Universidad de las Ciencias Informáticas

Facultad 6

Título: "Simulador del Sistema Inmune J-ImmSim"

Trabajo de Diploma para optar por el título de

Ingeniero Informático

Autores: Adriel Acosta Reyes

René Vega Gorgoso

Tutores: Dr. Kalet León Monzón.

Msc. Noel Moreno Lemus

Lic. Yaniela Fernández Mena

Junio 2008

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 firmo la presente a los ____ días del mes de _____ del año _____.

Adriel Acosta Reyes

_____

Firma del Autor

Yaniela Fernández Mena

_____

Firma del Tutor

René Vega Gorgoso

_____

Firma del Autor

Noel Moreno Lemus

_____

Firma del Tutor

Datos de Contacto

Tutores:

Dr. Kalet León Monzón.

Centro de Inmunología Molecular, Habana, Cuba.

Email: kalet@ict.cim.sld.cu

Msc. Noel Moreno Lemus.

Universidad de las Ciencias Informáticas, Habana, Cuba.

Email: noel@uci.cu

Lic. Yaniela Fernández Mena

Universidad de las Ciencias Informáticas, Habana, Cuba.

Email: yaniela@uci.cu

Agradecimientos

De René:

Quiero agradecer especialmente a Fidel y a la Revolución Cubana por lograr que mis sueños se convirtieran en realidades.

Agradecerles a mis padres y a mi hermana, por todo el apoyo que me han dado, por su confianza y amor, por haberme conducido siempre por el camino correcto.

A nuestra tutora Yaniela por su paciencia, apoyo y dedicación en cada uno de nuestros encuentros a lo largo de este curso.

A todos los maestros y profesores que me han formado a lo largo de las diferentes enseñanzas.

A Noel, Hugo, José Albert, Javier, Yunet y Edel por su ayuda incondicional.

A mis abuelos por ser un baúl de enseñanzas y cariño.

A mis tíos por haberme dado el consejo oportuno o el cocotazo necesario.

A Arasay, Yadi, Martha, Ana, Emily, Kenia y demás amigas por haberme soportado durante cinco años.

A Maovys y Denia por tener la capacidad de hacerme sonreír cada día.

A todos mis amigos y compañeros de la universidad, nunca los olvidaré.

A mis demás familiares y a todos los que en algún momento me preguntaron: ¿Y la tesis qué?

Muchas Gracias

De Adriel:

Agradezco a la Revolución y a Fidel, por darme esta oportunidad.

Agradezco a mis padres por el amor, el apoyo, la exigencia y la confianza que siempre me han tenido.

A mis hermanos, por estar en mi ausencia.

A mi novia, por su amor y su confianza.

A Menéndez, Zule y el Clan, por darme siempre aliento y ser mi familia.

A nuestra tutora Yaniela, por sus conocimientos, y por su paciencia con nosotros.

A Noel por incluirme en este proyecto.

A Edel y Javier por su ayuda en programación.

A José Albert por sus consejos.

A Hugo, por su contribución con las EDE.

A todos los maestros y profes que han contribuido en mi formación.

A Tica, Yady, Ana, Emily, Kary, Kenia, Diana y demás chicas, por compartir conmigo estos cinco años, las voy a extrañar.

A mis compañeros de apto, por convivir conmigo, y soportarme.

Gracias a todos

Dedicatoria

A mis padres y hermana

A mi abuela Dora

A mi tía Any (Estamos en paz)

René

A mis padres y hermanos.

A mis abuelos

A Lisi.

A mi familia.

Adriel

Resumen.

Actualmente el Sistema Inmune es uno de los sistemas complejos más estudiados por su estrecha relación con enfermedades como el Cáncer y el VIH/SIDA que causan la pérdida de numerosas vidas humanas cada año. De ahí la importancia de cualquier aporte que favorezca las investigaciones en este sentido. Producto del auge de la Bioinformática en los últimos años se han desarrollado en el mundo herramientas computacionales que simulan el Sistema Inmune. Sin embargo la mayoría de estas herramientas presentan la limitante de que no se ajustan a las necesidades reales de los investigadores, pues son muy específicas o responden a un modelo rígido, es decir, no permiten cambios por parte de los usuarios. El presente Trabajo de Diploma tiene como objetivo desarrollar un software que permita simular el Sistema Inmune de una manera más flexible, es decir, que el biólogo pueda definir sus propios entes biológicos y sus propias interacciones.

