Universidad de las Ciencias Informáticas

FACULTAD 6

Título: Sistema informático para el ensamblaje y edición de secuencias de ácidos nucleicos.

Trabajo de Diploma para optar por el título de
Ingeniero Informático

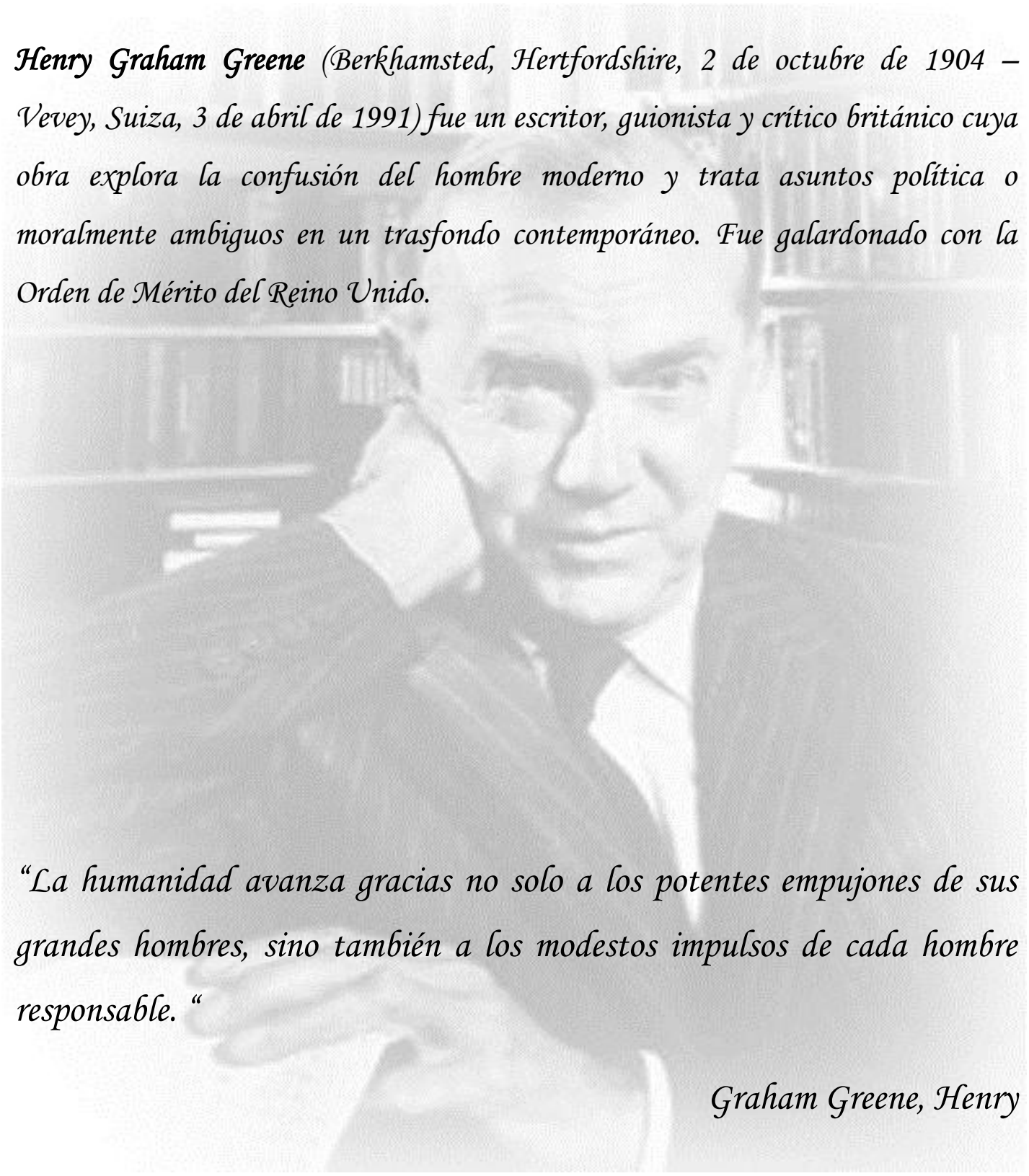
Autor: Janier José Ramírez Landaburo

Tutores: MSc. Orlando Martínez Pérez

Ing. Dayana Joseph Smarth

La Habana, Junio 2013
"Año 54 de la Revolución"

Henry Graham Greene *(Berkhamsted, Hertfordshire, 2 de octubre de 1904 – Vevey, Suiza, 3 de abril de 1991) fue un escritor, guionista y crítico británico cuya obra explora la confusión del hombre moderno y trata asuntos política o moralmente ambiguos en un trasfondo contemporáneo. Fue galardonado con la Orden de Mérito del Reino Unido.*



"La humanidad avanza gracias no solo a los potentes empujones de sus grandes hombres, sino también a los modestos impulsos de cada hombre responsable. "

Graham Greene, Henry

DECLARACIÓN DE AUTORÍA

Declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

Janier José Ramírez Landaburo

_____

Firma del Autor

MSc. Orlando Martínez Pérez

_____

Firma del Tutor

Ing. Dayana Joseph Smarth

_____

Firma del Tutor

AGRADECIMIENTOS

A todos mis amigos del pre, especialmente a Raúl, Dayron, Tomás, Ariel y Gretel junto a quienes aprendí cosas inolvidables y juntos vivimos momentos que perdurarán en mi memoria por siempre.

A mis compañeros del 6108, 6207,6309, 6409, 6509, especialmente a Karel, Yasel y a otros que ya no se encuentran entre nosotros, a todos con los que compartí ese pequeño espacio que es el aula, donde pasamos momentos alegres, tristes y donde convivimos la mayoría del tiempo aquí en la Universidad, pero fue ahí en realidad donde aprendimos todo lo que hoy sabemos. Hoy es imposible nombrarlos a todos, pero del mismo modo es imposible olvidar aquellos momentos.

A todos mis profesores por su esmero en mi formación como profesional, a profesores como Lacoste, Yosuán, Manolo y Niurka por su especial dedicación, entrega y amor en esa difícil tarea que es enseñar.

Hacerse cargo de mí es bastante complicado…jajaja, sin embargo de repente apareció alguien en mi vida que no se separó ni un solo día cuando pasé por momentos difíciles con mi brazo y a quien le debo prácticamente terminar todo este trabajo en tiempo. Gracias por hacerme sentir orgulloso de haberte conocido, gracias por tu cariño y comprensión. Gracias Lily, mi tusita.

Cuando aún era niño, apareció de repente en mi vida una persona de la que hoy estoy muy orgulloso de tener como hermano, aunque siempre lo esté molestando es una de las personas que más admiro y quiero. Espero que se esfuerce mucho en el camino que le queda por recorrer que todavía es bastante y aproveche esa inteligencia que tiene. Gracias a mi hermano Alejandro.

Hubo dos personas que un día queriéndole hacer un bien a la humanidad terminaron haciéndome a mí, algunos dicen que nací con 43 semanas, pero bueno, nunca es tarde si la dicha es buena. Gracias por crearme, por aceptarme, por indicarme siempre el camino correcto, por ser ambos mi ejemplo a seguir. Gracias a mis padres Ivette y Jochy.

A veces me siento la persona más afortunada del universo, creo que tener dos madres no es una posibilidad que tiene todo el mundo. Gracias por ser mi mejor aliada, la persona que más me mima, la que más me comprende, la que verdaderamente sabe cuando me pasa algo. Gracias por tus consejos. Gracias a mi tía Yudelkis.

No sé cómo se agradece la dedicación de toda una vida, desde que di mis primeros pasos y aprendí mis primeras letras hasta hoy. Son ustedes la principal inspiración de estos últimos años de mi vida por ser promotores de mi educación, por estar ahí siempre conmigo, en las buenas y en las malas, en la luz y en la oscuridad, en el calor y en el frio y por esperar siempre lo mejor de mí. Gracias a mis abuelos Nancy y Eduardo, a quienes cariñosamente llamamos Nanita y Kinde.

A mis bisabuelos Edita y Rigoberto, quienes fueron partícipes de mi educación y aunque hoy Rigoberto ya no esté entre nosotros, lo tengo más presente que nunca.

A mi primo Iván, mi tía Yoli y a Roberto, quienes de una manera u otra pusieron su granito de arena y no fue poca la ayuda que recibí de ellos durante mi estancia en la universidad.

A los desarrolladores de Biojava, a los del proyecto AMOS, al Centro de Bioinformática y Biología Computacional de la Universidad de Maryland y a los miembros de los foros de debate de las comunidades por todo el soporte que me brindaron.

A mis tutores por su impulso en el desarrollo de este trabajo.

A todas aquellas personas que de algún modo formaron parte de mi formación profesional, de mi vida privada y de mis buenos y malos momentos dentro y fuera de la universidad.

MUCHAS GRACIAS A TODOS...

DEDICATORIA

A toda mi familia, quienes al costo de un sacrificio enorme han formado la persona en la que me he convertido.

Especialmente a mis abuelos Nanita y Kinde quienes han sido mi único camino a seguir y ahora siento que estoy un escalón más cerca de ser un poco más como ellos.

EL Autor

RESUMEN

La Bioinformática es una de las esferas de la ciencia que más explota los sistemas de cómputo debido a la inmensa cantidad de datos que se hace necesario procesar durante un proceso investigativo. Una de las tareas más comunes en los proyectos de Bioinformática es el uso de herramientas matemáticas para extraer información útil de datos producidos por técnicas biológicas, como la secuenciación del genoma y en particular, el ensamblado de secuencias genómicas de alta calidad desde fragmentos obtenidos tras la secuenciación del ADN. La presente investigación se centra en el desarrollo de un sistema para realizar el ensamblaje y edición de secuencias de ácidos nucleicos, el cual es un proceso investigativo que se lleva a cabo en muchos centros científicos del país, específicamente en el Instituto Pedro Kourí. El objetivo de la investigación es crear un sistema capaz de realizar dicho proceso debido al alto costo que implican las herramientas de este tipo a nivel internacional. Para ello se definieron una serie de tecnologías, entre las cuales se encuentran metodologías de desarrollo, lenguajes de programación, arquitecturas de software y patrones de diseño, las cuales se describen a lo largo del presente documento. El resultado fue la obtención de un sistema robusto y escalable capaz de realizar el ensamblaje y edición de secuencias de ácidos nucleicos, entre otras funcionalidades que se incorporaron con el objetivo de incrementar la usabilidad del mismo.

PALABRAS CLAVE: Bioinformática, ADN, Secuenciación, Ensamblaje de secuencias

TABLA DE CONTENIDOS

INTRODUCCIÓN 1
CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA 6
Introducción 6
1.1. Proceso de ensamblaje de secuencias 6
1.2. Algoritmos para el ensamblaje de secuencias 10
1.2.1. Algoritmos Voraces 10
1.2.2. Grafos de Brujin 10
1.2.3. Transformación de Burrows-Wheeler 11
1.2.4. Smith–Waterman 11
1.3. Problemas del ensamblaje de secuencias 11
1.4. Códigos de Ambigüedad 13
1.5. Sistemas para el ensamblaje de secuencias de ácidos nucleicos. 13
1.6. Metodología de desarrollo 15
1.7. Herramientas y Tecnologías 18
1.7.1. Herramienta CASE 18
1.7.2. Entorno de Desarrollo Integrado 19
1.8. Lenguajes 20
1.8.1. Lenguaje de programación 20
1.8.2. Lenguaje de modelado 21
1.9. Librerías 22
Conclusiones del capítulo 23
CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA 25
Introducción 25
2.1. Modelo de Dominio 25
2.1.1. Definición de las Clases de Modelo del Dominio 25
2.2. Requisitos del sistema. 26
2.2.1. Requisitos Funcionales: 26
2.2.2. Requisitos no Funcionales 27
2.3. Definición de los actores del sistema 28

2.4. Definición de los casos de uso del sistema 29
2.4.1. Patrones de Caso de Uso 29
2.4.2. Listado de Casos de Uso 29
2.5. Diagrama de Casos de Uso del Sistema 30
2.5.1. Descripción de Casos de Uso 31
Conclusiones del Capítulo 32
CAPÍTULO 3: ANÁLISIS Y DISEÑO DEL SISTEMA 33
Introducción 33
3.1. Arquitectura de Software 33
3.1.1. Patrón Arquitectónico 33
3.1.2. Patrones de Diseño 36
3.2. Modelo del Diseño 38
3.3. Diagrama de Secuencia 39
Conclusiones del Capítulo 40
CAPÍTULO 4: IMPLEMENTACIÓN Y PRUEBAS 41
4.1. Modelo de Implementación 41
4.1.1. Diagrama de Componentes 41
4.1.2. Modelo de Despliegue 43
4.2. Pruebas de Software 44
4.2.1. Pruebas de Caja Blanca 45
5.1.1. Pruebas de Caja Negra 47
Conclusiones del Capítulo 51
CONCLUSIONES 52
RECOMENDACIONES 53
REFERENCIAS BIBLIOGRÁFICAS 54
BIBLIOGRAFÍA 55
ANEXOS 60

ÍNDICE DE TABLAS

Tabla 1: Diferencias entre el Ensamblaje tipo De Novo y el Ensamblaje Comparativo o por Referencia.....7
Tabla 2: Códigos de Ambigüedad. 13
Tabla 3: Definición de los actores del sistema. 28
Tabla 4: Descripción del CU Ensamblar Secuencias de Ácidos Nucleicos.....32
Tabla 5: Patrones GRASP y sus funciones dentro del sistema. 37
Tabla 6: Patrones GoF y sus funciones dentro del sistema.....38
Tabla 7: Descripción de los componentes del CU “Gestionar Secuencias”. 43
Tabla 8: Descripción de los nodos del Modelo de Despliegue. 44
Tabla 9: Diseño de casos de prueba CU “Mostrar Secuencia de Ácidos Nucleicos”. 49
Tabla 10: Descripción de variables CU “Mostrar Secuencia de Ácidos Nucleicos”.....49
Tabla 11: Diseño de casos de prueba CU “Exportar Secuencia de Ácidos Nucleicos”.....50
Tabla 12: Descripción de variables CU “Exportar Secuencia de Ácidos Nucleicos”. 50
Tabla 13: No conformidades detectadas a través de los casos de prueba aplicando la técnica de pruebas de caja negra.....51

Tabla A 1: Descripción del CU “Gestionar Secuencias de Ácidos Nucleicos”.....63
Tabla A 2: Descripción del CU “Modificar Parámetros de Ensamblaje”.....64
Tabla A 3: Descripción del CU “Mostrar Cromatogramas”.....65
Tabla A 4: Descripción del CU “Exportar Secuencia”.....66
Tabla A 5: Descripción del CU “Establecer Intervalos de Confianza”. 67
Tabla A 6: Secuencias de Ácidos Nucleicos válidas. 67

ÍNDICE DE FIGURAS

Figura 1: Pasos en el proceso de ensamblaje de secuencias. 7
Figura 2: Cromatograma con lecturas de mala calidad. 8
Figura 3: Red de interacciones entre las secuencias que alinean entre sí. 9
Figura 4: Bases de las cadenas alineadas y la cadena consenso. 9
Figura 5: Diagrama de Casos de Uso del Sistema. 30
Figura 6: Patrón Modelo Vista Controlador 34
Figura 7: Vista Lógica del Sistema. 36
Figura 8: Diagrama de clases del diseño del CU “Ensamblar Secuencias”. 39
Figura 9: Diagrama de Secuencia del CU “Ensamblar Secuencias”. 40
Figura 10: Diagrama de Componentes de CU “Ensamblar Secuencias”. 42
Figura 11: Modelo de Despliegue. 44
Figura 12: Prueba del camino básico aplicada a la función que actualiza la calidad de las secuencias. 46
Figura 13: Grafo que representa el camino básico de la función Actualizar Calidad. 47

INTRODUCCIÓN

La biología es la ciencia que tiene como objeto de estudio a los seres vivos y específicamente, su origen, su evolución y sus propiedades. Se ocupa tanto de la descripción de las características y los comportamientos de los organismos individuales como de las especies en su conjunto, así como de la reproducción de los seres vivos y de las interacciones entre ellos y el entorno. La biología es una disciplina científica que abarca un amplio espectro de campos de estudio que, a menudo, se tratan como disciplinas independientes. Todas ellas juntas estudian la vida en un amplio rango de escalas (1). Cada célula de un organismo vivo contiene cromosomas compuestos por pares de bases de una secuencia de ADN. Esta secuencia, denominada "genoma[1]", representa un conjunto de instrucciones que controlan la replicación y función de cada organismo.

La biología, en la actualidad, tiene como gran aliado a la tecnología, por medio de ella, sus estudios y análisis, son más acabados y completos, por lo que su campo de observación y experimentación, se amplía enormemente al utilizarla (2).

A partir de la Biología y la Informática como ciencias, surge la Bioinformática que es una disciplina científica que combina biología, computación y tecnologías de la información, la cual tiene como objetivo facilitar nuevas percepciones biológicas y crear una perspectiva global que permita identificar los principios unificadores de la biología. Las principales áreas de la Bioinformática son por tanto: el desarrollo de herramientas que permitan el acceso, uso y actualización de distintos tipos de información biológica; el desarrollo de nuevos algoritmos y soluciones estadísticas para analizar grandes conjuntos de datos y resolver problemas biológicos complejos, tales como predecir la estructura de un gen en una secuencia genómica, predecir la estructura de proteínas e identificar familias de proteínas por su similitud de secuencia (3).

Una de las tareas más comunes en los proyectos de Bioinformática es el uso de herramientas matemáticas para extraer información útil de datos producidos por técnicas biológicas, como la secuenciación del genoma y en particular, el montaje o ensamblado de secuencias genómicas de alta calidad desde fragmentos obtenidos tras la secuenciación del ADN.

[1]Conjunto de genes que especifican todos los caracteres que pueden ser expresados en un organismo.

En Bioinformática el proceso de ensamblaje de secuencias se refiere a cuando los cromosomas de muchas células se rompen en pequeños fragmentos, los cuales son secuenciados y reensamblados en secuencias largas o contigs[2]. Los contigs pueden ensamblarse en secuencias más largas llamadas scaffolds[3] y a veces si la calidad de la secuenciación es lo suficientemente alta, puede haber información suficiente para ensamblar los scaffolds en cromosomas. La colección resultante de secuencias después del ensamblaje se llama un "genoma de ensamblaje" (4). Actualmente existen dos tipos fundamentales de ensamblado de secuencias, denominados ensamblaje comparativo y ensamblaje de novo.

La tarea de un ensamblador de secuencias es combinar todas las lecturas en contigs basado en la similitud de las secuencias individuales leídas. El proceso de secuenciación comienza rompiendo físicamente el ADN en millones de fragmentos al azar, que luego son leídos por una máquina secuenciadora. A continuación un programa llamado ensamblador de secuencias lee los solapamientos y reconstruye la secuencia original.

El ensamblaje de secuencias es actualmente una de las tareas más complejas computacionalmente, incluso hoy en día pocos centros cuentan con los recursos tanto en software como en hardware para ensamblar un genoma completo fraccionado en miles o millones de secuencias.

