un gen.

CP_Principal	Esta clase representa la vista principal a la que tendrá acceso el usuario.
CP_VisualizeInflSNPGen	Clase dedicada a recoger los datos que ingresó el usuario.
CP_VisualizarSNPGen	Diseñada para mostrar el resultado al usuario.
SP_dispatcher	Clase controladora principal, maneja todos los eventos y acciones de la aplicación.
CC_ControladorVisualizarInfSNPGen	Concebida para interactuar con los datos ingresados en la página cliente “VisualizeInflSNPGen”.
SirnaPortType.java	Interfaz destinada para declarar las funciones que invocan los servicios web.
SirnaPortTypeProxy.java	Interfaz creada para definir las funciones, y funciona en dependencia de los valores que necesite el servicio web.
Sirna.java	Contiene los atributos necesarios para la ejecución del servicio web.

3.3.2. Diagrama de secuencia

Un diagrama de secuencia muestra la interacción de un conjunto de objetos en una aplicación a través del tiempo y se modela para cada caso de uso.

Se indican los módulos o clases que forman parte del programa y las llamadas que se hacen en cada uno de ellos para realizar una tarea determinada. El detalle que se muestre en el diagrama de secuencia debe estar en correspondencia con lo que se intenta mostrar o bien con la fase de desarrollo en la que esté el proyecto. En el diagrama de secuencia no se ponen situaciones erróneas, puesto que poner todos los detalles puede dar lugar a un diagrama que no se entiende o sea difícil de leer. (43)

El diagrama de secuencia de la Figura 10 muestra la comunicación entre diferentes componentes del *software*. Se determinó una clase controladora principal “SP_dispatcher”, la cual gestionará todas las peticiones del usuario. El resto de los diagramas de secuencia de la aplicación aparecen en los anexos del documento.

3.3.3. Diagrama de despliegue

Los diagramas de despliegue muestran a los nodos procesadores, la distribución de los procesos y de los componentes. (41) Los diagramas de despliegue son los complementos de los diagramas de componentes que, unidos, proveen la vista de implementación del sistema. Describen la topología del sistema, la estructura de los elementos de *hardware* y el *software* que ejecuta cada uno de ellos. Los diagramas de despliegue representan a los nodos y sus relaciones. Los nodos son adheridos por asociaciones de comunicación tales como enlaces de red, conexiones TCP/IP, JDBC o RMI.El diagrama de despliegue está representado en la Figura 11.