Palabras Claves: Sistema Inmune, Sistemas Complejos, Bioinformática, Modelo, Entes Biológicos, Interacciones

Índice

Introducción

En los últimos años la investigación en el campo de las ciencias biológicas ha estado sufriendo drásticas transformaciones. Debido al creciente volumen de información y la complejidad de los cálculos se ha hecho necesario el uso de herramientas que sean capaces de extraer información útil. El momento cumbre tuvo lugar con el inicio del proyecto Genoma Humano donde se crearon grandes bases de datos internacionales de uso compartido. A consecuencia de esto la computadora se incorpora como herramienta fundamental en los procesos investigativos propiciando el surgimiento de una nueva disciplina: la Bioinformática, como un área de investigación multidisciplinaria. La Bioinformática utiliza tecnología de la información para organizar, analizar y distribuir información biológica.

Unido al desarrollo de la Bioinformática las ciencias biológicas han sufrido un cambio de paradigma. Los científicos se han percatado de que el enfoque reduccionista[1] es ineficiente para investigar aquellos sistemas complejos que existen en la naturaleza, lo que ha propiciado el surgimiento de un nuevo enfoque, la Biología de Sistemas (BS), que pretende conocer el funcionamiento de los sistemas biológicos como un todo y estudiar el comportamiento de cada una de sus partes.

Dentro de la BS, el procesamiento de datos, la modelación y la simulación de procesos biológicos han sido las líneas de investigación por las que más interés han mostrado los científicos de esta área. A diferencia de los métodos clásicos que se basan en la confirmación o refutación de hipótesis guiados por resultados experimentales, la BS emplea la modelización como una técnica que permita estudiar o comprender el sistema en análisis y predecir su comportamiento. La simulación y los modelos están estrechamente vinculados; podemos pensar en ella como la realización de experimentos en un modelo, con la finalidad de entender el comportamiento del sistema o evaluar las distintas estrategias (dentro de los límites impuestos por un criterio o conjunto de criterios) para la operación del sistema [1].

Dentro de los sistemas biológicos el Sistema Inmune (SI) es uno de los más estudiados por su estrecha relación con enfermedades como el Cáncer y el VIH/SIDA que causan la pérdida de numerosas vidas humanas cada año. Constituye una red de células, tejidos y órganos que trabajan en conjunto para defender el cuerpo contra ataques de agentes extraños(bacterias, virus, parásitos), generando una respuesta inmune que puede tener dos tipos de clasificaciones: innata o adquirida y humoral o mediada por células.[2]

[1] Este enfoque consiste en descomponer los sistemas biológicos en sus componentes más simples y estudiar estos por separado.

El SI presenta gran cantidad de células (linfocitos) y biomoléculas que interactúan entre sí y participan en diferentes procesos constituyendo un sistema complejo que maneja un conjunto considerable de variables; por tanto, realizar experimentos *in vitro*[1] resulta una labor difícil y costosa, además de enfrentar disyuntivas éticas; la simulación resulta entonces una poderosa alternativa para los científicos, que en años recientes ha estado incorporada de forma creciente en sus investigaciones.

Para simular el sistema inmune se necesitan herramientas que por lo general se basan en el uso de diferentes técnicas de modelación como Ecuaciones Diferenciales, Redes Neuronales, Autómatas Celulares (AC) y Sistemas Basados en Agentes. Actualmente existen diversas aplicaciones que permiten simular el SI basándose fundamentalmente en modelos de ecuaciones diferenciales y autómatas celulares

Uno de los modelos más utilizados constituye el de Seiden y Celada [3], los cuales desarrollaron el software IMMSIM en el lenguaje APL2. Una variante de esta implementación corresponde a C-IMMSIM v.6 desarrollado por Filippo Castiglione en el Instituto de Computación Aplicada en Roma, Italia. Ambas herramientas implementan un modelo particular de autómata celular y son ejemplos que constituyen aportes a los estudios de la teoría del SI.

