Universidad de las Ciencias Informáticas

Facultad 6

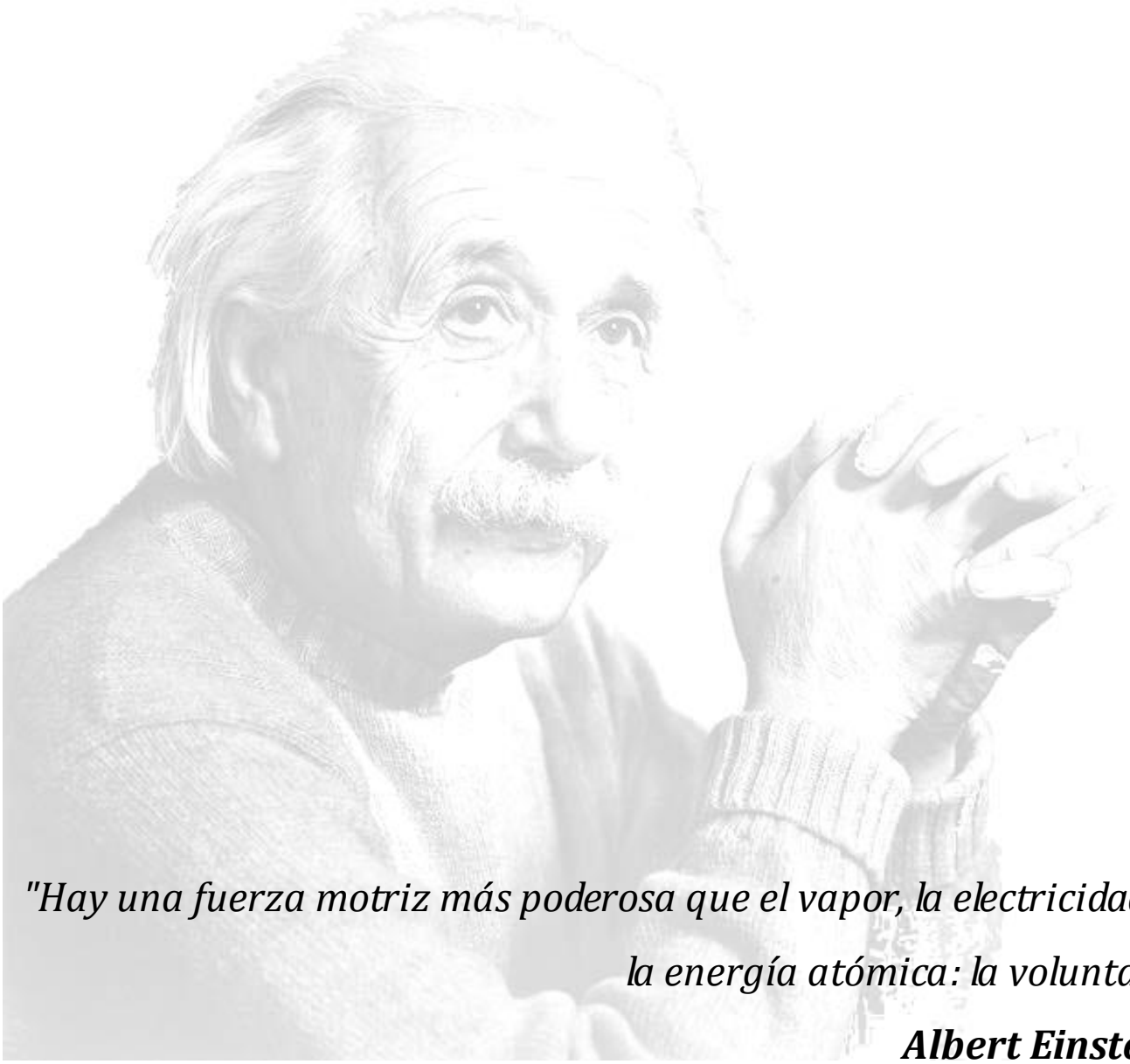
Título: “BioSyS: Desarrollo del módulo Modelación Gráfica”.

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

Autores: Arsenio Llamo Castilla
Ingrid Viamontes de Armas

Tutores: Ing. Yuniesky Armentero Moreno
Ing. Dismey Saavedra López

Ciudad de la Habana, junio 2009
¨Año del 50 Aniversario del Triunfo de la Revolución¨



"Hay una fuerza motriz más poderosa que el vapor, la electricidad y la energía atómica: la voluntad."

Albert Einstein

Declaración de Autoría

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

Ingrid Viamontes de Armas **Arsenio Llamo Castilla**

_____ _____

Firma del Autor Firma del Autor

Ing. Yuniesky Armentero Moreno

_____

Firma del Tutor

Ing. Dismey Saavedra López

_____

Firma del Tutor

Datos de contacto

Tutores: Ing. Yuniesky Armentero Moreno.

Universidad de las Ciencias Informáticas, Habana, Cuba.

Correo electrónico: yarmentero@uci.cu

Ing. Dismey Saavedra López

Universidad de las Ciencias Informáticas, Habana, Cuba.

Correo electrónico: dsaavedra@uci.ci

Agradecimientos

A la Revolución por crear la oportunidad única de formar personas y profesionales.

A nuestros padres por su dedicación y amor.

A nuestra familia por su ayuda y preocupación.

A nuestros amigos que conforman la inmensa familia de la UCI, por brindarnos siempre lo mejor.

A nuestros compañeros de proyecto que pusieron un granito de arena y contribuyeron con sus críticas.

A nuestros profesores por haber contribuido con nuestra formación profesional.

A nuestros tutores Dismey y Armentero por sus consejos y colaboración.

A todos muchas gracias…

Dedicatoria

A mi mamá y mi papá que siempre han confiado en mí, que me han dado su apoyo sin condiciones y que cada día me impulsan a ser mejor, a mi abuela que es la artífice principal de de una familia maravillosa, a mis tías y tíos, a mi abuelo y Prado por estar siempre pendientes y poner cientos de de granos de arena y bloques en esta construcción.

A mis primos, mi hermano y Kika por siempre preguntar, por querer saber que estoy haciendo.

A mis amigos y compañeros de estudios y proyecto por estar juntos estos 5 años y ser la familia que ha estado más cerca en los buenos y malos momentos, a todos los que en mayor o menor medida contribuyeron a que este momento se realizara. A Yoslainy, Arnolis, Nosbel, Liusmila, Dayana Dioleysis, Lino, Raúl, Jorge Carlos, Norlen, Landy, Michel, Claritza, Anelis, a mis amigos de la facultad 5 y muchos más que quiero y extrañaré cuando no los tenga cerca pero harían esta lista interminable, a Ingrid por el empeño puesto en la realización de esta tesis.

A los amigos de la secundaria y el pre.

A todas las personas que un día me brindaron su ayuda por pequeña que fuera, que me dieron un consejo aunque fuera sencillo, que me dieron ánimo cuando no lo tenía, que estuvieron estos 5 años o solo un momento, que me preguntaron por los estudios, las pruebas o finalmente por la tesis .

Arsenio

Dedicatoria

A mi mamá por estar a mi lado en cada uno de los pasos de mi vida, por hacer realidad todos mis sueños, por su paciencia, su amor y dedicación a mi formación como persona.

A mi papá, que aunque no pudo estar a mi lado físicamente durante estos cinco largos años, logró con su cariño, sus palabras de aliento, su apoyo y su preocupación constante que no notara su ausencia.

A mi hermana que quiero con la vida, por todo el apoyo que me dio durante todo este tiempo.

A mi familia, mamita, mi abuela, mis tíos Nieves y Kike, a mi prima Grettel por su preocupación.

A mi amigo Tito que más que un amigo se convirtió en mi hermano, a ti por toda la ayuda que nos diste en la tesis y durante mi carrera, por tu preocupación, por siempre encontrarle la solución a mis problemas y por estar a mi lado en todo momento.

A Luis Miguel por su amor, ayuda y comprensión en los momentos más difíciles.

A mi amiga del alma Aurora.

A mi compañero de tesis Arsenio que se convirtió en mi amigo, por su perseverancia y paciencia.

A mis amigos y compañeros de los cinco años de la UCI: Naty , Yanet , Landy, Titi, Maide y Elezky.

A Maritza por soportar mis malcriadeces durante todo este año y ayudarme en todo lo que necesité como si fuese mi madre.

A todos los que me ayudaron y estuvieron a mi lado, a los que me dieron un consejo o aclararon alguna duda.

Ingrid

Resumen

Hoy en día los retos propuestos por las diferentes comunidades de científicos dedicados a la Biología de Sistemas están encaminados a normalizar las notaciones gráficas utilizadas para describir las interacciones biológicas y estandarizar el intercambio de modelos computacionales, los cuales constituyen un importante paso hacia la comunicación eficaz de los conocimientos biológicos entre las distintas comunidades.

El presente trabajo tiene como objetivo desarrollar de una aplicación informática para la modelación de sistemas biológicos que cumpla con dos de los estándares propuestos actualmente para los sistemas biológicos el de representación gráfica y el de intercambio de modelos computacionales. La aplicación constituirá el nuevo módulo de Modelación gráfica del software BioSyS y ampliará las potencialidades del mismo.

Palabras claves
Sistemas biológicos
Biología de Sistemas

Índice

Introducción

La propuesta de la estructura de doble hélice del ADN por James Dewey Watson y Francis Crick, el secuenciamiento de la proteína insulina bovina y la construcción del primer circuito integrado en la década del 50 fueron los hechos que marcaron el surgimiento de la bioinformática.[1]

Puede definirse bioinformática como la aplicación de la informática, las matemáticas y la estadística para organizar, analizar y entender problemas que involucren secuencias de nucleótidos, aminoácidos o cualquier otra información biológica relacionada [2] . Provee herramientas que facilitan el análisis de la información biológica; actividad que solo es posible realizar de forma automatizada. La bioinformática se aplica en la gestión de datos biológicos, la automatización de experimentos, el trabajo con secuencias genéticas, las búsquedas en bases de datos de estructuras de proteínas, la determinación y predicción de la estructura de las macromoléculas, la evolución molecular y los árboles filogenéticos. Posee amplios conocimientos que han sido divididos en diferentes disciplinas para su mejor estudio, una de las ramas surgidas producto del desarrollo de tecnologías de larga escala y alto rendimiento a nivel experimental y computacional es la Biología de Sistemas.

La Biología de Sistemas integra la investigación experimental y computacional para entender la complejidad de los sistemas biológicos[2]. La unidad fundamental de esta ciencia es un modelo computacional (o in-silico) que es construido a partir de componentes celulares y sus mecanismos de comunicación. La Biología de Sistemas brinda una oportunidad única de entrelazar moléculas y sus mensajes dentro de un aparato computacional que puede ser probado contra un amplio conjunto de condiciones y combinaciones[3].

Los sistemas biológicos pueden modelarse de forma gráfica o matemática. A través de la modelación gráfica se reducen las dificultades propias del estudio y análisis de los sistemas biológicos a diferentes niveles de abstracción, desde el nivel molecular hasta los ecosistemas, por la comprensión y expresividad de los gráficos. En cambio la modelación matemática presenta un mayor grado de complejidad producto a que los sistemas de ecuaciones utilizados para modelar los sistemas biológicos son más complejos. Aunque con el método gráfico se pueden visualizar fácilmente las interacciones entre los elementos, no permite describir con precisión los sistemas estudiados. Para ello, son necesarios los modelos matemáticos, que permiten describir los sistemas en términos cuantitativos, simular mejor su comportamiento y, por tanto, predecir las propiedades que no son

evidentes al investigador.

Para el manejo de los modelos computacionales la Biología de Sistemas utiliza la técnica de modelación la cual permite representar un sistema real y responder preguntas acerca de la estructura y funcionamiento de un fenómeno dado.

Modelar sistemas biológicos en los últimos años ha sido el reto de la comunidad de científicos dedicados a la Biología de Sistemas. En este sentido han surgido empresas y grupos de investigación en todo el mundo dedicados a desarrollar softwares encaminados a dar solución a esta problemática. Entre las empresas pioneras de este campo se encuentran Entelos y Gene Network Scienes además de otras compañías de nueva creación:

- Cellnomica
- BioSeek
- Genstruct

Dos de los institutos de investigación destacados creados por grupos de investigación y universidades son:

- Institute for Systems Biology
- Institute for Advanced Biosciences, Keio University

Las empresas y grupos de investigación existentes ofrecen softwares para el estudio y comprensión de los sistemas biológicos. En su mayoría estas herramientas no están libres de pagos o centran su solución en un problema específico. Otro reto de la comunidad de científicos ha sido la creación de una notación que permita la representación de los sistemas biológicos y elimine las ambigüedades existentes maximizando la correcta comprensión de los modelos. Además de un formato de intercambio legible para las diferentes herramientas de software que modelan sistemas biológicos.

En el marco del programa de informatización que se lleva a cabo en la sociedad cubana y como parte

de las investigaciones llevadas a cabo por el Centro de Inmunología Molecular (CIM) del Polo Científico surge una alternativa a la problemática planteada anteriormente, el proyecto Software para la Simulación y Análisis de Sistemas Biológicos (BioSyS) compuesto por cuatro módulos fundamentales: **Modelación** encargado de crear un modelo gráfico, **Editor de ecuaciones** responsable de gestionar todo lo referente a la edición de las ecuaciones matemáticas, **Simulación** que se encarga de la gestión de simulaciones de los modelos matemáticos y **Análisis** que se ocupa del análisis de los resultados luego de generadas las simulaciones. El módulo Modelación en la versión BioSyS 1.0 no está regido por el estándar de representación gráfica de sistemas biológicos, lo que dificulta la comprensión de los modelos creados a investigadores que no estén familiarizados con la representación gráfica utilizada. Otro factor que minimiza las potencialidades de esta versión es que no se ajusta al formato para el intercambio de modelos de sistemas biológicos, los modelos generados por BioSyS 1.0 no pueden ser visualizados por otras herramientas de software similares.

Para dar continuidad al trabajo realizado en el proyecto Software para la Simulación y Análisis de Sistemas Biológicos, que desarrollan actualmente el CIM y el polo de bioinformática de la Universidad de las Ciencias Informáticas (UCI) el presente trabajo de diploma desarrollará un nuevo módulo Modelación con vista a incorporarle nuevas funcionalidades y estándares necesarios para la representación y distribución de los modelos de sistemas biológicos.

Ante tal situación se plantea el siguiente **problema científico**: ¿Cómo modelar gráficamente sistemas biológicos utilizando el estándar de notación gráfica y el formato de intercambio de modelos biológicos?
El problema planteado se enmarca en el **objeto de estudio**: Modelación de sistemas biológicos, delimitado por el **campo de acción**: Modelación gráfica de sistemas biológicos.

En aras de solucionar el problema planteado se define como **objetivo general**: Desarrollar una aplicación informática que permita modelar sistemas biológicos utilizando los estándares SBGN y SBML.
Para alcanzar el mismo se trazan los siguientes **objetivos específicos**:

- Definir las funcionalidades que tendrá la herramienta para la modelación gráfica.
- Diseñar las funcionalidades para la modelación gráfica de sistemas biológicos con los estándares requeridos.

- Implementar las funcionalidades diseñadas.

Para dar cumplimiento a los objetivos trazados se desarrollarán las siguientes **tareas**:

- Investigación del estado del arte para la modelación de sistemas biológicos.
- Investigación de las tendencias y tecnologías actuales para diseñar e implementar herramientas de modelado gráfico.
- Análisis y selección de las tecnologías para desarrollar la aplicación.
- Modelación del negocio y las actividades del flujo de trabajo de requerimientos.
- Realización de las actividades del flujo de trabajo de análisis y diseño.
- Realización de las actividades del flujo de trabajo de implementación.

Capítulo 1: Fundamentación teórica

Este capítulo contiene una descripción acerca de los principales aspectos relacionados con la Biología de sistemas, su origen, desarrollo, características esenciales, retos, perspectivas y nuevas iniciativas surgidas para dar solución a sus problemas. Además se realiza un análisis del software utilizado y desarrollado actualmente y a describe la metodologías a ser utilizadas para la solución del problema.

Capítulo 2: Características del sistema

El presente capítulo contiene una breve descripción de las características básicas del sistema, en el que se identifican las clases del dominio, los requerimientos funcionales y no funcionales y se definen los actores que intervienen en el mismo, así como los diagramas de casos de uso del sistema y una descripción detallada de cada uno de ellos.

Capítulo 3: Diseño del sistema

En este capítulo se plantea como quedaría la arquitectura mediante una descripción basada ya en la aplicación, se ilustra el modelo de diseño definiéndose las clases y los diagramas de interacción que componen cada modelo, especificándose además los patrones de diseño aplicados.

Capítulo 4: Implementación del sistema

En este capítulo se muestran los diagramas de componentes definidos para la implementación del sistema, así como fragmentos de códigos de algunas de las principales clases del mismo, la validación del sistema a nivel de desarrollador y varias pantallas de la aplicación.

Capítulo 1. Fundamentación Teórica

Comprender la necesidad actual de la modelación gráfica de sistemas biológicos requiere de un estudio de las características generales de la biología de sistemas como área interdisciplinaria, sus principales retos, perspectivas, estándares y tendencias de desarrollo de software de la comunidad internacional para dar solución a los problemas que se plantean en el estudio de los sistemas biológicos.

Es importante además la acertada selección de herramientas y tecnologías que permitan ofrecer una solución eficaz al problema planteado.

1.1 Biología de Sistemas

La evolución de la biología molecular, el fraccionamiento del conocimiento, las nuevas técnicas para el acceso a un volumen de dato constantemente creciente y el desarrollo de la computación dieron paso al nacimiento de una nueva disciplina: la Biología de Sistemas, que comenzó a desarrollarse en los años 60 aunque su institucionalización académica no se produjo hasta el año 2000.

La Biología de Sistemas representa una estrategia analítica que permite relacionar los elementos de un sistema, con el objetivo de comprender sus propiedades emergentes. Por tanto, el análisis de sistemas puede aplicarse a moléculas, células, órganos, individuos o incluso ecosistemas. En cada uno de estos casos, se pretende describir todos los elementos que componen el sistema, definir las redes biológicas que interrelacionan los elementos del sistema y caracterizar el flujo de información que determina la puesta en marcha de un proceso biológico[4].

La Biología de Sistemas es un área interdisciplinaria que agrupa conocimientos de varias ciencias como la biología, la matemática, la física y la informática no pudiéndose definir el alcance de cada una de ellas en la interrelación.

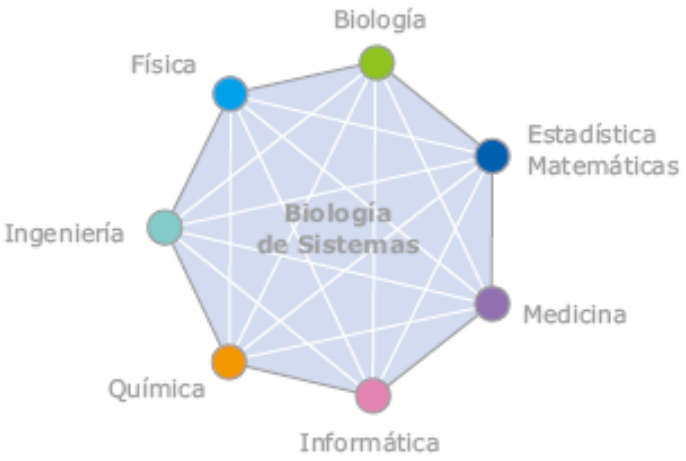


Figura 1. Disciplinas que componen la Biología de Sistemas.
Fuente: Munich Systems Biology Forum (MSBF).

1.1.1 Características de la Biología de Sistemas:

- Maneja una gran colección de datos procedentes de estudios experimentales.
- Propone modelos matemáticos que pueden explicar algunos de los fenómenos biológicos estudiados.
- Proporciona soluciones matemáticas que permiten obtener predicciones para los procesos biológicos.

A diferencia de los métodos clásicos de estudio usados por los biólogos basados en el método científico, que se refieren a la confirmación o refutación de hipótesis vía resultados experimentales, la Biología de Sistemas emplea técnicas de predicción rigurosas. Estas técnicas surgen fundamentalmente del uso de modelos matemáticos que describen el comportamiento del sistema. Se caracteriza por la integración de aproximaciones experimentales y computacionales en la comprensión de los sistemas biológicos.[5]

La Biología de Sistemas es una disciplina relativamente nueva, por lo que necesita solucionar problemas y afrontar retos para alcanzar los objetivos esperados.

1.1.2 Retos, restricciones y perspectivas de la Biología de Sistemas

- La formulación de modelos matemáticos más robustos.
- La estandarización de experimentos in-vitro e in-vivo.
- La creación y consolidación de grupos multidisciplinarios de excelencia.
- El desarrollo de nanosensores y dispositivos microfluídicos que permitan la medida simultánea de múltiples parámetros.
- La integración de los modelos cuantitativos y cualitativos es uno de los principales retos de la Biología de Sistemas.

Alcanzar estos retos requiere de potentes herramientas de software que entre otras funcionalidades permitan la integración de modelos gráficos y matemáticos.

1.2 Estándares y formatos

A pesar de las dificultades mencionadas la comunidad internacional ha ideado un conjunto de iniciativas para la estandarización de bases de datos y softwares de modelación entre las que aparecen:

- Lenguaje de hipertexto para la Biología de Sistemas (SBML, por sus siglas en inglés).
- Notación gráfica para la Biología de Sistemas (SBGN, por sus siglas en inglés).
- Ontología para la Biología de Sistemas (SBO, por sus siglas en inglés).
- BioModels.

1.2.1 Lenguaje de hipertexto para la Biología de Sistemas

El Lenguaje de hipertexto para la Biología de Sistemas es un formato de archivo estándar para la descripción de modelos computacionales en la Biología de Sistemas, SBML no es un intento de definir un lenguaje universal de modelos cuantitativos, el propósito de SBML es servir como lengua franca un formato de intercambio utilizado actualmente por las diferentes herramientas para comunicar los

aspectos esenciales de un modelo computacional. Los archivos SBML contienen la información de los modelos representados. Para el trabajo con estos ficheros se puede utilizar la librería LibSBML que permite la lectura, escritura, traducción y la validación de los ficheros SBML.