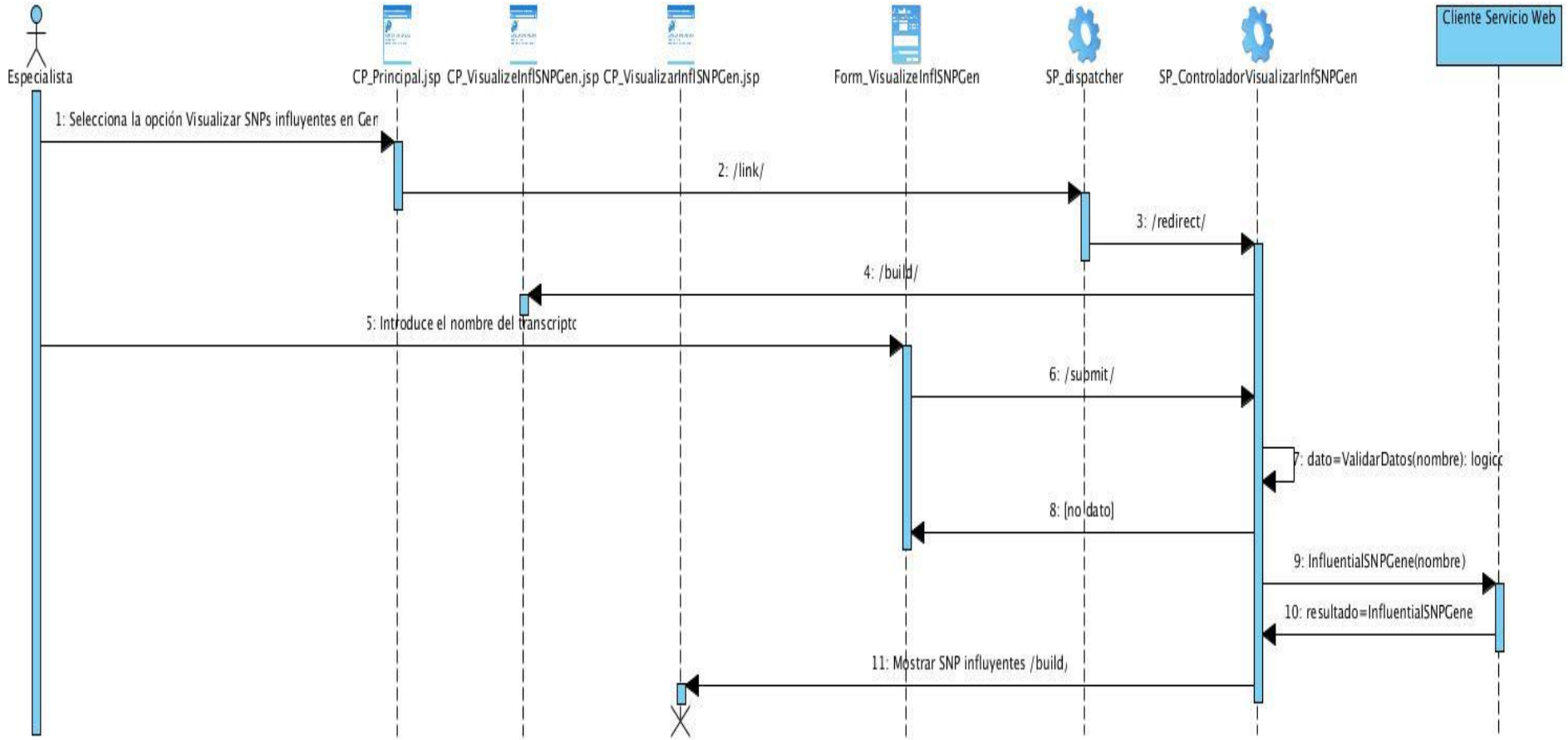

Figura. 10 Diagrama de Secuencia del CU Visualizar SNPs influyentes de un gen.

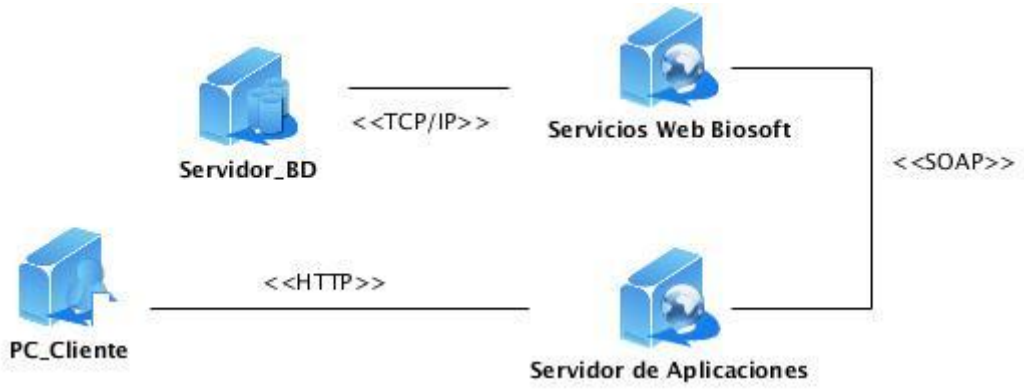

Figura. 11 Diagrama de despliegue de la aplicación. Se representan los componentes necesarios para realizar el despliegue del sistema y los protocolos de comunicación entre ellos.

PC_Cliente: representa una computadora (PC) que mediante un navegador web realiza las peticiones a través del protocolo HTTP al servidor de aplicaciones.

Servidor_BD: representa un servidor de base de datos qué permite consultar la información de los genes de mirna a través del servidor de servicios web utilizando el protocolo TCP/IP.

Servidor de Aplicaciones: representa un servidor de aplicaciones destinado a responder las peticiones hecha por la PC_Cliente a través de los servicios que brinda el servidor web utilizando el protocolo SOAP.

Servidor Web Biosoft: representa un servidor que contiene los servicios web necesarios para dar respuesta a las peticiones realizadas por el servidor de aplicaciones utilizando para ello la información que brinda el servidor de base de datos.

Conclusiones

En el presente capítulo se confeccionó el diagrama de secuencia para el caso de uso “Visualizar SNPs influyentes de un gen”, posibilitando de esta forma mejor comprensión del proceso ejecución del mismo. Para la elaboración del diagrama de secuencia se utilizaron los patrones arquitectónicos y de diseño, estos proveen mayor robustez al producto final, posibilitando alcanzar una arquitectura menos vulnerable al cambio.

Para la implementación de la herramienta se utilizó el patrón arquitectónico Modelo Vista Controlador, el cual está presente en los diagramas de clases del diseño, así como los patrones GRASP: Bajo Acoplamiento, Alta Cohesión, Creador, Experto y Controlador. Por último se realizó el diagrama de despliegue, representando los cuatro nodos que necesita el sistema para realizar su despliegue: una PC_Cliente, un servidor de aplicaciones, un servidor de servicios web y un servidor de Base de Datos.

CAPÍTULO 4. IMPLEMENTACIÓN Y PRUEBAS

En este capítulo, se confeccionará el Diagrama de Componentes, en el mismo se describen los elementos físicos del sistema y sus relaciones. Se muestran las clases e implementaciones más importantes y las pruebas realizadas para validar la solución.