El Instituto de Medicina Tropical Pedro Kourí (IPK) es una institución científica cubana, reconocida internacionalmente por los numerosos resultados relevantes y premios obtenidos por sus investigadores. Además, algunos de sus Laboratorios constituyen Centros Colaboradores de la Organización Mundial de la Salud y, a nivel nacional, es la máxima autoridad en las disciplinas de Microbiología, Parasitología, Medicina Tropical así como Clínica y Epidemiología de Enfermedades Transmisibles (5).

En el IPK, se llevan a cabo investigaciones, que incluyen el proceso de ensamblaje de secuencias de ácidos nucleicos. Dicho Instituto cuenta con un software de licencia comercial (Sequencher) capaz de gestionar dichos procesos, pero debido al alto costo para la adquisición y mantenimiento del mismo se hace complicada la renovación de la licencia la cual oscila sobre los 1800 euros para uso académico y 2600 para uso investigativo y comercial , además de que esta licencia es física y solo puede utilizarse en

[2]Se trata de una secuencia contigua que puede ser leída a través de varias secuencias individuales solapantes.

3 Contigs que no solapan entre si (tienen gaps) pero que se sabe que van juntos gracias a la información de los extremos apareados.

una sola computadora a la vez, o sea consiste en un dispositivo USB el cual es reconocido por el Sequencher para que el sistema brinde el 100% de sus funcionalidades.

Existen varias herramientas para el ensamblaje de secuencias a la misma vez que existen otras para la visualización de los ensamblajes, pero no existe una solución libre que integre ambos procesos.

En Cuba no se ha desarrollado una herramienta de este tipo, por lo que su desarrollo sería una buena opción para los centros científicos del país que realicen este tipo de ensamblaje, en materia de ahorro de recursos.

La Universidad de las Ciencias Informáticas (UCI) es una institución cubana creada con el objetivo de formar profesionales comprometidos y altamente calificados cuyo método de formación consiste en vincular el estudio a la producción de software. En la UCI se encuentra el Centro de Tecnologías y Gestión de Datos (DATEC), en el cual está presente la Bioinformática como una rama de investigación y producción de software. El proyecto de Bioinformática del Centro DATEC tiene entre sus objetivos el abastecimiento al país de herramientas para la gestión y análisis de datos biológicos. Debido a ello se ha encomendado a la UCI la tarea de solucionar dicho problema, poniendo a disposición del IPK los servicios y la gran capacidad que posee la Universidad para el desarrollo de software con calidad.

Dada esta situación el **problema a resolver** del presente trabajo de diploma se resume en: ¿Cómo integrar el proceso de ensamblaje de ácidos nucleicos con una interfaz gráfica de usuario (GUI) que permita la edición de las secuencias y el análisis de las mismas por parte de los investigadores del IPK?

El **objeto de estudio** se centra en la integración del proceso de ensamblaje y edición de secuencias de ácidos nucleicos con una GUI.

El **campo de acción** se enmarca en la integración del proceso de ensamblaje de secuencias de tipo comparativo con una GUI.

Para dar solución a dicho problema se ha planteado el siguiente **objetivo general**: Desarrollar un Sistema Informático para el ensamblaje y edición de secuencias de ácidos nucleicos.

Objetivos específicos

- Realizar análisis y diseño al Sistema Informático para el Ensamblaje y Edición de Secuencias de Ácidos Nucleicos.
- Implementar el Sistema Informático para el Ensamblaje y Edición de Secuencias de Ácidos Nucleicos.
- Evaluar el Sistema Informático para el Ensamblaje y Edición de Secuencias de Ácidos Nucleicos.

Para dar cumplimiento a los objetivos planteados, se identificaron las siguientes **tareas de investigación**.

1. Análisis del estado del arte sobre los sistemas de código libre para el ensamblaje de secuencias de ácidos nucleicos.
2. Estudio y selección de la metodología de desarrollo, herramientas y tecnologías a utilizar para la implementación del Sistema
3. Identificación de los requisitos funcionales del Sistema.
4. Selección de los patrones de diseño a utilizar.
5. Implementación de los requisitos funcionales identificados.

El documento está estructurado en cuatro capítulos que incluyen los siguientes elementos:

CAPITULO 1. **Fundamentación Teórica:** Muestra un estudio detallado sobre el estado actual del proceso de ensamblaje y edición de secuencias de ácidos nucleicos, o sea herramientas que se utilizan, sus características y algoritmos existentes para este proceso, también se especifican las herramientas a utilizar, metodología de desarrollo, lenguaje de programación y modelado y tecnologías.

CAPITULO 2. Características del Sistema: Incluye el análisis y modelado de los procesos del negocio, los requisitos funcionales y no funcionales identificados, además de los diagramas y descripción del caso de uso identificado como arquitectónicamente significativo.

CAPITULO 3. Análisis y Diseño: En este capítulo se describen los patrones de diseño utilizados, además de la vista lógica del sistema, así como el diagrama de secuencia y el modelo de clases del diseño para el caso de uso identificado como arquitectónicamente significativo.

CAPITULO 4. Implementación y Pruebas: Se presenta el modelo de implementación en el cual se define el modelo de despliegue y el diagrama de componentes para el caso de uso identificado como arquitectónicamente significativo, además de la descripción de cada uno de ellos, también se muestran los modelos de prueba usados para comprobar el correcto funcionamiento del sistema.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Introducción

En este capítulo se abordarán temas relacionados con el proceso de ensamblaje y edición de secuencias de ácidos nucleicos así como algoritmos que se emplean en dicho proceso y algunos sistemas que lo realizan. Además se realizará un estudio de las herramientas, metodologías y tecnologías disponibles para llevar a cabo el desarrollo del software. Con el objetivo de lograr una mayor comprensión del negocio también se definirán algunos conceptos fundamentales.

1.1. Proceso de ensamblaje de secuencias

El proceso de ensamblaje de secuencias está concebido por una serie de fases, esto depende del tipo de ensamblaje que se desea realizar, actualmente existen dos tipos fundamentales:

Ensamblado de Novo: Intenta reconstruir la secuencia de ADN completa a partir de las lecturas sin ningún tipo de conocimiento previo a cerca del genoma a ensamblar. Busca lecturas cuyo final coincida con el principio, de otra de forma que se puedan unir para formar fragmentos mayores hasta completar el genoma (6).

Ensamblado Comparativo o por Referencia: Basándose en un genoma secuenciado previamente y que suponemos sea similar al que se quiere ensamblar. El procedimiento básico tratará de colocar cada una de las lecturas en la posición adecuada utilizando el genoma de referencia como guía (6).

Por Referencia	De Novo
Las lecturas se alinean contra una referencia.	No existe información previa del genoma.
Usado ampliamente para el estudio del genoma humano.	Necesario para genomas nuevos o donde difiere de la secuencia de referencia, por ejemplo el cáncer.
Genera contigs que se ensamblan en supercontigs.	Las lecturas se asignan en su posición correspondiente.
Se buscan similitudes entre las lecturas y su referencia.	Se buscan solapamientos entre las lecturas.
Se utilizan más lecturas para proporcionar mayor cobertura.	Se busca un consenso entre las lecturas.

Se generan más repeticiones y se utilizan extremos apareados.	Mayor consumo de CPU y memoria.

Tabla 1: Diferencias entre el Ensamblaje tipo De Novo y el Ensamblaje Comparativo o por Referencia.

El problema identificado se centra sobre el ensamblado de tipo comparativo, debido a que en las investigaciones llevadas a cabo en el IPK se centran en tomar el genoma de algún individuo, ya sea un virus o cualquier otro organismo como referencia y alinear las lecturas obtenidas del secuenciador contra esta referencia, obteniendo una nueva cadena llamada “consenso”, la cual debe ser similar o igual al genoma de referencia. El proceso de ensamblaje comparativo consiste en lo siguiente:

El proceso de ensamblaje comparativo consta de tres fases:

1. Limpieza de secuencias
2. Alineamiento de secuencias
3. Ensamblaje.

Figura 1: Pasos en el proceso de ensamblaje de secuencias.

Limpieza de secuencias

Es el primer paso a realizar ante cualquier proyecto de secuenciación. Los secuenciadores devuelven las secuencias crudas y es necesario eliminar o bloquear diferentes regiones que:

- Contengan bases con mala calidad de lectura.
- Contengan bases que pertenezcan a vectores o adaptadores.
- Contengan bases que estén altamente repetidas.

Estas regiones interfieren con el ensamblaje y es necesario no tenerlas en cuenta (7).

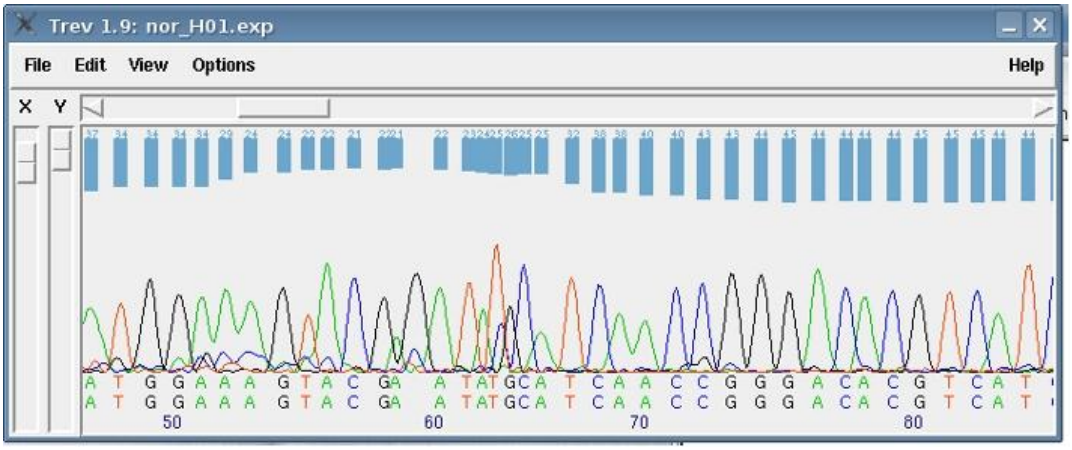


Figura 2: Cromatograma con lecturas ruidosas.

Todos los sistemas de secuenciación valoran la calidad de sus lecturas asignando un valor a cada posición. Esta estimación del error es específica de cada tecnología y es calculada por el software del equipo. Para facilitar el análisis y la interpretación de los resultados, estos valores suelen cambiar de escala a una escala normalizada y utilizable para todas las tecnologías de secuenciación (7).

Alineamiento de secuencias

El siguiente paso es alinear todas las secuencias dos a dos para identificar que secuencias se solapan entre sí. El alineamiento se realizará utilizando una secuencia como referencia. Serán identificados pares o grupos de secuencias con fragmentos de secuencias comunes. Primero se realiza una comparación rápida para identificar que secuencias comparten palabras o ventanas iguales. En un segundo paso estas parejas de secuencias se alinean mediante programas de alineación dinámica. El resultado es una red de interacciones entre las secuencias que se alinean entre sí, en el que en cada borde está localizado una secuencia (7).

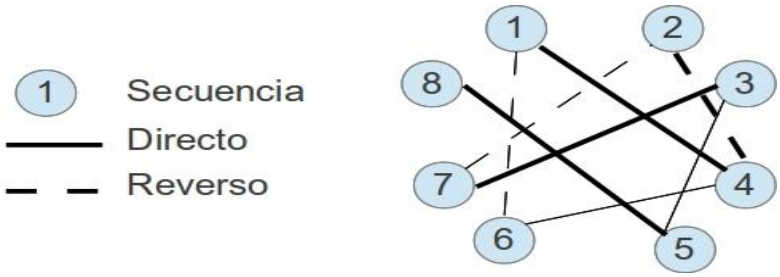


Figura 3: Red de interacciones entre las secuencias que alinean entre sí.

Ensamblaje de secuencias

A partir de estos grafos realizados con las secuencias identificadas que comparten regiones comunes y la secuencia de referencia, se crea el ensamblaje. A diferencia de un alineamiento múltiple los ensambladores no fuerzan que todas las secuencias cubran la totalidad del alineamiento, este se va alargando con la adición de nuevas secuencias. Existen diferentes algoritmos que a partir de la información almacenada en el grafo ordenan y ensamblan las secuencias utilizando alineación múltiple para obtener la mejor distribución de las secuencias solapantes. En muchos programas estos dos últimos pasos (alineamientos de secuencias y ensamblaje) son cíclicos. A partir de estos alineamientos se obtiene la secuencia consenso que representa a cada conjunto de secuencias o contigs (7).

Figura 4: Bases de las cadenas alineadas y la cadena consenso.

A continuación se describen los algoritmos más utilizados en la actualidad para el ensamblaje de secuencias.

1.2. Algoritmos para el ensamblaje de secuencias

La evolución de los programas de ensamblaje de secuencias ha ido acompañada de la mejora y propuesta de distintos enfoques para resolver el problema. Aunque comienzan a surgir nuevas tecnologías de secuenciación, actualmente las principales se basan en la secuenciación del genoma mediante el método de "perdigonada"[4] a través del cual el genoma se parte en varios fragmentos (lecturas) que posteriormente se alinean obteniendo la secuencia original (4).

La mayoría de los programas de ensamblaje de secuencias utilizan diferentes aproximaciones algorítmicas de las que se puede destacar: aproximaciones voraces o greedy, solapamiento–diseño–consenso, grafos de Brujin y basados en la transformación de Burrows-Wheeler. Existen, no obstante, varios algoritmos que se utilizan en el ensamblaje de secuencias como puede ser el algoritmo de Smith-Waterman para alineamientos locales y que se utiliza en muchos programas de ensamblaje (4).

1.2.1. Algoritmos Voraces

Los algoritmos voraces son implícitamente algoritmos de grafos ya que simplifican drásticamente el grafo al considerar solo los enlaces con más puntuación, como optimización pueden instanciar solo un solapamiento por cada lectura que examinan y pueden también descartar cada solapamiento inmediatamente después de la extensión del contig. Al igual que todos los ensambladores, los algoritmos voraces necesitan mecanismos para evitar incorporar solapamientos falsos positivos en los contigs (4).

1.2.2. Grafos de Brujin

Esta aproximación para ensamblaje de secuencias se utiliza principalmente para lecturas cortas de las plataformas de SOLiD y Solexa. Se basa en grafos de K-mer que los hacen muy eficientes para tratar grandes cantidades de este tipo de lecturas, ya que los grafos de K-mer no requieren un descubrimiento de todos los solapamientos mediante la comprobación de todos contra todos, tampoco es necesario almacenar las lecturas individuales en sus solapamientos y, adicionalmente, comprime las secuencias

[4] Método de secuenciación semi-automatizado que involucra fragmentos aleatorios de un genoma, sin previo conocimiento de su localización dentro del mismo.

redundantes. Por el contrario, los grafos K-mer no mantienen secuencias en un momento determinado y son grandes consumidores de memoria para grandes genomas, aunque con los sistemas de memoria distribuida se comportan bastante bien (4).

1.2.3. Transformación de Burrows-Wheeler

La transformación de Burrows-Wheeler es un algoritmo de permutación reversible utilizado en programas de ensamblaje de secuencias ya que permite dar una solución parcial a uno de los principales problemas del ensamblaje de secuencias que es el consumo de memoria. Este algoritmo permuta el orden de las bases y consigue que la misma base se repita varias veces de manera consecutiva lo cual es útil como paso previo para la compresión y almacenamiento de los datos. La transformación de Burrows-Wheeler se utiliza en las tareas de alineación contra un genoma de referencia ya que permite reducir la memoria necesaria para indexar el genoma (4).

1.2.4. Smith–Waterman

El algoritmo Smith-Waterman es uno de los algoritmos más importantes de alineación local de secuencias de ácidos nucleicos, es decir, determina si dos secuencias biológicas tienen algún fragmento en común o son de una gran similitud teniendo en cuenta las posibles mutaciones, inserciones o eliminaciones de elementos dentro de una cadena. Por tal motivo este algoritmo es usado en diferentes ámbitos bioinformáticos con el fin de localizar alineamientos de carácter local (8). Este algoritmo posee la atractiva propiedad que garantiza encontrar el alineamiento local óptimo.

1.3. Problemas del ensamblaje de secuencias

Hay varios factores que afectan la calidad del ensamblaje:

- La calidad de las secuencias.
- El número de veces que se lee cada posición (cobertura).
- La longitud de las secuencias.

- La existencia de secuencias repetidas.

Para obtener buenos ensamblajes hay que obtener buenos solapamientos entre nuestras secuencias. Si dos secuencias se solapan entonces se puede suponer que provienen de la misma región del fragmento, cuanto mayor y con mayor similitud sea el solapamiento mejor. Pero se pueden producir solapamientos debido al azar, regiones repetidas o genes duplicados. Para evitar estos falsos positivos podemos tener en cuenta una serie de factores:

- Similitud de la región solapada.
- Longitud de la región solapada.
- Porcentaje de secuencia que solapa.
- Ignorar secuencias que solapan con muchas secuencias (repetitivas).

Los errores causados por la mala calidad, escasa cobertura y longitud de secuencias son evitables mediante la limpieza de secuencias o la adición de nuevas secuencias al ensamblaje. Pero el problema de las secuencias repetidas no puede obviarse ya que en los genomas, sobre todo en los eucariotas, existe un alto porcentaje de secuencias repetidas (7).

Secuencias repetidas

Las lecturas que únicamente contienen el mismo ADN repetitivo ensamblarán entre sí aunque provengan de regiones distintas del genoma, solo podremos diferenciarlas si contienen alguna parte de secuencia única. Esto hace que si la longitud del ADN repetitivo es mayor que el tamaño de las lecturas los ensambladores no puedan ensamblar estas regiones ya que solapan con multitud de copias, esto crea gaps y diferentes contigs. Las regiones repetidas más pequeñas que la longitud de las secuencias se podrán ensamblar perfectamente siempre que se tenga una lectura que la cruce en su totalidad, es decir que tenga en sus extremos secuencias de copia única (7).

1.4. Códigos de Ambigüedad

Los códigos de ambigüedad estándar para los nucleótidos son proporcionados por la IUPAC (Unión Internacional de Química Pura y Aplicada) e indican los nucleótidos posibles que pueden ocurrir en una determinada posición. Los símbolos son válidos tanto para el ADN y el ARN y se muestran en la siguiente tabla: (9)

Código	Secuencia(s)Posible(s)	Nombre
A	A	adenina
C	C	citosina
G	G	guanina
T	T	timina
R	AG	purina
Y	CT	primidina
K	GT	ceto
M	AC	amino
S	GC	enlaces fuertes
W	AT	enlaces débiles
B	GTC	todos excepto A
D	GAT	todos excepto C
H	ACT	todos excepto G
V	GCA	Todos excepto T
N	GACT	cualquiera

Tabla 2: Códigos de Ambigüedad.

1.5. Sistemas para el ensamblaje de secuencias de ácidos nucleicos.

Internacionalmente se han desarrollado numerosos programas para el ensamblaje de secuencias. Se puede destacar por ejemplo ABBA, MAQ, AMOS y Mosaik. Estos programas han sido diseñados y ampliamente utilizados para el ensamblaje de genomas durante los últimos años. A continuación se realiza una breve descripción de cada uno de ellos.