Al apoyar SBML como formato de entrada y salida, las diferentes herramientas de software logran la misma representación de un modelo posibilitando la eliminación de errores en la traducción y la garantía de un punto de partida común para los análisis y simulaciones. [6]

1.2.2 Notación gráfica para la Biología de Sistemas

Un proyecto recién creado es la Notación gráfica para la Biología de Sistemas en el que aborda la cuestión de la comunicación humana coherente, tratando de añadir más rigor y coherencia a la forma gráfica de los diagramas de red, que a menudo acompañan a la investigación que es publicada sobre los modelos de reacción de los sistemas biológicos. El objetivo de SBGN es estandarizar la representación gráfica visual de los principales procesos bioquímicos y celulares estudiados en la Biología de Sistemas. SBGN define un amplio conjunto de símbolos con precisión semántica, sintáctica, junto con normas que definan su uso. También describe la forma en que tal información gráfica debe ser interpretada. [7]

La normalización de notaciones gráficas para describir las interacciones biológicas es un paso importante para la transmisión eficiente de los conocimientos biológicos entre las diferentes comunidades. Tradicionalmente, los diagramas que representan las interacciones entre los genes y las moléculas se han establecido de manera informal, utilizando formas simples y bordes sin restricciones, como flechas. Hasta el desarrollo de SBGN, no existía un estándar acordado que definiera exactamente cómo dibujar diagramas, la notación gráfica permite a los lectores interpretar de forma coherente, correcta y sin ambigüedades dichos diagramas. [7]

1.2.3 Ontología para la Biología de Sistemas

La Ontología para la Biología de Sistemas es una colección de términos desarrollados por la comunidad de modelización de sistemas biológicos, utilizados para describir las partes del modelo de forma comprensible y controlada. SBML provee un atributo en todos sus componentes destinados a enlazar cada componente con un término SBO. [8]

Modelar sistemas biológicos competentemente requiere de la visualización estándar brindada por

SBGN, del uso controlado de los términos definidos para cada componente de un sistema biológico propuesto por SBO y la posibilidad de exportar e importar los modelos en el formato SMBL por parte de los investigadores.

1.3 Software que modelan sistemas biológicos utilizando SBML, SBGN y SBO

1.3.1 BioUML

BioUML es un framework diseñado para el trabajo con sistemas biológicos que brinda una notación gráfica formalizada para describir el funcionamiento y la estructura de los sistemas biológicos, su visualización y simulación, así como acceder a bases de datos con datos experimentales relevantes[5].

Características:

- Soporte para ecuaciones algebraicas con todas las funciones de MathML.
- Modelación según el estándar SBGN.
- Soporte para SBML nivel 2.

1.3.2 CellDesigner

CellDesigner es un editor de diagramas estructurados para la modelación de genes y redes bioquímicas. Las redes son modeladas a través de diagramas de procesos con la notación propuesta por Hiroaki Kitano y almacenados siguiendo el formato del estándar SBML (Systems Biology Markup Language).
Características del CellDesigner:

- Representación de semánticas bioquímicas.
- Descripción detallada de las transiciones de estados de las proteínas.
- Utiliza el estándar SBML.
- Portabilidad extrema siendo una aplicación desarrollada en Java.

1.4 Necesidad

La mayoría de las herramientas existentes y disponibles en la actualidad poseen los estándares definidos para la modelación de sistemas biológicos por lo que se hace necesario que el software BioSyS incorpore los mismos con el fin de ofrecer un software competitivo y compatible con otras herramientas de modelación a la comunidad de investigadores de la biología de sistemas.

A continuación se describen las características de la metodología de desarrollo, herramientas y lenguajes de programación utilizados para el análisis, diseño e implementación de este tipo de software.

1.5 Metodología de desarrollo

Las metodologías de desarrollo de software describen los pasos para desarrollar un producto de software. Definen detalladamente qué es lo que se debe hacer, cómo y quién debe hacerlo. Precisan cuáles son las tareas a desarrollar durante el ciclo de vida del proyecto, los artefactos que han de ser construidos, cómo deben hacerse y el orden en que serán realizados, además de designar responsables por cada tarea. De esta forma se obtiene como producto final, una solución eficaz, factible y de calidad al problema que le dio origen. La selección de las metodologías a utilizar se hace en base a las necesidades específicas de cada situación y a las características del equipo de desarrollo.

1.5.1 OpenUp

OpenUP es un proceso unificado, abierto, mínimo y suficiente, lo que brinda la posibilidad de solo incluir el contenido fundamental y necesario. Conserva las características principales de la metodología de desarrollo RUP por lo que se aplica de forma iterativa e incremental, provee los componentes básicos que pueden servir de base a procesos específicos a pesar de no tener lineamientos para todos los elementos que se manejan en un proyecto. Es una forma de desarrollo ágil y ligero donde la mayoría de sus elementos fomentan el intercambio entre los equipos de desarrollo, y de colaboración sincronizando intereses y compartiendo conocimientos, principio fundamental que promueve las buenas prácticas para propiciar un ambiente saludable y de colaboración. Maximiza los beneficios al equilibrar las prioridades, permitiendo a los desarrolladores llevar a cabo una solución que cumpla con los requerimientos del proyecto, minimice el riesgo al centrarse en la arquitectura y proporcione la retroalimentación y mejoramiento continuo al realizar prácticas que permitan incrementos progresivos a

las funcionalidades. [9]

Teniendo en cuenta lo antes explicado la metodología que se utilizará para guiar el desarrollo de las funcionalidades es OpenUp debido a sus características y a ser éste modelo el que más se ajusta a las necesidades del desarrollo de la aplicación, además de ser la metodología escogida y utilizada en el desarrollo de versiones anteriores del software BioSys y una decisión del polo de Bioinformática de la facultad.

1.6 Roles y Artefactos

Un rol es una definición abstracta de un conjunto de actividades realizadas y de artefactos obtenidos. Los roles son realizados típicamente por un individuo, o un conjunto de individuos, trabajando juntos en equipo. [10]
La metodología de desarrollo OpenUP define los siguientes roles:

- Jefe de Proyecto es el encargado de planificar y administrar el proyecto y cada una de las iteraciones por las que transita la solución final, se hace responsable por los artefactos Plan de Proyecto, Plan de Iteración, Lista de Riesgos y Lista de Artefactos.

- Arquitecto es el encargado de elaborar el Documento de la arquitectura producto de la definición y refinado la arquitectura.

- Analista es el responsable de encontrar, especificar y detallar los requerimientos, de delimitar el sistema, encontrar los actores y los casos de usos y asegurar que el modelo de casos de usos sea completo y consistente. Los artefactos que serán producidos por el analista serán: Glosario de términos, Especificación de requerimientos, Modelo de Caso de Uso y Documento Visión generados a partir de la definición de la visión del proyecto.

- Desarrollador es el responsable de diseñar e implementar la solución en términos de código fuente y las pruebas de desarrollador, actividades que generan la implementación de la solución. Los artefactos que serán producidos por el analista serán: elementos de implementación que constituyen la parte física de la implementación, incluyen los archivos y directorios. Incluyen ficheros de código (fuentes, binarios o ejecutables), de datos y de documentación como son ficheros de ayuda online.

En el desarrollo del presente trabajo de diploma y de acuerdo a las necesidades actuales del proyecto se desarrollarán los roles de: analista, arquitecto y desarrollador.

1.7 Arquitectura

La arquitectura de software es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos, el ambiente y los principios que orientan su diseño y evolución. Es una vista estructural de alto nivel, ocurre muy tempranamente en el ciclo de vida y define los estilos o grupos de estilos adecuados para cumplir con los requerimientos no funcionales. [11]

1.7.1 Estilo arquitectónico

Un estilo arquitectónico define las reglas generales de organización en términos de un patrón y las restricciones en la forma y la estructura de un grupo numeroso y variado de sistemas de softwares. En una forma más específica, un estilo determina el vocabulario de componentes y conectores que pueden ser utilizados en instancias de este estilo, con un conjunto de restricciones en las descripciones arquitectónicas. [11]

1.7.1.1 Estilos de llamada y retorno

El estilo arquitectónico de llamada y retorno encapsula la Arquitectura en capas, que tiene como objetivo primordial la separación de la lógica de negocios de la lógica de diseño; un ejemplo básico de esto consiste en separar la capa de datos de la capa de presentación al usuario. [11]

La ventaja principal de este estilo es que el desarrollo se puede llevar a cabo en varios niveles y, en caso de que sobrevenga algún cambio, sólo se ataca al nivel requerido sin tener que revisar entre código mezclado. Además, permite distribuir el trabajo de creación de una aplicación por niveles; de este modo, cada grupo de trabajo está totalmente abstraído del resto, de forma que basta con conocer la API que existe entre niveles. [11]

1.8 Patrones

Un patrón es una solución a un problema en un contexto, codifica conocimiento específico acumulado por la experiencia en un dominio. Cada patrón describe un problema que ocurre una y otra vez en el o ambiente, y luego describe el núcleo de la solución a ese problema, de tal manera que puedes usar

esa solución un millón de veces más, sin hacer jamás la misma cosa dos veces. [11]

1.8.1 Patrones arquitectónicos

Los patrones arquitectónicos son aquellos que expresan un esquema organizativo estructural fundamental para sistemas software. [11]

1.8.1.1 Modelo Vista Controlador

Es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. El patrón Modelo Vista Controlador (MVC) se ve frecuentemente en aplicaciones Web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página, el modelo es el Sistema de Gestión de Base de Datos y el controlador representa la Lógica de negocio. [11]

- **Modelo:** Esta es la representación específica de la información con la cual el sistema opera. La lógica de datos asegura la integridad de estos y permite derivar nuevos datos; por ejemplo, no permitiendo comprar un número de unidades negativo, calculando si hoy es el cumpleaños del usuario o los totales, impuestos o importes en un carrito de la compra.
- **Vista:** Este presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario.
- **Controlador:** Este responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista.

Aunque se pueden encontrar diferentes implementaciones de MVC, el flujo que sigue el control generalmente es el siguiente:

- El usuario interactúa con la interfaz de usuario de alguna forma (por ejemplo, el usuario pulsa un botón, enlace)
- El controlador recibe (por parte de los objetos de la interfaz-vista) la notificación de la acción solicitada por el usuario. El controlador gestiona el evento que llega, frecuentemente a través de un gestor de eventos handler o callback.
- El controlador accede al modelo, actualizándolo, posiblemente modificándolo de forma adecuada a la acción solicitada por el usuario. Los controladores complejos están a menudo

estructurados usando un patrón de comando que encapsula las acciones y simplifica su extensión.

- El controlador delega a los objetos de la vista la tarea de desplegar la interfaz de usuario. La vista obtiene sus datos del modelo para generar la interfaz apropiada para el usuario donde se refleja los cambios en el modelo. El modelo no debe tener conocimiento directo sobre la vista. En algunas implementaciones la vista no tiene acceso directo al modelo, dejando que el controlador envíe los datos del modelo a la vista.
- La interfaz de usuario espera nuevas interacciones del usuario, comenzando el ciclo nuevamente.

Para el desarrollo de la aplicación se utilizará el patrón Modelo Vista Controlador por las amplias posibilidades que brinda.

1.8.2 Patrones de casos de uso

La experiencia en la utilización de casos de uso ha evolucionado en un conjunto de patrones que permiten con más precisión reflejar los requisitos reales, haciendo más fácil el trabajo con los sistemas, y mucho más simple su mantenimiento. [12]

1.8.2.1 CRUD

Es un patrón que se basa en la fusión de casos de uso simples para formar una unidad conceptual. [12] Modela las operaciones que pueden ser realizadas sobre una parte de la información de un tipo específico, tales como creación, recuperación, actualización y eliminación[12]. Existen dos variantes de este patrón, Crud completo y Crud parcial.

1.8.2.1.1 CRUD Completo

Con esta variante se modelan las cuatro operaciones básicas que propone el patrón CRUD: crear, recuperar, actualizar y eliminar en secciones separadas dentro de la descripción del caso de uso.

1.8.2.1.2 CRUD Parcial

Este patrón alternativo modela una de las vías de los casos de uso como un caso de uso separado. Es preferiblemente utilizado cuando una de las alternativas de los mismos es más significativa, larga o más compleja que las otras.[12]

Para la modelación de los casos de usos descritos en el presente trabajo de diploma se utilizará la segunda variante del patrón CRUD.

1.8.3 Patrones de diseño

Un patrón de diseño es una descripción de clases y objetos comunicándose entre sí adaptada para resolver un problema de diseño general en un contexto particular. [11]
Los patrones de diseño son clasificados en:

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

1.8.3.1 Patrones Creacionales

1.8.3.1.1 Factory Method

Centraliza en una clase constructora la creación de objetos de un subtipo de un tipo determinado, ocultando al usuario la casuística para elegir el subtipo que crear. [11]

1.8.3.1.2 Singleton

Garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia. [11]

1.8.3.2 Patrones GRASP

Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. GRASP es un acrónimo que significa General Responsibility Asignment Software Patterns (patrones generales de software para asignar responsabilidades). [11]

1.8.3.2.1 Experto

Propone asignar la responsabilidad a la clase que cuenta con la información necesaria para cumplir la responsabilidad. Permitiendo que se conserve el encapsulamiento, soportando un bajo acoplamiento y

una alta cohesión.[13]

1.8.3.2.2 Creador

Propone asignarle a una clase la responsabilidad de crear los objetos de la otra en los casos de contener, agregar, registrar o utilizar. Brindando soporte de bajo acoplamiento, lo cual supone menos dependencias entre clases y posibilidades.[13]

1.8.3.2.3 Bajo Acoplamiento

Brinda como solución asignar responsabilidades de manera que las clases no dependan fuertemente de otras. Ofreciendo como beneficio que son fáciles de entender por separadas, fáciles de reutilizar y no se afectan por cambios de otros componentes.[13]

1.8.3.2.4 Alta Cohesión

Este patrón propone asignar la responsabilidad de manera que la complejidad se mantenga dentro de límites manejables asumiendo solamente las responsabilidades que deben manejar, evadiendo un trabajo excesivo. Su utilización mejora la claridad y facilidad con que se entiende el diseño, simplifica el mantenimiento y las mejoras de funcionalidad, generan un bajo acoplamiento, soporta mayor capacidad de reutilización.[13]

1.8.3.2.5 Controlador

Sugiere asignar la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase que represente un manejador de los eventos del sistema. Este patrón ofrece mayor potencial de los componentes reutilizables garantizando que los procesos de dominio sean manejados por la capa de los objetos del dominio.[13]

1.10 Herramienta CASE

Las herramientas CASE (Computer Aided Software Engineering, Ingeniería de Software Asistida por Ordenador) de modelado con UML brindan la posibilidad de aplicar una metodología de análisis y diseño con abstracción del código fuente, de manera que la arquitectura y el diseño son más obvios y fáciles de comprender y modificar. En la medida que el proyecto es más grande y complejo, resulta más necesario utilizar una herramienta CASE ya que aporta grandes beneficios para todos los involucrados en un proyecto, por ejemplo, jefe del proyecto, analistas, arquitectos, desarrolladores y

otros.

1.10.1 Visual Paradigm

Visual Paradigm es una herramienta CASE que brinda la posibilidad a los desarrolladores de visualizar y diseñar los elementos de software, utiliza como lenguaje de modelado UML y permite a los desarrolladores diseñar un producto de forma rápida. Facilita la interoperabilidad con otras herramientas CASE y se integra con entorno de desarrollo Eclipse.

Debido a que el sistema operativo que se está utilizando es Ubuntu (distribución de Linux) se decide utilizar el Visual Paradigm para visualizar y diseñar los elementos de software, debido a que corre en este sistema operativo y utiliza el Lenguaje Unificado de Modelado (UML). Tiene disponibilidad para disímiles versiones y para integrarse en múltiples plataformas. Esta herramienta necesita de altos requerimientos computacionales para su óptima ejecución.

Características Visual Paradigm:

- Genera código en PHP, Java, C++, y otros. Permite incorporar los dibujos del Visio en cualquier diagrama de UML.
- No solo realiza diagramas de UML normales, sino también posibilita modelar hardware, dominio específico, el software, conectando una red de computadoras los componentes usando sus iconos, más allá de las anotaciones de UML normales.
- Es multiplataforma, está disponible en las plataformas Windows y Linux.
- Está orientada a la creación de diseños usando el paradigma de programación orientado a objetos.

1.11 Lenguaje de modelado

Los lenguajes de modelado permiten la modelación de objetos mediante un conjunto estandarizados de símbolos y técnicas de modelado. Modelan una simplificación de la realidad donde se toma de los objetos los aspectos significativos brindando así una mejor comprensión del sistema.

1.11.1 UML

El UML (Unified Modeling Language) está considerado como el lenguaje de modelado gráfico y visual

por excelencia, utilizado para especificar, visualizar, construir y documentar los componentes de un sistema de software. Está pensado para poder aplicarse en cualquier medio que necesite capturar requerimientos y comportamientos del sistema que se desee construir. Ayuda a comprender y a mantener de una mejor forma un sistema basado en un área que el analista o desarrollador puede desconocer. UML surgió con el propósito de lograr una estandarización universal al crecido número de metodologías de desarrollo que habían estado apareciendo.

Este lenguaje permite captar información sobre la estructura estática y dinámica de un sistema, en donde la estructura estática proporciona información sobre los objetos que intervienen en determinado proceso y las relaciones que existen entre ellos, y la estructura dinámica define el comportamiento de los objetos a lo largo de todo el tiempo que estos interactúan hasta llegar a su o sus objetivos. Una característica sobresaliente del UML es que no es un método, sino un lenguaje de modelado. Un método define su notación (lenguaje) y su proceso a seguir durante el ciclo de vida de desarrollo del software. Se encarga de la notación gráfica y su significado, a partir de la cual se crearán los diseños de sistemas y no depende de un proceso, el cual sería el encargado de orientar los pasos a seguir para elaborar el diseño. Mejorar la comunicación es la idea a seguir con la creación de los diagramas mediante UML, sin dudas ayuda a que los desarrolladores de software se comuniquen con un mismo lenguaje de modelado independiente de las metodologías empleadas. Por esta razón la mayoría de las fuentes coinciden en que la principal ventaja de UML sobre otras notaciones OO es que elimina las diferencias entre semánticas y notaciones.

UML tiene como características:

- Tecnología orientada a objetos
- Viabilidad en la corrección de errores.
- Permite especificar todas las decisiones de análisis, diseño e implementación, construyéndose así modelos precisos, no ambiguos y completos.
- Puede conectarse con lenguajes de programación (Ingeniería directa e inversa).
- Permite documentar todos los artefactos de un proceso de desarrollo (requisitos, arquitectura, pruebas, versiones, etc.).
- Cubre las cuestiones relacionadas con el tamaño propio de los sistemas complejos y críticos.
- Es un lenguaje muy expresivo que cubre todas las vistas necesarias para desarrollar y luego desplegar los sistemas.

- Existe un equilibrio entre expresividad y simplicidad, pues no es difícil de aprender ni de utilizar.

El lenguaje de modelado escogido para la modelación del sistema es UML debido sus características y su alto grado de comprensión. Además de ser el lenguaje de modelado utilizado actualmente por el equipo de desarrollo del software BioSyS.

1.12 Lenguaje de programación

1.12.1 Java

Java ofrece todas las ventajas de un lenguaje potente y robusto, pues fue diseñado para crear software altamente fiable. Es un lenguaje de programación basado en clases y orientado a objetos. Sus características de memoria liberan a los programadores de responsabilidades y errores. Es compilado en un código intermedio más abstracto que el código de máquina que es ejecutado por la máquina virtual de Java. Hereda su sintaxis de los lenguajes C y C++, pero cuenta con un modelo de objetos mucho más sencillo y elimina elementos de trabajo a bajo nivel como los punteros. [14]

Una de las características más significativas de Java es que posee una arquitectura neutral, es decir, el compilador Java compila su código a un fichero objeto de formato independiente de la arquitectura de la máquina en que se ejecutará. La independencia de la arquitectura representa solo una parte de su portabilidad. Java especifica los tamaños de sus tipos de datos básicos y el comportamiento de sus operadores aritméticos, de manera que los programas son iguales en todas las plataformas. [14]

El lenguaje de programación a utilizar para la implementación de las nuevas funcionalidades es java por todas las ventajas que ofrece y por ser el lenguaje escogido y utilizado actualmente en el desarrollo del software BioSyS.

1.13 Entorno de Desarrollo Integrado (IDE)

Un Entorno de Desarrollo Integrado (IDE, en inglés, Integrated Development Environment) es un programa compuesto por un conjunto de herramientas para un programador, como puede ser: un editor de código, un compilador, un depurador y un constructor de interfaz gráfica. Los mismos proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación, y cabe la posibilidad que un mismo IDE pueda funcionar con varios lenguajes de programación. Existen diferentes IDE como Zend Studio, PHP Editor y Eclipse.

1.13.1 Eclipse

El IDE Eclipse una potente herramienta universal de entorno de desarrollo de software desarrollado en Java, usa java como lenguaje de programación aunque permite plugins para varios lenguajes. La plataforma está construida en base a plugins, mecanismo que permite desarrollar, integrar y correr nuevos plugins. [15]

Otros beneficios que aporta el uso de Eclipse son:

- Es una herramienta de código abierto.
- Eclipse es neutral y adaptable a cualquier tipo de lenguaje, por ejemplo C/C++, Cobol, C#, XML, etc. La característica clave de Eclipse es la extensibilidad.
- Soporta la programación orientada a objetos (POO).
- La depuración e implementación de aplicaciones resultan mucho más sencillas.
- Corre en una gran cantidad de sistemas operativos incluyendo Windows y Linux.
- Provee a los desarrolladores, herramientas (ej.- PDE) que facilitan la creación de plugins.

El IDE de desarrollo escogido para la realización del presente trabajo es Eclipse por sus grandes potencialidades y sus posibilidades de extensión mediante el uso de plugins. Otra razón es el conocimiento y experiencia del equipo de desarrollo de software BioSyS en el trabajo con este IDE, que ha usado este IDE en múltiples aplicaciones, por lo que se tiene experiencia en su uso.

1.14 Bibliotecas

1.14.1 Jgraph

Jgraph es una biblioteca de código abierto en java para la visualización de gráficos. Sigue los patrones de diseño Swing. Tiene capas jerárquicas para volúmenes de trabajo ejemplo capas de árbol para gráficos de organizaciones. Entre sus características se destaca su capacidad para gestionar diagramas de múltiples tipos, zoom, visualización en árbol, deshacer, arrastrar y soltar. Puede trabajar

con diferentes sistemas como: XML, ficheros y bases de datos. Es compatible con las convenciones de código de Java, y puede funcionar tanto en aplicaciones de escritorio como en aplicaciones de servidor.

1.14 Framework

1.14.1 JGraph Editor Framework

Para el desarrollo de la nueva aplicación que permitirá modelar de forma gráfica sistemas biológicos se utilizará JGraph Editor 6.0.7 pues es un framework que simplifica el desarrollo de la misma al presentar elementos que garantizan un trabajo factible con los elementos visuales. JGraph Editor es un framework de código abierto implementado con el lenguaje java y extiende las funcionalidades de la librería JGraph. Su diseño está encaminado a estandarizar la interfaz de usuario, y a separar la estructura del contenido. Contiene las partes comunes de todos los editores que usan JGraph y define las acciones y las herramientas para la creación de una interfaz de usuario configurable atreves de archivos XML. Utiliza el patrón arquitectónico Modelo Vista Controlador lo que posibilita obtener varias vistas de un mismo modelo. Brinda la facilidad de añadir acciones, herramientas y componentes configurables a la aplicación a través de plug-in.

Características:

- La interfaz del usuario, incluyendo todos los menús, las barras de herramientas y las cajas de herramientas está definida en archivos XML facilitando la configuración.
- Las aristas pueden estar relacionadas con otras aristas, característica puede ser usada para implementar casos especiales en diversos diagramas.
- Incluye un componente que exhibe una vista minúscula del diagrama.
- Posibilita mover los puertos de conexión alrededor del perímetro de la figura.
- Las vistas las celdas pueden ser redefinidas.