4.1. Modelo de Implementación

El Modelo de Implementación describe cómo los elementos del Modelo de Diseño, se implementan en términos de Componentes, Ficheros de Código Fuente y Ejecutables. Describe también cómo se organizan los Componentes de acuerdo a los mecanismos de estructuración disponibles en el entorno de implementación y lenguajes empleados, y cómo dependen los componentes unos de otros. (47)

4.1.1. Diagrama de componentes

Un diagrama de componentes muestra las organizaciones y dependencias lógicas entre componentes de software: código fuente, binarios o ejecutables. Desde el punto de vista del diagrama de componentes se tienen en consideración los requisitos relacionados con la facilidad de desarrollo, la gestión del software, la reutilización, y las limitaciones imputadas por los lenguajes de programación y las herramientas utilizadas en el desarrollo.

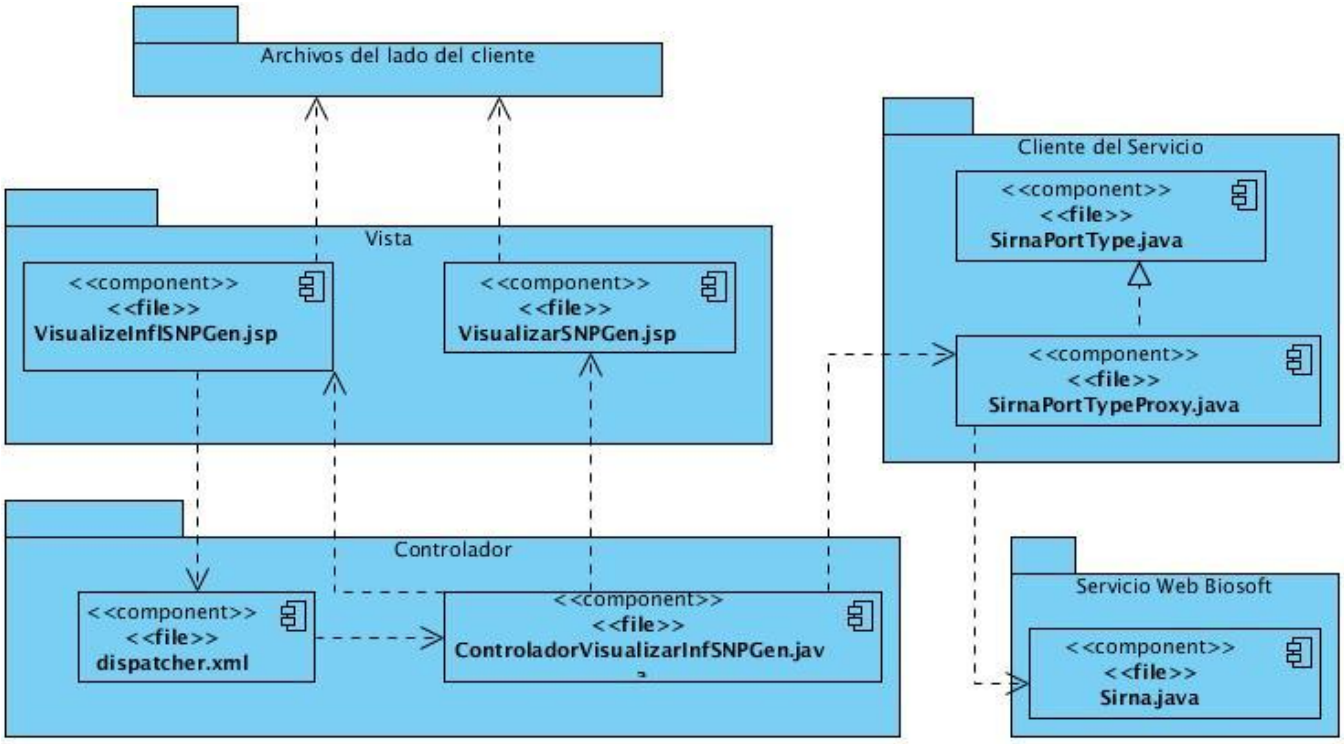

Figura. 12 Diagrama de componentes del Visualizar SNPs influyentes de un gen.

Tabla 5 Descripción de los componentes del diagrama.

VisualizeInfISNPGen.jsp	Página JSP que permite al usuario ingresar los datos para visualizar la influencia de los SNPs en un gen.
dispacher.xml	Archivo XML encargado de atender todas las peticiones del usuario y delegar estas al controlador indicado.
VisualizarSNPGen.jsp	Página JSP utilizada para visualizar el resultado de la influencia de los SNPs en un gen.
ControladorVisualizarInfSNPGen.java	Clase Java dedicada a procesar los datos enviados por la JSP.
SirnaPortType.java	Clase interfaz que declara todas las operaciones que invocan los servicios web.
SirnaPortTypeProxy.java	Clase que implementa las operaciones de la interfaz SirnaPortType.java.

El resto de los diagramas de componentes se ubican en la sección de anexos del documento.