ABBA (Assembly Boosted By Amino acid sequences): Es un ensamblador denominado comparativo para lecturas cortas que utiliza una secuencia de aminoácidos para dirigir el ensamblaje de genes. Las

secuencias de aminoácidos están más conservadas que el ADN lo cual permite aumentar la distancia relativa que se usa como referencia (4).

MAQ (Mapping and Assembly with Quality): Ensambla mediante un proceso de identificación lecturas cortas contra secuencias de referencia. El programa está específicamente diseñado para Illumina-Solexa/AB-SOLiD pero no para 454, fue desarrollado por Heng Li del centro Sanger en C++ y utiliza secuencias en formato FASTA. MAQ proporciona una puntuación de calidad a cada alineamiento (4).

AMOS-AMOScmp: AMOS consiste en una colección de módulos que operan sobre una estructura de datos llamada banco. Un banco es en realidad un directorio que contiene una base de datos que comprende los objetos de ensamblado relacionados, tales como lecturas, contigs y scaffolds. Los módulos de este modo se comunican entre sí mediante cambios en el banco. AMOScmp es un paquete para el ensamblaje comparativo que se encuentra incluido dentro del proyecto AMOS, una herramienta para la manipulación de la entrada y salida de ficheros relacionados con el ensamblaje de genomas completos (4).

Mosaik: Es un ensamblador de secuencias guiado por referencia que utiliza el algoritmo Smith–Waterman para el alineamiento de las lecturas. Mosaik está compuesto por 4 módulos principales (10).

- MosaikBuild
- MosaikAligner
- MosaikSort
- MosaikAssembler

MosaikBuild convierte varios formatos de secuencias en un formato nativo de Mosaik, luego MosaikAligner alinea cada lectura a una serie de referencias especificadas, MosaikSort ordena dicho alineamiento para que finalmente MosaikAssembler analice el alineamiento ordenado y produzca un fichero con la salida estándar del ensamblaje (10). De esta manera, se puede decir que el flujo de trabajo consiste en proporcionar secuencias de entrada en formato FASTA, FASTQ y producir ficheros de ensamblaje.

Luego del análisis de los softwares para el ensamblaje de secuencias disponibles, ha sido seleccionado Mosaik para el desarrollo del Sistema, para ello fueron tenidas en cuenta las siguientes características:

- Es un ensamblador de tipo comparativo.
- Soporta la varias tecnología de secuenciación (Sanger, 454, Illumina/Solexa, Solid y Helicos).
- Permite ajustar parámetros fundamentales en el alineamiento de las secuencias, lo que permite realizar un alineamiento completamente configurable.
- Utiliza el algoritmo Smith-Waterman en el alineamiento de las secuencias contra la referencia, el cual es óptimo en alineamientos locales, garantizando de este modo mejores solapamientos entre las lecturas y la referencia.
- Posee suficiente documentación.
- Puede ser ejecutado en Windows.
- Es una herramienta libre.

1.6. Metodología de desarrollo

Una metodología de desarrollo se refiere a un marco de trabajo usado para estructurar, planificar y controlar el proceso de desarrollo de sistemas computacionales. Así, se espera que al utilizar una metodología para desarrollo de software, esta pueda proveer un conjunto de prácticas y herramientas que faciliten el proceso de desarrollo, ofreciendo un producto con alta calidad, seguro y que satisfaga las expectativas del cliente (11). Actualmente existen muchas metodologías, las cuales se pueden dividir en dos tipos principales: Ágiles y Tradicionales.

Algunos ejemplos de Metodologías Tradicionales son: RUP (Rational Unified Process, Proceso Racional Unificado) y OMT (Object Modeling Technique, Técnica de Modelado de Objetos). En este tipo de metodologías se considera la importancia de la documentación del sistema, lo cual permite entender, extender y darle mantenimiento al software. Además, estas metodologías proporcionan un orden y una estructura bien definida para el desarrollo del sistema (11).

Por otro lado, están las Metodologías Ágiles, las cuales presentan respuestas rápidas y efectivas al cambio; tienen un plan de proyecto flexible, y muestran simplicidad, de manera general, en el desarrollo.

Sin embargo, tienen la desventaja de generar poca documentación y no hacer uso de métodos formales. Algunos ejemplos de este tipo de metodologías, son: XP (Programación Extrema), Scrum y OpenUp (11), esta última cuenta con gran prestigio a nivel internacional aplicándose a proyectos pequeños. OpenUp evita generar documentación innecesaria de diagramas que son exigidos por otras metodologías, además de que es un proceso modelo extensible, dirigido a la gestión y desarrollo de proyectos de software basados en desarrollo iterativo, ágil e incremental apropiado para proyectos pequeños y de bajos recursos; y es aplicable a un conjunto amplio de plataformas y aplicaciones de desarrollo. Por esas y otras razones que se describen a continuación, ha sido seleccionada para el desarrollo del sistema.

OpenUp

OpenUp es un marco del proceso del desarrollo del software Open Source que en un cierto plazo, se espera que cubra un amplio sistema de necesidades para los proyectos de desarrollo. OpenUP es un proceso iterativo para el desarrollo de software que es (12):

- Mínimo: Solo incluye el contenido del proceso fundamental.
- Completo: Puede ser manifestado como proceso entero para construir un sistema.
- Extensible: Puede ser utilizado como base para agregar o para adaptar más procesos.

Beneficios en el uso de OpenUP

- Ya que es apropiado para proyectos pequeños y de bajos recursos permite disminuir las probabilidades de fracaso en los proyectos pequeños e incrementar las probabilidades de éxito.
- Permite detectar errores tempranos a través de un ciclo iterativo.
- Evita la elaboración de documentación, diagramas e iteraciones innecesarios requeridos en la metodología RUP.
- Por ser una metodología ágil tiene un enfoque centrado al cliente y con iteraciones cortas.

Principios de OpenUP

- Colaborar para alinear intereses y para compartir conocimiento.
- Balancear las prioridades.
- Centrado en la Arquitectura.
- Desarrollo Iterativo.

Fases de OpenUP

Concepción: La primera de las cuatro fases en el ciclo de vida del proyecto. Esta es acerca de entender el alcance y los objetivos del proyecto y obtener suficiente información para confirmar que el proyecto debería continuar o convencerse de que este no debería continuar. El propósito en esta fase es lograr concurrencia entre todos los involucrados sobre los objetivos del ciclo de vida para el proyecto.

Elaboración: La segunda de cuatro fases en el ciclo de vida del proyecto, cuando los riesgos arquitecturalmente significativos son dirigidos. El propósito de esta fase es establecer la línea base de la arquitectura del sistema y proporcionar una base estable para el gran esfuerzo de desarrollo de la siguiente fase.

Construcción: La tercera de cuatro fases en el ciclo de vida del proyecto, la Construcción se enfoca en diseño, implementación y prueba de las funcionalidades para desarrollar un sistema completo. El propósito de esta fase es completar el desarrollo del sistema basado en la arquitectura.

Transición: Cuarta y última fase en el ciclo de vida del proyecto. El propósito en esta fase es asegurarse que el software está listo para entregarse a los usuarios.

OpenUp es un proceso modelo y extensible, dirigido a la gestión y desarrollo de proyectos de software basado en el desarrollo iterativo, ágil e incremental, apropiado para proyectos pequeños y es aplicable a un conjunto amplio de plataformas y aplicaciones de desarrollo (12).

1.7. Herramientas y Tecnologías

1.7.1. Herramienta CASE

Se puede definir a las herramientas de Ingeniería de Software Asistida por Computadora (CASE) como un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del ciclo de vida de desarrollo de un software. La realización de un nuevo sistema requiere que las tareas sean organizadas y completadas en forma correcta y eficiente. Las Herramientas CASE fueron desarrolladas para automatizar esos procesos y facilitar las tareas de coordinación de los eventos que necesitan ser mejorados en el ciclo de desarrollo (13).

Existe una gran variedad de herramientas CASE para el modelado de sistemas y negocios, como son el Erwin, Poseidón y Visual Paradigm, esta última fue la seleccionada debido a las siguientes características.

Visual Paradigm para UML

Visual Paradigm para UML es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación (14).

Algunas características:

- Soporta Diagramas UML.
- Soporta aplicaciones de escritorio.
- Generación de código para Java.
- Se integra con el IDE NetBeans, para desarrolladores de Java.
- Ingeniería inversa - Código a modelo, código a diagrama.

1.7.2. Entorno de Desarrollo Integrado

Un Entorno Integrado de Desarrollo (IDE, Integrated Development Environment) es un sistema que facilita el trabajo del desarrollador de software, integrando sólidamente la edición orientada al lenguaje, la compilación o interpretación, la depuración, las medidas de rendimiento, la incorporación de las fuentes a un sistema de control de fuentes, etc., normalmente de forma modular (15).

Existen varios IDEs para el desarrollo de aplicaciones de escritorio en Java, entre los más utilizados se encuentran: Eclipse, Jbuilder, Sun Java Studio Enterprise y NetBeans, este último ha sido seleccionado para desarrollar el sistema debido a las características que se describen a continuación.

NetBeans IDE 7.1

El IDE NetBeans es un entorno de desarrollo integrado, una herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas. Está escrito en Java pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extender el IDE NetBeans. NetBeans es un producto libre y gratuito sin restricciones de uso. A continuación se describen algunas de las principales características de NetBeans.

Características de NetBeans (16)**:**

- **Ofrece un mejor soporte para las últimas tecnologías Java:** NetBeans IDE proporciona un soporte completo y de primera clase para las últimas tecnologías Java y las mejoras más recientes de Java antes de otros IDEs.

- **Edición de código rápida e inteligente:** Un IDE es mucho más que un editor de texto ya que proporciona plantillas de código, sugerencias de codificación y herramientas de refactorización.

- **Gestión de Plugins:** NetBeans es un IDE extensible por lo que se pueden adicionar, eliminar o actualizar una serie de funcionalidades gracias a este mecanismo.

- **Administración de proyectos fácil e inteligente:** Mantener una visión clara de las aplicaciones grandes, con miles de carpetas y archivos, y millones de líneas de código, es complicado.

NetBeans IDE proporciona diferentes vistas de los datos, a través de múltiples ventanas de proyectos.

- **Rápido desarrollo de la interfaz de usuario:** Diseño de interfaces gráficas de usuario para aplicaciones JavaEE, JavaSE y JavaME de forma rápida y sin problemas al arrastrar y colocar componentes GUI de una paleta en el editor NetBeans.

1.8. Lenguajes

1.8.1. Lenguaje de programación

Un lenguaje de programación actúa como un traductor entre el usuario y el equipo. En lugar de aprender el lenguaje nativo del equipo (conocido como lenguaje máquina), se puede utilizar un lenguaje de programación para dar instrucciones al equipo de un modo que sea más fácil de aprender y entender. Un programa especializado conocido como compilador toma las instrucciones escritas en el lenguaje de programación y las convierte en lenguaje máquina (17).

Existen varios lenguajes de programación para el desarrollo de aplicaciones de escritorio robustas y seguras tales como C++, C#, Phyton y Java, sin embargo Java será el utilizado para el desarrollo del sistema debido a las características que se muestran a continuación.

Java

Java es un lenguaje de propósito general, concurrente, simple, orientado a objetos, distribuido, interpretado, robusto, seguro, de arquitectura neutra, portátil, de alto desempeño, dinámico, basado en clases y de hilos múltiples (18).

Las características principales que ofrece Java son (18):

- **Lenguaje Simple:** Basado en el lenguaje C++, pero donde se eliminan muchas de las características de la Programación Orientada a Objetos (POO) que se utilizan esporádicamente y que creaban frecuentes problemas a los programadores. Esta eliminación de causas de errores y problemas de mantenimiento, facilita y reduce el coste del desarrollo del software.

- **Orientado a Objetos:** Java da buen soporte a las técnicas de desarrollo orientadas a objetos y en resumen a la reutilización de componentes de software.

- **Interpretado:** El compilador Java traduce cada fichero fuente de clases a código de bytes (Bytecode), que puede ser interpretado por todas las máquinas que den soporte a un visualizador que funcione con Java. Este Bytecode no es específico de una máquina determinada, por lo que no se compila y enlaza como en el ciclo clásico, sino que se interpreta.

- **Sólido:** El código Java no se quiebra fácilmente ante errores de programación. Así el relaje que existe en la declaración y manejo de tipos en C y C++ se torna en restricciones en Java, donde no es posible la conversión forzada de enteros en punteros y no ofrece soporte a los punteros que permitan saltarse reglas de manejo de tipos. Así en Java no es posible escribir en áreas arbitrarias de memoria ni realizar operaciones que corrompan el código. En resumen se eliminan muchas de las posibilidades de "trucos" que ofrecía el C y C++.

- **Portable:** Al ser de arquitectura neutral es altamente portable, pero esta característica puede verse de otra manera: Los tipos de datos estándares están igualmente implementados en todas las máquinas por lo que las operaciones aritméticas funcionarán igual en todas las máquinas.

Java fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible. Su intención es permitir que los desarrolladores de aplicaciones escriban el programa una vez y lo ejecuten en cualquier dispositivo (conocido en inglés como WORA, o "write once, run anywhere"), lo que quiere decir que el código que es ejecutado en una plataforma no tiene que ser recompilado para correr en otra. Java es, a partir de 2012, uno de los lenguajes de programación más populares en uso (18).

1.8.2. Lenguaje de modelado

Un lenguaje de modelado de objetos es un conjunto estandarizado de símbolos y de modos de disponerlos para modelar parte de un diseño de software orientado a objetos. Algunas organizaciones los usan extensivamente en combinación con una metodología de desarrollo de software para avanzar de una especificación inicial a un plan de implementación y para comunicar dicho plan a todo un equipo de desarrolladores. El uso de un lenguaje de modelado es más sencillo que la auténtica programación, pues existen menos medios para verificar efectivamente el funcionamiento adecuado del modelo. Esto puede

suponer también que las interacciones entre partes del programa den lugar a sorpresas cuando el modelo ha sido convertido en un software funcional (19).

Para el modelado de sistemas y negocios existen varios lenguajes de modelado como: Business Process Modeling Notation (BPMN) y Unified Modeling Language (UML), este último ha sido seleccionado debido a las siguientes características.

Leguaje Unificado de Modelado (UML)

El Lenguaje Unificado de Modelado es un lenguaje gráfico para visualizar, especificar, construir y documentar cada una de las partes que comprende el desarrollo de software. UML entrega una forma de modelar cosas conceptuales como lo son procesos de negocio y funciones de sistema, además de cosas concretas como lo son escribir clases en un lenguaje determinado, esquemas de base de datos y componentes de software reusables (19).

Para una mejor comprensión de que es en realidad UML, basta solo con analizar cada una de las palabras que lo componen por separado (20):

Lenguaje: el UML es precisamente un lenguaje, lo que implica que este cuenta con una sintaxis y una semántica. Por lo tanto al modelar un concepto en UML, existen reglas sobre cómo deben agruparse los elementos del lenguaje y el significado de esta agrupación.

Modelado: el UML es visual, mediante su sintaxis se modelan distintos aspectos del mundo real, que permiten una mejor interpretación y entendimiento de este.

Unificado: unifica varias técnicas de modelado en una única.

UML tiene como objetivo fundamental entregar un material de apoyo que le permita al lector poder definir diagramas propios como también poder entender el modelamiento de diagramas ya existentes (20).

1.9. Librerías

En términos informáticos una librería o biblioteca es un conjunto de subprogramas utilizados para desarrollar aplicaciones. Las bibliotecas contienen código y datos, que proporcionan servicios a

programas independientes, es decir, pasan a formar parte de estos. Esto permite que el código y los datos se compartan y puedan modificarse de forma modular (21).

BioJava

BioJava es un proyecto de código abierto dedicado a proveer un marco de Java para el procesamiento de datos biológicos. Se proporciona rutinas analíticas y estadísticas, analizadores de formatos de archivo comunes y permite la manipulación de secuencias y estructuras 3D. El objetivo del proyecto BioJava es facilitar el desarrollo rápido de aplicaciones bioinformáticas, esta librería se encuentra actualmente bajo licencia GPL (22). Actualmente Biojava mantiene en desarrollo dos versiones diferentes las cuales son descritas a continuación.

Biojava 1.x: Entre otras funcionalidades, esta versión está enfocada hacia la secuenciación, así como la gestión de ficheros generados por equipos de este tipo como los SCF y ABI. También se puede realizar lectura y escritura de secuencias en formato fasta y construir Cromatogramas a partir de ficheros tipo SCF.

Biojava 3.x: Entre otras funcionalidades, esta versión también hace parte del trabajo de secuenciación pero esta vez dirigido hacia los ficheros en formato fastq en cualquiera de sus variantes (Sanger, Illumina, Solexa), los cuales no se pueden leer a través de Biojava 1.x. También con esta versión se puede visualizar el alineamiento de secuencias.

Durante el desarrollo del sistema, serán empleadas ambas versiones de Biojava con el objetivo de dar soporte a funcionalidades como la visualización de cromatogramas, la lectura de varios formatos provenientes de equipos de secuenciación de última generación y la captura de sus valores de calidad, entre otras funciones.

Conclusiones del capítulo

Luego de haber realizado un estudio de las herramientas y tecnologías disponibles, se han seleccionado aquellas que brindan un conjunto de características que se acoplan de la mejor manera a la problemática en cuestión, o sea al sistema a desarrollar. Como metodología de desarrollo fue seleccionada OpenUP, el sistema será implementado en el lenguaje de programación Java utilizando NetBeans 7.1 como Entorno

de Desarrollo Integrado, los diagramas serán modelados en Visual Paradigm 8.0 utilizando UML como lenguaje de modelado de objetos. Como ensamblador comparativo de secuencias de ácidos nucleicos se empleará Mosaik 1.032. Todas estas herramientas serán utilizadas bajo el patrón MVC el cual facilitará el desarrollo de una herramienta robusta, extensible y completamente libre.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

Introducción

En el presente capítulo se realiza una propuesta del sistema a desarrollar, representando los artefactos definidos por la metodología de desarrollo OpenUP. Se presenta el modelo de dominio con la descripción de cada objeto, la captura de los requisitos funcionales, la definición de los requisitos no funcionales y el diagrama de casos de uso del sistema, además de la descripción de los casos de uso.

2.1. Modelo de Dominio

El modelo de dominio (MD) se aplica cuando no es posible aplicar el modelo de negocio. En este modelo se representan las clases fundamentales del negocio y sus relaciones. El mismo representa una parte visual de las clases y sus interacciones con el entorno del proyecto (23). El diagrama que se presenta a continuación muestra los objetos utilizados para el desarrollo del sistema.