1.15 Conclusiones

Para el desarrollo del sistema se utilizará como metodología OpenUP, la misma define cuales serán las tareas por roles y los artefactos que serán obtenidos por cada unos de ellos, el lenguaje de modelado UML debido su alto grado de comprensión y seguir el paradigma de programación orientado a objetos, la herramienta CASE a utilizar para visualizar y diseñar los elementos del sistema será Visual Paradim ya que es multiplataforma y utiliza como lenguaje de modelado UML, como entorno

de desarrollo Eclipse y lenguaje de programación Java.

Capítulo 2. Características del Sistema

El presente capítulo contiene la descripción del modelo de dominio perteneciente a la representación de un modelo biológico, los requisitos funcionales y no funcionales del sistema, el diagrama de casos de uso y las descripciones textuales detalladas de los casos de uso.

2.1 Modelado del Dominio

Para comprender la estructura y la dinámica de la organización, los problemas actuales, sus posibles mejoras y derivar los requerimientos del sistema que va a soportar la organización, se hace necesario hacer una evaluación de la organización, para ello se realizará el modelamiento de dominio. El modelo del dominio captura los tipos más importantes de objetos que existen o los eventos que suceden en el entorno donde estará el sistema.

2.1.1 Descripción de Clases del Modelo del Dominio

La representación de un sistema biológico, esta expresada mediante un modelo biológico que puede contener compartimientos, especies, reacciones químicas, funciones matemáticas y operadores lógicos.
El modelo biológico es único para cada sistema biológico que se desee modelar, los compartimientos se encuentran dentro del modelo biológico y son los contenedores de los procesos bioquímicos y celulares ocurrentes entre las diferentes especies del sistema biológico.

Las especies representan: químicos simples, macromoléculas, ácidos nucleícos o entidades no especificadas de las cuales el tipo no es conocido o simplemente el propósito no es relevante para el modelo. El complejo es la forma abstracta utilizada para agrupar diferentes especies y otros complejos, el resultado final será una entidad que posee las propiedades de especie.

Las reacciones químicas e interacciones ocurrentes entre las diferentes especies pueden ser: de consumo, producción, catálisis, inhibición, modulación, disparadoras y de estimulación. Las funciones matemáticas pueden ser de diferentes tipos: reglas, que describen las restricciones que existen entre variables de distintas, leyes cinéticas, regulan la velocidad a las que se producen las reacciones químicas. Los parámetros son variables utilizadas en las formulas matemáticas que pueden ser expresados en diferentes unidades.
Tanto los complejos y las especies en ocasiones pueden relacionarse dando como resultado final un

nuevo grupo de complejos o especies, tales acciones son modeladas mediante los operadores lógicos AND, OR y NOT. El operador AND es utilizado cuando las dos especies relacionadas dan como resultado el tercero, el OR se utiliza cuando una de las especies relacionadas puede dar como resultado un tercero y el NOT para denotar que la especie no puede producir ninguna salida.

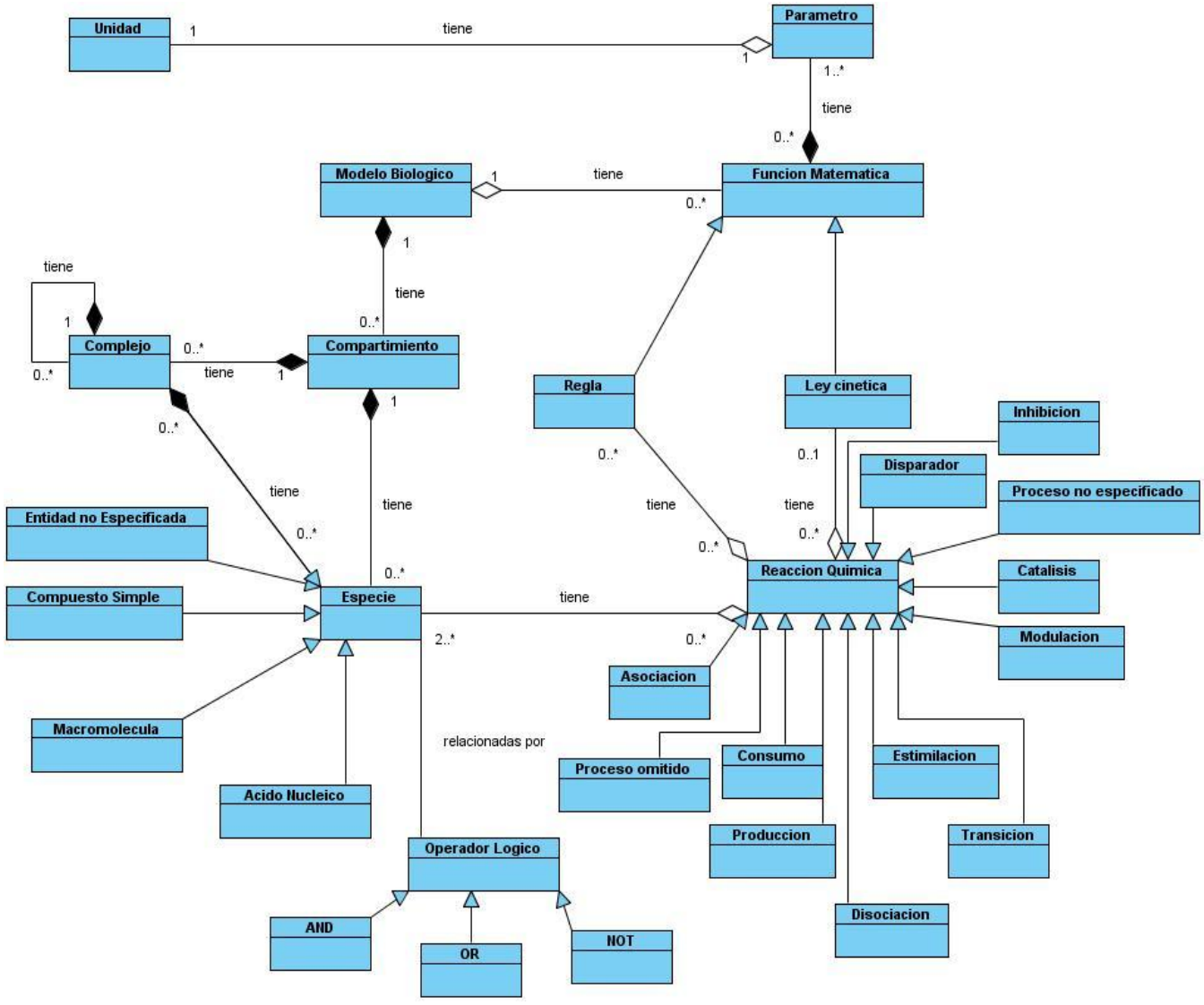


Figura 2. Modelo de Dominio.

2.2 Actores del Sistema

Los actores son usuarios externos que interactúan con el sistema, pueden representar un rol o intercambiar información y no son parte de él. El actor identificado para interactuar con la herramienta a desarrollar es el Investigador.

Actor(s)	Justificación
Investigador	Es la persona que interactúa con el sistema. Elabora el modelo gráfico del sistema biológico, procesa sus datos y realiza los estudios correspondientes.

Tabla 1: Actores del sistema.

2.3 Modelado del Sistema

2.3.1 Requisitos funcionales

Los requisitos funcionales son las capacidades o condiciones que el sistema tiene que cumplir para su correcto funcionamiento, la herramienta a desarrollar debe cumplir con los siguientes requisitos funcionales:

RF 1 Modelar gráficamente sistemas biológicos.
RF 1.1 Modelar gráficamente compartimientos.
RF 1.1.1 Insertar compartimientos.
RF 1.1.2 Modificar las propiedades de los compartimientos.
RF 1.1.3 Eliminar compartimientos.

RF 1.2 Modelar gráficamente especies.
RF 1.2.1 Insertar especies.
RF 1.2.2 Modificar las propiedades de las especies.
RF 1.2.3 Eliminar especies.

RN 1.3 Modelar gráficamente reacciones químicas entre las distintas especies.
RF 1.3.1 Insertar reacciones.
RF 1.3.2 Modificar las propiedades de las reacciones.

RF 1.3.3 Eliminar reacciones.
RF 1.4 Modelar gráficamente operadores lógicos.
RF 1.4.1 Insertar operadores.
RF 1.4.2 Eliminar operadores.

RF 1.5 Modelar gráficamente modificadores.
RF 1.5.1 Insertar modificador.
RF 1.5.2 Eliminar modificador.

RF 1.6 Modelar gráficamente reactantes.
RF 1.6.1 Insertar reactante.
RF 1.6.2 Eliminar reactante.

RF 1.7 Modelar gráficamente productos.
RF 1.7.1 Insertar producto.
RF 1.7.2 Eliminar producto.

RF 2 Mostrar propiedades de los componentes del sistema biológico.
RF 3 Guardar la información del sistema biológico en formato SBML.
RF 4 Permitir el trabajo con una Biblioteca de funciones.

2.3.2 Requisitos no funcionales

Los requerimientos no funcionales son las propiedades o las cualidades que el producto debe tener, para que sea más atractivo, usable, rápido y confiable. Entre los requisitos no funcionales que la herramienta debe tener se encuentran:

2.3.2.1 Usabilidad

RNF1 El sistema podrá ser usado por aquellos usuarios que posean conocimientos básicos en el campo de la modelación de sistemas biológicos.

2.3.2.2 Soporte

RNF 2 Mantenimiento: El sistema debe estar bien documentado de forma tal que el tiempo de mantenimiento sea mínimo en caso de necesitarse.

RNF 3 Instalación: La instalación del sistema debe caracterizarse por su facilidad, claridad y sencillez.

RNF 4 Portabilidad: El sistema debe ser multiplataforma (ser capaz o caracterizarse por poder funcionar o mantener una interoperabilidad de forma similar en diferentes sistemas operativos o plataformas).

2.3.2.3 Restricciones de diseño

RNF 5 El sistema debe cumplir con el estándar de Notación Gráfica de Sistemas Biológicos y utilizar las funcionalidades de la librería libsbml.jar. El lenguaje de programación a utilizar en la implementación debe ser Java.

2.3.2.4 Requisitos de software

RNF 6 Para el uso del sistema es necesario tener instalado la máquina virtual de java 1,5 o una versión superior.
Es necesaria la utilización de esta versión de la maquina virtual debido a que la librería libsbml.jar la requiere.

2.3.2.5 Interfaces de usuario

RNF 7 Se necesita una interfaz amigable, legible, interactiva, fácil de usar, profesional, clara y sencilla.

2.3.3 Diagrama de Casos de Uso del Sistema

Los casos de uso son procesos que dan un resultado de valor para un actor determinado, son artefactos narrativos que describen en forma de acciones y reacciones, el comportamiento del sistema desde el punto de vista del usuario. Los diagramas de casos de uso del sistema representan gráficamente los procesos y su interacción con los actores. El siguiente diagrama corresponde al sistema a desarrollar.

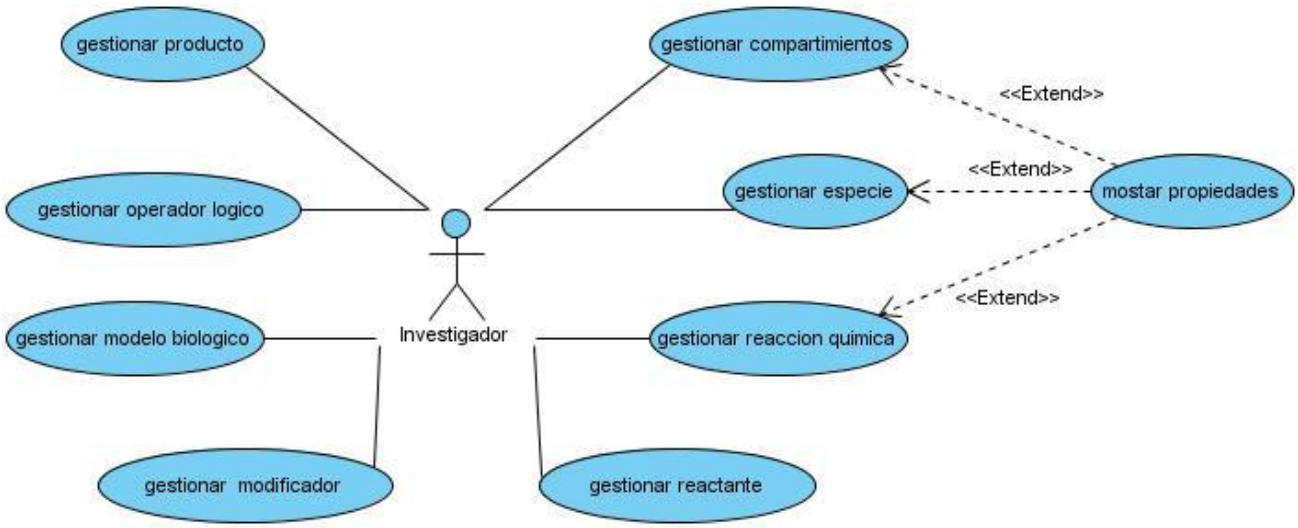


Figura 3. Diagrama de Casos de Uso del Sistema.

2.3.3.1 Descripción textual de Casos de Uso del Sistema

La comprensión de las funcionalidades del sistema no es suficiente con la representación gráfica, Diagrama de casos de uso, es necesaria también una descripción textual de forma detallada de cada una de las interacciones ocurrentes dentro de cada sección. Para ello se describen de forma textual los casos de uso correspondientes al sistema.

2.3.3.1.1 Caso de uso gestionar compartimiento

Caso de Uso:	Gestionar compartimento
Actores:	investigador (Inicia)
Resumen:	El caso de uso se inicia cuando el investigador va a realizar una de las siguientes acciones: • Insertar compartimiento: cuando el investigador inserta un nuevo compartimiento al modelo y finaliza cuando el sistema visualiza el compartimiento y actualiza el modelo. • Modificar Propiedades de compartimiento: cuando el investigador modifica las propiedades del compartimiento finalizando así el CU. • Eliminar compartimiento: cuando el investigador elimina el compartimiento del modelo finalizando así el CU.

<table>
<tr><td>Precondiciones:</td><td></td></tr>
<tr><td>Referencias</td><td>RF 1, RF 1.1 , RF 1.1.2 y RF 1.1.3</td></tr>
<tr><td>Prioridad</td><td>Crítico.</td></tr>
<tr><td colspan="2">Flujo Normal de Eventos</td></tr>
<tr><td colspan="2">Sección “General”</td></tr>
<tr><td>Acción del Actor</td><td>Respuesta del Sistema</td></tr>
<tr><td>1. El investigador selecciona la acción que desea realizar:
Insertar compartimiento
Modificar propiedades de compartimiento
Eliminar compartimiento</td><td>2. El sistema en dependencia de la acción que el investigador realice:
Insertar compartimiento: ir a la sección “Insertar compartimiento”
Mostrar propiedades de compartimiento: ir a la sección “Modificar propiedades compartimiento”.
Eliminar compartimiento: ir a la sección “Eliminar compartimiento”</td></tr>
<tr><td colspan="2">Sección “Insertar compartimiento”</td></tr>
<tr><td>1. El investigador selecciona el componente nuevo modelo biológico en la barra de herramientas.</td><td>2. El sistema agrega el compartimiento al modelo biológico y lo visualiza. Finalizando así el CU.</td></tr>
<tr><td colspan="2">Sección “Modificar propiedades compartimiento”</td></tr>
<tr><td></td><td>1. El sistema muestra una venta con los campos editables (Name, Spatial Dimensions, Size, Unit, Constant).</td></tr>
<tr><td>2. El investigador modifica los datos que desea del compartimiento.</td><td>3. El sistema guarda los cambios y finaliza el CU.</td></tr>
</table>

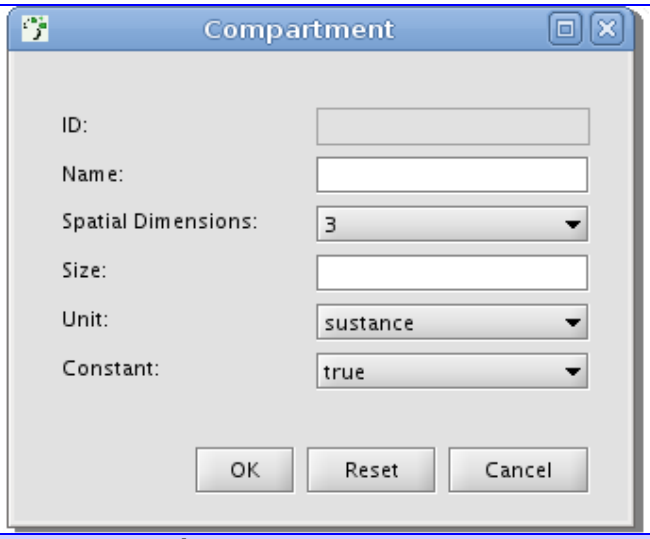
<table>
<tr><td colspan="2"> 
</td></tr>
<tr><td colspan="2">Sección“Eliminar compartimiento”</td></tr>
<tr><td></td><td>1. El sistema elimina el compartimiento y todo lo que contiene, actualiza el modelo y visualiza el cambio finalizando así el CU.</td></tr>
<tr><td>Poscondiciones</td><td>En dependencia de la acción del investigador:
Se inserta un compartimiento al modelo.
Se modifican las propiedades de un compartimiento.
Se elimina un compartimiento.</td></tr>
</table>

Tabla 2: Descripción del Caso de Uso gestionar compartimiento.

2.3.3.1.2 Caso de Uso gestionar especie

Caso de Uso:	Gestionar especie.
Actores:	investigador (Inicia)
Resumen:	El caso de uso se inicia cuando el investigador va a realizar una de las siguientes acciones: - Insertar especie: El caso de uso se inicia cuando el investigador inserta una nueva especie al modelo y finaliza cuando el sistema visualiza la especie y actualiza el modelo. - Modificar propiedades de la especie: cuando el investigador modifica las propiedades de la especie finalizando así el CU. - Eliminar especie: cuando el investigador elimina la especie del modelo

	finalizando así el CU.
Precondiciones:	Que exista un compartimiento
Referencias	RF 1, RF 1.2.1, RF 1.2.2 y RF 1.2.3
Prioridad	Crítico.

Flujo Normal de Eventos	
Sección "General"	
Acción del Actor	**Respuesta del Sistema**
1. El investigador selecciona la acción que desea realizar: • Insertar especie. • Modificar propiedades de la especie. • Eliminar especie.	**2.** El sistema en dependencia de la acción que el investigador realice: • Insertar especie: ir a la sección "Insertar especie". • Modificar propiedades de la especie: ir a la sección "Modificar propiedades de la especie". • Eliminar especie: ir a la sección "Eliminar especie".
Sección "Insertar especie**"**	
1. El investigador selecciona el componente especie del conjunto de especies que se muestran en la paleta de componentes y da clic en el área de modelado.	**2.** El sistema agrega la especie al modelo biológico y lo visualiza finalizando así el CU.
Sección "Modificar propiedades de la especie**"**	
	1. El sistema muestra una ventana con los campos editables (Name, Compartment, Initial Amount, Initial Concentration, Substance Units, Has Only Substance Units, Boundary Condition, Charge y Constant).
2. El investigador modifica los datos que desea de la especie seleccionada.	**3.** El sistema guarda los cambios y finaliza el CU.

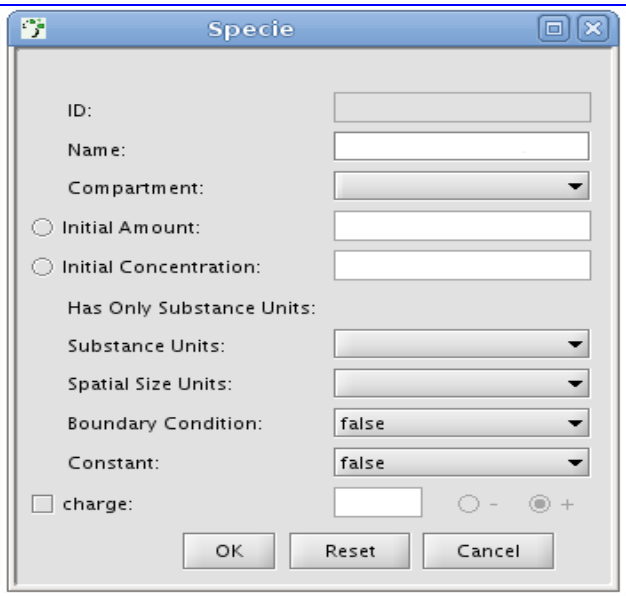
<table>
<tr><td colspan="3"> 
</td></tr>
<tr><td colspan="3">Sección“Eliminar especie”</td></tr>
<tr><td colspan="2"></td><td>1. El sistema verifica si la especie seleccionada es parte de una reacción química.</td></tr>
<tr><td colspan="2"></td><td>2. El sistema elimina la especie, actualiza el modelo y visualiza el cambio finalizando así el CU.</td></tr>
<tr><td colspan="3">Flujos Alternos</td></tr>
<tr><td colspan="2">Acción del Actor</td><td>Respuesta del Sistema</td></tr>
<tr><td colspan="2">1.1 Si se parte del punto 1 de la sección Eliminar Especie del flujo normal de eventos.</td><td>1.2 El sistema elimina las reacciones modificadoras y la especie seleccionada.
1.3 El sistema elimina la referencia de la especie seleccionada de la reacción química en la que participa.</td></tr>
<tr><td>Poscondiciones</td><td colspan="2">En dependencia de la acción del Investigador:
Se inserta una especie.
Se modifican las propiedades de una especie.
Se elimina una especie.</td></tr>
</table>

Tabla 3: Descripción del Caso de Uso gestionar especie.

2.3.3.1.3 Caso de uso gestionar reacción química

Caso de Uso:	Gestionar reacción química.
Actores:	investigador (Inicia)
Resumen:	El caso de uso se inicia cuando el investigador va a realizar una de las siguientes acciones: • Insertar reacción química: cuando el investigador decide insertar una nueva reacción química al modelo y finaliza cuando la reacción queda incluida en el modelo y este se actualiza. • Modificar propiedades de la reacción química: cuando el investigador modifica las propiedades de la reacción química finalizando así el CU. • Eliminar reacción química: cuando el investigador elimina la reacción química del modelo finalizando así el CU.
Precondiciones:	
Referencias	RF 1, RF 1.3, RF 1.3.1, RF 1.3.2, RF 1.3.3
Prioridad	Crítico.