4.2. Pruebas de software

La fase de pruebas es una de las más costosas del ciclo de vida de un sistema. En sentido estricto, deben realizarse pruebas de todos los artefactos generados durante la construcción de un producto, lo que incluye especificaciones de requisitos, casos de uso, diagramas de diversos tipos incluyendo el código fuente y el resto de productos que forman parte de la aplicación. (35)

Las pruebas son fundamentalmente usadas para descubrir los errores en el *software,* tanto el código fuente como en la interfaz de usuario, cosas tan sencillas como faltas de ortografía y errores en mensajes de alerta.

El éxito en las pruebas de software determina la satisfacción final del cliente, pruebas mal ejecutadas a una aplicación, puede generar daños económicos muy serios. Los métodos para validar las aplicaciones informáticas más utilizados son: el método de Caja Negra y el de Caja Blanca.

4.2.1. Pruebas de caja blanca

La prueba de caja blanca, denominada a veces prueba de caja de cristal es un método de diseño de casos de pruebas que usa la estructura de control del diseño procedimental para obtener los casos de prueba. (23)

Mediante los métodos de prueba de caja blanca, el ingeniero de *software* puede obtener casos de prueba que: garanticen que se ejercita por lo menos una vez todos los caminos independientes de cada módulo; ejerciten todas las decisiones lógicas en sus vertientes verdadera y falsa; ejecuten todos los bucles en sus límites y con sus límites operacionales; y ejerciten las estructuras internas de datos para asegurar su validez.

Uno de los métodos de prueba de caja blanca es el camino básico, la cual determina la complejidad ciclomática de una porción de código. La complejidad ciclomática es una métrica del *software* que proporciona una medición cuantitativa de la lógica de un programa (23).

Cuando se usa el camino básico, el valor calculado como complejidad ciclomática define el número de caminos independientes del conjunto básico de un programa y nos da un límite superior para el número de pruebas que se deben realizar. Esta complejidad se puede calcular de tres formas:

1. El número de regiones del grafo de flujo coincide con la complejidad ciclomática.

2. La complejidad ciclomática, V (G), de un grafo de flujo G, se define como: V (G)=A-N+2.

Donde A es el número de aristas del grafo de flujo y N es el número de nodos del mismo.

3. La complejidad ciclomática, V (G), de un grafo de flujo G también se define como V (G) = P +1. Donde P es el número de nodos predicado contenido en el grafo de flujo G.

4.2.2. Pruebas de Caja Negra

Las pruebas de caja negra, también denominadas pruebas de comportamiento, se centran en los requisitos funcionales del *software*. La prueba de caja negra intenta encontrar errores dentro de las siguientes categorías: funciones incorrectas o ausentes, errores de interfaz, errores en estructuras de datos o accesos a bases de datos externas, errores de rendimiento, inicialización y terminación. (44)

La partición equivalente es un método de prueba de caja negra que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. Un caso de prueba ideal descubre de forma inmediata una clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico (23). La partición equivalente se dirige a la definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar.

Las clases de equivalencia se pueden definir de acuerdo con las siguientes directrices:

1. Si una condición de entrada especifica un rango, se define una clase de equivalencia váliday dos no válidas.

2. Si una condición de entrada requiere un valor específico, se define una clase de equivalencia válida y dos no válidas.

3. Si una condición de entrada especifica un miembro de un conjunto, se define una clase de equivalencia válida y una no válida.

4. Si una condición de entrada es lógica, se define una clase de equivalencia válida y una no válida.

4.2.3. Aplicación del método de Caja Blanca

Como se mencionó en acápites anteriores uno de las técnicas de prueba de caja blanca es la del camino simple, que se aplica a fragmentos de código de la aplicación. Se determinó aplicar este método a la función que busca un transcrito por su nombre en la base de datos.

Para la función “findTranscriptName” se identificaron los bloques de ejecución y se enumeraron para identificarlos (Figura.13). Se obtuvieron 15 bloques y se determinó el camino básico ilustrado en la Figura.14. Identificando en cada sentencia condicional un nodo predicado del cual se derivan más de un camino a seguir, tal es el caso de los nodos 2, 7, 10 y 13.

Con el camino básico determinado, se aplica una de las tres formas para calcular la complejidad ciclomática, se utilizó la fórmula V (G) = A-N + 2, para la cual se obtuvo 11 artistas y nueve nodos, por lo tanto: V (G) = 18-15 + 2, quedando V (G) = 5. De la misma forma se pueden comprobar que las otras variantes explicadas de calcular la complejidad ciclomática arriban al mismo resultado. La complejidad ciclomática indica los posibles casos de ejecución para la función, lo que es muy útil a la hora de diseñar los casos de prueba de caja negra.