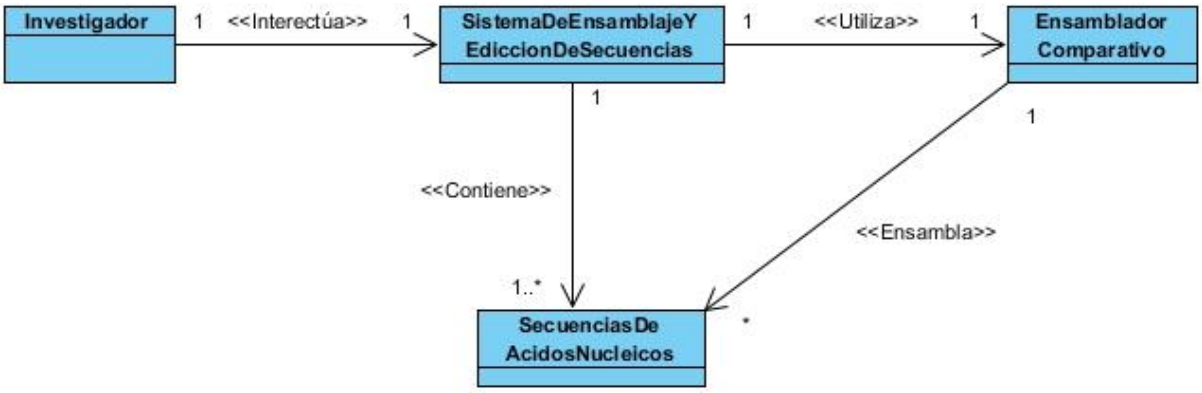


Figura 2.1 Diagrama de Modelo de dominio.

2.1.1. Definición de las Clases de Modelo del Dominio

Investigador: Usuario que interactúa con el sistema.

SistemaDeEnsamblajeYEdicionDeSecuencias: Sistema a desarrollar. Interfaz en la que el investigador crea nuevos proyectos para el ensamblaje y edición de secuencias de ácidos nucleicos.

EnsambladorComparativo: Sistema ensamblador de secuencias de ácidos nucleicos de tipo comparativo, el cual será utilizado para realizar el ensamblaje.

SecuenciasDeÁcidosNucleicos: Conjunto de secuencias de ácidos nucleicos obtenidas a través de ficheros importados en varios formatos.

2.2. Requisitos del sistema.

Los requisitos de software son aquellas funcionalidades que debe realizar o alcanzar un sistema (24). A continuación se describen los mismos separados en Requisitos Funcionales y Requisitos no Funcionales.

2.2.1. Requisitos Funcionales:

Los requisitos funcionales surgen de las entrevistas realizadas al cliente, los cuales representan las funcionalidades que debe presentar el software, a continuación se enumeran los mismos y se realiza una breve descripción de cada uno de ellos.

RF1: Importar Secuencia de Ácidos Nucleicos.

- Consiste en cargar un fichero el cual contiene una o varias secuencias, dependiendo del formato que posea.

RF2: Mostrar Secuencia de Ácidos Nucleicos.

- Consiste en mostrar la cadena correspondiente a una secuencia previamente importada.

RF3: Modificar Secuencia de Ácidos Nucleicos.

- Permite al usuario realizar cambios en las secuencias, así como la inserción, eliminación y cambio de bases o gaps.

RF4: Eliminar Secuencia de Ácidos Nucleicos.

- Permite al usuario eliminar completamente una secuencia del sistema.

RF5: Modificar Parámetros de Ensamblaje.

- Permite al usuario modificar los parámetros que se utilizan en el ensamblaje de las secuencias.

RF6: Ensamblar Secuencias de Ácidos Nucleicos.

- Permite al usuario realizar el ensamblaje comparativo de secuencias.

RF7: Mostrar Cromatogramas.

- Permite al usuario visualizar el cromatograma de una secuencia obtenida a partir de la secuenciación.

RF8: Exportar Secuencias de Ácidos Nucleicos.

- Permite al usuario exportar una secuencia hacia un directorio determinado.

RF9: Establecer Intervalos de Confianza.

- Permite al usuario establecer un intervalo de confianza para evaluar el porcentaje de calidad de una secuencia.

2.2.2. Requisitos no Funcionales

Por otro lado los Requisitos no Funcionales (RNF) son requisitos que imponen restricciones en el diseño o la implementación. Son propiedades o cualidades que el producto debe tener y además importantes para que los usuarios valoren las características no funcionales del sistema (24). A continuación se muestran los RNF identificados durante la investigación.

RNF1: Legales

Se usarán herramientas de software libre, o que funcionen bajo las licencias GNU/GPL, por lo que el sistema estará basado también sobre los términos de la licencia GNU/GPL.

RNF2: Software

Se debe disponer de las siguientes aplicaciones para hacer uso del sistema.

- Windows XP o Windows 7.
- Mosaik 1.032
- JDK 7 o superior.

RNF3: Hardware

Para un desempeño eficaz del sistema las estaciones de trabajo donde se instalará deben poseer una memoria RAM de 2GB o superior, además de un procesador Celeron a 2.5 GHz o superior y una capacidad de almacenamiento mínima de 40 GB.

RNF4: Apariencia

El sistema tendrá un ambiente sencillo y similar al que presenta el Sequencher lo que facilitará a los investigadores del IPK su interacción con el mismo.

RNF5: Escalabilidad

Para el desarrollo del sistema fueron empleadas herramientas y tecnologías completamente libres, además de su código libre, al cual se le pueden agregar, eliminar u optimizar funcionalidades con el objetivo de aumentar su uso y rendimiento.

2.3. Definición de los actores del sistema

Un actor es una entidad externa que interactúa con el sistema participando en un Caso de Uso (CU). Los actores pueden ser personas, otros ordenadores o eventos externos.

Los actores no representan a personas físicas o a sistemas, sino su rol. Esto significa que cuando una persona interactúa con el sistema de diferentes maneras (asumiendo diferentes papeles), estará representando varios actores (25).

Actor	Definición
Investigador	Representa el usuario final del sistema, el cual podrá realizar todas las operaciones sin restricción alguna en el sistema.
Mosaik	Sistema para el ensamblaje comparativo de secuencias de ácidos nucleicos.

Tabla 3: Definición de los actores del sistema.

2.4. Definición de los casos de uso del sistema

Un caso de uso es una secuencia de interacciones entre un sistema y alguien o algo que usa alguno de sus servicios. Los casos de uso representan requisitos funcionales por lo que representan funcionalidades para el sistema.

2.4.1. Patrones de Caso de Uso

Con el objetivo de lograr un sistema más fácil de comprender y mantener, se aplican patrones en el diseño de los casos de uso, los cuales permiten construir un producto mucho más enfocado hacia el diseño y las técnicas de alta calidad. A continuación se describe el patrón utilizado.

CRUD: Este patrón se utiliza en los casos donde se quiere realizar altas, bajas, cambios y consultas a alguna entidad del sistema. Su nombre es un acrónimo de las palabras en inglés Create (Crear), Read (Leer), Update (Actualizar) y Delete (Eliminar).

2.4.2. Listado de Casos de Uso

CU1: Gestionar Secuencias de Ácidos Nucleicos.

- RF1: Importar Secuencias de Ácidos Nucleicos.
- RF2·: Mostrar Secuencias de Ácidos Nucleicos.
- RF3: Modificar Secuencias de Ácidos Nucleicos.
- RF4: Eliminar Secuencias de Ácidos Nucleicos.

CU2: Modificar Parámetros de Ensamblaje.

- RF5: Modificar Parámetros de Ensamblaje.

CU3: Ensamblar Secuencias de Ácidos Nucleicos.

- RF6: Ensamblar Secuencias de Ácidos Nucleicos.

CU4: Mostrar Cromatogramas

- RF7: Mostrar Cromatogramas.

CU5: Exportar Secuencias de Ácidos Nucleicos.

- RF8: Exportar Secuencias de Ácidos Nucleicos.

CU6: Establecer Intervalos de confianza

- RF9: Establecer Intervalos de confianza.

2.5. Diagrama de Casos de Uso del Sistema

El diagrama de casos de uso representa la forma de como un actor opera con el sistema en desarrollo, además de la forma, tipo y orden de como los elementos interactúan.

Los diagramas de casos de uso describen las relaciones y las dependencias entre un grupo de casos de uso y los actores participantes en el proceso.

Los diagramas de casos de uso no están pensados para representar el diseño y no pueden describir los elementos internos de un sistema. 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. En otras palabras, los diagramas de casos de uso describen qué es lo que debe hacer el sistema, pero no como (25).

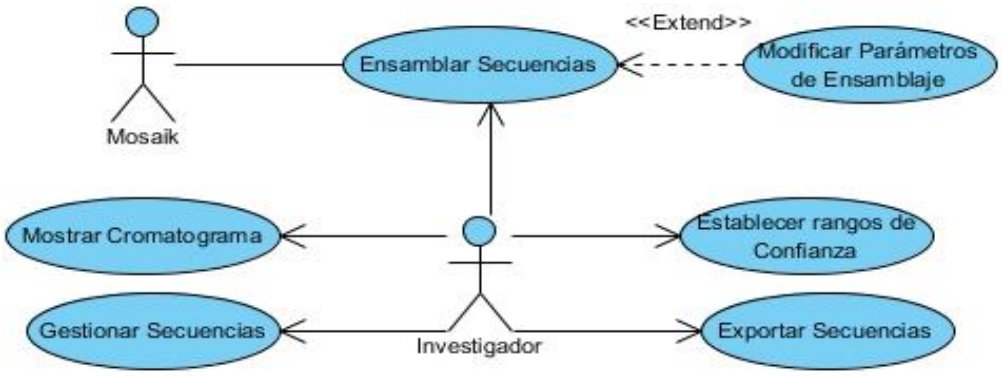


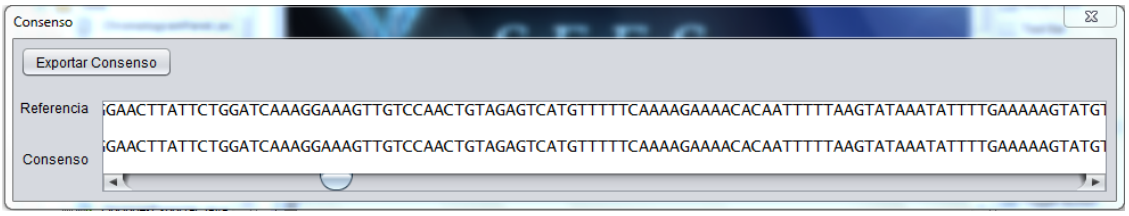
Figura 5: Diagrama de Casos de Uso del Sistema.

2.5.1. Descripción de Casos de Uso

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

A continuación se realiza la descripción del CU Ensamblar Secuencias de Ácidos Nucleicos, el cual ha sido identificado como el CU más crítico del sistema. El resto de las descripciones de los CU se pueden encontrar en el *Anexo 1.*

Descripción del CU Ensamblar Secuencias de Ácidos Nucleicos.

<table>
<tr><td>Caso de Uso:</td><td colspan="2">Ensamblar Secuencias de Ácidos Nucleicos.</td></tr>
<tr><td>Actores:</td><td colspan="2">Investigador, Mosaik</td></tr>
<tr><td>Resumen:</td><td colspan="2">El CU se inicia cuando el investigador selecciona la opción “Ensamblar Secuencias”, luego Mosaik realiza el ensamblaje de las secuencias seleccionadas y se visualiza el consenso en el Sistema, finalizando así el caso de uso.</td></tr>
<tr><td>Precondiciones:</td><td colspan="2">Seleccionar una secuencia como referencia</td></tr>
<tr><td>Referencias</td><td colspan="2">RF6</td></tr>
<tr><td>Prioridad</td><td colspan="2">Crítico.</td></tr>
<tr><td colspan="3">Flujo Normal de Eventos</td></tr>
<tr><td colspan="2">Acción del Actor</td><td>Respuesta del Sistema</td></tr>
<tr><td colspan="2">1. El Investigador selecciona las secuencias a ensamblar y la opción “Ensamblar Secuencias”.</td><td>2. El sistema ensambla las secuencias seleccionadas.
3. El sistema muestra la Secuencia consenso obtenida a partir del ensamblaje.</td></tr>
<tr><td colspan="3">Prototipo de interfaz
 
</td></tr>
<tr><td colspan="3">Flujos Alternos</td></tr>
</table>

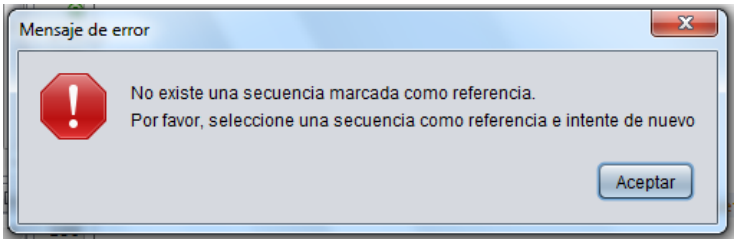
<table>
<tr><th>Acción del Actor</th><th>Respuesta del Sistema</th></tr>
<tr><td></td><td>3.1. Muestra el mensaje “No existe una Secuencia marcada como referencia en la selección”.</td></tr>
<tr><td colspan="2">Prototipo de Interfaz
 
</td></tr>
<tr><td>Poscondiciones</td><td>Se obtiene la Secuencia Consenso.</td></tr>
</table>

Tabla 4: Descripción del CU Ensamblar Secuencias de Ácidos Nucleicos.

Conclusiones del Capítulo

En este capítulo se dio una visión general del sistema a construir. Con el objetivo de esclarecer como se desarrolla todo el proceso se definió el modelo de dominio donde se representaron las entidades que participan en la solución propuesta y sus relaciones entre sí. Además se capturaron 9 requisitos funcionales a partir de entrevistas con el cliente los cuales se agruparon en 6 casos de uso, además se plantearon 5 requisitos no funcionales y se identificaron dos actores. De este modo quedan sentadas las bases para continuar con las siguientes fases del proceso de desarrollo.

CAPÍTULO 3: ANÁLISIS Y DISEÑO DEL SISTEMA

Introducción

Luego de haber analizado las características del sistema a desarrollar, se procede a realizar el análisis y diseño de la propuesta planteada en el capítulo anterior. En este capítulo se definirán los patrones de arquitectura y diseño que regirán el desarrollo del sistema. Se presentarán los diagramas de clases del diseño y de interacción, los cuales representan al sistema desde un punto de vista ingenieril para proseguir con la implementación del mismo.

3.1. Arquitectura de Software

La arquitectura de software es una forma de representar sistemas complejos mediante el uso de la abstracción. Independientemente de la metodología implementada, la intención es obtener una arquitectura con la documentación necesaria, y asegurar que el sistema cumple con los requisitos y la funcionalidad que espera el usuario, además de los atributos de calidad asociados que deben cumplirse, y que dirigen las decisiones al momento de la construcción de la arquitectura del sistema (26).

La arquitectura de software puede considerarse entonces como el “puente” entre los requisitos del sistema y la implementación. Las actividades que culminan en la definición de la arquitectura pueden ubicarse en las fases tempranas del ciclo de desarrollo del sistema. Desde esta perspectiva, la arquitectura constituye un artefacto de la actividad de diseño, que servirá de medio de comunicación entre los miembros del equipo de desarrollo, los clientes y usuarios finales, dado que contempla los aspectos que interesan a cada uno. Además, pasa a ser la base del diseño del sistema a desarrollar, debido a que es usada como guía para el resto de las tareas del desarrollo.

3.1.1. Patrón Arquitectónico

Modelo Vista Controlador (MVC)

Uno de los patrones que ha ganado un gran auge en los últimos años debido a los resultados que se pueden obtener mediante su uso, es el Modelo-Vista-Controlador (MVC) cuyo objetivo es dividir una

aplicación interactiva en tres componentes. El modelo contiene la información central y los datos. Las vistas despliegan información al usuario. Los controladores capturan la entrada del usuario. Las vistas y los controladores constituyen la interfaz del usuario (27).

Dicho patrón fue diseñado para reducir el esfuerzo de programación necesario en la implementación de sistemas garantizando un gran desacople entre sus partes. Sus características principales están dadas por el hecho de que, el Modelo, las Vistas y los Controladores se tratan como entidades separadas; esto hace que cualquier cambio producido en el Modelo se refleje automáticamente en cada una de las Vistas.

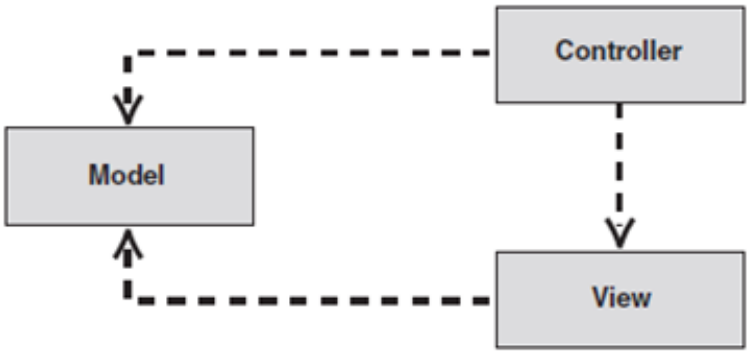


Figura 6: Representación del Patrón Modelo-Vista-Controlador.

Este modelo de arquitectura presenta varias ventajas

- Separación clara entre los componentes de un programa; lo cual permite su implementación por separado.
- Interfaz de Programación de Aplicaciones API (Aplication Programming Interface) muy bien definida; cualquiera que use el API, podrá reemplazar el Modelo, la Vista o el Controlador, sin aparente dificultad.
- Conexión entre el Modelo y sus Vistas dinámica; se produce en tiempo de ejecución, no en tiempo de compilación.

A continuación se presenta la descripción de cada paquete representado en la *Figura 7.*

Modelo: Los modelos están compuestos por entidades, las cuales dominan la información que se maneja en el sistema, estos encapsulan los datos y las funcionalidades. El paquete Modelo, cuenta con tipos de datos: Secuencia SecuenciaAbstracta, SecuenciaSimple, SecuenciaCompuesta, SecuenciaFasta,

SecuenciaSCF, SecuenciaABI y SecuenciaFastQ, donde SecuenciaAbstracta es la clase base la cual tiene como hijas a SecuenciaSimple y SecuenciaCompuesta, donde las Secuencias de tipo SecuenciaSimple (SecuenciaFasta, SecuenciaSCF, SecuenciaABI), tienen una sola secuencia y las de tipo SecuenciaCompuesta (SecuenciaFastQ) tienen varias.

Vista: El paquete vista contiene la interfaz principal del proyecto, además de un conjunto de ventanas con las que interactúa el investigador, a través de las cuales se pueden obtener datos relacionados con las secuencias así como realizar cambios sobre las mismas.

Controlador: El paquete controlador contiene la clase controladora del proyecto, esta clase contiene las secuencias cargadas en el Sistema. La clase controladora se encarga a la vez de actualizar el modelo correspondiente con la petición realizada.

Además de los paquetes que representan la arquitectura MVC se representaron subsistemas que influyen en el funcionamiento del software.

Mosaik: Sistema para el ensamblaje de secuencias de ácidos nucleicos. Este sistema recibe dos ficheros, uno con el genoma de referencia en formato “fasta” y otro con las lecturas a ensamblar en formato “fastq”. A partir de esta entrada Mosaik genera como salida un fichero en formato “ace”, el cual contiene la cadena consenso.

Biojava: Librería usada para la lectura de ficheros generados por equipos de secuenciación, la construcción de cromatogramas y la obtención de la cadena reversa y complementaria de una secuencia.