Flujo Normal de Eventos	
Sección "General"	
Acción del Actor	**Respuesta del Sistema**
1. El investigador selecciona la acción que desea realizar: • Insertar reacción química. • Modificar propiedades de la reacción química • Eliminar reacción química.	**2.** El sistema en dependencia de la acción que el investigador realice: • Insertar reacción química: ir a la sección "Insertar reacción química" . • Modificar propiedades de la reacción química: ir a la sección "Modificar propiedades de la reacción química". • Eliminar reacción química: ir a la sección "Eliminar reacción química"
Sección "Insertar reacción química**"**	
1. El investigador selecciona en el menú, correspondiente a las reacciones químicas, que se	**2.** El sistema agrega la reacción química al modelo biológico y lo visualiza. Finalizando así

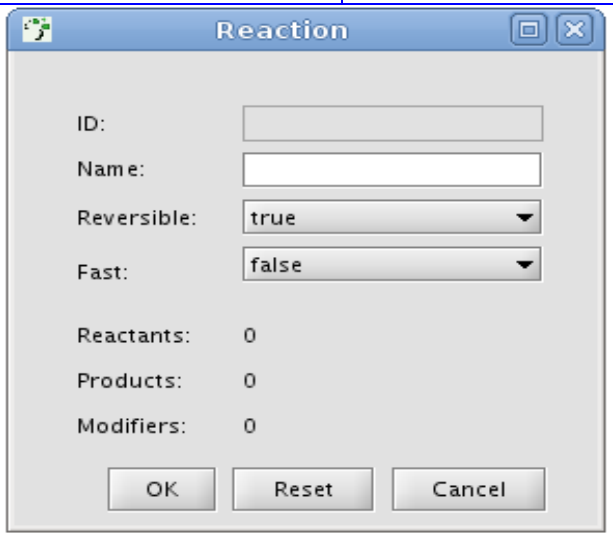
encuentra en la paleta de componentes, la reacción química y da clic en el área de modelado.	el CU.
Sección "Modificar propiedades de la reacción química**"**	
	1. El sistema muestra una ventana con los campos editables (name, Reversible, fast).
2. El investigador modifica los datos que desea de la reacción química seleccionada.	**3.** El sistema guarda los cambios y finaliza el CU.
	
Sección"Eliminar reacción química**"**	
	1. El sistema elimina la reacción química, actualiza el modelo y visualiza el cambio finalizando así el CU.
Poscondiciones	En dependencia de la acción del Investigador: • Se inserta una reacción química. • Se modifican las propiedades de una reacción química. • Se elimina una reacción química.

Tabla 4: Descripción del Caso de Uso gestionar reacción química.

2.3.3.1.4 Caso de Uso gestionar operador lógico

Caso de Uso:	gestionar operador lógico
Actores:	investigador (Inicia)

<table>
<tr><td>Resumen:</td><td colspan="2">El caso de uso se inicia cuando el investigador va a realizar una de las siguientes acciones:
Insertar operador lógico: cuando el investigador inserta un operador y finaliza cuando el sistema visualiza el operador lógico.
Eliminar operador lógico: cuando el investigador elimina un operador lógico, el sistema visualiza el cambio finalizando así el CU</td></tr>
<tr><td>Precondiciones:</td><td colspan="2"></td></tr>
<tr><td>Referencias</td><td colspan="2">RF1, RF 1.4, FR 1.4.1, RF 1.4.2</td></tr>
<tr><td>Prioridad</td><td colspan="2">Secundario.</td></tr>
<tr><td colspan="3">Flujo Normal de Eventos</td></tr>
<tr><td colspan="3">Sección “General”</td></tr>
<tr><td>Acción del Actor</td><td colspan="2">Respuesta del Sistema</td></tr>
<tr><td>1. El investigador selecciona la acción que desea realizar:
Insertar operador lógico.
Eliminar operador lógico.</td><td colspan="2">2. El sistema en dependencia de la acción que el investigador realice:
Insertar operador lógico: ir a la sección “Insertar operador lógico”.
Eliminar operador lógico: ir a la sección “Eliminar operador lógico”.</td></tr>
<tr><td colspan="3">Sección “Insertar operador lógico”</td></tr>
<tr><td>1. El investigador selecciona el componente operador lógico y da clic en el área de modelado.</td><td colspan="2">2. El sistema agrega el operador lógico y lo visualiza. Finaliza el CU</td></tr>
<tr><td colspan="3">Sección “Eliminar operador lógico”</td></tr>
<tr><td>1. El investigador da clic derecho sobre el operador lógico en el área de modelado y selecciona que acción eliminar.</td><td colspan="2">2. El sistema elimina el operador lógico y visualiza el cambio finalizando así el CU.</td></tr>
<tr><td>Poscondiciones</td><td colspan="2">En dependencia de la acción del Investigador:
Se elimina un operador lógico.
Se agrega un operador lógico.</td></tr>
</table>

Tabla 5: Descripción del Caso de Uso gestionar operador lógico.

2.3.3.1.5 Caso de Uso gestionar modificador

Caso de Uso:	gestionar modificador
Actores:	investigador (Inicia)
Resumen:	El caso de uso se inicia cuando el investigador va a realizar una de las siguientes acciones: • Insertar modificador: cuando el investigador va a insertar una especie como modificador a una reacción química y finaliza cuando la especie es insertada como modificadora de la reacción el sistema visualiza el cambio. • Eliminar modificador: cuando el investigador va a eliminar la reacción que hace que una especie actué como modificador en una reacción química y finaliza cuando dicha reacción es eliminada.
Precondiciones:	
Referencias	RF 1, RF 1.5, RF 1.5.1, RF 1.5.2
Prioridad	Secundario.

Flujo Normal de Eventos	
Sección "General"	
Acción del Actor	**Respuesta del Sistema**
1. El investigador selecciona la acción que desea realizar: • Insertar modificador. • Eliminar modificador.	El sistema en dependencia de la acción que el investigador realice: • Insertar modificador: ir a la sección "Insertar modificador". • Eliminar modificador: ir a la sección "Eliminar modificador".
Sección "Insertar modificador"	
	3. El sistema visualiza la reacción, agrega la especie a la lista de modificadores de la reacción actualiza el modelo y finaliza el CU
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**

2.1 si se parte del punto 2 de la sección "Insertar modificador" del flujo normal de eventos.	**2.2** El sistema muestra un mensaje de error " Incorrect Association" y elimina la reacción insertada **2.3** El sistema va a la acción 1.1 de esta sección.
Sección "Eliminar modificador"	
1. El investigador da clic derecho sobre la reacción que hace que una especie actué como modificador en una reacción química en el área de modelado y selecciona la acción eliminar.	**2.** El sistema elimina la especie de la lista de modificadores de la reacción, actualiza el modelo, elimina la reacción seleccionada y visualiza el cambio y finaliza el CU.
Poscondiciones	En dependencia de la acción del Investigador: • Se inserta un modificador a la reacción química. • Se elimina un modificador de la reacción química.

Tabla 6: Descripción del Caso de Uso gestionar modificador.

2.3.3.1.6 Caso de Uso gestionar reactante

Caso de Uso:	gestionar reactante
Actores:	investigador (Inicia)
Resumen:	El caso de uso se inicia cuando el investigador va a realizar una de las siguientes acciones: • Insertar reactante: cuando el investigador va a insertar una especie como reactante a una reacción química • Eliminar reactante: cuando el investigador va a eliminar la arista que hace que una especie actué como reactante en una reacción química y finaliza cuando dicha arista es eliminada.
Precondiciones:	
Referencias	RF 1, RF 1.6, RF 1.6.1, RF 1.6.2
Prioridad	Secundario.
Flujo Normal de Eventos	
Sección "General"	
Acción del Actor	**Respuesta del Sistema**
1. El investigador selecciona la acción que desea realizar:	El sistema en dependencia de la acción que el investigador realice:

• Insertar reactante. • Eliminar reactante.	• Insertar reactante: ir a la sección "Insertar reactante". • Eliminar reactante: ir a la sección "Eliminar reactante".
Sección "Insertar reactante"	
1. El investigador selecciona en la paleta de componentes la arista que permite agregar una especie como reactante a la reacción existente. **1.1** El investigador selecciona la especie origen y la reacción química destino.	**2.** El sistema comprueba que el origen y destino son validos
	3. El sistema visualiza la arista, agrega la especie a la lista de reactantes de la reacción actualiza el modelo y finaliza el CU
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
2.1 si se parte del punto 2 de la sección "Insertar reactante" del flujo normal de eventos.	**2.2** El sistema muestra un mensaje de error "Incorrect Association" y elimina la arista insertada **2.3** El sistema va a la acción 1.1 de esta sección.
Sección "Eliminar reactante"	
1. El investigador da clic derecho sobre la arista que une la especie con la reacción química y selecciona la acción eliminar.	**2.** El sistema elimina la especie de la lista de reactantes de la reacción, actualiza el modelo, elimina la arista seleccionada y visualiza el cambio y finaliza el CU.
Poscondiciones	En dependencia de la acción del Investigador: • Se inserta un reactante a la reacción química. • Se elimina un reactante de la reacción química.

Tabla 7: Descripción del Caso de Uso gestionar reactante.

2.3.3.1.7 Caso de Uso gestionar producto

Caso de Uso:	gestionar producto
Actores:	investigador (Inicia)

<table>
<tr><td>Resumen:</td><td colspan="2">El caso de uso se inicia cuando el investigador va a realizar una de las siguientes acciones:
Insertar producto: cuando el investigador va a insertar una especie como producto a una reacción química
Eliminar producto: cuando el investigador va a eliminar la arista que hace que una especie sea el producto en una reacción química y finaliza cuando dicha arista es eliminada.</td></tr>
<tr><td>Precondiciones:</td><td colspan="2"></td></tr>
<tr><td>Referencias</td><td colspan="2">RF 1, RF 1.7, RF 1.7.1, RF 1.7.2</td></tr>
<tr><td>Prioridad</td><td colspan="2">Secundario.</td></tr>
<tr><td colspan="3">Flujo Normal de Eventos</td></tr>
<tr><td colspan="3">Sección “General”</td></tr>
<tr><td>Acción del Actor</td><td colspan="2">Respuesta del Sistema</td></tr>
<tr><td>1. El investigador selecciona la acción que desea realizar:
Insertar producto.
Eliminar producto.</td><td colspan="2">El sistema en dependencia de la acción que el investigador realice:
Insertar producto: ir a la sección “Insertar producto”.
Eliminar producto: ir a la sección “Eliminar producto”.</td></tr>
<tr><td colspan="3">Sección “Insertar producto”</td></tr>
<tr><td>1. El investigador selecciona en la paleta de componentes la arista que permite agregar una especie como producto a la reacción existente.
1.1 El investigador selecciona la reacción origen y la especie destino.</td><td colspan="2">2. El sistema comprueba que el origen y destino son validos</td></tr>
<tr><td></td><td colspan="2">3. El sistema visualiza la arista, agrega la especie a la lista de productos de la reacción actualiza el modelo y finaliza el CU</td></tr>
<tr><td colspan="3">Flujos Alternos</td></tr>
<tr><td>Acción del Actor</td><td colspan="2">Respuesta del Sistema</td></tr>
<tr><td>2.1 si se parte del punto 2 de la sección “Insertar</td><td colspan="2">2.2 El sistema muestra un mensaje de error</td></tr>
</table>

producto" del flujo normal de eventos.	"Incorrect Association" y elimina la arista insertada **2.3** El sistema va a la acción 1.1 de esta sección.
Sección "Eliminar producto"	
1. El investigador da clic derecho sobre la arista que une la especie con la reacción química y selecciona la acción eliminar.	**2.** El sistema elimina la especie de la lista de productos de la reacción, actualiza el modelo, elimina la arista seleccionada y visualiza el cambio y finaliza el CU.
Poscondiciones	En dependencia de la acción del Investigador: • Se inserta un producto a la reacción química. • Se elimina un producto de la reacción química.

Tabla 8: Descripción del Caso de Uso gestionar producto.

2.3.3.1.8 Caso de Uso mostrar propiedades

Caso de Uso:	mostrar propiedades.
Actores:	investigador (Inicia)
Resumen:	El caso de uso se inicia cuando el investigador decide ver las propiedades de cualquier componente del modelo biológico el sistema muestra las propiedades y finaliza el CU.
Precondiciones:	
Referencias	RF2
Prioridad	Secundario
Flujo Normal de Eventos	
Sección "General"	
Acción del Actor	**Respuesta del Sistema**
1. El actor selecciona el componente deseado en el área de modelado haciendo sobre el mismo.	2. El sistema muestra las propiedades. Finalizando el CU.
Poscondiciones	Se elimina el operador lógico.

Tabla 9: Descripción del Caso de Uso mostrar propiedades.

2.3.3.1.9 Caso de Uso gestionar modelo biológico

Caso de Uso:	gestionar modelo biológico
Actores:	investigador (Inicia)

<table>
<tr><td>Resumen:</td><td>El caso de uso se inicia cuando el investigador va a realizar una de las siguientes acciones:
Abrir modelo biológico: el investigador decide abrir un modelo existente, el sistema abre el modelo seleccionado y finaliza el CU.
Guardar modelo biológico: el investigador decide guardar el modelo biológico previamente creado, el sistema guarda el modelo y finaliza el CU.</td></tr>
<tr><td>Precondiciones:</td><td>Si el investigador decide “Guardar modelo biológico” debe haber un modelo biológico creado.</td></tr>
<tr><td>Referencias</td><td>RF3</td></tr>
<tr><td>Prioridad</td><td>Critico</td></tr>
<tr><td colspan="2">Flujo Normal de Eventos</td></tr>
<tr><td colspan="2">Sección “General”</td></tr>
<tr><td>Acción del Actor</td><td>Respuesta del Sistema</td></tr>
<tr><td>1. El investigador se dirige al menú principal y selecciona una de las siguientes opciones:
Abrir modelo biológico
Guardar modelo biológico</td><td>2. El sistema en dependencia a la acción solicitada por el investigador muestra la interfaz correspondiente:
Abrir modelo biológico: ir a la Sección “Abrir modelo biológico”
Guardar modelo biológico: ir a la Sección “Guardar modelo biológico”</td></tr>
<tr><td colspan="2">Sección “Abrir modelo biológico”</td></tr>
<tr><td>1. El investigador se dirige al menú principal y selecciona la opción Abrir.</td><td>2. El sistema muestra una ventana para seleccionar el lugar donde se encuentra el modelo.</td></tr>
<tr><td>3. El investigador indica la ruta.</td><td>4. El sistema abre el modelo y finaliza el CU.</td></tr>
<tr><td colspan="2">Sección“Guardar modelo biológico”</td></tr>
<tr><td>1. El investigador se dirige al menú principal y selecciona la opción guardar.</td><td>2. El sistema muestra una ventana para seleccionar el nombre y el lugar donde se guardara el modelo.</td></tr>
<tr><td>3. El investigador indica la ruta y el nombre.</td><td>4. El sistema comprueba que en el lugar seleccionado no exista otro modelo de igual</td></tr>
</table>

	nombre. 4.2 El sistema guarda el modelo finalizando el CU.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
3.1 si se parte del punto 3 de la sección General del flujo normal.	3.2 el sistema muestra un mensaje de error "Existe un modelo con el mismo nombre" el sistema se dirige a la acción 2 de esta sección.
Poscondiciones	En dependencia de la acción del Investigador: • Se abre un modelo biológico existente. • Se guarda el modelo biológico creado.

Tabla 10: Descripción del Caso de Uso Gestionar modelo biológico

2.4 Conclusiones

Al finalizar el capítulo quedó conformado el sistema en el modelo de dominio definiéndose los requisitos funcionales y no funcionales que el sistema debe ser capaz de alcanzar, diagrama de casos de uso del sistema y las descripciones textuales correspondientes a cada caso de uso.

Capítulo 3. Diseño del Sistema

El diseño del sistema tiene como propósito formular los modelos centrados en los requisitos y en el dominio de la solución y prepara el sistema para la implementación. En el presente capítulo se desarrolla el diseño, a través del cual se modela la aplicación la que debe ser capaz de soportar todos los requerimientos mencionados en el capítulo anterior. Para ello se utilizarán diferentes artefactos como son: los diagramas de interacción, los diagramas de clases del diseño y las descripciones de algunas de las principales clases del diseño.

3.1 Estilo arquitectónico

La arquitectura del sistema estará orientada a objetos y el estilo arquitectónico utilizado para la construcción del Módulo Modelación es el estilo de llamada y retorno que encapsula la arquitectura en capas y tiene como objetivo fundamental la separación de la lógica de negocios de la lógica de diseño. La ventaja fundamental de dicho estilo es que el desarrollo se puede llevar a cabo en varios niveles.

El patrón Modelo Vista Controlador (MVC) que sigue la filosofía del estilo arquitectónico de llamada y retorno, es el seleccionado para el desarrollo de la aplicación debido a que es utilizado por el framework JgraphEditor Editor y sus características brindan la posibilidad de descomponer el sistema en varios niveles de abstracción al tener soporte para múltiples vistas, la vista se separa del modelo y no hay ninguna dependencia directa entre vista y modelo, la interfaz de usuario puede mostrar múltiples vistas de los mismos datos simultáneamente. Además de un mayor soporte a los cambios producto a que los requisitos de interfaz tienden a cambiar más rápidamente que las reglas de negocio. Puesto que el modelo no depende de la vista, la adición de nuevos tipos de interfaces visuales al sistema generalmente no afectan al modelo. Por tanto, el ámbito del cambio se limita a la vista.

Para el diseño de la aplicación además de los paquetes que define el patrón MVC, para la organización de los elementos, se adicionó al paquete Controlador el paquete SBMLController que contiene las clases controladoras para el trabajo con modelos biológicos y archivos SBML. De forma paralela al patrón se encuentra el subsistema LibSBML que contiene la librería de igual nombre y que exporta las funcionalidades utilizadas por el paquete SBMLController, la adición de este subsistema a la aplicación no interfiere con la organización propuesta por el patrón dado que solo se realizan llamadas a un ente externo.

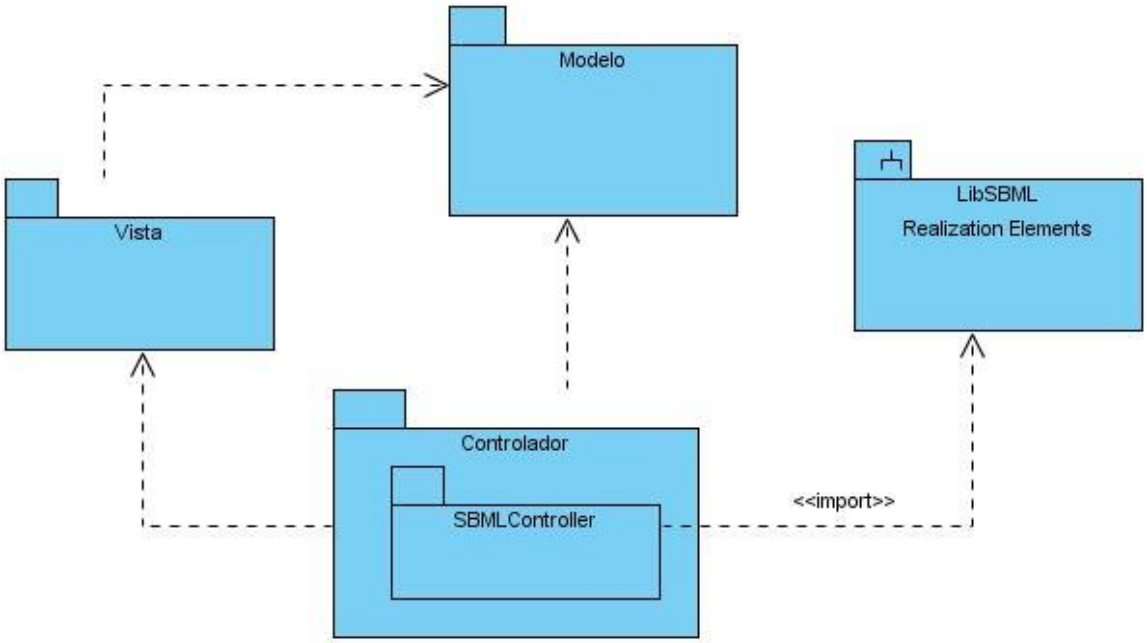


Figura 4. Diagrama de Paquetes del Diseño.

En el framework Jgraph Editor el modelo almacena la estructura lógica de la gráfica y provee métodos diversos para acceder a esa información. Las vistas son una o más capas lógicas sobre el modelo que realizan la tarea de visualizar la gráfica y actualizarla automáticamente cuando ocurren cambios en el modelo.

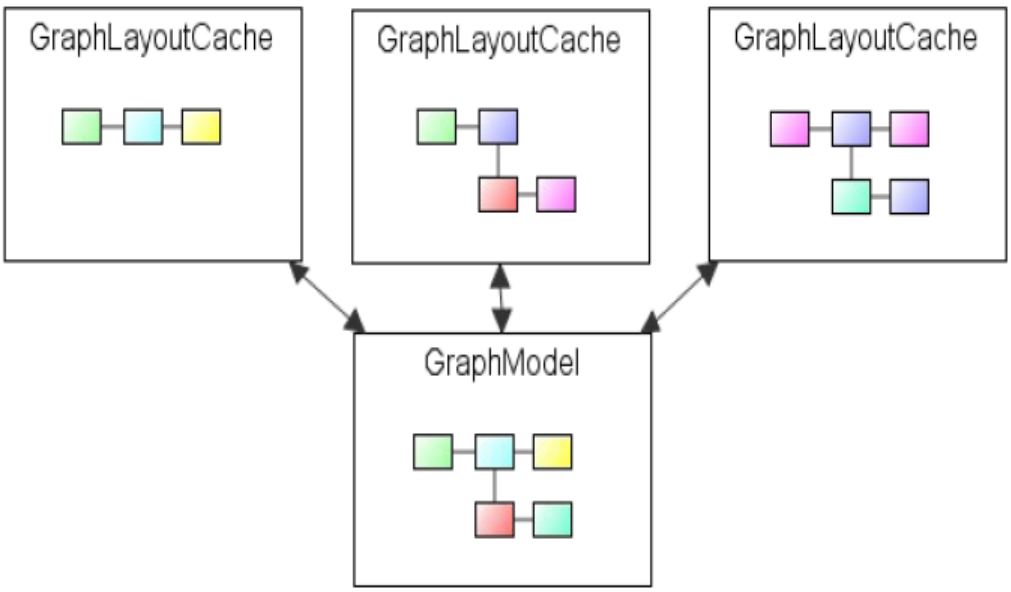


Figura 5. Esquema de múltiples vistas de un modelo en JGraph Editor.

3.2 Patrones de diseño

3.2.1 Factory Methods

El objetivo principal del patrón Factory Methods es el de centralizar en una clase la responsabilidad de crear objetos de un sbtipo determinado ocultándole al usuario la casuística seguida para elegir el subtipo a crear. Por lo que se tiene factoría "genérica" que es una clase abstracta y sirve para "fabricar" un producto, también genérico (un interfaz o bien una clase abstracta), permitiendo extender el sistema de una forma más elegante. [16]

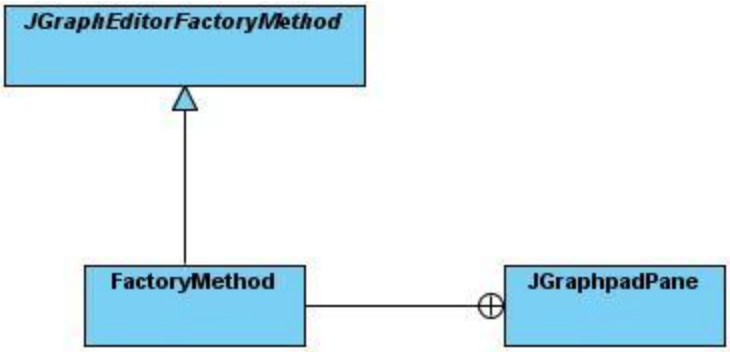


Figura 6. Subdiagrama donde se utiliza el patrón Factory Methods.