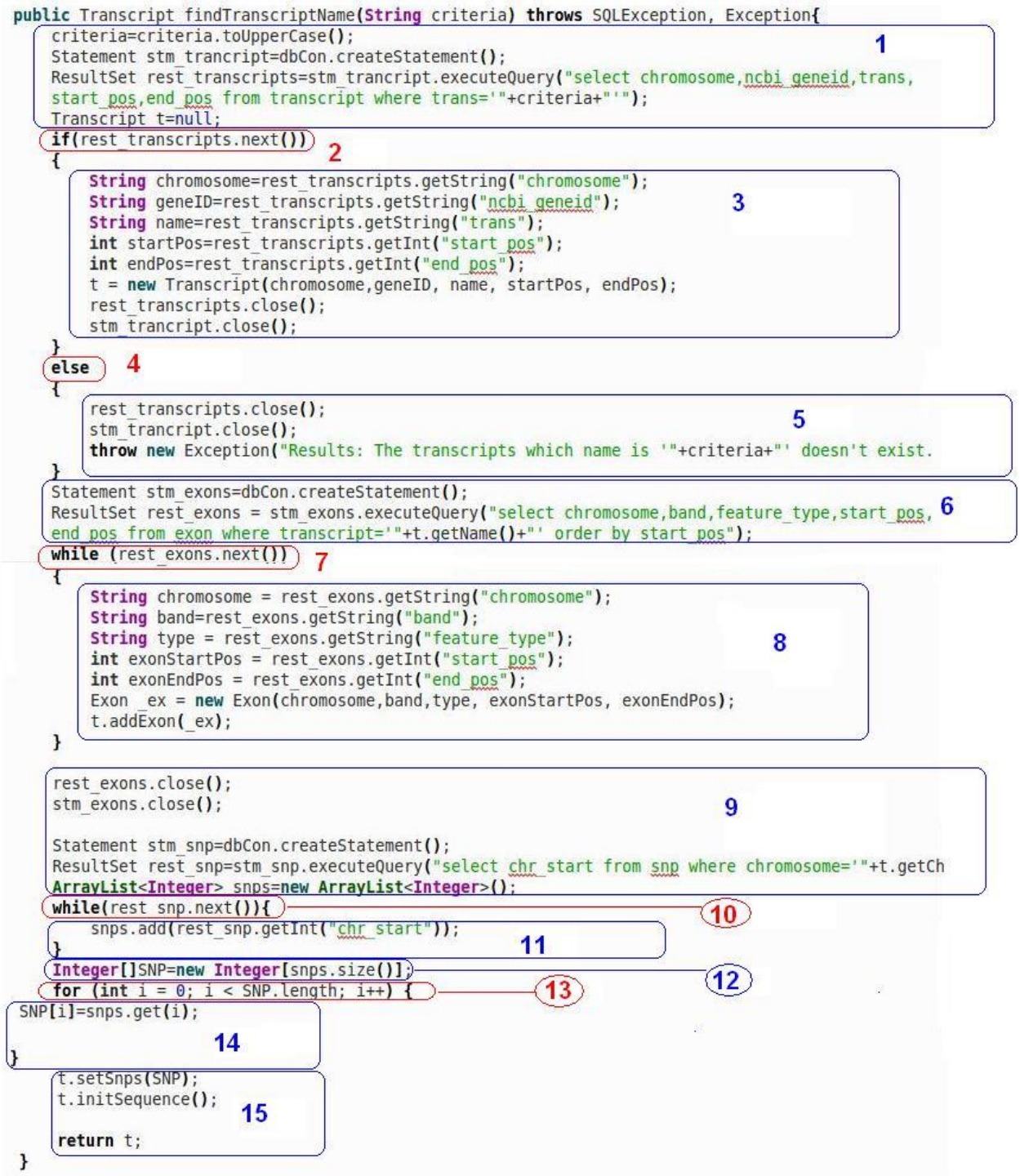

Figura. 13 Función que busca un transcrito por su nombre. El código se divide por bloques de ejecución, los cuales están enumerados y constituyen los nodos del camino básico.

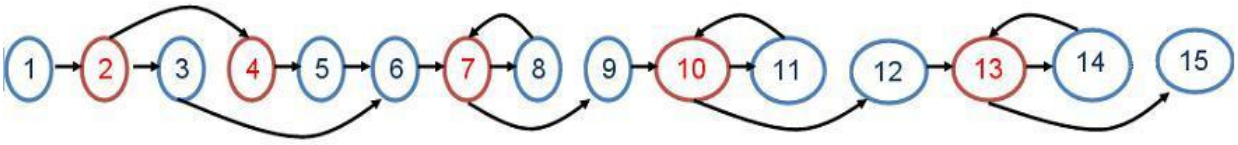

Figura. 14 Camino básico de la función "findTranscriptName".Los nodos resaltados en color verde corresponden a las condicionales del código de la función y por tanto son nodos predicados. Las aristas indican los posibles caminos a seguir a partir del nodo correspondiente.

4.2.4. Aplicación del método de Caja Negra

Para aplicar el método se obtuvieron las variables, las cuales se definen en conjunto de entrada, condiciones de ejecución y resultados esperados al finalizar.