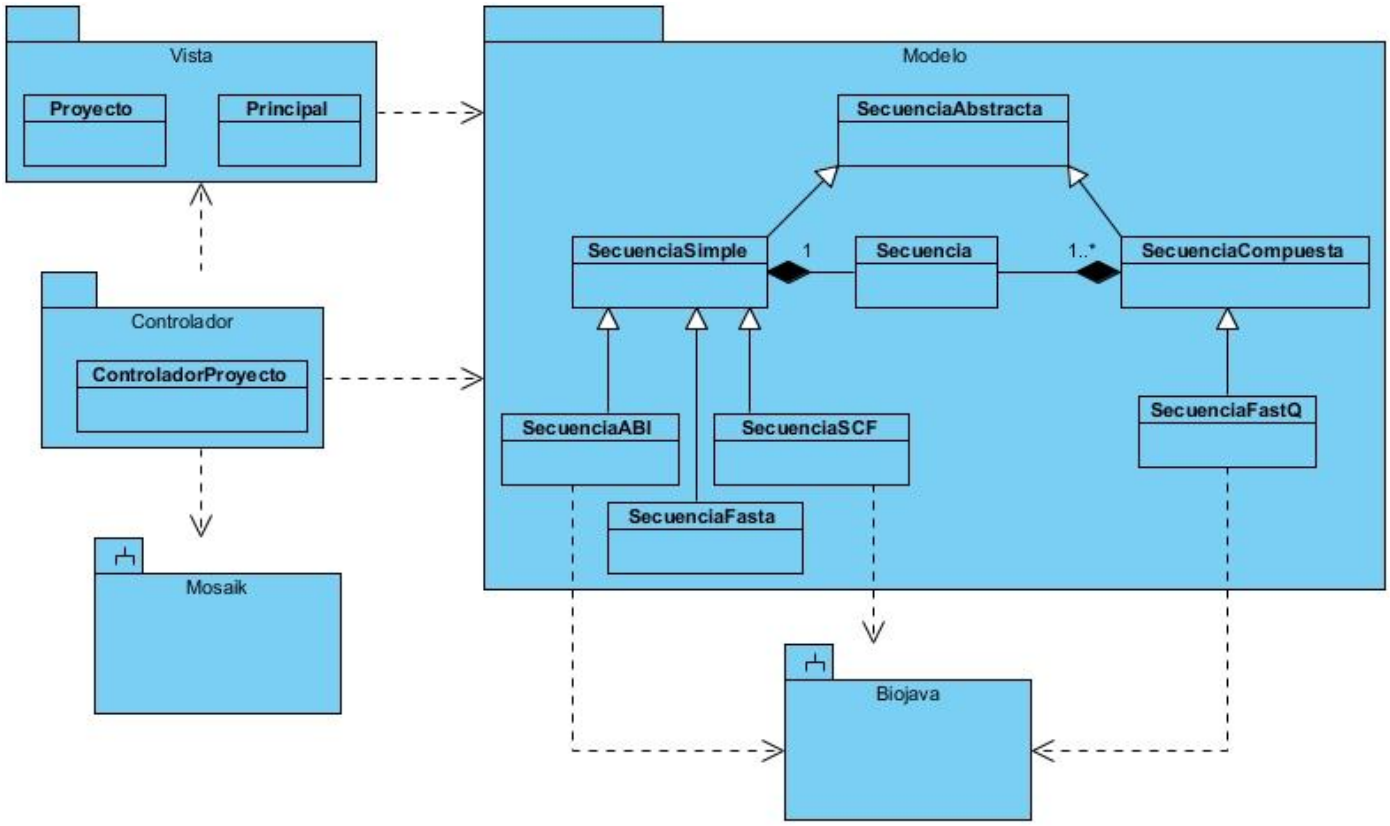


Figura 7: Vista Lógica del Sistema.

Al incorporar el modelo de arquitectura MVC a un diseño, las piezas de un programa se pueden construir por separado y luego unirlas entiempo de ejecución. Si uno de los componentes, posteriormente, se observa que funciona mal, puede reemplazarse sin que las otras piezas se vean afectadas.

3.1.2. Patrones de Diseño

Un patrón de diseño provee un esquema para refinar los componentes de un sistema de software, o las relaciones entre ellos. Describe la estructura comúnmente recurrente de los componentes en comunicación, que resuelve un problema general de diseño en un contexto particular (27).

Una muy buena práctica a la hora de desarrollar aplicaciones orientadas a objetos es el uso de los patrones GRASP. Dichos patrones describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. GRASP es un acrónimo que significa

General Responsibility Assignment Software Patterns. El nombre se eligió para indicar la importancia de captar estos principios, si se quiere diseñar eficazmente el software orientado a objetos.

Patrón GRASP	Función dentro del sistema
Experto	Este patrón asigna una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad. Dentro del sistema se aplica para asignar una responsabilidad al experto en información, por ejemplo, todas las operaciones a realizar sobre una secuencia se encuentran en la clase “Secuencia”.
Creador	Este patrón explica que clase es la encargada de crear objetos, en determinados escenarios de ejecución. Dentro del sistema se aplica para asignar la creación de objetos a la clase adecuada, por ejemplo, la clase principal de proyecto es la encargada de crear los objetos de cada secuencia cargada en el sistema.
Bajo Acoplamiento	Este patrón explica como asignar una responsabilidad para mantener un bajo acoplamiento. Se aplica para dar soporte a una dependencia escasa y a un aumento de la reutilización. Dentro del sistema está presente en las clases del modelo, las cuales son completamente independientes de la vista y el controlador, lo que garantiza una amplia reutilización de las mismas.
Alta Cohesión	El objetivo de este patrón es asignar una responsabilidad de modo que la cohesión siga siendo alta. Explica cómo mantener la complejidad dentro de límites manejables. Dentro del sistema se aplica para mantener una alta coherencia entre las clases del mismo, por ejemplo, cada vista accede a los datos del controlador a través de una instancia recibida como parámetro.

Tabla 5: Patrones GRASP y sus funciones dentro del sistema.

Por otro lado se encuentran los patrones GoF (Gang of Four), los cuales pueden ser vistos como complementos de los patrones GRASP y en ocasiones se puede encontrar una contraposición entre este tipo de patrones, e incluso, podría inferirse que algunos patrones GoF son variantes de los patrones GRASP, es por ello que la decisión de utilizar uno u otro debe tomarse con precaución y aplicarse solo en el ámbito necesario. Los patrones GoF proponen soluciones a ciertos problemas a la hora de construir softwares complejos. Para el desarrollo del Sistema, fueron utilizados 2 de estos patrones, los cuales se describen a continuación.

Patrón	Responsabilidad
Fachada	Este patrón proporciona una interfaz unificada de alto nivel que, representando a todo un subsistema, facilite su uso. La "fachada" satisface a la mayoría de los clientes, sin ocultar las funciones de menor nivel a aquellos que necesiten acceder a ellas. Dentro del sistema el patrón fachada se utiliza para que el usuario acceda a la gran mayoría de las interfaces del sistema a través de la interfaz principal del proyecto. Este patrón especifica que puede darse el caso de vistas a las que se deba acceder a través de otras vistas, como es el caso de las opciones de exportación de secuencias.
Método Plantilla	Este patrón define el esqueleto de un algoritmo para una operación, dejando para sus subclases la capacidad de redefinir el funcionamiento de los pasos de este algoritmo, siempre y cuando la estructura del mismo permanezca intacta. Dentro del sistema el método plantilla se emplea en las clases del modelo para llenar la cadena de una secuencia, varios datos que se llenan de igual forma para todas las secuencias se llenan en la clase padre, los demás datos se llenan en la clase hija correspondiente.

Tabla 6: Patrones GoF y sus funciones dentro del sistema.

3.2. Modelo del Diseño

El diagrama de clases se utiliza para describir la estructura de un sistema, mostrando sus clases, sus atributos y sus relaciones entre ellos. Los diagramas de clases son utilizados durante el proceso de análisis y diseño de los sistemas. A continuación se presenta el diagrama de clases del diseño del CU "Ensamblar Secuencias". Los restantes diagramas de clases del diseño se encuentran en la planilla Modelo de Diseño.

En el presente diagrama se muestra una clase interfaz de usuario "Proyecto", la cual se encarga de mostrar otra clase con interfaz "Parámetros" en caso de que se desee modificar algún parámetro de ensamblaje. Ambas clases pertenecen al paquete "Vista" dentro de la arquitectura MVC. Por otro lado se tiene la clase "ControladorProyecto", la cual se encarga de gestionar las secuencias cargadas en el sistema y en este caso de llamar al sistema Mosaik facilitándole la secuencia marcada como referencia y las lecturas a ensamblar, para que finalmente se muestre una nueva interfaz "Ensamblaje", la cual representa la secuencia consenso resultante del ensamblaje.

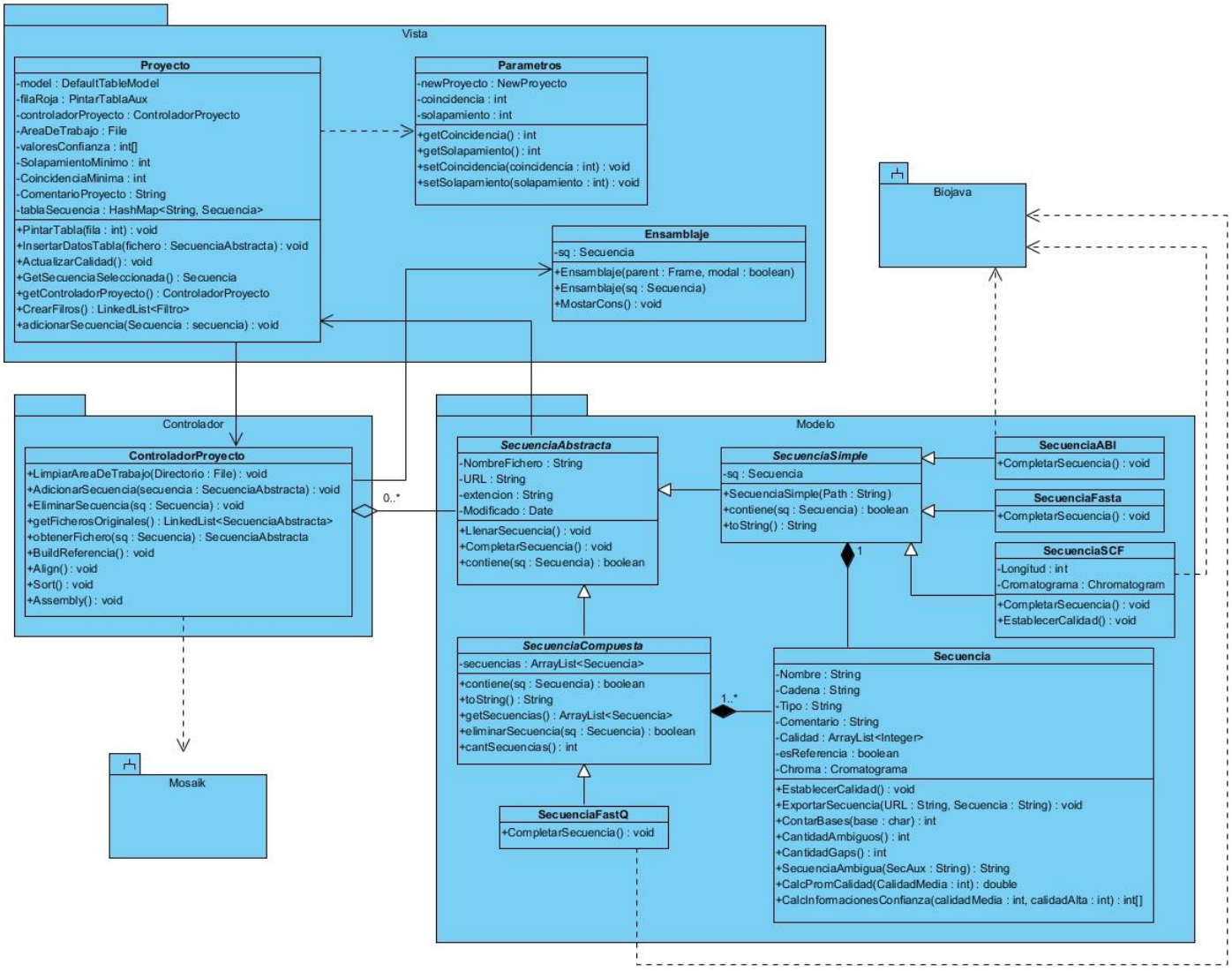


Figura 8: Diagrama de clases del diseño del CU “Ensamblar Secuencias”.

3.3. Diagrama de Secuencia

El diagrama de secuencia es un esquema conceptual que permite representar el comportamiento de un sistema, para lo cual emplea la especificación de los objetos que se encuentran en un escenario y la secuencia de mensajes intercambiados entre ellos, con el fin de llevar a cabo una transacción del sistema. A continuación se presenta el Diagrama de Secuencia de CU “Ensamblar Secuencias”.

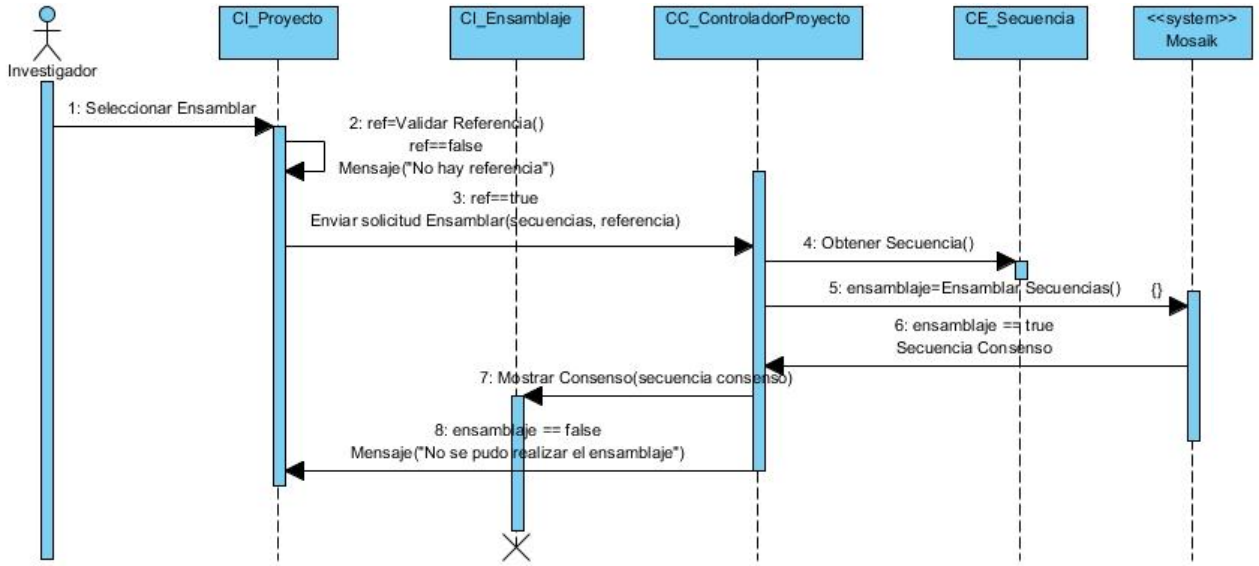


Figura 9: Diagrama de Secuencia del CU “Ensamblar Secuencias”.

Los restantes diagramas de secuencias se encuentran en la planilla Modelo de Diseño.

Conclusiones del Capítulo

En el presente capítulo se definieron los diagramas de interacción del sistema para el CU de mayor impacto en la arquitectura: “Ensamblar Secuencias”. Se tuvo en cuenta la utilización de varios patrones arquitectónicos y de diseño, los cuales posibilitaron una mejor arquitectura y una mayor robustez. Dentro de estos patrones se seleccionaron el MVC, los patrones experto, creador, bajo acoplamiento y alta cohesión de los GRASP, y así como los patrones fachada y método plantilla de los GoF.

Por último se definió el diagrama de clases del diseño para el CU: “Ensamblar Secuencias”, en la elaboración del mismo se dividió el sistema en paquetes los cuales representan la arquitectura en capas MVC, obteniendo una capa para la presentación (Vista), una para la lógica de aplicación (Controlador) y otra para la representación de las entidades (Modelos).

CAPÍTULO 4: IMPLEMENTACIÓN Y PRUEBAS

Una vez realizado el análisis y diseño del sistema, entonces ya se cuenta con los detalles necesarios para proceder con la implementación del mismo y una vez concluida esta, la verificación del cumplimiento de los requisitos funcionales a través de las pruebas de software. En el presente capítulo se muestra el modelo de implementación, el cual se representa a través de un diagrama de componentes y el modelo de despliegue del sistema, también se describen las pruebas realizadas y los resultados obtenidos.

4.1. Modelo de Implementación

El Modelo de Implementación está comprendido por un conjunto de componentes y subsistemas que constituyen la composición física de la implementación del sistema. Entre los componentes podemos encontrar datos, archivos, ejecutables, código fuente y los directorios. Fundamentalmente, se describe la relación que existe desde los paquetes y clases del modelo de diseño a subsistemas y componentes físicos. Este artefacto describe como se implementan los componentes, congregándolos en subsistemas organizados en capas y jerarquías, y señala las dependencias entre estos (28).

4.1.1. Diagrama de Componentes

Un Diagrama de Componentes muestra las interacciones y relaciones de los componentes de un modelo. Entendiéndose como componente a una clase de uso específico, que puede ser implementada desde un entorno de desarrollo, ya sea de código binario, fuente o ejecutable; dichos componentes poseen tipo, que indican si pueden ser útiles en tiempo de compilación, enlace y ejecución. Este tipo de diagrama se representa a través de componentes unidos mediante relaciones de dependencia. A continuación se muestra el Diagrama de Componentes del CU “Ensamblar Secuencias”.