Este patrón es utilizado por el framework JGraph Editor para crear objetos de los subtiopos menubar, toolbar y toolbox. Como muestra la Figura 6 la clase FactoryMethod es la encargada de crear objetos del subtipo JGraphPane. La clase JGraphEditorFactoryMethod es la clase abstracta que fábrica productos genérico.

3.2.2 Singleton

El patrón Singleton permite asegurar que de una clase habrá solo una instancia, y proporciona un punto de acceso a ella global a todo el código. [16]

Si el constructor de una clase es público, podrá llamarse desde cualquier otra. Por tanto, para asegurar el control de la creación de instancias, el constructor debe ser privado. En este caso el constructor solo se podrá llamar desde un objeto de la propia clase. Pero no es posible instanciar objetos de esa clase, porque el constructor es privado. La solución consiste en utilizar un método estático para llamar al constructor así es posible controlar el acceso al constructor y que pueda ser llamarlo desde fuera de la clase. En todo momento debe existir una única instancia de la clase por lo que es necesario almacenar dicha instancia dentro de la propia clase (como una variable static).

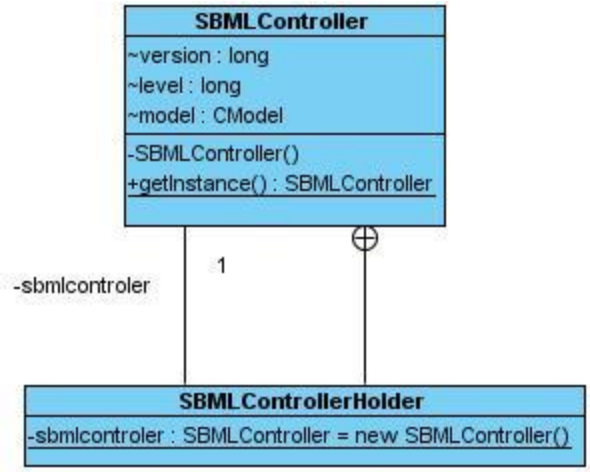


Figura 7. Subdiagrama donde se utiliza el patrón Singleton.

Como muestra la Figura 7 el constructor de la clase SBMLController es privado, por lo que ninguna otra clase podrá crear instancias de la misma. El método estático getInstance() es el punto de acceso al único objeto de SBMLController creado por la clase SBMLControllerHolder contenida dentro de la misma SBMLController.

3.2.3 Experto

Si las responsabilidades a las clases se asignan en forma adecuada, los sistemas tienden a ser más fáciles de entender, mantener y ampliar, lo que permite reutilizar los componentes en futuras aplicaciones. Por lo que este patrón brinda la posibilidad de asignar una responsabilidad al experto en información es decir asignar dicha responsabilidad la clase que cuenta con la información necesaria para cumplirla. [13]

3.2.4 Creador

El diseño, bien asignado, puede soportar un bajo acoplamiento, una mayor claridad, el encapsulamiento y la reutilización. El patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos. El propósito fundamental de este patrón es encontrar un creador que se debe conectar con el objeto producido en cualquier evento ya sea agregar, registrar o utilizar. Al escogerlo como creador, se da soporte al bajo acoplamiento. [13]

3.2.5 Bajo Acoplamiento

El acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases, con que las conoce y con que recurre a ellas. Acoplamiento bajo significa que una clase no depende de muchas clases. Como solución este patrón brinda la posibilidad de asignar responsabilidades de manera que las clases no dependan fuertemente de otras. [13]

3.2.6 Alta Cohesión

La cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme. Como solución el patrón permite asignar una responsabilidad de modo que la cohesión siga siendo alta. [13]

3.2.7 Controlador

Un evento del sistema generado por un actor externo; es un evento de entrada externa. Se asocia a operaciones del sistema: las que se emite en respuesta a los eventos del sistema. Un Controlador es un objeto de interfaz no destinado al usuario que se encarga de manejar un evento del sistema. Define además el método de su operación. El patrón Controlador sugiere asignar la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase que represente un manejador de los eventos del sistema. [13]

3.3 Diagrama de clases del diseño

Los diagramas de clases del diseño describen gráficamente las especificaciones de las clases de software, de las interfaces, así como sus relaciones en una aplicación. [17]
Contienen:

- Clases, atributos y métodos.
- Interfaces, con sus operaciones y constantes.
- Asociaciones: navegabilidad, multiplicidad.
- Dependencias.
- Paquetes o subsistemas.

Muestran las clases del diseño enfocadas a un lenguaje en específico, ya en el diagrama se manejan especificaciones más técnicas y detalladas. Son la primicia para entender lo que posteriormente se va

a implementar. [17]

Para la construcción de la aplicación se diseñó un diagrama de clases del diseño para cada caso de uso del sistema.

3.3.1 Diagrama de clases del diseño

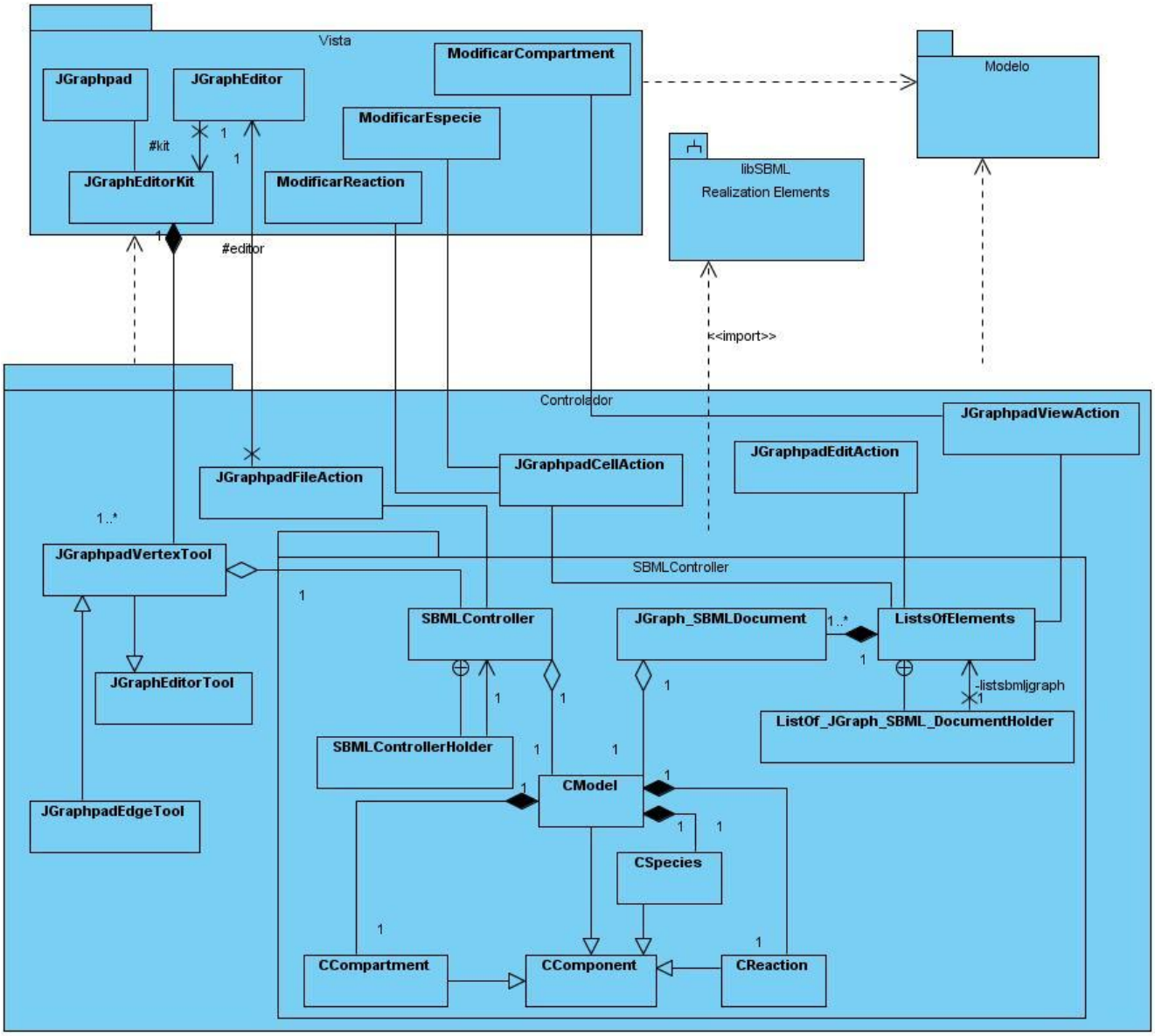


Figura 8 Diagrama de clases del Caso de uso gestionar compartimiento.

3.4 Descripción de las clases del diseño

Las clases contenidas en el paquete SBMLController son las encargadas de gestionar las funcionalidades que exporta la librería libSBML:

- SBMLController:
  Clase controladora de la que es posible crear una única instancia para tener un control mayor de las operaciones que realiza, contiene las funcionalidades para crear y salvar documentos SBML. En esta clase se encuentran los atributos versión y nivel de dichos documentos.

- CComponent:
  Clase que contiene las variables countid, countname, countmetaid de tipo entero que se utilizan para generar los nombres, id y metaid de los elementos que serán insertado en el modelo biológico y los métodos para incrementar las variables mencionadas anteriormente. De esta superclase heredan las clases CModel, CCompartment, CReaction y CSpecies.

- CModel:
  Clase controladora que contiene las funcionalidades para la creación de modelos biológicos, adición de compartimientos, reacciones y especies, de modificación y acceso a los atributos de los mismos. A través de los métodos **public Model createModel(),public Compartment addModelCompartment(Model mod, Compartment comp), public Reaction addModelReaction(Model mod), public Species addModelSpecie(Model mod, Compartment comp)** y los métodos getX y setX donde X es el nombre del atributo del compartimiento que será accedido o modificado. Utiliza las funcionalidades brindadas por la librería libsbml.jar.

- CCompartment:
  Clase controladora que contiene las funcionalidades para la creación de compartimientos, modificación y acceso a los atributos de los mismos. A través de los métodos **public Compartment createCompartment()**, y los métodos getX y setX donde X es el nombre del atributo del compartimiento que será accedido o modificado. Utiliza las funcionalidades brindadas por la librería libsbml.jar.

- CReaction:

  Clase controladora que contiene las funcionalidades para la creación de reacciones químicas, adición de reactantes, productos y modificadores, modificación y acceso a los atributos de las mismas. A través de los métodos **public Reaction CreateReaction(),public void addReactionReactant(Species esp, Reaction react), public void addReactionProduct(Species esp, Reaction react), public void addReactionModifier(Species esp, Reaction react)** y los métodos getX y setX donde X es el nombre del atributo de la reacción que será accedido o modificado. Utiliza las funcionalidades brindadas por la librería libsbml.jar.

- CSpecies:

  Clase controladora que contiene las funcionalidades para la creación de especies, modificación y acceso a los atributos de las mismas. A través de los métodos **public Species CrearSpecies(Compartment compart)**, donde compart es el compartimiento donde estará ubicada la especie, y los métodos getX y setX donde X es el nombre del atributo de la especie que será accedido o modificado. Utiliza las funcionalidades brindadas por la librería libsbml.jar.

- ListOfElements:

  Clase que contiene las listas **LinkedList<JGraph_SBMLDocument> list, LinkedList<SBase_Cell> list_sc y LinkedList<LogicalOperator_Cell> list_lc** que mantienen unidos a los elementos gráficos con los elementos SBase que ellos representan. Esta clase contiene también las funcionalidades para obtener el par SBase de cualquier elemento gráfico.

3.5 Diagrama de interacción

Los diagramas de interacción se utilizan para modelar los aspectos dinámicos de un sistema, muestran una interacción, que consiste en un conjunto de objetos y sus relaciones, incluyendo los mensajes que se pueden enviar entre ellos. Un diagrama de secuencia es un diagrama de interacción que destaca la ordenación temporal de los mensajes [18]. Los diagramas que se muestran en este epígrafe están divididos en secciones para una mejor comprensión.

3.5.1 Diagrama de secuencia del Caso de Uso gestionar especie sección insertar especie

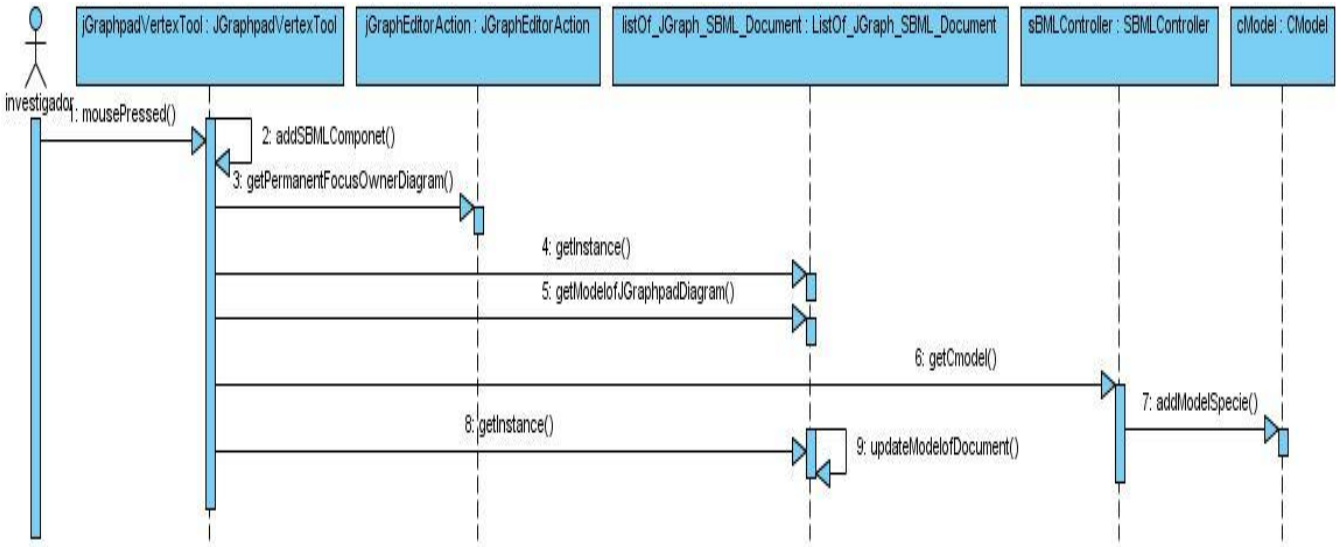


Figura 9. Diagrama de secuencia del Caso de Uso gestionar especie sección insertar especie.

3.5.2 Diagrama de secuencia del Caso de Uso gestionar especie sección eliminar especie

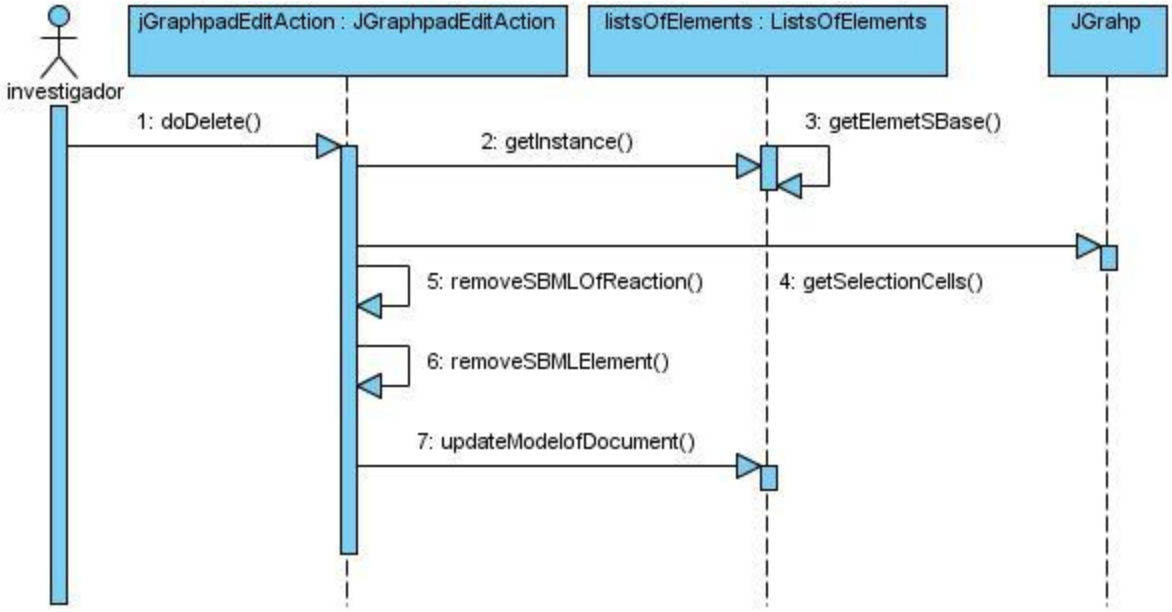


Figura 10. Diagrama de secuencia del Caso de Uso gestionar especie sección eliminar especie.

3.5.3 Diagrama de secuencia del Caso de Uso mostrar propiedades

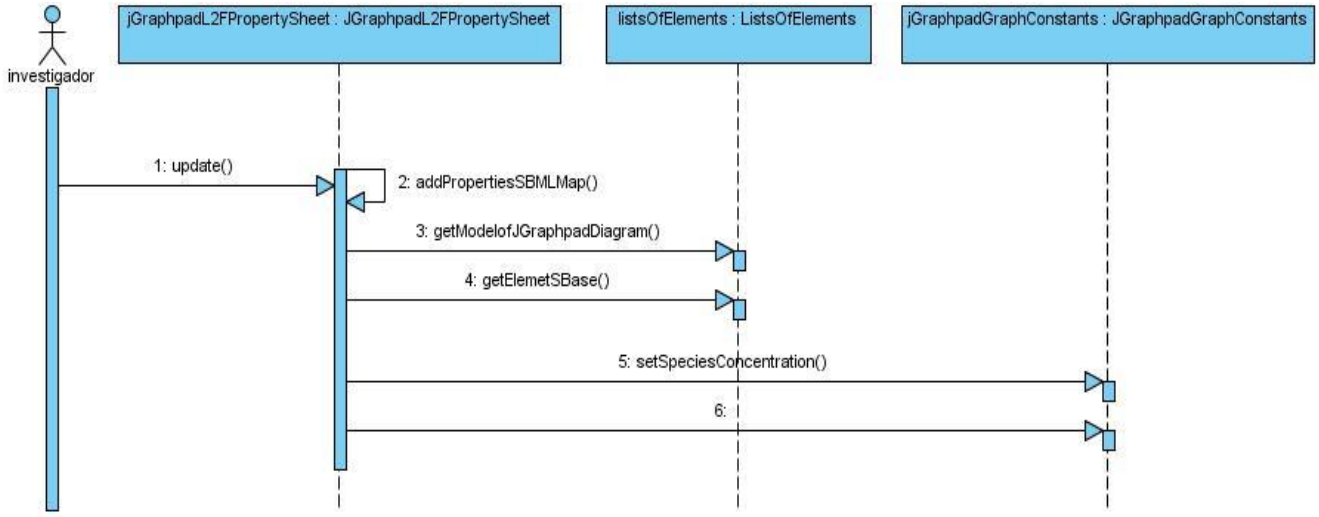


Figura 11. Diagrama de secuencia del Caso de Uso mostrar propiedades

3.5.4 Diagrama de secuencia del Caso de Uso gestionar modelo biológico

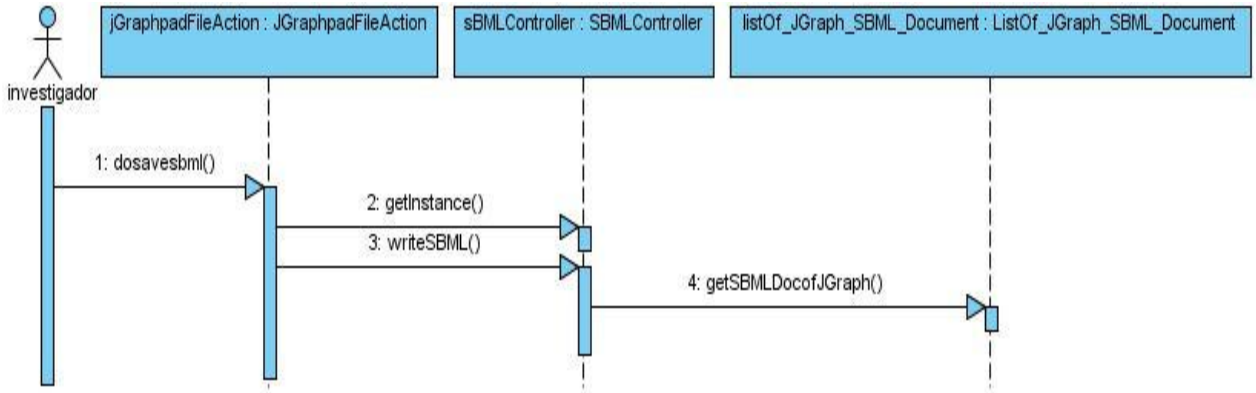


Figura 12. Diagrama de secuencia del Caso de Uso gestionar modelo biológico.

3.6 Conclusiones

En el presente capítulo fue descrito el estilo arquitectónico que se utilizó en el diseño de la aplicación así como los patrones de diseño utilizados y la fundamentación de su uso. También se realizaron los diferentes diagramas de clases del diseño para los casos de uso y los diagramas de interacción.

Capítulo 4. Implementación del Sistema

La implementación comienza con el resultado del diseño y se implementa el sistema en términos de componentes. El propósito principal es desarrollar la arquitectura y el sistema como un todo. En el presente capítulo se describe cómo quedará implementada la aplicación y se muestran las principales pantallas de la misma con sus descripciones.

4.1 Diagramas de Componentes

Los diagramas de componentes muestran las organizaciones y dependencias lógicas entre componentes software, sean estos componentes de código fuente, binarios o ejecutables es un grafo donde los componentes están unidos por medio de relaciones de dependencia: compilación y ejecución. [19]

Son utilizados para estructurar el modelo de implementación en términos de subsistemas de implementación y mostrar las relaciones entre los elementos de implementación. [19]
El uso más importante de estos diagramas es mostrar la estructura de alto nivel del modelo de implementación, especificando:

- Los subsistemas de implementación y sus dependencias a la hora de importar código.
- Organizar los subsistemas de implementación en capas.