Tabla 6 Variables para el caso de prueba Búsqueda de Transcrito por el nombre

No.	Nombre del campo	Clasificación	Valor Nulo	Descripción
1	criterio	Campo de texto	No	Secuencia de caracteres alfanuméricos:NM_79501

Tabla 7 Caso de prueba 1: Búsqueda de Transcrito por el nombre

Escenario	Descripción	criterio	Respuesta del sistema	Flujo central
EC 1.1 Introducir criterio de búsqueda	Se llena el campo correctamente y se realiza la búsqueda de transcrito	V[4]: NM_002908	Mostrar información del criterio introducido	1- Búsqueda 2- Se introduce el valor. 3- Se selecciona búsqueda por nombre de transcrito. 4-Buscar
EC 1.2 Introducir	Se desea realizar la	I[5]: ( )	Muestra un	1- Búsqueda

[4]Entrada valida
[5]Entrada invalida

valor en blanco	búsqueda de transcrito		mensaje de error en el campo vacío.	2- Se introduce el valor. 3- Se selecciona búsqueda por nombre de transcrito. 4-Buscar

Conclusiones

En el presente capítulo se creó el modelo de implementación, que arrojó como resultado el diagrama de componentes del caso de uso “Visualizar SNPs influyentes de un gen”, posibilitando un acercamiento a los componentes más importantes para el funcionamiento del sistema y encapsulados en paquetes para una mejor comprensión.

Se definieron los conceptos de Pruebas de Caja Blanca y Pruebas de Caja Negra. Con la aplicación del la técnica del camino básico se obtuvo una complejidad ciclomática de valor cinco para la función encargada de buscar transcritos a través del nombre, además se aplicó la técnica de partición de equivalencia a las interfaces mostradas en el Portlet. Las pruebas permitieron validar las funcionalidades implementadas para obtener un producto de mayor calidad.

CONCLUSIONES

Una vez concluida la investigación se puede afirmar que se incorporó las funcionalidades de la aplicación Alas-SiRNA_Design a la Plataforma de Servicios Bioinformáticos de la Universidad de las Ciencias Informáticas. Para ello:

- Se determinó el uso de componentes de interfaz web denominados portlets, que proveen información en formato HTML y pueden ser utilizados por otras aplicaciones indistintamente de las tecnologías usadas en su confección. Ello implica que el sistema puede ser trasladado o utilizado fácilmente por otras aplicaciones.

- Se definieron los requisitos del módulo SiRNA_Design y se agruparon en cuatro Caso de Uso, identificando como arquitectónicamente significativo el CU “Visualizar SNPs influyentes en un gen”. La arquitectura se confeccionó con la utilización de varios patrones arquitectónicos y de diseño, destacándose SOA y el MVC. Los mismos fueron aplicados en la realización de los diagramas de interacción y del modelo del diseño, obteniendo un sistema separado en capas, altamente escalable y menos vulnerable al cambio.

- Se implementaron los requisitos funcionales identificados, obteniendo un sistema capaz de realizar el diseño de siRNAs mediante el análisis de secuencias de mRNA. El sistema permite realizar búsqueda de datos de genes y transcritos, análisis de secuencias de mRNA, ensamblaje de secuencias comunes a un gen o específicas de un transcrito de un gen, visualización de SNPs influyentes en un gen y en un mRNA.

- Se validaron las funcionalidades implementadas a través de la interfaz del sistema, mediante el diseño de pruebas de caja negra de partición de equivalencia y de análisis de valores al límite. Las pruebas permitieron encontrar inconformidades en la validación y en la búsqueda de información sobre la base de datos, estas fueron corregidas con inmediatez y contribuyeron a aumentar la calidad del sistema.

- Se recomienda para futuras versiones que la aplicación pueda brindar un flujo de trabajo directo haciendo posible que la información mostrada sea menos engorrosa y más independiente del especialista.