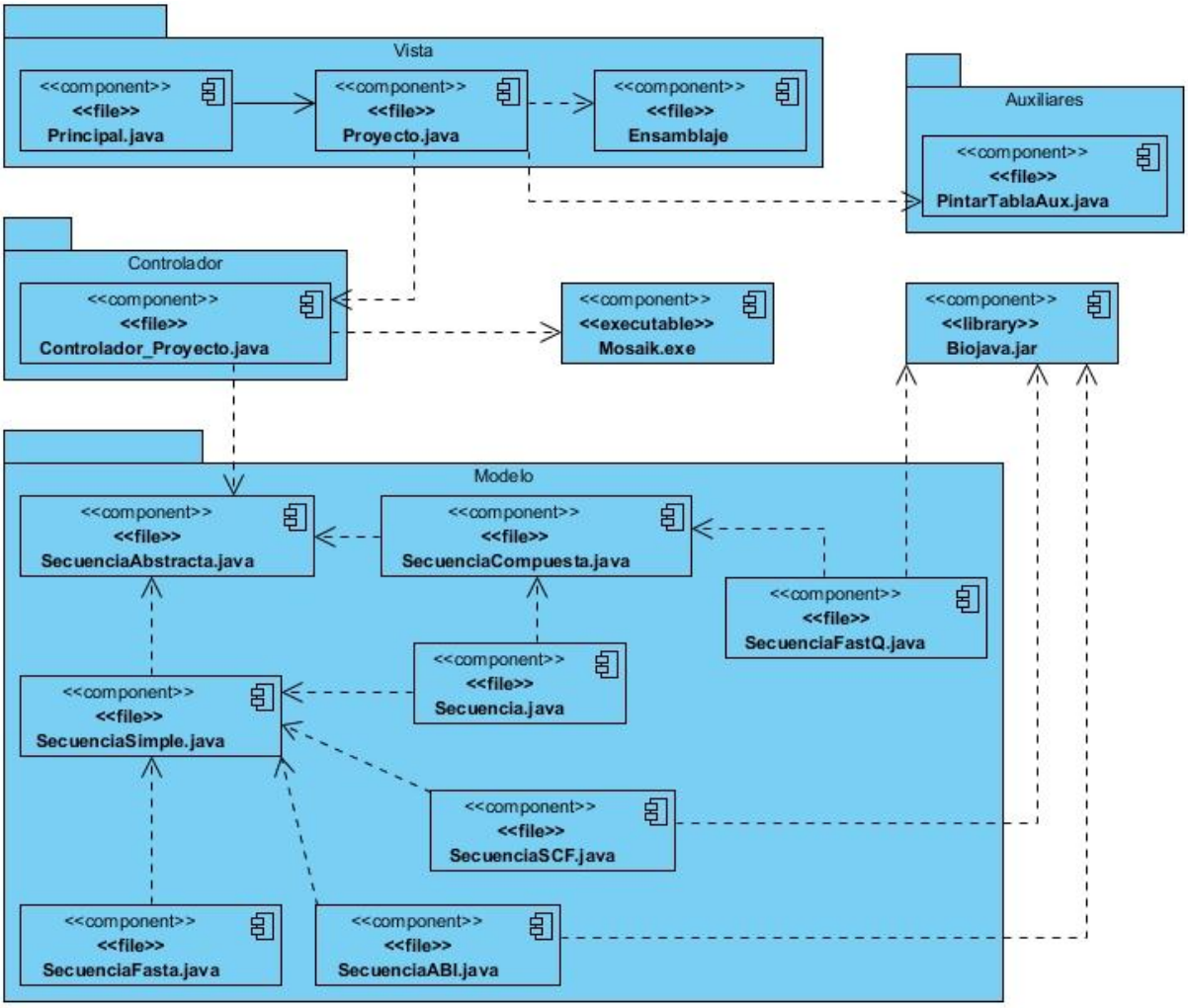


Figura 10: Diagrama de Componentes de CU “Ensamblar Secuencias”.

Los componentes constituyen partes físicas de un sistema, los cuales interactúan entre sí para lograr un flujo que cumpla con un requisito determinado. A continuación se describen los componentes mostrados en el diagrama de la *Figura 10.*

Componente	Propósito
Paquete Vista	Contiene todas las vistas con las que interactúa el usuario.
Paquete Modelo	Contiene las clases que representan entidades dentro del sistema, las cuales contienen toda la información referente a las secuencias cargadas en el mismo.

Paquete Controlador	Contiene la clase controladora del sistema, encargada de atender las peticiones del usuario.
Paquete Auxiliares	Contiene un conjunto de clases que intervienen en el desempeño del sistema.
Principal.java	Esta clase contiene la interfaz principal del proyecto, donde se realizan todas las operaciones dentro del sistema.
Proyecto.java	Contiene la tabla que muestra las secuencias cargadas en el sistema.
Ensamblaje.java	Muestra el resultado del ensamblaje, dígase secuencia consenso y secuencia referencia.
PintarTablaAux.java	Repinta la tabla y almacena la fila de la secuencia marcada como referencia.
Controlador_Proyecto.java	Clase controladora del sistema, almacena las secuencias cargadas y se encarga de la gestión de las mismas, así como de dar cumplimiento a las peticiones del investigador.
Mosaik.exe	Sistema para el ensamblaje de secuencias de ácidos Nucleicos, el cual recibe la referencia y las lecturas y devuelve el consenso.
Biojava.jar	Librería utilizada para la lectura y gestión de ficheros obtenidos a partir de máquinas de secuenciación.
SecuenciaAbstracta.java	Clase base de todos los tipos de secuencias.
SecuenciaCompuesta.java	Clase para gestionar la información de ficheros con múltiples secuencias.
SecuenciaFastQ.java	Entidad para almacenar los datos de una secuencia compuesta de tipo SecuenciaFastQ.
SecuenciaSimple.java	Clase para gestionar la información de ficheros con una sola secuencia.
SecuenciaFasta.java	Entidad para almacenar los datos de una secuencia de tipo SecuenciaFasta
SecuenciaABI.java	Entidad para almacenar los datos de una secuencia de tipo SecuenciaABI.
SecuenciaSCF.java	Entidad para almacenar los datos de una secuencia de tipo SecuenciaSCF.
Secuencia.java	Clase para almacenar los datos de una secuencia.

Tabla 7: Descripción de los componentes del CU “Gestionar Secuencias”.

4.1.2. Modelo de Despliegue

Un diagrama de despliegue es un grafo de nodos unidos por conexiones. Un nodo puede contener instancias de componentes de software y objetos. En general un nodo es una unidad de computación de algún tipo, por ejemplo un sensor o un dispositivo. Las instancias de componentes de software pueden estar unidas por relaciones de dependencia, estas relaciones pueden ser múltiples ya que un componente

puede tener más de una interfaz de comunicación (29). A continuación se muestra el modelo de despliegue del sistema.

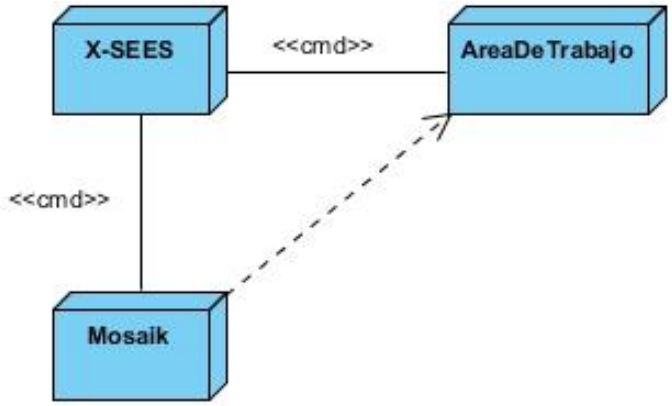


Figura 11: Modelo de Despliegue.

Para el despliegue de X-SEES se requiere una computadora, en la que se encuentre el programa Mosaik en el directorio C:\, la cual permita copiar ficheros en el área de trabajo (Home del usuario), a través de dicho directorio se realizará el intercambio de datos entre Mosaik y X-SEES.

Nodo	Descripción
X-SEES	Representa el sistema con la GUI para el ensamblaje y edición de secuencias, el cual una vez realizada la petición de "Ensamblar Secuencias" copia las secuencias a ensamblar hacia el área de trabajo y ejecuta a Mosaik a través de la línea de comandos (cmd).
Mosaik	Sistema que ensambla las secuencias que se encuentran en el área de trabajo
Área de Trabajo	Representa un directorio de la computadora (directorio home del usuario), hacia el cual se copian las secuencias que van a ser ensambladas.

Tabla 8: Descripción de los nodos del Modelo de Despliegue.

4.2. Pruebas de Software

Una vez generado el código fuente es necesario probar el software para descubrir y corregir la mayor cantidad de errores posible antes de entregarlo al cliente. El objetivo fundamental de las pruebas de software es diseñar casos de prueba que tengan una alta probabilidad de encontrar errores (24).

El único instrumento adecuado para determinar el grado de la calidad de un software es el proceso de pruebas. En este proceso se ejecutan pruebas dirigidas a componentes o al sistema en su totalidad, con el objetivo de medir el grado en que el sistema cumple con los requisitos. En las pruebas se usan casos de prueba, especificados de forma estructurada mediante Técnicas de prueba. El proceso de pruebas, sus objetivos y los métodos y técnicas usados se describen en el plan de prueba (30).

Las pruebas definen el grado de aceptación del sistema que pueden tener los clientes del mismo, e incluso determinar si lo aceptan o no. Para ser más eficaces, las pruebas deberían ser realizadas por un equipo independiente, esto garantiza pruebas con alta probabilidad de encontrar errores, que es el objetivo principal de las pruebas. Existen dos tipos de pruebas, las de caja blanca y las de caja negra, las cuales se describen y ejecutan a lo largo de este capítulo (24).

4.2.1. Pruebas de Caja Blanca

Las pruebas de caja blanca se basan en el minucioso examen de los detalles procedimentales. Se comprueban los caminos lógicos del software proponiendo casos de prueba que examinen que están correctas todas las condiciones y bucles para determinar si el estado real coincide con el esperado o afirmado. Esto genera gran cantidad de caminos posibles por lo que hay que dedicar esfuerzos a la determinación de las condiciones de prueba que se van a verificar. Mediante la prueba de la caja blanca el ingeniero del software puede obtener casos de prueba que (24):

1. Garanticen que se ejerciten por lo menos una vez todos los caminos independientes de cada módulo, programa o método.
2. Ejerciten todas las decisiones lógicas en las vertientes verdadera y falsa.
3. Ejecuten todos los bucles en sus límites operacionales.
4. Ejerciten las estructuras internas de datos para asegurar su validez.

Es por ello que se considera a la prueba de caja blanca como uno de los tipos de pruebas más importantes que se le aplican a los software, logrando como resultado que disminuya en un gran porciento el número de errores existentes en los sistemas y por ende una mayor calidad y confiabilidad.

Prueba del camino básico

La prueba del camino básico es una técnica que permite obtener una medida de la complejidad lógica de un diseño y usar esta medida como guía para la definición de un conjunto básico. La idea es derivar casos de prueba a partir de un conjunto dado de caminos independientes por los cuales puede circular el flujo de control. Para obtener dicho conjunto de caminos independientes se construye el Grafo de Flujo asociado y se calcula su complejidad ciclomática. Los pasos que se siguen para aplicar esta técnica son (24):

1. A partir del diseño o del código fuente, se dibuja el grafo de flujo asociado.
2. Se calcula la complejidad ciclomática del grafo.
3. Se determina un conjunto básico de caminos independientes.
4. Se preparan los casos de prueba que obliguen a la ejecución de cada camino del conjunto básico.

Los casos de prueba derivados del conjunto básico garantizan que durante la prueba se ejecuta por lo menos una vez cada sentencia del programa.

A continuación se muestra una de las pruebas realizadas a una de las funcionalidades del sistema, la cual actualiza la calidad de las secuencias cargadas, luego de haber sido modificados los valores de confianza. Para ello se identificaron las sentencias de ejecución y se enumeraron para representarlas en un grafo.

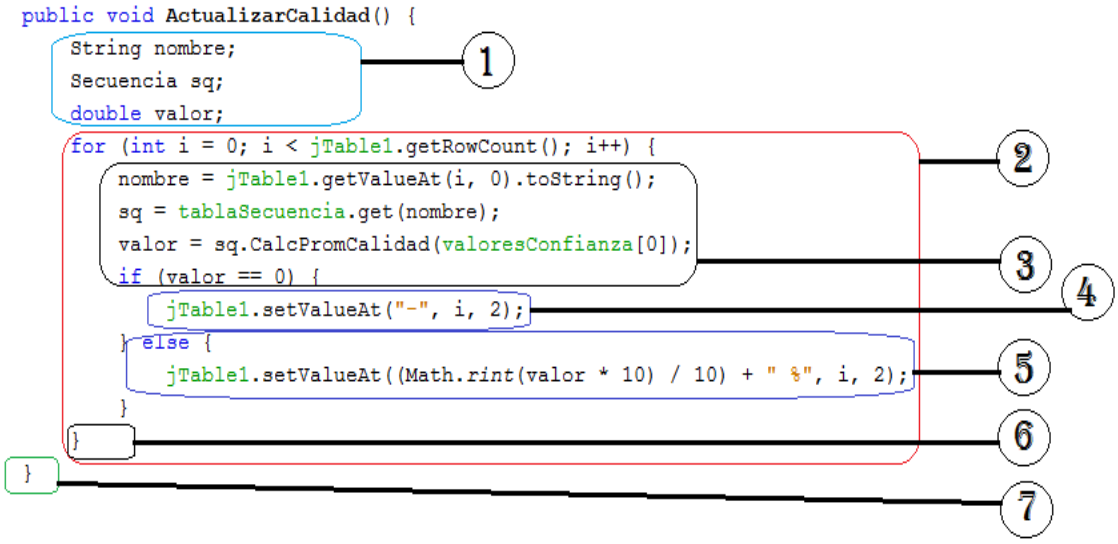


Figura 12: Prueba del camino básico aplicada a la función que actualiza la calidad de las secuencias.

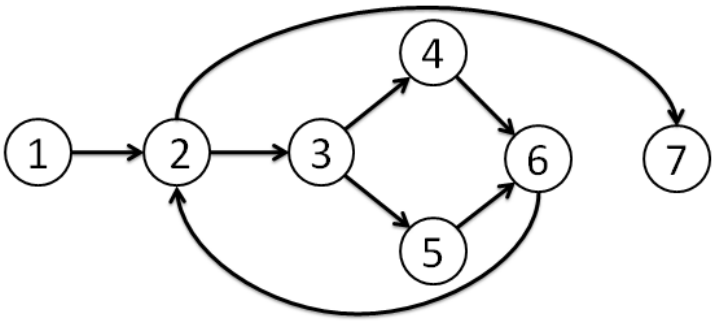


Figura 13: Grafo que representa el camino básico de la función Actualizar Calidad.

Una vez hallado el grafo de la *Figura 13*, 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 ocho artistas y siete nodos, por lo tanto: $V(G) = 8 - 7 + 2$ quedando $V(G) = 3$. La complejidad ciclomática indica los posibles casos de ejecución para la función.

5.1.1. Pruebas de Caja Negra

Las pruebas de caja negra son las que se realizan a la interfaz del software. Una prueba de este tipo examina algún aspecto funcional de un sistema que tiene poca relación con la estructura lógica interna del software. Este tipo de pruebas a veces también conocidas como pruebas de comportamiento se centran en los requisitos funcionales, es decir, permiten obtener conjuntos de condiciones de entrada que ejercitarán por completo todos los requisitos funcionales de un sistema. Las pruebas de caja negra no es una opción frente a las pruebas de caja blanca, es más bien un enfoque complementario que tiene probabilidades de encontrar otra clase de errores de los que se descubrirán con las técnicas de caja blanca (24).

Las pruebas de caja negra intentan identificar errores de las siguientes categorías: funciones incorrectas o ausentes, errores de interfaz, errores en estructuras de datos o en accesos a bases de datos externas, errores de rendimiento y errores de inicialización y de terminación (24).

Método de Partición Equivalente

La partición equivalente es el método de prueba de caja negra que divide el dominio de entrada de un prgrama en clases de datos a partir de las cuales pueden derivarse casos de prueba. Un caso de prueba ideal de manejo simple descubre una clase de errores (por ejemplo manejo incorrecto de todos los datos de caracteres) que, de otra menera requeriría la ejecución de muchos casos antes de que se observe el error general. La partición equivalente se esfuerza por definir un caso de prueba que descubra cierta clase de errores, reduciendo así el número total de casos de prueba que deben desarrollarse (24).

A continuación se realiza el diseño de casos de prueba para los CU Mostrar Secuencia y Exportar Secuencia utilizando las secuencias especificadas en el *Anexo 2*.

Caso de Uso Mostrar Secuencia.

Descripción:

El CU se inicia cuando el usuario selecciona la opción “Mostrar Secuencia” en la interfaz principal. El sistema muestra una nueva interfaz con la secuencia, finalizando así el CU.

Condiciones de ejecución:

Debe estar la secuencia cargada en el sistema y seleccionada.

Sección a probar: CU Mostrar Secuencia

Escenario	Descripción	Secuencia	Calidad	Respuesta del Sistema	Flujo Central
EC 1.1 Mostrar Secuencia	Se muestra la secuencia que el usuario desea visualizar.	Anexo 2, Tabla A6.	Valores de calidad correspondientes a la secuencia.	El Sistema muestra una nueva interfaz con la secuencia correspondiente y los gaps en color azul fuerte, indicando un bajo índice de calidad.	1. Seleccionar la secuencia a visualizar. 2. Hacer clic derecho sobre la secuencia y seleccionar la opción Mostrar Secuencia.

					3. El Sistema muestra una nueva interfaz con la secuencia seleccionada.
EC 1.2 Modificar Secuencia	Se elimina o inserta una base, en caso de insertarse se asigna valor máximo de calidad para dicha base.	Anexo 2, Tabla A6.	Valores de calidad correspondientes a la secuencia.	El Sistema no actualiza la lista de valores de calidad.	1. Insertar una o varias bases. 2. Cerrar la ventana.

Tabla 9: Diseño de casos de prueba CU “Mostrar Secuencia de Ácidos Nucleicos”.

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	Secuencia	Área de texto	No	Muestra el contenido de la secuencia, coloreando cada base de acuerdo a su valor de calidad.
2	Calidad	Lista de enteros	No	Contiene el valor de calidad asociado a cada base.

Tabla 10: Descripción de variables CU “Mostrar Secuencia de Ácidos Nucleicos”.

Caso de Uso Exportar Secuencia.

Descripción:

El CU se inicia cuando el usuario selecciona la opción “Exportar Secuencia” en la interfaz principal. El sistema muestra una nueva interfaz para exportar la secuencia.

Condiciones de ejecución:

Debe estar la secuencia cargada en el sistema y seleccionada.

Sección a probar:

CU Exportar Secuencia.

Escenario	Descripción	Salida	SecuenciaFich	Respuesta del Sistema	Flujo Central
EC 1.1 Exportar Secuencia	Se exporta la secuencia en formato fasta	Anexo 2, Tabla A6	C:\Documents and Settings\Janier\ Desktop	El Sistema exporta la secuencia con gaps.	1. Seleccionar la secuencia a exportar. 2. Hacer clic derecho sobre la secuencia y seleccionar la opción Exportar Secuencia. 3. El sistema muestra una nueva interfaz con opciones para exportar la secuencia. 4. Seleccionar la opción Exportar
EC 1.1 Exportar Secuencia	Se exporta la secuencia en formato fasta	Anexo 2, Tabla A6	C:\Documents and Settings\Janier\ Desktop	Si existe un fichero con el mismo nombre en el directorio seleccionado, el Sistema lo sobrescribe.	1. Seleccionar la secuencia a exportar. 2. Hacer clic derecho sobre la secuencia y seleccionar la opción Exportar Secuencia. 3. El sistema muestra una nueva interfaz con opciones para exportar la secuencia. 4. Seleccionar la opción Exportar

Tabla 11: Diseño de casos de prueba CU “Exportar Secuencia de Ácidos Nucleicos”.

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	Salida	Cadena de Texto	No	Contiene el resultado de la cadena a exportar con todos los cambios acordes a las opciones seleccionadas.
2	SecuenciaFich	Fichero	No	Fichero con la dirección seleccionada por el usuario.

Tabla 12: Descripción de variables CU “Exportar Secuencia de Ácidos Nucleicos”.