El diagrama de componentes quedó estructurado como se muestra en la Figura 13 donde:

- El subsistema de implementación Vista: contiene los archivos de código fuente de pertenecientes a la capa Vista.
- El subsistema Controlador contiene el código fuente distribuido en archivos de las clases controladoras
- Subsistema de implementación SBMLController: contiene las clases que implementan las funciones para el trabajo con los componentes de los modelos biológicos y los documentos SBML.
- Librería libSBML: contiene las funcionalidades implementadas para el trabajo con archivos SBML.
- Interfaz libFuctions: exporta las funcionalidades de la librería libSBML

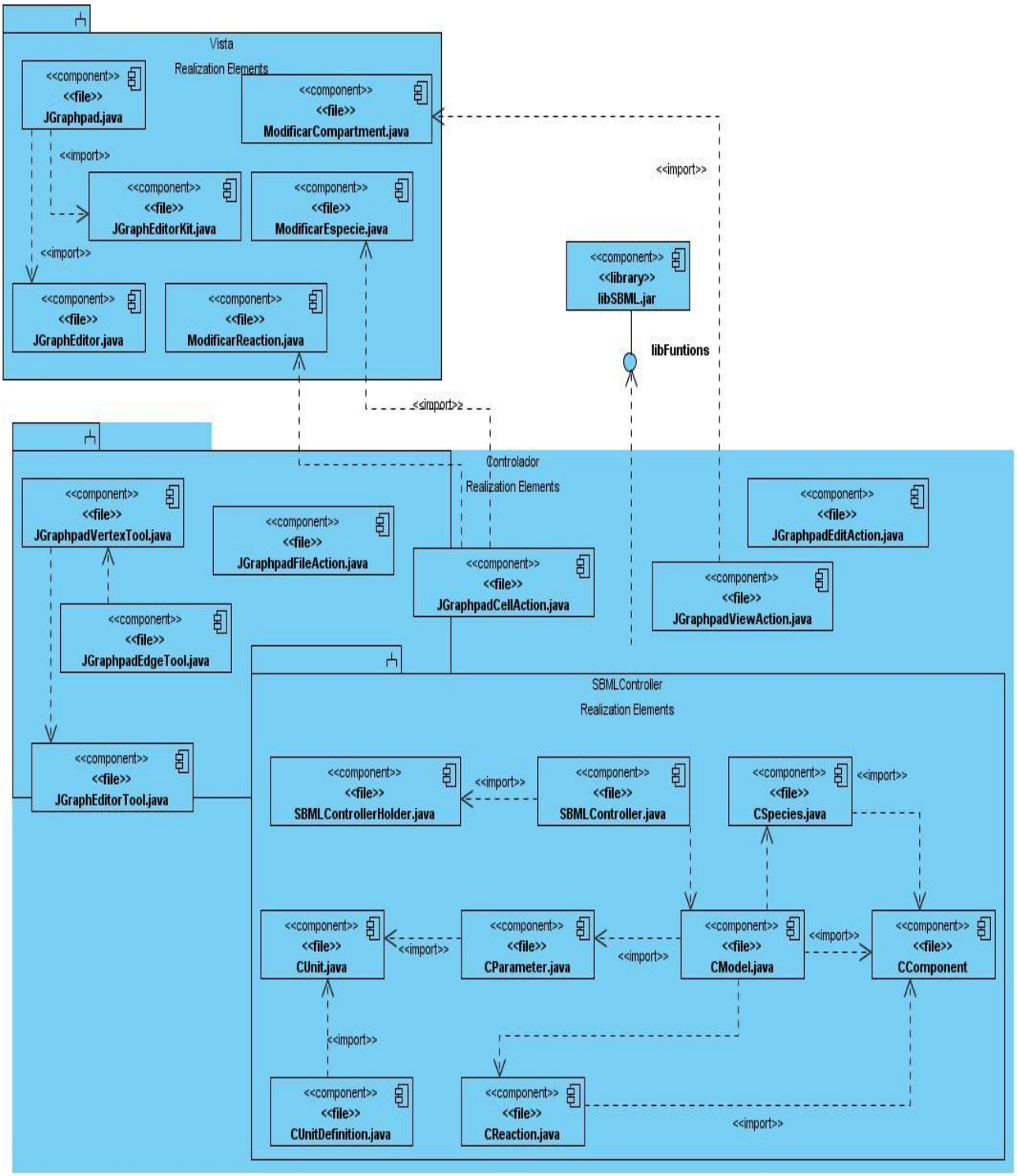


Figura 13. Diagrama de Componentes.

4.2 Código fuente de los principales métodos y su descripción

A continuación se describen algunos de los principales métodos que fueron desarrollados para conformar la aplicación que permite modelar gráficamente sistemas biológicos regida por el estándar SBGN y SBML. Las descripciones se centran principalmente en los métodos que permiten dar cumplimiento a los requisitos funcionales soportados por la aplicación y aquellos que implementan las funcionalidades que brinda la librería libSBML, para el trabajo con el estándar SBML.

El siguiente fragmento de código perteneciente al método **protected void paintBackground(Graphics g)** que se encarga de dibujar la forma gráfica de los símbolos propuestos por SBGN, pinta en el área de modelado la figura que representa un *Nucleic acid Feature*.

```
else if (shape == SHAPE_NUCLEIC_ACID_FEATURE){

    int height, width;

    height = d.height - b;
    width = d.width - b;

    Graphics2D g2 = (Graphics2D) g;
    Graphics2D g3 = (Graphics2D) g;
    Graphics2D g4 = (Graphics2D) g;
    Graphics2D g5 = (Graphics2D) g;
    Graphics2D g6 = (Graphics2D) g;
    Graphics2D g7 = (Graphics2D) g;

    int h4 = (int) (d.getHeight()/4);
    int r = d.width;

    g2.fillRect(0, 0, width, 0);
    g3.fillRect(0, 0, 0, height);
    g4.fillRect(width, 0, width,height);
    g5.fillRect(0,height, width,height);

    g6.fillArc(b, d.height ,d.height, d.width,0,180);
    g7.fillArc(b, d.height - h4 - b , r-2, h4,0,-46);
}
```

Resultado

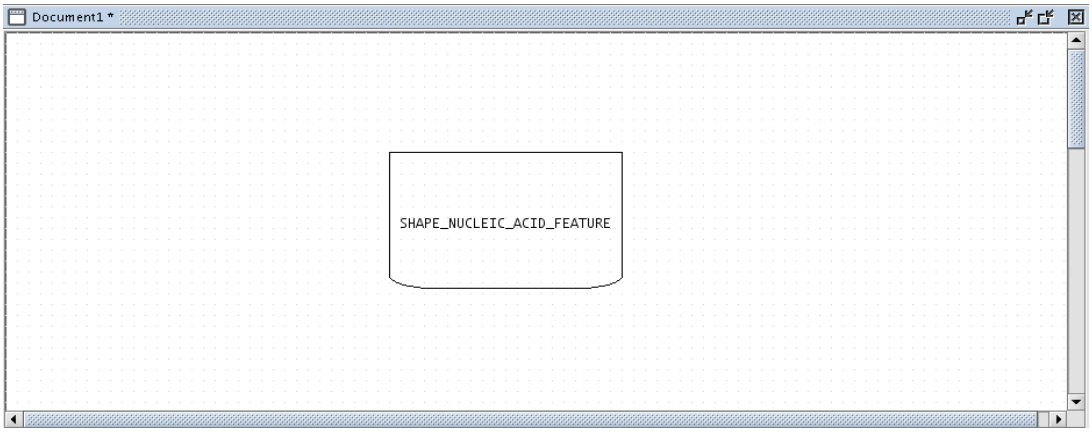


Figura 14. Resultado 1.

El siguiente código brinda la funcionalidad a la aplicación de poder salvar el modelo biológico en el formato SBML.

```
protected void doSaveFileSBML(JGraphEditorDiagram file,
        boolean forceFilenameDialog) throws IOException {

    String filename = file.getName();

    if (forceFilenameDialog ) {
         {
             ListsOfElements.
                getInstance().setSBMLDocumentIsNew(file);
            // Strip extension as it will be added by the dialog
            // based on the chooseable filter.
            if (filename.toLowerCase().endsWith(".xml.gz"))
                filename = filename.substring(0, filename.length() - 7);
            else if (filename.toLowerCase().endsWith(".xml"))
                filename = filename.substring(0, filename.length() - 4);

            filename = dlgs.editorFileDialog(
                    getPermanentFocusOwnerOrParent(),
                    getString("SaveSBML"), filename, false,
                    lastDirectory);
        }
    }
    if (filename != null) {
        SBMLController.getInstance().
         writeSBML(ListsOfElements.getInstance().
                 getSBMLDocofJGraph(file), filename);
    }

}
```

Resultado:

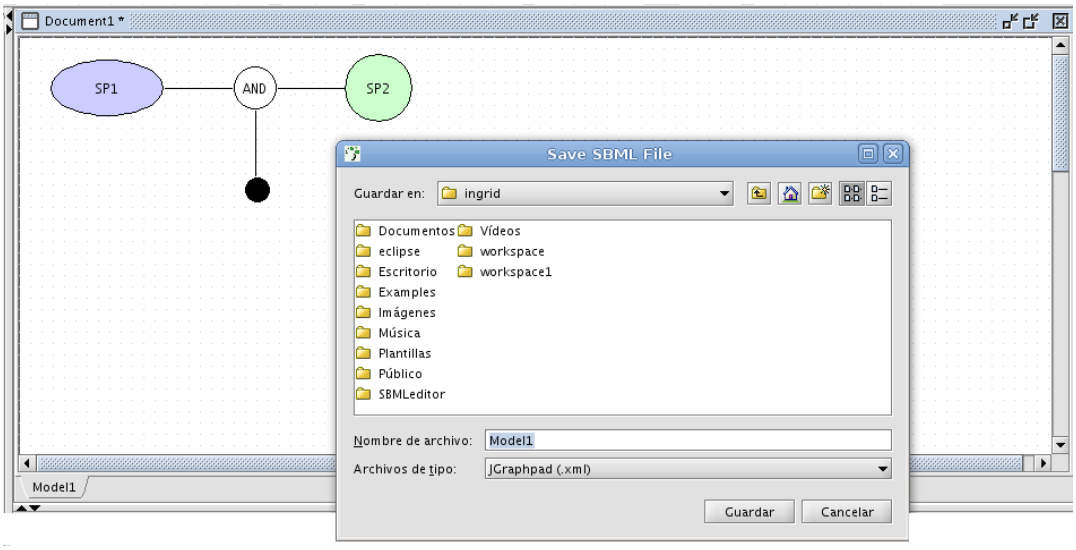


Figura 15 Resultado 2.

El formato SBML codifica el modelo a XML.

Model1.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<sbml xmlns="http://www.sbml.org/sbml/level2" level="2" version="1">
  <model metaid="_mo1" id="MOD1" name="Model1">
    <listOfCompartments>
      <compartment metaid="_cp1" id="COMP1" name="Compartment1"/>
    </listOfCompartments>
    <listOfSpecies>
      <species metaid="_sp1" id="SPE1" name="SP1" compartment="COMP1"/>
      <species metaid="_sp2" id="SPE2" name="SP2" compartment="COMP1"/>
    </listOfSpecies>
    <listOfReactions>
      <reaction metaid="_rc1" id="REAC1" name="Asociation1">
        <listOfModifiers>
          <modifierSpeciesReference species="SPE1"/>
          <modifierSpeciesReference species="SPE2"/>
        </listOfModifiers>
      </reaction>
    </listOfReactions>
  </model>
</sbml>
```

Figura 16. Código XML del modelo salvado.

El modelo ha sido salvado en formato SBML y puede ser abierto por otros softwares, el siguiente ejemplo muestra como el modelo generado por la aplicación es cargado por el software SBML Editor.

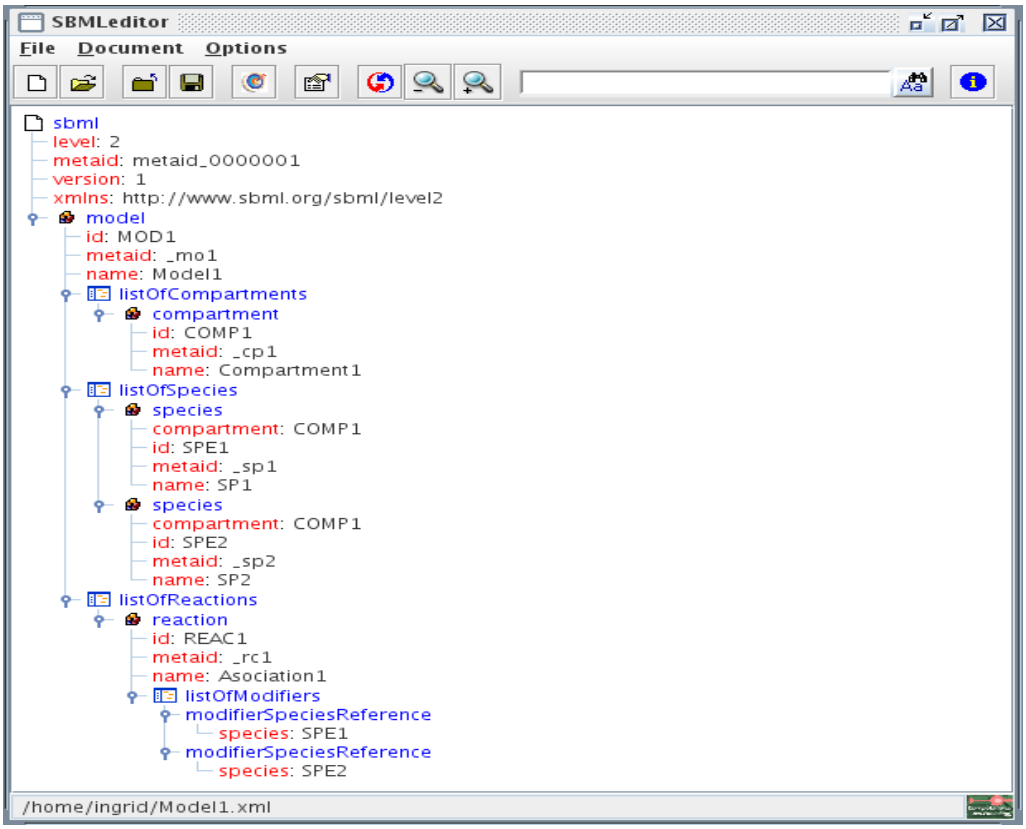


Figura 17. SBML Editor.

4.3 Validación a nivel de desarrollador

Los estándares SBGN y SBML proponen un conjunto de reglas que son necesarias a la hora de realizar la implementación de aplicaciones que modelen sistemas biológicos, SBGN propone un conjunto de símbolos con precisión semántica y sintáctica que ilustran la forma gráfica de los componentes del sistema biológico junto a normas que define su utilización, por su parte SBML establece una definición para cada componente que integra el sistema biológico y los tipos de operaciones que son posible realizar ente los diferentes componentes además de permitir que los modelos sean codificados en XML. Por lo que se realiza la validación de la aplicación a nivel de desarrollador para dar cumplimiento a las reglas y normas establecidas por los dos estándares.

En el siguiente fragmento de código perteneciente al método **public void reactionController (DefaultGraphCell cell_first,DefaultGraphCell cell_second)** se validan las operaciones realizadas entre los componentes especies y reacción química. Se lanza la excepción para validar que no se pueda realizar la asociación ente la especie seleccionada y la reacción química si asociación es incorrecta.

```
else if(ListsOfElements.getInstance().
        getElemetSBase(temp_mod, cell_first) instanceof Reaction &&
        ListsOfElements.getInstance().
        getElemetSBase(temp_mod, cell_second) instanceof Reaction &&(name.equals(JGraphpad.NAME_CATALYSIS)
                || name.equals(JGraphpad.NAME_MODULATION)|| name.equals(JGraphpad.NAME_PRODUCTION)
                || name.equals(JGraphpad.NAME_STIMULATION) || name.equals(JGraphpad.NAME_INHIBITION)
                || name.equals(JGraphpad.NAME_TRIGGER) || name.equals(JGraphpad.NAME_EDGETOOL))){
    throw new Exception("Incorrect association");
}
```

A través de la funcionalidad **showMessageDialog()** se muestra al usuario un mensaje de error indicándole que la asociación que ha hecho es incorrecta.

```
try {
    DefaultGraphCell sour = (DefaultGraphCell) graph.getModel().
    getParent(graph.getModel().getSource(edge));
    DefaultGraphCell targ = (DefaultGraphCell) graph.getModel().
    getParent(graph.getModel().getTarget(edge));
    reactionController(sour, targ);
    }