REFERENCIAS BIBLIOGRÁFICAS

1. **Martínez Usero, J. Á.** Análisis de los usuarios, contenidos y servicios de los servicios públicos electrónicos. [En línea] 2007. http://eprints.ucm.es/6253/1/2007-BAAB-administracion.pdf.
2. **Altschul, S., Boguski, M., Gish, W., & Wootton, J.** *Issues in searching molecular Sequence databases.* s.l. : Nat Genet. 6. 119-29 .
3. **Lic. Cañedo, R. y Téc. Arencibia, R.** Bioinformática: en busca de los secretos moleculares de la vida. [En línea] http://scielo.sld.cu/pdf/aci/v12n6/aci02604.pdf. ISSN: 1561-2880.
4. Bases de datos y tecnologías Web. [En línea] 2013. [Citado el: 25 de Mayo de 2013.] http://riveracaballero.wikispaces.com/Bases+de+datos+y+tecnolog%C3%ADas+Web..
5. **Pérez Capdevila, J.** Las Tecnologías Web para la Gestión del Conocimiento. [En línea] Septiembre de 2007. http://www.sociedadelainformacion.com/9/las_tecnologias_web.htm . ISSN: 1578-326x..
6. DNA Data Bank of Japan (DDBJ). [En línea] http://www.ddbj.nig.ac.jp/intro-e.html.
7. National Center for Biotechnology Information (NCBI). [En línea] http://www.ncbi.nlm.nih.gov/About/glance/ourmission.html..
8. Bioinformatics Resource Portal (Expasy). [En línea] http://expasy.org.
9. European Bioinformatics Institute (EBI). [En línea] http://www.ebi.ac.uk/Information..
10. **Cobas, Ana Rosa Montone.** *Módulo de visualización de servicios para la plataforma.* Habana : s.n., 2011.
11. **Lara, Beatriz.** *Desarrollo de algoritmos para SiRNA Web Services para el análisis de la influencia de las mutaciones puntuales en los miRNAs.* Ciudad de la Habana : s.n., 2011.
12. **Pelechano, V.** Servicios Web. Estándares, Extensiones y Perspectivas de Futuro. [En línea] ftp://jano.unicauca.edu.co/cursos/Enfasis_III/Documentacion/Referencias/ServiciosWeb.pdf.
13. **Arboleda, L. M.** Servicios web: distribución e integración. [En línea] http://www.icesi.edu.co/biblioteca_digital/bitstream/10906/403/1/larboled_servicios-web.pdf..
14. **Services, IBM Global Business.** *Arquitectura orientada a servicios.* Somers, NY : s.n., . 10589.
15. **Allamaraju, Subrahmanyam, y otros.** *Programación Java Server con J2EE Edition.* pág. 1206.
16. **Berberl, Antonio J.** Portlet Gestor de Comentarios de Contenidos. [En línea] Julio de 2009. http://www.iit.upcomillas.es/pfc/resumenes/4aaebc47f2717.pdf.
17. **Moreño, Juan J.** Curso Java y Tecnologías Java EE. [En línea] 2009. http://www.bibliocomunidad.com/web/libros/Curso Java y J2EE.pdf.
18. **Scamercio, Francesco.** Introducion a Liferay Portal. [En línea] 10 de Diciembre de 2010. http://francescoscamarcio.com/2010/12/10/introducion-a-liferay-portal/.
19. **Jonas, J.** *Liferay Portal Enterprise Intranets.* Birmingham, E.E. U.U. : Packt Publishing., 2008. ISBN 978-1-847192-72-1.
20. *Revista Digital de Innovación y experiencias educativas.* **Peréz, Teresa Garzón.** 2010, págs. 1-15.
21. PostgreSQL. [En línea] http://www.postgresql.org.es/.
22. Ventajas y desventajas de PostgreSQL. [En línea] Enero de 2012. http://www.aplicacionesempresariales.com/ventajas-y-desventajas-de-postgresql.html.
23. **Pressman, Roger S.** *Ingeniería de software. Un enfoque práctico.* s.l. : McGraw-Hill Companies. págs. 574-585. ISBN: 8448132149.
24. Visual Paradigm for UML. [En línea] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%28M%C3%8D%29_14720_p/.