No	No Conformidad	Ubicación	Estado
1	Se visualiza de forma errónea el color de los gaps.	EC1.1 CU Gestionar Secuencias de Ácidos Nucleicos	Resuelta
2	No se asigna valor de calidad máxima a las bases insertadas.	EC1.2 CU Gestionar Secuencias de Ácidos Nucleicos	Resuelta
3	No se eliminan los gaps de la secuencia, si esta opción fue seleccionada.	EC1.1 CU Exportar Secuencia	Resuelta
5	Por defecto se sobrescribe el fichero si existe uno con el mismo nombre en el directorio seleccionado por el investigador	EC1.1 CU Exportar Secuencia	Resuelta

Tabla 13: No conformidades detectadas a través de los casos de prueba aplicando la técnica de pruebas de caja negra.

La aplicación de pruebas de caja negra permitió verificar el cumplimiento de los requisitos funcionales del sistema, la realización de dichas pruebas arrojó 4 no conformidades en 2 CU las cuales fueron resueltas satisfactoriamente.

Conclusiones del Capítulo

En este capítulo se mostró el modelo de implementación del sistema, para lo que se construyó el diagrama de componentes correspondiente al CU Ensamblar Secuencias, en el cual se describieron los componentes necesarios para el funcionamiento de dicho CU.

Se llevó a cabo el proceso de pruebas de software, aplicando pruebas de caja blanca y caja negra. Las pruebas de caja blanca se desarrollaron teniendo en cuenta la técnica del camino básico la cual arrojó una complejidad ciclomática de 3 para la operación “Actualizar Calidad”. Del mismo modo se diseñaron casos de prueba de caja negra utilizando la técnica de partición de equivalencia. Dichas pruebas se realizaron mediante la interfaz del sistema en tiempo de ejecución a tevés de la cual se identificó un total de 4 no conformidades que fueron solucionadas rápidamente corrigiendo el código fuente en las clases correspondientes.

CONCLUSIONES

Una vez concluida la investigación se puede afirmar que se desarrolló un sistema para el ensamblaje y edición de secuencias de ácidos nucleicos y además se obtuvieron los siguientes resultados:

- Se definieron los requisitos del sistema y se agruparon los RF en 6 CU, identificando como arquitectónicamente significativo el CU “Ensamblar Secuencias”. La arquitectura se confeccionó con la utilización del patrón MVC. El mismo fue aplicado 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 obtuvo un sistema capaz de gestionar secuencias de ácidos nucleicos provenientes de máquinas de secuenciación de última generación y a partir de ellos realizar cualquier análisis, edición, visualización de cromatogramas y exportación de secuencias, además de que quedó plasmada una arquitectura favorable para continuar el desarrollo del software hasta su despliegue final.

- Se validaron las funcionalidades implementadas a través de la interfaz del sistema, mediante el diseño de pruebas de caja negra de análisis de valores al límite. Las pruebas permitieron encontrar inconformidades en la validación de datos y el tratamiento de errores que contribuyeron a aumentar la calidad del sistema.

- Se agregaron funcionalidades a modo de valor agregado, tales como obtención de información de una secuencia determinada y un conjunto de opciones las cuales el investigador puede seleccionar a la hora de exportar una secuencia. Todo esto con el objetivo de obtener un software más usable por el usuario.

RECOMENDACIONES

1. Debido al tamaño que puede presentar un genoma de referencia, la cantidad de lecturas a ensamblar y teniendo en cuenta que las comparaciones de las lecturas con el genoma de referencia se realizan de manera independiente, se recomienda paralelizar el proceso de alineamiento.
2. Agregar la visualización del alineamiento de las lecturas hacia la referencia, debido a la información que este provee.
3. Debido a su gran uso en la actualidad se recomienda mostrar los cromatogramas de las secuencias tipo ABI, así como gestionar sus valores de calidad.

REFERENCIAS BIBLIOGRÁFICAS

1. **Universidad Industrial de Santander.** *Sitio web de la Universidad Industrial de Santander* http://orientacion.universia.net.co/informacion_carreras/pregrado/biologia-codigo-snies-704-1893/universidad-industrial-de-santander-67.html

2. **Revista Publispain.** *Sitio del Grupo Publispain* http://www.publispain.com/revista/seccion/ecologismo/que_es_la_biologia.html

3. **Universidad de Navarra.** *sitio de la Universidad de Navarra.* [En línea] [Citado el: 2013 de 1 de 5.] http://www.unav.es/acienciacierta/concepto.html .

4. **Hormigo Ruiz, Miguel.** *ESTUDIO Y EVALUACIÓN DE ALGORITMOS Y PROGRAMAS DE ENSAMBLAJE DE SECUENCIAS.* Universidad Internacional de Andalucía. 2011. Tesis de Maestría. ISBN 978-84-7993-943-4.

5. **Instituto de Medicina Tropical Pedro Kourí.** *sitio del Instituto de Medicina Tropical Pedro Kourí.* [En línea] [Citado el: 2013 de Enero de 6.] http://www.ipk.sld.cu/infogen.htm.

6. **Niebla Velasco, Consuelo María.** *La aplicación de NGS como técnicas para el estudio del Microbioma Humano.* Málaga : s.n., 2012. Trabajo Fin de Máster.

7. *sitio web de Bioinformatica. Grado de Biotecnologia.* http://personales.upv.es/jcanizar/bioinformatica/ensamblaje.html

8. **González, Germán.** [En línea] 2013. http://www.bioinformaticos.com.ar/algoritmo-smith-waterman/.

9. **Ensembl Project.** *sitio de Ensembl Project.* [En línea] 1 de 2013. [Citado el: 6 de Enero de 2013.] http://www.ensembl.org/common/Help/Glossary?db=core.

10. **Strömberg, Michael.** *Mosaik Assembler Guide.* 2009.

11. *Metodologías híbridas para desarrollo de software.* **Jiménez Hernández, Eréndira Miriam y Orantes Jiménez, Sandra Dinora.** 1, México : Revista Digital Universitaria, 2012, Vol. 13.

12. **EPF.** Eclipse.org. [En línea] 2010. [Citado el: 9 de Enero de 2013.] http://epf.eclipse.org/wikis/openupsp/openup_basic/deliveryprocesses/openup_basic_process_0uyGoMlgEdmt3adZL5Dmdw_desc.html.

13. **Reyna, Rafael.** Capítulo 1: Herramientas CASE. *Ingeniería de Software 1.* 2008.

14. **VisualParadigm.com.** *sitio web de Visual Paradigm.* [En línea] 15 de 1 de 2013. [Citado el: 17 de Enero de 2013.] http://www.visual-paradigm.com/product/vpuml/.

15. Capítulo 8: Entornos y tecnologías de desarrollo*Introducción al software libre*2007

16. **NetBeans.org.** *sitio web de Net Beans.* [En línea] 2012. [Citado el: 12 de Enero de 2013.] http://netbeans.org/features/index.html.

17. **Microsoft.** *sitio web de Microsoft.* [En línea] 2013. [Citado el: 13 de Enero de 2013.] http://msdn.microsoft.com/es-es/library/ms172579%28v=vs.80%29.aspx.

18. **Universidad de Valladolid.** *sitio Web del Departamento de Informática de la Universidad de Valladolid.* [En línea] 21 de 5 de 2012. [Citado el: 14 de Enero de 2013.] http://www.infor.uva.es/~jmrr/TAD2003/Sesiones/TADONJava/JAVA.html.

19. **Salinas Caro, Patricio y Histchfeld K., Nancy.** Tutorial de UML. *Tutorial de UML.*

20. **Herrera Hernandez, Alberto y Armas Hernández, Jorge Humberto.** *Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas: Integración de un Sistema de Almacenamiento Distribuido al Portal de Servicios Bioinformáticos.* Ciudad Habana : s.n., 2011.

21. **Franco Martínez, Edgardo Adrián.** *Construcción de bibliotecas en C.*

22. **biojava.org.** *The BioJava Tutorial.*

23. **Ambler, Scot.** ActiveData. [En línea] [Citado el: 25 de Febrero de 2013.] http://www.agiledata.org/essays/agileDataModeling.html#InitialDomainModel.

24. **Pressmas, Roger S.** Ingeniería de Software. Un enfoque práctico. *Ingeniería de Software. Un enfoque práctico.* 1998.

25. **Elementos de UML.** Elementos de UML. *KDE Documentation.* [En línea] [Citado el: 25 de Febrero de 2013.] http://docs.kde.org/stable/es/kdesdk/umbrello/uml-elements.html.

26. **Camacho, Erika, Cardeso, Fabio y Gabriel, Nuñez.** *Arquitectura de Software, Guía de Estudio.* 2004.

27. **Gamma, Erich.** *Design patterns: elements of reusable object-oriented software.* Bostos, USA : Addison-Wesley Longman Publishing Co., 1995.

28. **Metodología de la Red Nacional de Integración y Desarrollo de Software Libre (MeRinde).** [En línea] 2013. [Citado el: 10 de 4 de 2013.] http://merinde.net/index.php?option=com_content&task=view&id=96.

29. **Quisbert Limachi, Nancy Susana y Marca Huallpara, Hugo Michael.** *Análisis y Diseño de Sistemas. Modelo de Despliegue.*

30. **Pruebas de Software.** Pruebas de Software. *sitio web de Gestión de Calidad y Pruebas de Software.* [En línea] 2013. [Citado el: 5 de Mayo de 2013.] http://pruebasdesoftware.com/laspruebasdesoftware.htm.

BIBLIOGRAFÍA

1. **Universidad Industrial de Santander.** *Sitio web de la Universidad Industrial de Santander.* [En línea] 22 de 1 de 2013. [Citado el: 5 de Enero de 2013.] http://orientacion.universia.net.co/informacion_carreras/pregrado/biologia-codigo-snies-704-1893/universidad-industrial-de-santander-67.html.

2. **Revista Publispain.** *Sitio del Grupo Publispain.* [En línea] [Citado el: 5 de Enero de 2013.] http://www.publispain.com/revista/seccion/ecologismo/que_es_la_biologia.html.

3. **Universidad de Navarra.** *sitio de la Universidad de Navarra.* [Online] [Cited: 1 2013, 5.] http://www.unav.es/acienciacierta/concepto.html .

4. **Hormigo Ruiz, Miguel .** *ESTUDIO Y EVALUACIÓN DE ALGORITMOS Y PROGRAMAS DE ENSAMBLAJE DE SECUENCIAS.* Universidad Internacional de Andalucía. 2011. Tesis de Maestría. ISBN 978-84-7993-943-4.

5. **Instituto de Medicina Tropical Pedro Kourí.** *sitio del Instituto de Medicina Tropical Pedro Kourí.* [Online] [Cited: Enero 2013, 6.] http://www.ipk.sld.cu/infogen.htm.

6. **Niebla Velasco, Consuelo María.** *La aplicación de NGS como técnicas para el estudio del Microbioma Humano.* Málaga : s.n., 2012. Trabajo Fin de Máster.

7. **Cañizares, Joaquín y Blanca, José Miguel.** *sitio web de Bioinformatica. Grado de Biotecnologia.* [En línea] 2011. [Citado el: 7 de Enero de 2013.] http://personales.upv.es/jcanizar/bioinformatica/ensamblaje.html.

8. **González, Germán.** [Online] 2013. http://www.bioinformaticos.com.ar/algoritmo-smith-waterman/.

9. **Ensembl Project.** *sitio de Ensembl Project.* [Online] 1 2013. [Cited: Enero 6, 2013.] http://www.ensembl.org/common/Help/Glossary?db=core.

10. **Strömberg, Michael.** *Mosaik Assembler Guide.* 2009.

11. *Metodologías híbridas para desarrollo de software.* **Jiménez Hernández, Eréndira Miriam and Orantes Jiménez, Sandra Dinora.** 1, México : Revista Digital Universitaria, 2012, Vol. 13.

12. **EPF.** Eclipse.org. [Online] 2010. [Cited: Enero 9, 2013.] http://epf.eclipse.org/wikis/openupsp/openup_basic/deliveryprocesses/openup_basic_process_0uyGoMlgEdmt3adZL5Dmdw_desc.html.

13. **Reyna, Rafael.** Capítulo 1: Herramientas CASE. *Ingeniería de Software 1.* 2008.

14. **VisualParadigm.com.** *sitio web de Visual Paradigm.* [Online] 1 15, 2013. [Cited: Enero 17, 2013.] http://www.visual-paradigm.com/product/vpuml/.

15. Capítulo 8: Entornos y tecnologías de desarrollo. [aut. libro] Jesus M. González Barahoma, Joaquín Seoane Pascual y Gregorio Robles. *Introducción al software libre.* 2007.

16. **NetBeans.org.** *sitio web de Net Beans.* [Online] 2012. [Cited: Enero 12, 2013.] http://netbeans.org/features/index.html.

17. **Microsoft.** *sitio web de Microsoft.* [Online] 2013. [Cited: Enero 13, 2013.] http://msdn.microsoft.com/es-es/library/ms172579%28v=vs.80%29.aspx.

18. **Universidad de Valladolid.** *sitio Web del Departamento de Informática de la Universidad de Valladolid.* [Online] 5 21, 2012. [Cited: Enero 14, 2013.] http://www.infor.uva.es/~jmrr/TAD2003/Sesiones/TADONJava/JAVA.html.

19. **Salinas Caro, Patricio and Histchfeld K., Nancy.** Tutorial de UML. *Tutorial de UML.*

20. **Herrera Hernandez, Alberto and Armas Hernández, Jorge Humberto.** *Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas: Integración de un Sistema de Almacenamiento Distribuido al Portal de Servicios Bioinformáticos.* Ciudad Habana : s.n., 2011.

21. **Franco Martínez, Edgardo Adrián.** *Construcción de bibliotecas en C.*

22. **biojava.org.** *The BioJava Tutorial.*

23. **Ambler, Scot.** ActiveData. [Online] [Cited: Febrero 25, 2013.] http://www.agiledata.org/essays/agileDataModeling.html#InitialDomainModel.

24. **Pressmas, Roger S.** Ingeniería de Software. Un enfoque práctico. *Ingeniería de Software. Un enfoque práctico.* 1998.

25. **Elementos de UML.** Elementos de UML. *KDE Documentation.* [Online] [Cited: Febrero 25, 2013.] http://docs.kde.org/stable/es/kdesdk/umbrello/uml-elements.html.

26. **Camacho, Erika, Cardeso, Fabio and Gabriel, Nuñez.** *Arquitectura de Software, Guía de Estudio.* 2004.

27. **Gamma, Erich.** *Design patterns: elements of reusable object-oriented software.* Bostos, USA : Addison-Wesley Longman Publishing Co., 1995.

28. **Metodología de la Red Nacional de Integración y Desarrollo de Software Libre (MeRinde).** [Online] 2013. [Cited: 4 10, 2013.] http://merinde.net/index.php?option=com_content&task=view&id=96.

29. **Quisbert Limachi, Nancy Susana and Marca Huallpara, Hugo Michael.** *Análisis y Diseño de Sistemas. Modelo de Despliegue.*

30. **Pruebas de Software.** Pruebas de Software. *sitio web de Gestión de Calidad y Pruebas de Software.* [Online] 2013. [Cited: Mayo 5, 2013.] http://pruebasdesoftware.com/laspruebasdesoftware.htm.

31. **Zerbino, Daniel.** [Online] 4 14, 2011. [Cited: Enero 8, 2013.] http://www.ebi.ac.uk/~zerbino/velvet/.

32. **Xiaoqiu Huang, Jianmin Wang, Srinivas Aluru.** *PCAP: A Whole-Genome Assembly Program.* 2012.

33. *AMOS Assembly Validation and Visualization.* **Schatz, Michael.** Abril 2006.

34. *Comparative genome.* **Pop Mihai, Phillippy Adam, Delcher Arthur, Salzberg Steven L.** 3, USA : s.n., Septiembre 2004, Vol. 5.

35. *Genome Sequence Assembly: Algorithms and Issues.* **Pop, Mihai , Salzberg, Steven L. and Shumway, Martin .** USA : s.n.

36. **Phillippy, Adam M.** *Comparative Genome Assembly.* USA : s.n.

37. *An Algorithm for Whole Genome Shotgun Sequencing.* **E. Anson, G. Myers.** USA : s.n.

38. Universidad de Navarra. *sitio web de la Universidad de Navarra.* [Online] 2013. [Cited: Enero 6, 2013.] http://www.unavarra.es/genmic/docbiomica/genoma.pdf.

39. **BC Cancer Agency | Canada's Michael Smith Genome Science Center.** *stio web de la BC Cancer Agency | Canada's Michael Smith Genome Science Center.* [Online] 5 30, 2012. [Cited: Enero 7, 2013.] http://www.bcgsc.ca/platform/bioinfo/software/abyss/.

40. **AMOS Sourceforge.** *Sitio web del proyecto AMOS.* [Online] 7 7, 2009. [Cited: Enero 8, 2013.] sourceforge.net/apps/mediawiki/amos/index.php?title=AMOScmp.

41. **Molecular Ecology Resources.** *sitio web de Molecular Ecology Resources.* [Online] 10 1, 2012. [Cited: Enero 8, 2013.] http://tomato.biol.trinity.edu/programs/index.php/Celera_Assembler.

42. **FILExt.** *sitio web de extenxiones de archivos.* [Online] [Cited: Enero 12, 2013.] http://filext.com/faq/using_an_emulator.php.

43. **BioJava.** *sitio web de BioJava.* [Online] 8 10, 2012. [Cited: Enero 15, 2013.] http://biojava.org/wiki/Main_Page.

44. **Binaryti.** *sitio web de Binaryti.* [Online] 2011. [Cited: Enero 13, 2013.] http://www.binaryti.com/2012/06/cygwin-un-linux-en-windows.html.

45. **Universidad de Navarra.** *sitio de la Universidad de Navarra.* [Online] [Cited: Enero 6, 2013.] http://www.unavarra.es/genmic/docbiomica/genoma.pdf.

46. **Gene Codes Corporation .** *Sequencher: Tour Guide for Windows and Macintosh.* 2007.

47. **The University of Michigan.** DNA Sequencing Core. *sitio web de The University of Michigan.* [Online] 2013. [Cited: Marzo 7, 2013.] http://seqcore.brcf.med.umich.edu/.

48. **Center for Bioinformatics an Computational Biology.** CBCB. *sitio web de Center for Bioinformatics an Computational Biology.* [Online] 2013. [Cited: Febrero 25, 2013.] http://www.cbcb.umd.edu/.

49. **Psicología2000.** *Biblioteca Informática.* [Online] 2011. [Cited: Enero 16, 2013.] http://www.psicologia2000.com/en/enciclopedia-general-psicologia-on-line-wiki-letra-b/28834-biblioteca-informatica.html.