catch (Exception e) {
    JOptionPane.showMessageDialog(JGraphEditorAction.getActiveFrame(),
            e.getMessage(),
            "Error",
            JOptionPane.ERROR_MESSAGE);
    Object [] array = new Object[1];
    array[0] = edge;
    cache.remove(array);
}
```

Resultado:

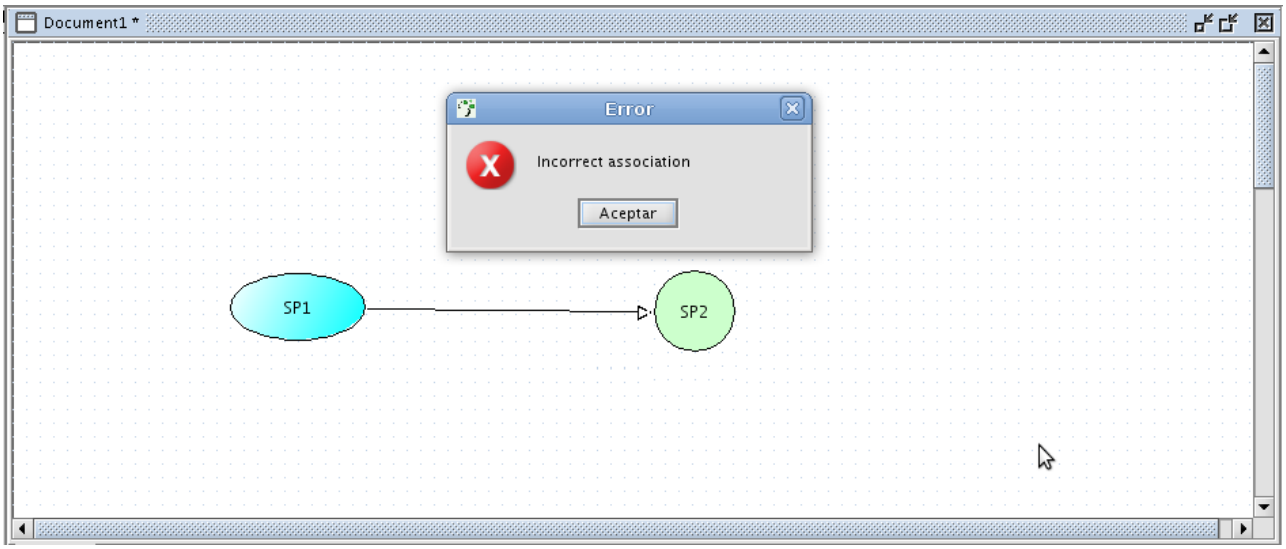


Figura 18. Resultado 3.

4.4 Pantallas de la Aplicación

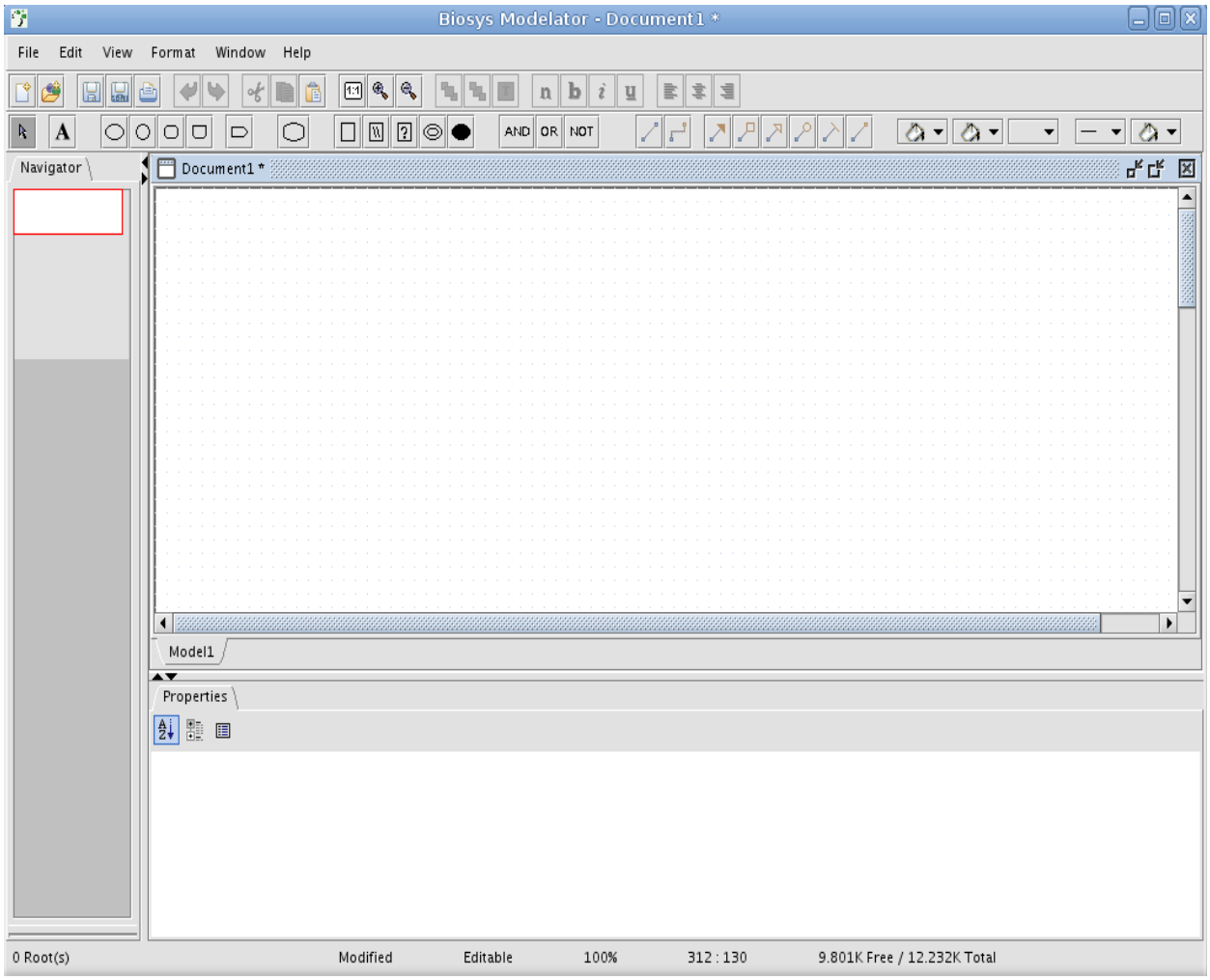


Figura 19. Interfaz Principal.

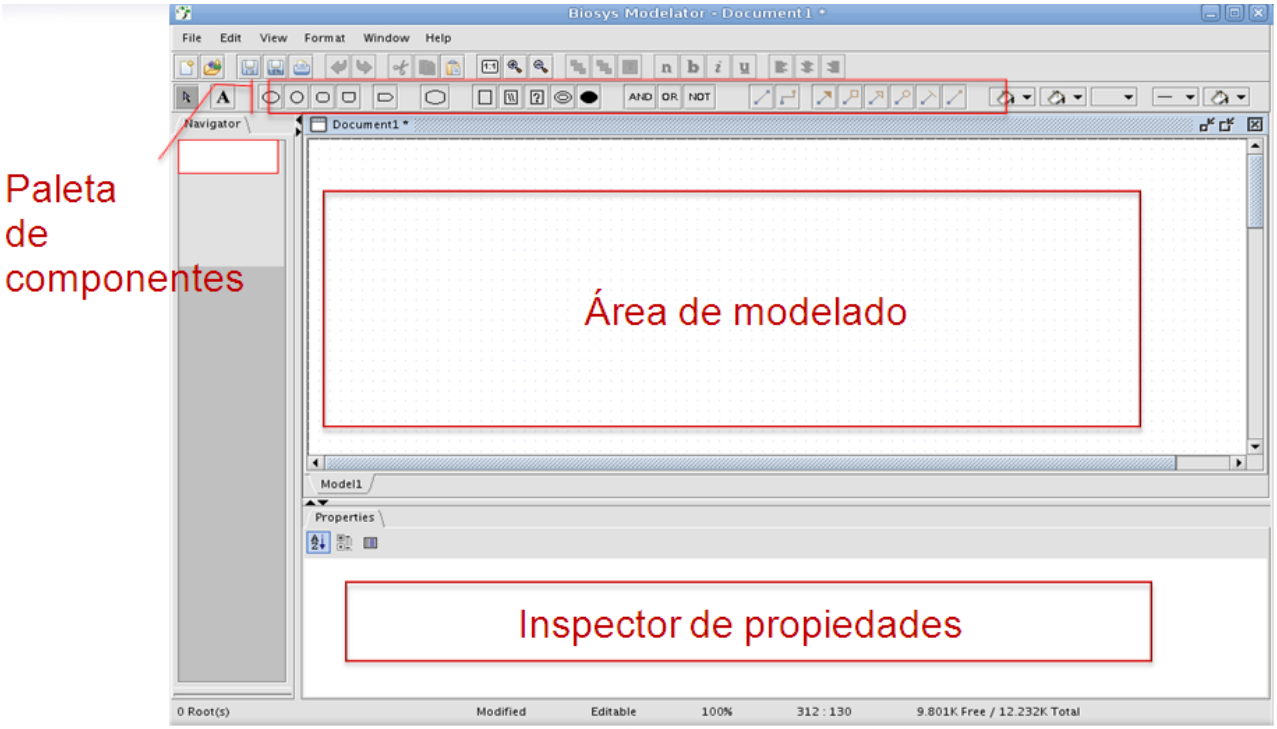


Figura 20. Principales áreas de la interfaz.

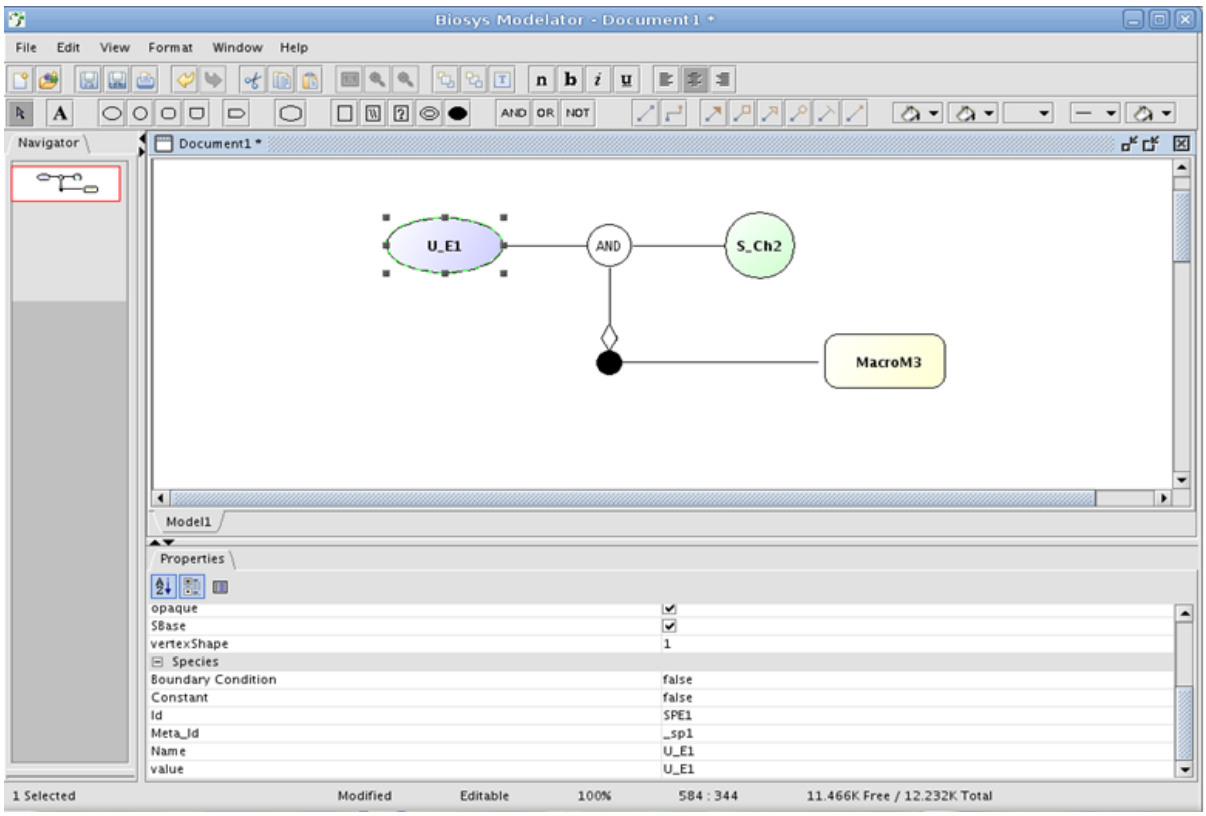


Figura 21 Interfaz Principapal.

4.5 Conclusiones

Mediante la realización del presente capítulo se describió como fue implementada la aplicación en términos de componentes además de fragmentos de códigos relevantes para la implementación y se mostraron las pantallas principales de las nuevas funcionalidades.

Conclusiones

Luego de todo lo expuesto en el presente trabajo, es posible concluir que:

- Se definieron las funcionalidades y características de la aplicación como resultado del proceso de captura de requisitos.
- Se realizó el diseño de una aplicación para modelar gráficamente sistemas biológicos.
- Se implementó una aplicación que permite modelar gráficamente sistemas biológicos, cumpliendo con el estándar de notación gráfica SBGN y el formato SBML.

Recomendaciones

- Integrar el nuevo módulo de Modelación Gráfica con el módulo Editor de Ecuaciones lo que permitirá editar las funciones correspondientes a los elementos SBML con mayor rigor.
- Continuar el estudio del formato SBML lo que permitirá incorporarle al módulo y al proyecto nuevas funcionalidades para el trabajo con sistemas biológicos.
- Incorporar el estándar SBO al módulo Modelación.

Referencias bibliográficas

1. Taguenca, Juan.B (2003) *Las nuevas biotecnologías en España* [Disponible en: http://www.tesisenxarxa.net/TESIS_UAB/AVAILABLE/TDX-1027104-172625//jtb1de2.pdf].

2. Giraldo, J.C.R (2006) *Estado del arte de la bioinformática*.[Disponible en: ftp.eia.edu.co/Sitios%20Web/bioinstrumentacion/docs/otros/BioinformaticsReviewUdeA.pdf]. [Consultado 22 de noviembre 2008].

3. Meloan, S (2001) *Exploring the New Frontier: Java Technology Powers the "Post-Genomic" Era. Sun Developer Network*. [Disponible en: http://java.sun.com/features/2001/10/genome2.html]. [Consultado 2 de dicembre 2008].

4. López, M., G.R. Romero, and M. Vega (2007) *Biología de Sistemas.* [Disponible en: http://www.gen-es.org/12_publ/docs/Biologiadeseistemas.pdf]. [Consultado 20 de diciembre 2008].

5. Sánchez, Y. and Y. Armentero, *"Software para la Simulación de Sistemas Biológicos: Módulo de modelación gráfica de Sistemas Biológicos"*. 2007, Universidad de las Ciencias Informáticas: Ciudad de la Habana.

6. Erato K. (2003) *Systems Biology Markup Language (SBML) Level 2: Structures and Facilities for Model Definitions* [Disponible en: sbml.org/images/b/b7/Finney_2003.pdf]. [Consultado noviembre 2008].

7. Kitano, H., N.L. Nov`ere, and S. Moodie. (2008) *Systems Biology Graphical Notation: Process Diagram Level 1 .* [Disponible en: www.icsb-2007.org/proceedings/abstracts/G19.pdf] [Consultado 5 de dicembre de 2008].

8. SBO, System Biology Ontology [Disponible en: http://sbml.org/images/4/4c/Hucka-sbo.pdf] [Consultado 15 de diciembre de 2008].

9. *Ayuda extendida OpenUP*. (2008). [Disponible en: http://10.34.20.5:5800/OpenUP] [Consultado 20 de noviembre de 2008].

10. [Disponible en: http://hanzcocchi.net/el-rol-del-analista-en-rup/]. [Consultado marzo de 2009].

11. Larman, C. (1999) *"UML y patrones".* [Disponible en: www.api-epsa.upv.es/Inform%E0tica/Temaris/6985.pdf]. [Consultado 20 de enero 2009].

12. Övergaard, G. and K. Palmkvlist (2004) *"Use Cases: Patterns and Blueprints".* [Disponible en: www.dcs.bbk.ac.uk/~gr/pdf/seke07_requirement_patterns_2.pdf]. [Consultado 2 febrero de 2009].

13. Gamma, E. and R. Helm, *Design Patterns. (1995) Elements of Reusable Object-Oriented Software*. [Dispònible en: www. cv.uoc.es/cdocent/RIB69XL1SCMBA5BYHM2N.pdf] [Consultado 11 febrero de 2009].

14. León, J.d.C.y.,(1999) *Guía de Iniciación al Lenguaje JAVA*. [Disponible en: http://pisuerga.inf.ubu.es/lsi/Invest/Java/Tuto/I_3.pdf] [Consultado 10 de enero de 2009].

15. Gutiérrez, J. (2001) *El entorno de desarrollo Eclipse.* [Disponible en: http://www.uv.es/~jgutierr/MySQL_Java/TutorialEclipse.pdf] [Consultado 10 de enero de 2009].

16. CCIA, D. (2007) *Introducción a los patrones de diseño. Algunos patrones básicos*. [Disponible en: http://www.jtech.ua.es/j2ee/2008-2009/doc/patrones-sesion01-apuntes.pdf]. [Consultado 3 de marzo de 2009].

17. Visconti, M. *Fundamentos de Ingeniería de Software*. Universidad Técnica Federico Santa María. [Disponible en: http://www.inf.utfsm.cl/~visconti/ili236/Documentos/10-DisenoOO.pdf]. [Consultado 15 de marzo de 2009].

18. Cueva.J.M. *Diseño Orientado a Objetos* [Disponible en: http://www.di.uniovi.es/~cernuda/pfc/doo.pdf]. [Consultado 15 de marzo de 2009].

19. Vilas, A.F. (2008) *Diagrama de Componentes*. [Disponible en: http://tvdi.det.uvigo.es/~avilas/UML/node49.html]. [Consultado 10 de abril de 2009].

Bibliografía

- Taguenca, Juan.B (2003) Las nuevas biotecnologías en España [Disponible en: http://www.tesisenxarxa.net/TESIS_UAB/AVAILABLE/TDX-1027104-172625//jtb1de2.pdf].

- Giraldo, J.C.R (2006) *Estado del arte de la bioinformática*.[Disponible en: ftp.eia.edu.co/Sitios%20Web/bioinstrumentacion/docs/otros/BioinformaticsReviewUdeA.pdf]. [Consultado 22 de noviembre 2008].

- Meloan, S (2001) *Exploring the New Frontier: Java Technology Powers the "Post-Genomic" Era. Sun Developer Network*. [Disponible en: http://java.sun.com/features/2001/10/genome2.html].
- [Consultado 2 de dicembre 2008].

- López, M., G.R. Romero, and M. Vega (2007) *Biología de Sistemas.* [Disponible en: http://www.gen-es.org/12_publ/docs/Biologiadeseistemas.pdf]. [Consultado 20 de diciembre 2008].

- Sánchez, Y. and Y. Armentero, *"Software para la Simulación de Sistemas Biológicos: Módulo de modelación gráfica de Sistemas Biológicos"*. 2007, Universidad de las Ciencias Informáticas: Ciudad de la Habana.

- Erato K. (2003) *Systems Biology Markup Language (SBML) Level 2: Structures and Facilities for Model Definitions* [Disponible en: sbml.org/images/b/b7/Finney_2003.pdf]. [Consultado noviembre 2008].

- Kitano, H., N.L. Nov`ere, and S. Moodie. (2008) *Systems Biology Graphical Notation: Process Diagram Level 1 .* [Disponible en: www.icsb-2007.org/proceedings/abstracts/G19.pdf] [Consultado 5 de dicembre de 2008].

- *SBO, System Biology Ontology* [Disponible en: http://sbml.org/images/4/4c/Hucka-sbo.pdf] [Consultado 15 de diciembre de 2008].

- *Ayuda extendida OpenUP*. (2008). [Disponible en: http://10.34.20.5:5800/OpenUP] [Consultado 20 de noviembre de 2008].

- [Disponible en: http://hanzcocchi.net/el-rol-del-analista-en-rup/]. [Consultado marzo de 2009].

- Larman, C. (1999) *"UML y patrones".* [Disponible en: www.api-epsa.upv.es/Inform%E0tica/Temaris/6985.pdf]. [Consultado 20 de enero 2009].