25. **Sierra, Daniel.** Visual Paradigm For Uml. [En línea] http://www.slideshare.net/vanquishdarkenigma/visual-paradigm-for-uml.
26. **Díaz León, Andry Daniel.** *Módulo básico de la Plataforma de Servicios Bioinformáticos.* La Habana : s.n., 2012.
27. Ventajas en la utilización de Eclipse. [En línea] Junio de 2012. http://redinertho.wordpress.com/2012/06/22/ventajas-en-la-utilizacion-de-eclipse/.
28. Apache Tomcat. [En línea] http://tomcat.apache.org/index.html.
29. **Durán, A. y Medel, R.** Introducción a Apache Tomcat 5.5. [En línea] http://www.lsi.us.es/docencia/get.php?id=1923.
30. **Minter, D.** *Beginning Spring 2. From Novice to Professional.* New York, E.E. U.U. : Apress, 2008.
31. **Walls, Craig y Breidebach, Ryan.** *Spring in action.* s.l. : Manning Publications, 2005. pág. 5. ISBN 1-932394-35-4.
32. Metodologia Open Up. [En línea] Junio de 2010. http://www.buenastareas.com/ensayos/Metodologia-Open-Up/446545.html.
33. **Mato García, Rosa María.** *Diseño de Bases de Datos.* Ciudad de la Habana : s.n., 2002.
34. OpenUP como alternativa metodológica para proyectos pequeños de software. [En línea] Septiembre de 2008. http://kasyles.blogspot.com/2008/09/openup-como-alternativa-metodolgica.html.
35. **Larman, C.** *ML y patrones. introducción al análisis y diseño orientado a objetos.* México : Prentice Hall, Inc. ISBN 970-1 7-0261-1.
36. **Martínez Jera, E y González Enríquez, L.** *AlasClínicas: Desarrollo de la Gestión de Datos de Ensayos Clínicos a partir del sistema OpenClinica.* La Habana : s.n., 2009.
37. Arquitectura de Software: ¿Qué es, y cómo funciona? . [En línea] http://www.ucci.edu.pe/blog/ingenieria_sistemas/?p=34.
38. Modelo Vista Controlador-Definición y características. [En línea] Noviembre de 2010. http://www.comusoft.com/modelo-vista-controlador-definicion-y-caracteristicas.
39. **Campo, G.** Patrones de Diseño, Refactorización y Antipatrones. Ventajas y Desventajas de su Utilización en el Software Orientado a Objetos. [En línea] 2009. http://www.ucasal.net/templates/unid-academicas/ingenieria/apps/4-p101-Campo.pdf.
40. **Trott, J. y Shalloway, A.** *Design Patterns Explained. MODELER.* 2000.
41. **Mendoza, J.** Diseño del sistema de tarjeta de crédito con UML. [En línea] http://sisbib.unmsm.edu.pe/bibvirtualdata/Tesis/Basic/mendoza_nj/Cap5.pdf.
42. **Visconti, M y Astudillo, H.** Fundamentos de Ingeniería de Software. [En línea] http://www.inf.utfsm.cl/~visconti/ili236/Documentos/08-Patrones.pdf.
43. **Tello, J.** Diagramas de Secuencia. [En línea] http://www2.uah.es/jcaceres/capsulas/DiagramaSecuencia.pdf.
44. **Jacobson, I., Booch, G. y J., Rumbaugh.** *El proceso unificado de desarrollo de software.* México. : Adison Wesley, 2000. ISBN: 84-7829-036-2..
45. **Dr. Usaola, M.** Mantenimiento Avanzado de Sistemas de Información. [En línea] http://alarcos.esi.uclm.es/doc/masi/doc/lec/parte5/polo-apuntesp5.pdf.
46. **Mendoza, J.** Diseño del sistema de tarjeta de crédito con UML. [En línea] [Citado el:] http://sisbib.unmsm.edu.pe/bibvirtualdata/Tesis/Basic/mendoza_nj/Cap5.pdf.
47. Scribd. [En línea] http://es.scribd.com/doc/53520152/9/Workflow-de-Implementacion.
48. Apache Software Foundation. [En línea] http://axis.apache.org/axis2/c/core.

ANEXOS

Anexos A. Descripción de Caso de Uso

Tabla 8 CU Guardar archivo Fasta con información de genoma humano

Caso de Uso	Guardar archivo Fasta con información de genoma humano.	
Actores	Especialista	
Propósito	Buscar información de todos los genes en el genoma humano y crear un archivo en formato Fasta con esta información.	
Resumen	El caso de uso se inicia cuando el especialista desea crear un archivo Fasta con información de todos los genes en el genoma humano.	
Precondiciones		
Referencias	RF1 , RF2	
Prioridad	Secundario.	
Flujo normal de eventos		
Acción del actor		**Respuesta del sistema**
1. El especialista escoge la opción crear un archivo Fasta con la información de todos los genes en el genoma humano.		2. Busca información de los genes en el genoma humano.
		3. Crea un archivo con formato Fasta con la información obtenida
Poscondiciones		

Tabla 9 Visualizar SNPs influyentes en el silenciamiento de un miRNA.

Caso de Uso	Visualizar SNPs influyentes en el silenciamiento de un miRNA.
Actores	Especialista
Propósito	Visualiza aquellos SNPs que sean influyentes en el poder silenciador de un determinado miRNA.
Resumen	El caso de uso se inicia cuando el especialista desea conocer los SNPs influyentes de un miRNA determinado.
Precondiciones	

Referencias	RF6
Prioridad	Alta.
Flujo normal de eventos	
Acción del actor	**Respuesta del sistema**
1. El especialista escoge la opción Visualizar SNPs influyentes de un gen determinado pasando como parámetro el ID del miRNA.	2. Busca los SNPs influyentes de los miRNAs maduros contenidos en el miRNA seleccionado y los visualiza.
Poscondiciones	

Tabla 10 Visualizar Mostrar información de miRNAs.

Caso de Uso	Mostrar información de miRNAs.
Actores	Especialista
Propósito	Buscar toda la información referente a los miRNAs.
Resumen	El caso de uso se inicia cuando el especialista desea conocer información de los miRNAs.
Precondiciones	
Referencias	RF3 , RF4
Prioridad	Alta.
Flujo normal de eventos	
Acción del actor	**Respuesta del sistema**
1. El especialista escoge la opción buscar información de un miRNA pasando como parámetro el ID del miRNA.	2. Busca los miRNAs maduros contenidos en dicho miRNA.
	3. Busca SNPs presentes en el miRNA maduro seleccionado.
	4. Busca genes blancos del miRNA maduro seleccionado.
Poscondiciones	