ANEXOS

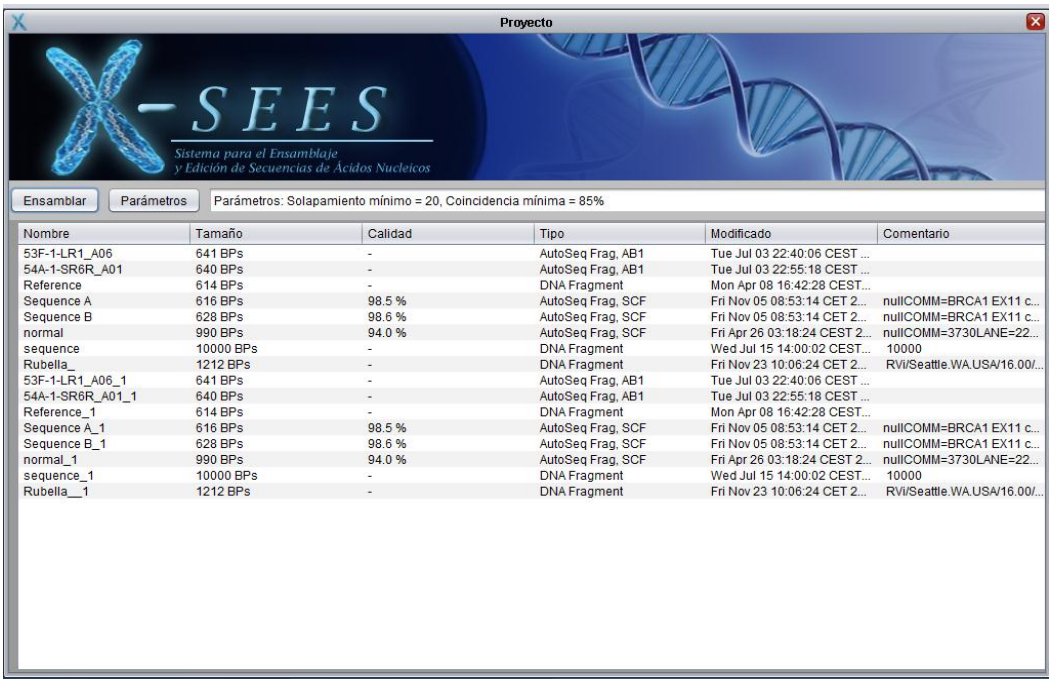
Anexo 1: Descripción de CU del Sistema

Descripción del CU Gestionar Secuencias de Ácidos Nucleicos.

<table>
<tr><td>Caso de Uso:</td><td colspan="2">Gestionar Secuencias de Ácidos Nucleicos</td></tr>
<tr><td>Actor:</td><td colspan="2">Investigador</td></tr>
<tr><td>Resumen:</td><td colspan="2">El CU se inicia cuando el Investigador selecciona la opción Nuevo Proyecto en la interfaz principal. El sistema muestra la interfaz correspondiente. El CU finaliza cuando se haya obtenido el resultado de la operación efectuada.</td></tr>
<tr><td>Precondiciones:</td><td colspan="2"></td></tr>
<tr><td>Referencias</td><td colspan="2">RF1, RF2, RF3, RF4</td></tr>
<tr><td>Prioridad</td><td colspan="2">Crítico.</td></tr>
<tr><td colspan="3">Flujo Normal de Eventos</td></tr>
<tr><td colspan="2">Acción del Actor</td><td>Respuesta del Sistema</td></tr>
<tr><td colspan="2">1. El investigador selecciona la opción Nuevo Proyecto en la Interfaz Principal.</td><td>2. El sistema muestra la interfaz del Proyecto.</td></tr>
<tr><td colspan="2">3. El investigador primeramente deberá cargar secuencias en el sistema y luego seleccionar una de las siguientes opciones:
• Importar Secuencias
• Mostrar Secuencia
• Eliminar Secuencias</td><td>4. El sistema en dependencia de la opción seleccionada, realizará lo siguiente:
• Si el investigador elige Importar Secuencias, ir a la sección Importar Secuencias.
• Si el investigador elige Mostrar Secuencias, ir a la sección Mostrar Secuencias.
• Si el investigador elige la opción Eliminar Secuencias, ir a la opción Eliminar Secuencias.</td></tr>
<tr><td colspan="3">Sección “Importar Secuencia de Ácidos Nucleicos”</td></tr>
<tr><td colspan="2">Acción del Actor</td><td>Respuesta del Sistema</td></tr>
<tr><td colspan="2">1. El investigador selecciona la opción “Importar Secuencias” en la interfaz</td><td>2. El sistema muestra un navegador de archivos para la selección de los ficheros.</td></tr>
</table>

Nuevo Proyecto.	
3. El investigador selecciona los ficheros que contienen las secuencias y selecciona la opción Aceptar.	4. El sistema actualiza el listado de las secuencias cargadas.

Prototipo de Interfaz



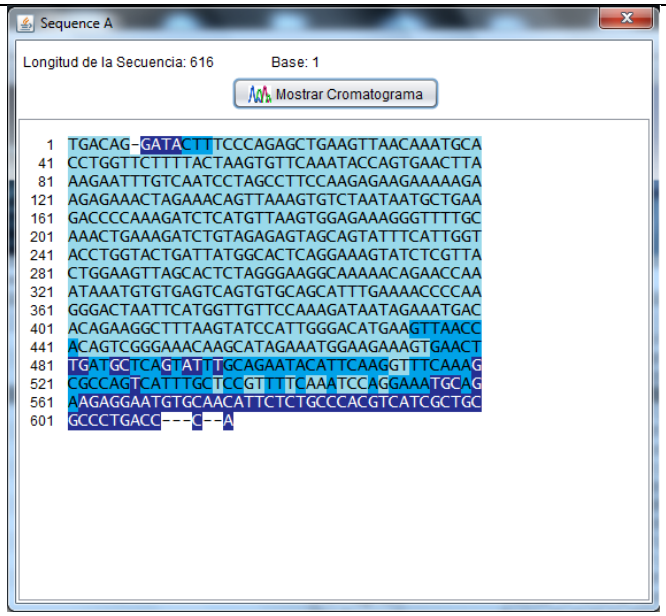
Flujo Alterno a la sección Importar Secuencia de Ácidos Nucleicos

4.1. El investigador selecciona la opción Cancelar.	4.2. El sistema cancela la importación de secuencias y cierra el navegador.

Sección “Mostrar Secuencias de Ácidos Nucleicos”

Acción del Actor	**Respuesta del Sistema**
1. El investigador selecciona la secuencia que desea mostrar y selecciona la opción “Mostrar Secuencia.”	3. El sistema muestra una nueva ventana con el contenido de la secuencia seleccionada.

Prototipo de Interfaz



Sección "Modificar Secuencias de Ácidos Nucleicos"	
Precondiciones	Se debe haber mostrado la secuencia a modificar
Acción del Actor	**Respuesta del Sistema**
1. El investigador modifica la secuencia.	2. En caso de que la secuencia sea de tipo SCF o Fastq, el sistema asigna valor de calidad máximo para la base adicionada, en caso contrario asigna valor de calidad -1.
Sección "Eliminar Secuencias de Ácidos Nucleicos"	
Acción del Actor	**Respuesta del Sistema**
1. El investigador selecciona las secuencias que desea eliminar y selecciona la opción "Eliminar Secuencias".	2. El sistema muestra un mensaje de confirmación.
3. El investigador confirma que desea eliminar las secuencias.	4. El sistema elimina las secuencias seleccionadas y actualiza la lista de secuencias.
Prototipo de Interfaz	

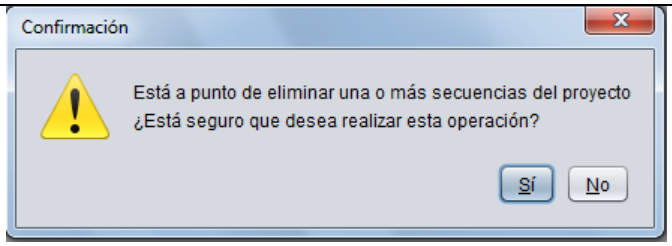
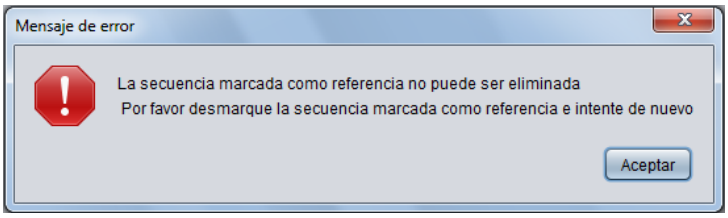
<table>
<tr><td colspan="2"> 
</td></tr>
<tr><td colspan="2">Flujo Alterno de la sección Eliminar Secuencias de Ácidos Nucleicos</td></tr>
<tr><td>4.1. El investigador elige “no” en la petición de eliminación de las secuencias.</td><td>4.2. El sistema no elimina las secuencias.</td></tr>
<tr><td></td><td>4.1. El sistema muestra un mensaje informando que la secuencia marcada como referencia no puede ser eliminada.</td></tr>
<tr><td colspan="2">Prototipo de Interfaz
 
</td></tr>
<tr><td>Poscondiciones</td><td>Se actualiza el listado de secuencias.</td></tr>
</table>

Tabla A 1: Descripción del CU “Gestionar Secuencias de Ácidos Nucleicos”.

Descripción del CU Modificar Parámetros de Ensamblaje.

Caso de Uso:	Modificar Parámetros de Ensamblaje
Actores:	Investigador
Resumen:	El CU se inicia cuando el actor selecciona la opción Modificar Parámetros, el actor modifica los parámetros deseados y presiona la opción aceptar, finalizando así el CU.
Precondiciones:	
Referencias	RF5
Prioridad	Normal
Flujo Normal de Eventos	

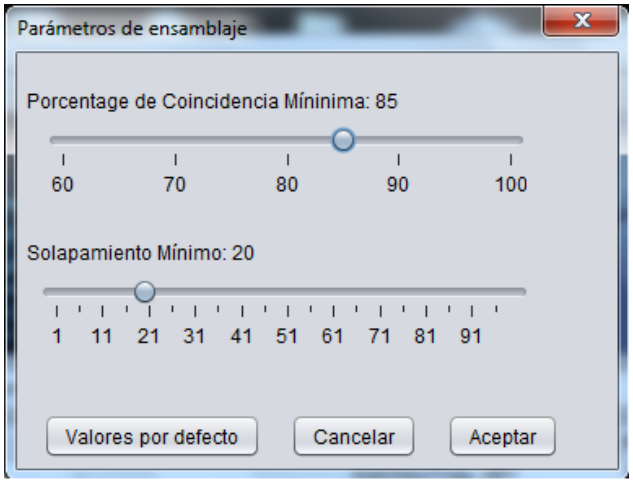
Acción del Actor	Respuesta del Sistema
1. El investigador modifica los parámetros de ensamblaje deseados y selecciona la opción Aceptar.	2. El sistema actualiza los parámetros de ensamblaje.
Prototipo de Interfaz 	
Poscondiciones	El sistema actualiza los parámetros de ensamblaje.

Tabla A 2: Descripción del CU “Modificar Parámetros de Ensamblaje”.

Descripción del CU Mostrar Cromatogramas

Caso de Uso:	Mostrar Cromatogramas
Actores:	Investigador
Resumen:	El CU se inicia cuando el actor selecciona la opción Mostrar Cromatograma, el CU finaliza una vez mostrado el Cromatograma.
Precondiciones:	El fichero de la secuencia de la cual se va a mostrar su Cromatograma tiene que estar en formato .SCF.
Referencias	RF7
Prioridad	Normal
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**

1. El investigador selecciona la opción “Mostrar Cromatograma”.	2. El sistema muestra una interfaz con el cromatograma correspondiente.
Prototipo de Interfaz	

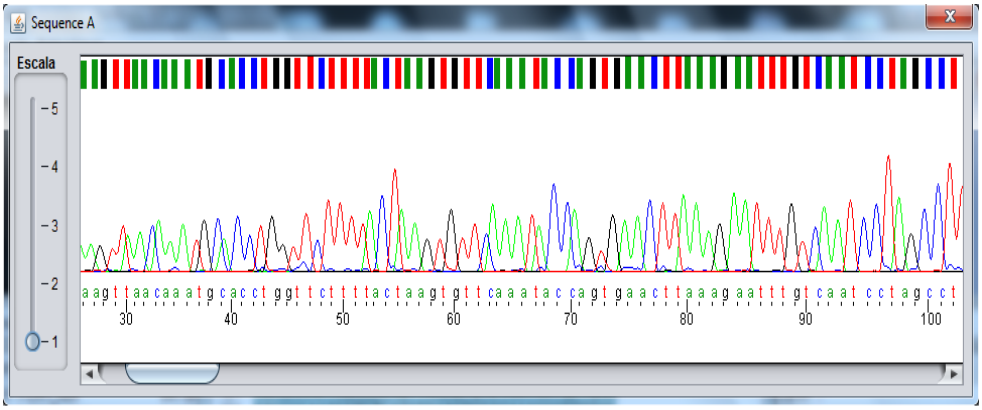


Tabla A 3: Descripción del CU “Mostrar Cromatogramas”.

Descripción del CU Exportar Secuencia

Caso de Uso:	Descripción del CU Exportar Secuencia
Actores:	Investigador
Resumen:	El CU se inicia cuando el actor selecciona la opción Exportar Secuencia y termina una vez exportada dicha secuencia.
Referencias	RF8
Prioridad	Normal
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El investigador selecciona la opción “Exportar Secuencia”.	2. El sistema muestra un navegador de archivos para seleccionar la dirección donde se desea guardar la secuencia.
3. El investigador selecciona la dirección donde desea guardar la secuencia.	4. El sistema guarda el resultado en la dirección definida por el investigador.
Prototipo de Interfaz	

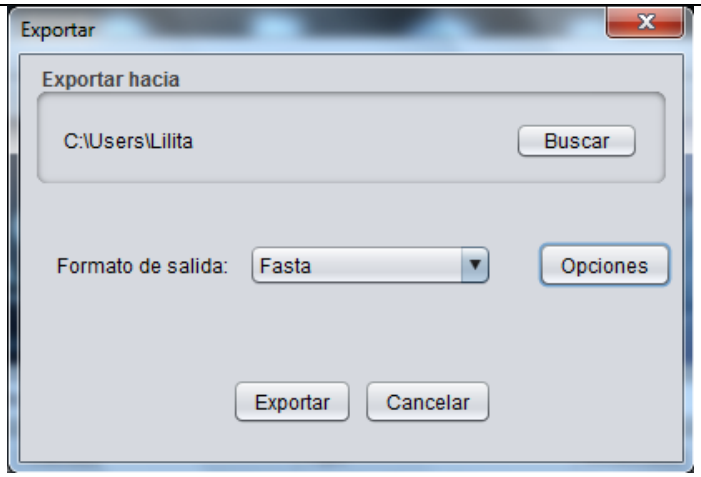


Tabla A 4: Descripción del CU “Exportar Secuencia”.

Descripción del CU Establecer Intervalos de Confianza

Caso de Uso:	Descripción del CU Establecer Intervalos de Confianza
Actores:	Investigador
Resumen:	El CU se inicia cuando el actor selecciona la opción Intervalos de Confianza y termina una vez cerrada dicha interfaz.
Referencias	RF9
Prioridad	Normal
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El investigador selecciona la opción “Intervalos de Confianza”.	2. El sistema muestra una vista en la cual se puede ajustar los rangos de confianza de las secuencias.
3. El investigador modifica los rangos de confianza y cierra la ventana.	4. El sistema calcula la confianza para cada secuencia y la actualiza en la interfaz principal del proyecto.
Prototipo de Interfaz	

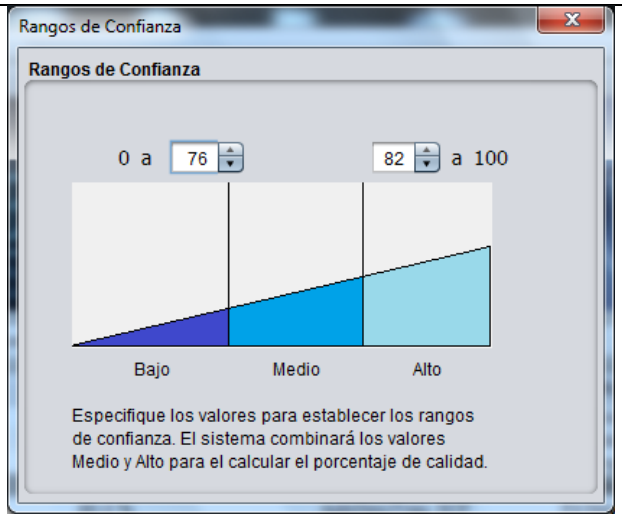


Tabla A 5: Descripción del CU “Establecer Intervalos de Confianza”.

Anexo 2: Secuencias empleadas para realizar los Casos de Pruebas

>SequenceA	>SequenceB
TGACAG-ATACTTTCCCAGAGCTGAAGTTAACAAATGCA	GC-AGACGAAGTAA-ACTGACG-GATACTTTCCCGAGCTG
CCTGGTTCTTTTACTAAGTGTTCAAATACCAGTGAACTT	AAGTTAACAAATGCACCTGGTTCTTTTACTAAGTGTTCAA
AAAGAATTTGTCAATCCTAGCCTTCCAAGAGAAGAAAAA	ATACCAGTGAACTTAAAGAATTTGTCAATCCTAGCCTTCC
GAAGAGAAACTAGAAACAGTTAAAGTGTCTAATAATGCT	AAGAGAAGAAAAAGAAGAGAAACTAGAAACAGTTAAAGTG
GAAGACCCCAAAGATCTCATGTTAAGTGGAGAAAGGGTT	TCTAATAATGCTGAAGACCCCAAAGATCTCATGTTAAGTG
TTGCAAACTGAAAGATCTGTAGAGAGTAGCAGTATTTCA	GAGAAAGGGTTTTGCAAACTGAAAGATCTGTAGAGAGTAG
TTGGTACCTGGTACTGATTATGGCACTCAGGAAAGTATC	CAGTATTTCATTGGTACCTGGTACTGATTATGGCACTCAG
TCGTTACTGGAAGTTAGCACTCTAGGGAAGGCAAAAACA	GAAAGTATCTCGTTACTGGAAGTTAGCACTCTAGGGAAGG
GAACCAAATAAATGTGTGAGTCAGTGTGCAGCATTTGAA	CAAAAACAGAACCAAATAAATGTGTGAGTCAGTGTGCAGC
AACCCCAAGGGACTAATTCATGGTTGTTCCAAAGATAAT	ATTTGAAAACCCCAAGGGACTAATTCATGGTTGTTCCAAA
AGAAATGACACAGAAGGCTTTAAGTATCCATTGGGACAT	GATAATAGAAATGACACAGAAGGCTTTAAGTATCCATTGG
GAAGTTAACCACAGTCGGGAAACAAGCATAGAAATGGAA	GACATGAAGTTAACCACAGTCGGGAAACAAGCATAGAAAT
GAAAGTGAACTTGATGCTCAGTATTTGCAGAATACATTC	GGAAGAAAGTGAACTTGATGCTCAGTATTTGCAGAATACA
AAGGTTTCAAAGCGCCAGTCATTTGCTCCGTTTTCAAAT	TTCAAGGTTTCAAAGCGCCAGTCATTTGCTCTGTTTTCAA
CCAGGAAATGCAGAAGAGGAATGTGCAACATTCTCTGCC	ATCCAGGAAATGCAGAAGAGGAATGTGCAACATTCTCTGC
CACGTCATCGCTGCGCCCTGACC---C--A	CCACGTCATCGCTGGCCCCTTGC-----

Tabla A 6: Secuencias de Ácidos Nucleicos válidas.