- Övergaard, G. and K. Palmkvlist (2004) *"Use Cases: Patterns and Blueprints".* [Disponible en: www.dcs.bbk.ac.uk/~gr/pdf/seke07_requirement_patterns_2.pdf]. [Consultado 2 febrero de 2009].

- Gamma, E. and R. Helm, *Design Patterns. (1995) Elements of Reusable Object-Oriented Software*. [Dispònible en: www. cv.uoc.es/cdocent/RIB69XL1SCMBA5BYHM2N.pdf] [Consultado 11 febrero de 2009].

- León, J.d.C.y.,(1999) *Guía de Iniciación al Lenguaje JAVA*. [Disponible en: http://pisuerga.inf.ubu.es/lsi/Invest/Java/Tuto/I_3.pdf] [Consultado 10 de enero de 2009].
- Gutiérrez, J. (2001) *El entorno de desarrollo Eclipse.* [Disponible en: http://www.uv.es/~jgutierr/MySQL_Java/TutorialEclipse.pdf] [Consultado 10 de enero de 2009].
- CCIA, D. (2007) *Introducción a los patrones de diseño. Algunos patrones básicos*. [Disponible en: http://www.jtech.ua.es/j2ee/2008-2009/doc/patrones-sesion01-apuntes.pdf]. [Consultado 3 de marzo de 2009].
- Visconti, M. *Fundamentos de Ingeniería de Software*. Universidad Técnica Federico Santa María. [Disponible en: http://www.inf.utfsm.cl/~visconti/ili236/Documentos/10-DisenoOO.pdf]. [Consultado 15 de marzo de 2009].
- Cueva.J.M. *Diseño Orientado a Objetos* [Disponible en: http://www.di.uniovi.es/~cernuda/pfc/doo.pdf]. [Consultado 15 de marzo de 2009].
- Vilas, A.F. (2008) *Diagrama de Componentes*. [Disponible en: http://tvdi.det.uvigo.es/~avilas/UML/node49.html]. [Consultado 10 de abril de 2009].
- Benson, D.(2009) *JGraph and JGraph Layout Pro User Manual.* [Disponible en: www.**jgraph**.com/pub/**jgraphmanual**.pdf]. [Consultado marzo de 2009].
- Gaudenz, A. (2005) *JGraph in Action Using JGraph in real-world applications.* [Disponible en: www.jgraph.com/pub/jgraphpadmanual.pdf] [Consultado marzo de 2009].
- En línea: http://www.biouml.org/. [Consultado noviembre 2008].
- En línea: http://sbml.org/Main_Page. [Consultado noviembre 2008].
- En línea: http://www.cellnomica.com/ . [Consultado noviembre 2008].

Anexos

Anexo 1 Diagrama de secuencia del CUS gestionar reacción química sección insertar

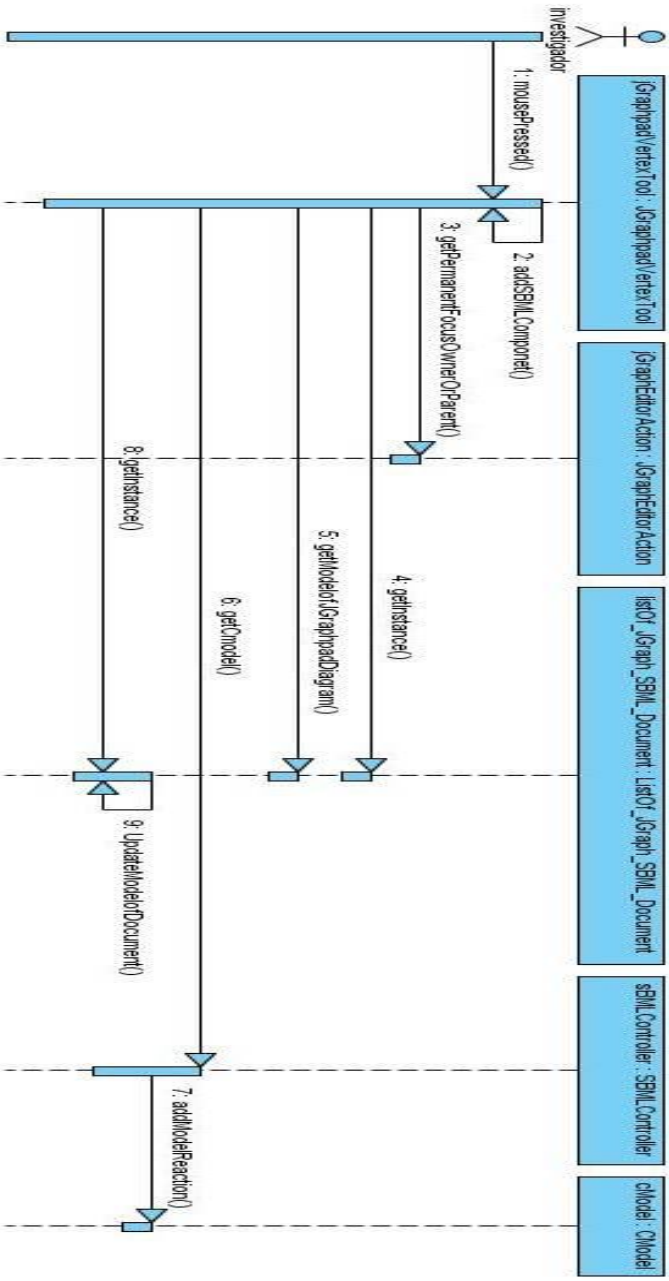


Figura 22. Diagrama de secuencia del CUS gestionar reacción química sección insertar

Anexo 2 Diagrama de secuencia del CUS gestionar reacción química sección eliminar

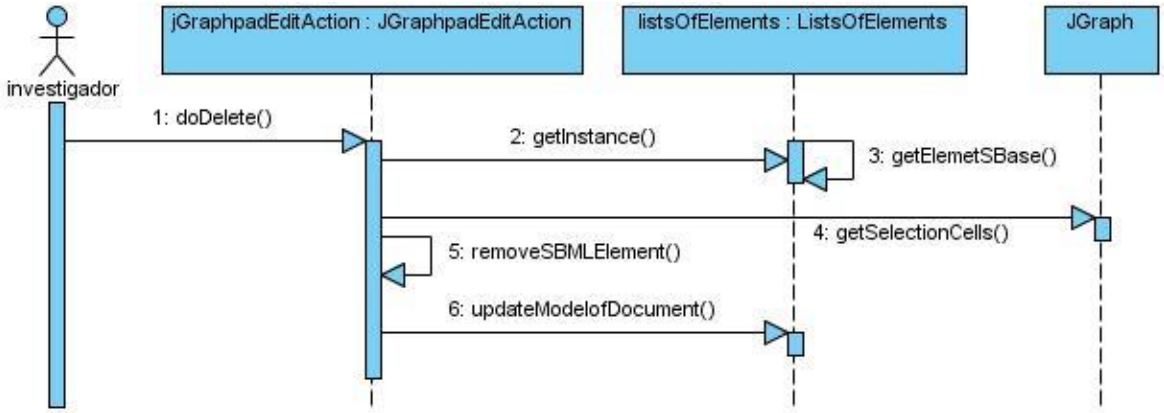


Figura 23. Diagrama de secuencia del CUS gestionar reacción química sección eliminar

Anexo 3 Diagrama de secuencia del CUS gestionar modificador sección eliminar

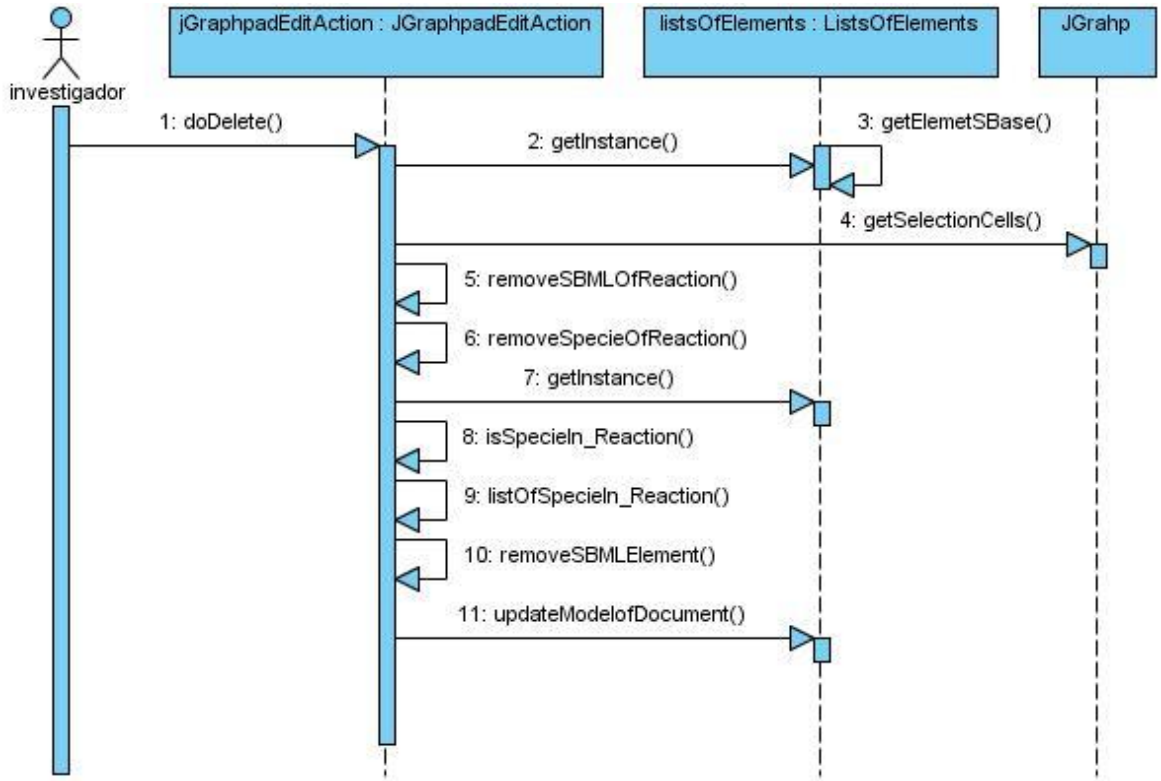


Figura 24. Diagrama de secuencia del CUS gestionar modificador sección eliminar

Anexo 4 Diagrama de secuencia del CUS gestionar modificador sección insertar

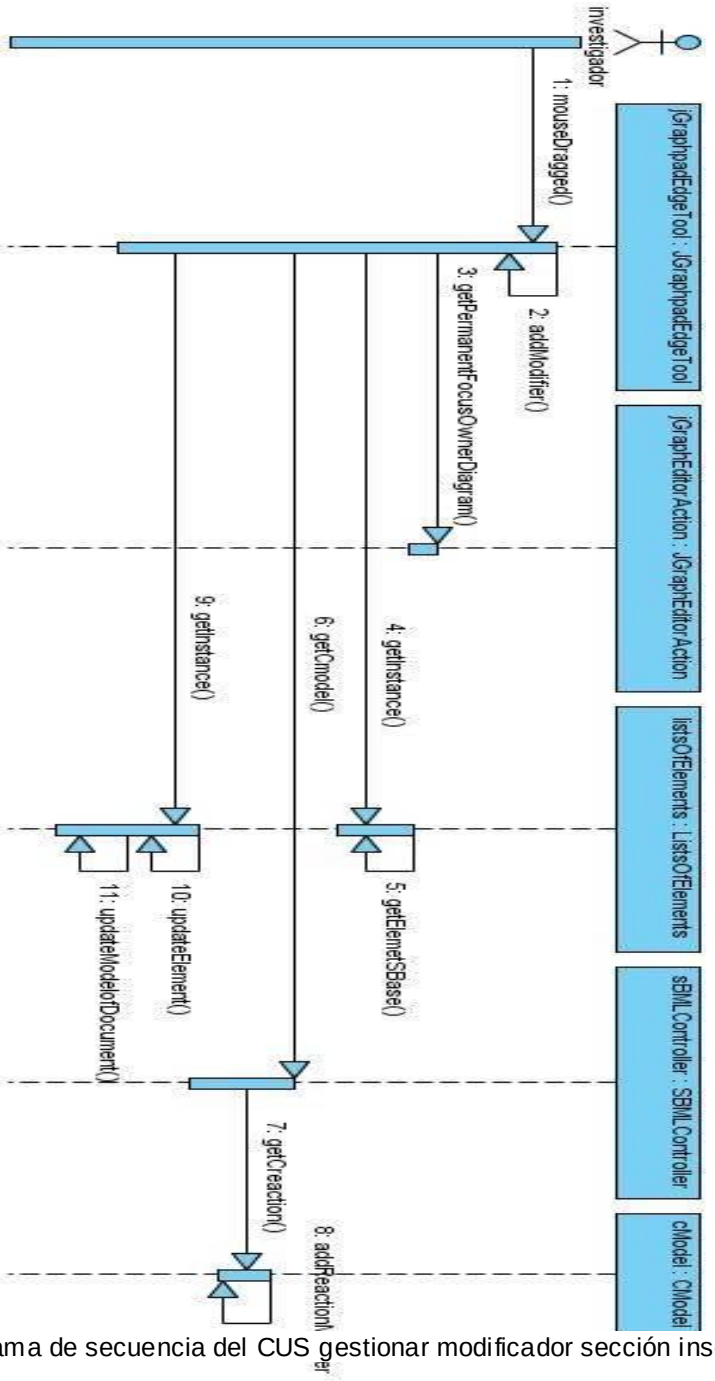


Figura 25. Diagrama de secuencia del CUS gestionar modificador sección insertar

Anexo 5 Diagrama de secuencia del CUS gestionar operador lógico sección eliminar

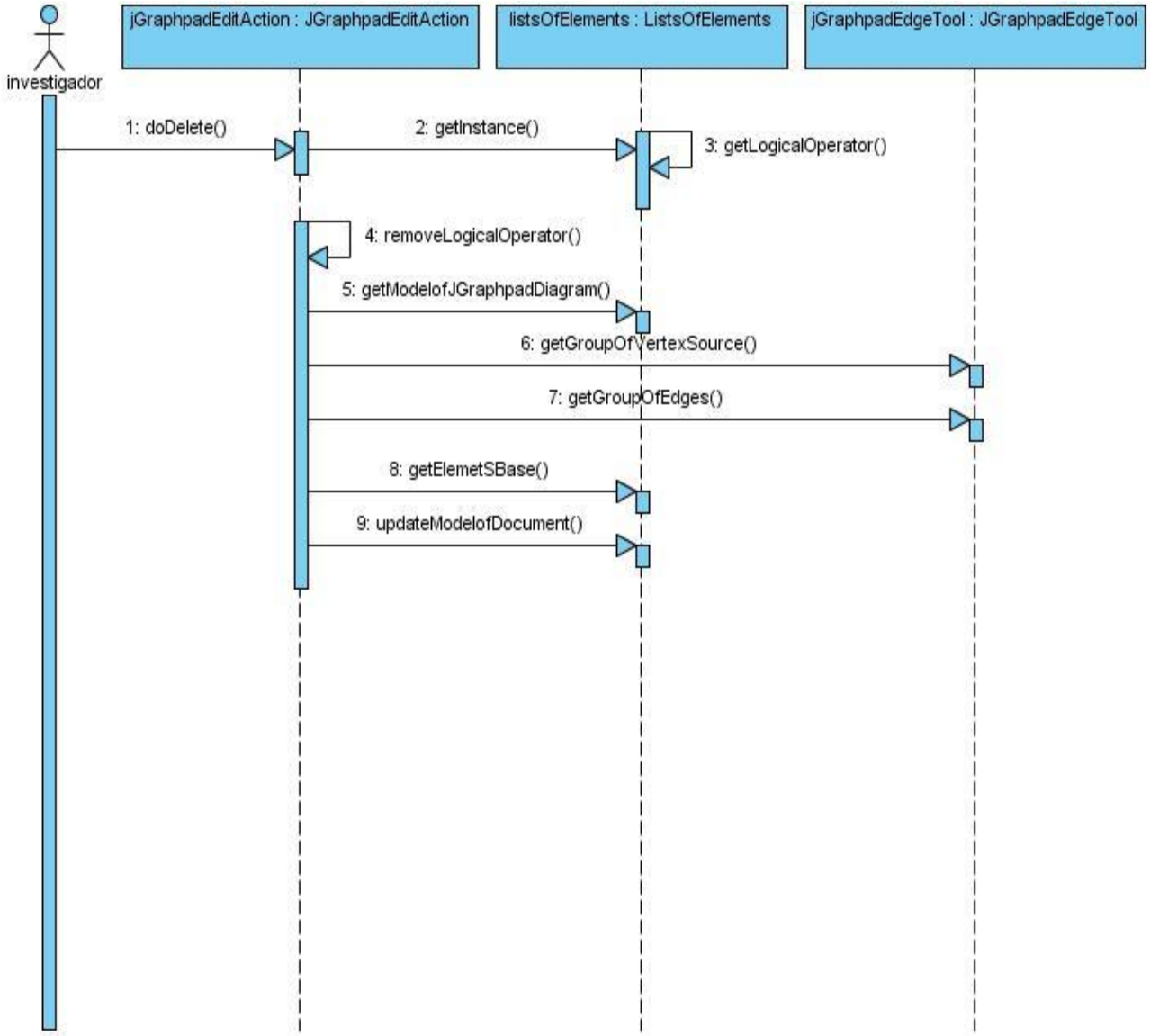


Figura 26. Diagrama de secuencia del CUS gestionar operador lógico sección eliminar

Anexo 6 Diagrama de secuencia del CUS gestionar reactante sección insertar

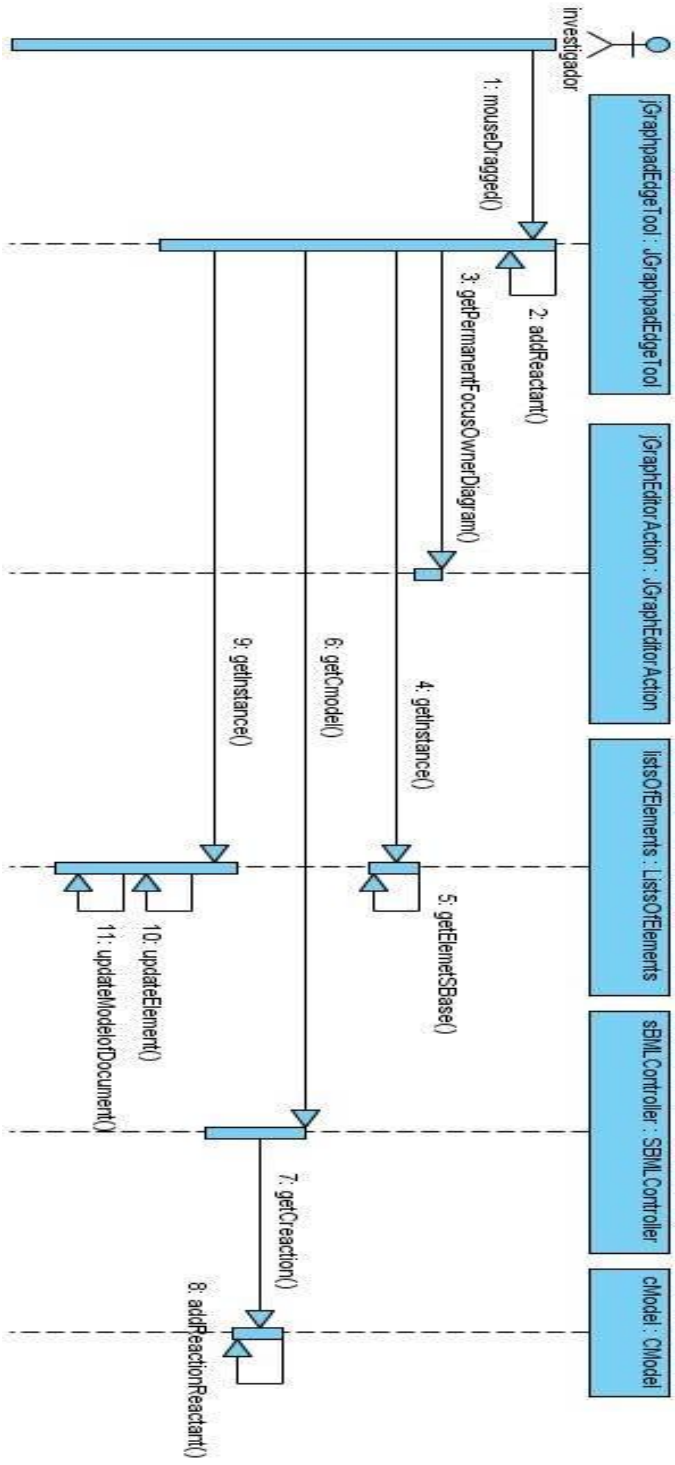


Figura 27. Diagrama de secuencia del CUS gestionar reactante sección insertar

Anexo 7 Diagrama de secuencia del CUS gestionar reactante sección eliminar

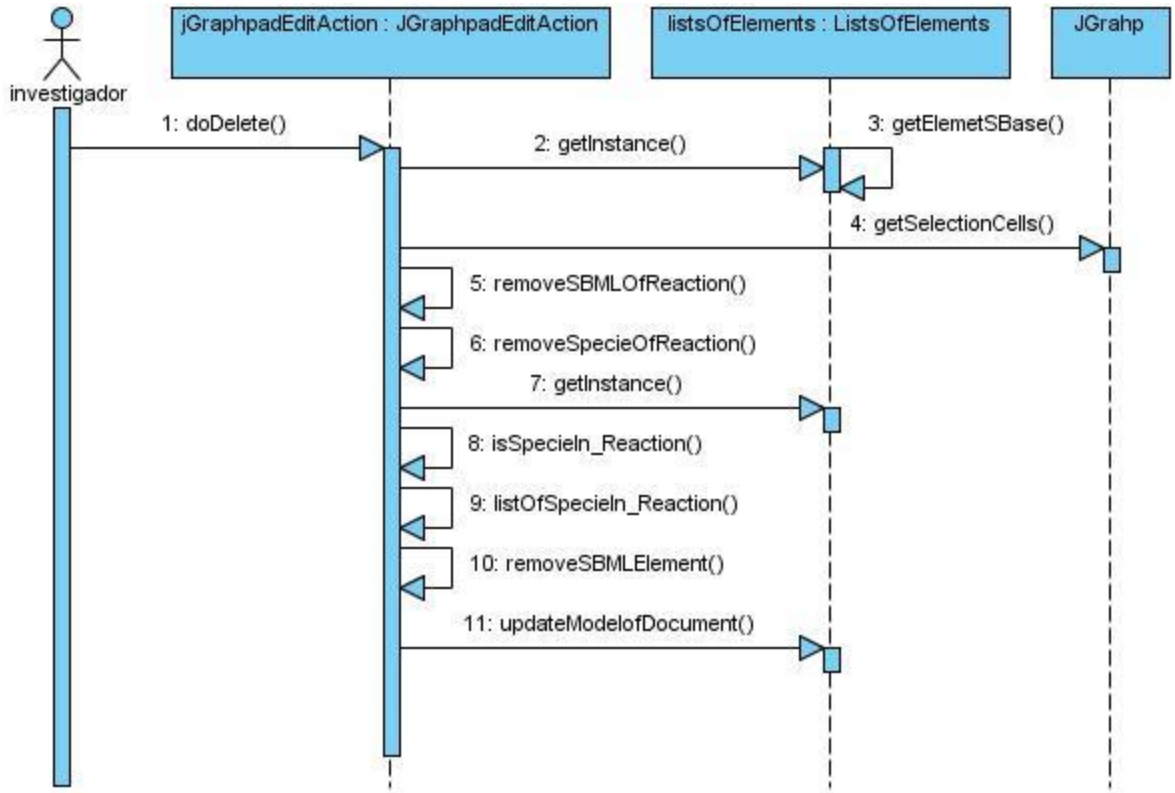


Figura 28. Diagrama de secuencia del CUS gestionar reactante sección eliminar

Anexo 8 Diagrama de secuencia del CUS gestionar producto sección insertar

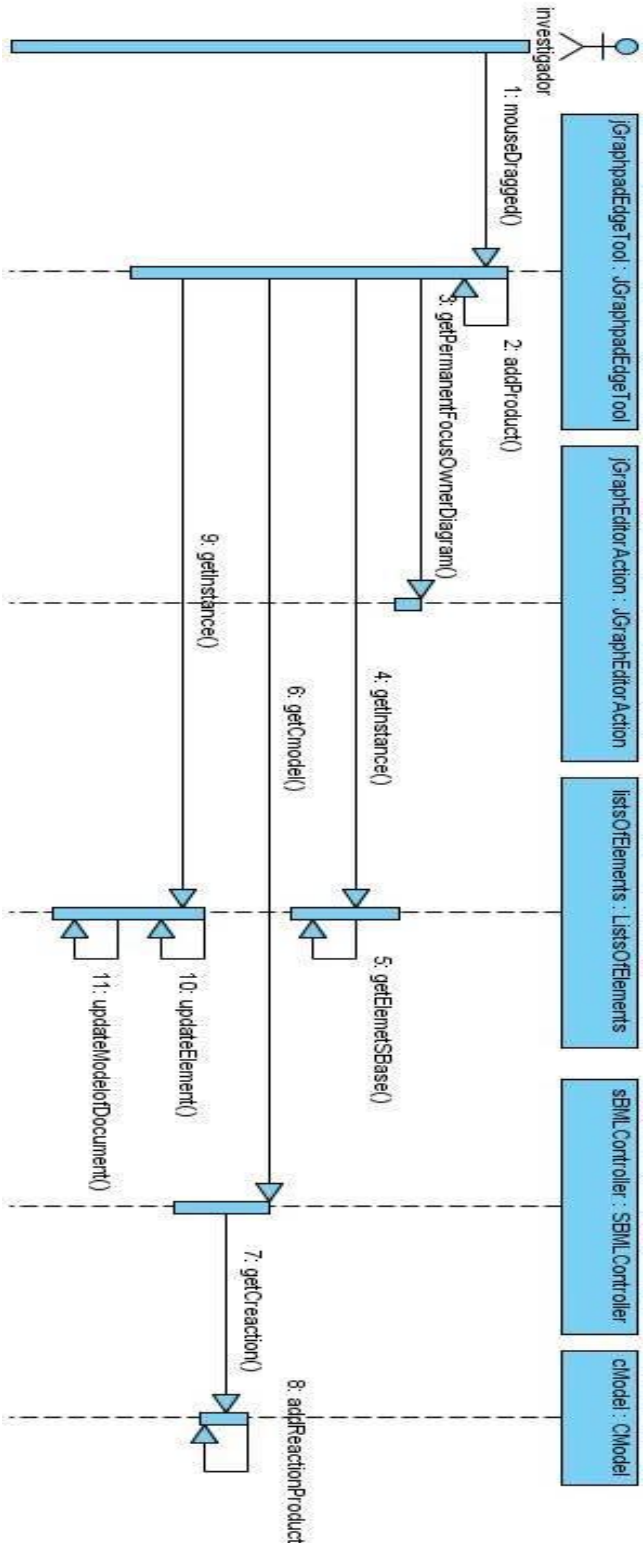


Figura 29. Diagrama de secuencia del CUS gestionar producto sección insertar

Anexo 9 Diagrama de secuencia del CUS gestionar producto sección eliminar

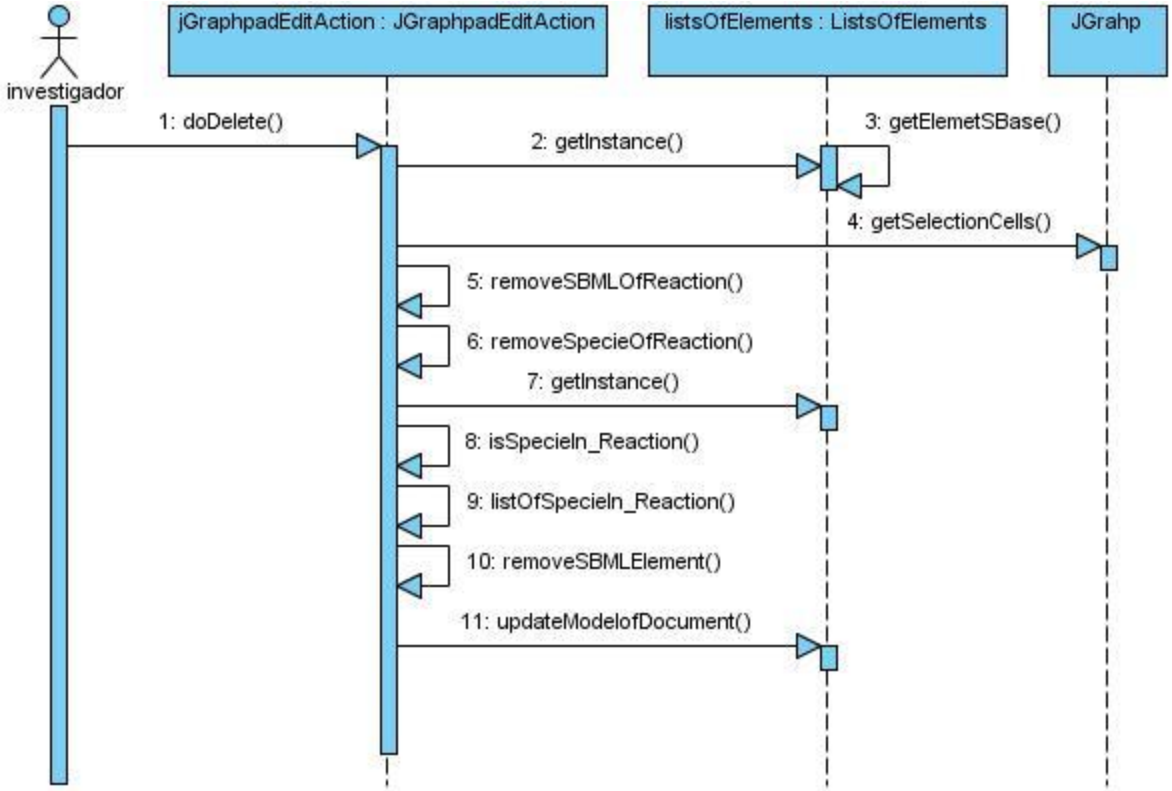


Figura 30. Diagrama de secuencia del CUS gestionar producto sección eliminar

Glosario de términos

Biología de Sistemas: Área de investigación científica que se preocupa del estudio de procesos biológicos usando un enfoque sistémico.

BioSyS: Biological System Simulator. (Simulador de Sistemas Biológicos, en ingles) Software para la simulación de sistemas biológicos.

BioSeek: Compañía dedicada al estudio de modelos de enfermedades humanas para el descubrimiento y desarrollo de fármacos. (http://www.bioseekinc.com)

BioSpice: Biological Simulation Program for Intra- and Inter-Cellular Evaluation. Framework de código abierto y conjunto de herramientas para ayudar a las investigaciones biológicas en el moldeamiento y simulación de procesos de células vivas.
https://biospice.org

Cellnomica:
(http://www.cellnomica.com)

Genstruct: Compañía dedicada a la biología de sistemas. Se centra en el descubrimiento y desarrollo de fármacos.
http://www.genstruct.com

Institute for Advanced Biosciences, Keio University: Instituto de Ciencias Biológicas de la Universidad de Keio, Mita, Japón, es un instituto de investigación académica en el área de la biología de sistemas, utiliza tanto la biología experimental como computacional.
http://www.iab.keio.ac.jp

Institute for Systems Biology: Instituto que reúne a un grupo multidisciplinario de académicos y científicos, de las ramas de la biología, matemática, ingeniería y física en un ambiente interactivo y colaborativo para el estudio de la biología de sistemas.
http://www.systemsbiology.org

In-vitro: (del latín: en vidrio) Procesos que se realizan en recipientes de vidrio, generalmente tubos de ensayo, bajo condiciones asépticas en el laboratorio.

In-vivo: Se refiere al proceso que tiene lugar en el organismo vivo.

Microfluídicos: dispositivos miniaturizados para el procesamiento de muestras de dimensiones celulares.

Modelación: Es un método de obtención del conocimiento, de aplicación en varias ciencias, en el cual se opera con un objeto, no en forma directa sino utilizando cierto sistema intermedio auxiliar conocido como modelo.

Modelo: Representación abstracta de la realidad.

Modelo biológico: Representación gráfica visual de un Sistema biológico.

Munich Systems Biology Forum (MSBF): Foro abierto para permitir el contacto y la colaboración entre grupos de investigación relacionados con la biología de sistemas en el área de Múnich.
http://www.msbf.mpg.de

Nanosensores nanotubos de carbón para el procesamiento de muestras celulares.

Proyecto E-CELL: Proyecto internacional de investigación cuyos objetivos son modelar y reconstruir fenómenos biológicos in-silico, el desarrollo de teorías de soporte necesarias, tecnologías y plataformas de software para permitir la simulación precisa de células enteras.
https://www.e-cell.org

Sistema biológico: sistemas abiertos que operan en condiciones alejadas del equilibrio termodinámico, con muchas y fuertes interacciones no lineales entre sus muchos elementos.

Software: Término genérico que designa al conjunto de programas que posibilitan realizar una tarea específica en un ordenador.

Systems Biology Markup Language (SBML): lenguaje basado en XML, para representar modelos de redes de reacción bioquímica. http://sbml.org/index.psp