Sin embargo estas herramientas presentan la limitante común de responder a un modelo específico de funcionamiento, que no es aceptado por la mayoría de los especialistas, sino que responde a la concepción de cada equipo de desarrollo. Surge entonces la necesidad de desarrollar una herramienta que permita simular el SI de manera que el investigador pueda definir su propio modelo.

Por lo cual en la investigación se plantea el siguiente **problema científico**: ¿Cómo realizar la simulación del Sistema Inmune de manera que el investigador pueda crear su propio modelo?

El **objeto de estudio** es la simulación del Sistema Inmune, lo cual define el **campo de acción** como la simulación del SI basado en Autómatas Celulares y Ecuaciones Diferenciales Estocásticas.

Por lo que la investigación se plantea como **objetivo general** desarrollar una herramienta computacional para simular el SI que le permita al investigador crear su propio modelo y como **objetivos específicos** se han definido:

[1] Literalmente "en vidrio", se aplica a los cultivos (o procesos) que se desarrollan en recipientes estériles. [4]

- Analizar el Simulador del SI.
- Diseñar el sistema analizado.
- Implementar el sistema diseñado.

Y para alcanzarlo se han propuesto como tareas las siguientes:

- Investigación del estado actual de los simuladores del SI.
- Obtención de los requerimientos funcionales y no funcionales del sistema.
- Elaboración de un modelo para la simulación del SI.
- Realización del diseño de la herramienta de simulación.
- Creación de los prototipos de interfaz visual.
- Implementación de la herramienta diseñada.
- Realización de las validaciones al simulador.

El resultado práctico que se espera del presente trabajo de diploma es brindar una herramienta capaz de simular la respuesta del sistema inmune a partir de modelos creados por el investigador.

La tesis se encuentra estructurada de la siguiente manera: resumen, introducción, cuatro capítulos, conclusiones, recomendaciones, bibliografía y referencias bibliográficas. Los cuatro capítulos describen los aspectos fundamentales relacionados con el contenido de la investigación y el desarrollo del software. A continuación se describe la distribución de dichos capítulos:

Capítulo 1: Fundamentación Teórica, se definen los principales conceptos en relación con el objeto de estudio, se describen las principales herramientas y metodologías a utilizar durante la investigación.

Capítulo 2: Características del Sistema, se construye y describe un modelo para simular el SI, se presentan los requerimientos funcionales y no funcionales así como el diagrama de casos de usos del sistema.

Capítulo 3: Diseño del Sistema, se describe el diseño de la aplicación, se presentan los diferentes diagramas que dan solución al problema y los patrones de diseño utilizados.

Capítulo 4: Implementación del Sistema, se presenta el modelo de implementación elaborado y los resultados de las validaciones realizadas al software construido.

Capítulo 1
Fundamentación Teórica

Introducción

En este capítulo se definen las bases conceptuales de la investigación. Se describen conceptos relacionados con los sistemas biológicos, la biología de sistemas y algunas técnicas que se utilizan dentro de la BS para modelar y simular los sistemas biológicos. Se exponen las principales características del SI y algunas aplicaciones existentes que lo simulan. Se describen además las diferentes herramientas y metodologías utilizadas y/o consideradas para desarrollar la investigación.

1.1 Biología de Sistemas y modelación de Sistemas Biológicos

1.1.1 Definición de Biología de Sistemas y Sistemas Biológicos

No existe un concepto generalizado de BS, sino que se manejan varios dentro de la comunidad científica. Es una ciencia emergente que trata de integrar diferentes niveles de información para entender el funcionamiento de los sistemas biológicos, estudiándolos como un todo, usando herramientas de modelación, simulación y comparación, integrando otras ciencias como la Física, Matemática, Biología, Química, y Computación. [5][6][7] La materia viva se organiza desde el nivel de molécula hasta el nivel de universo. Los sistemas biológicos pueden ser definidos como sistemas complejos que se pueden clasificar en uno de los niveles en que se organiza la materia viva, y tienen fuertes interacciones lineales o no entre sus muchos elementos. [8]

1.1.2 Modelación y simulación de sistemas biológicos.

El estudio científico teórico de una situación se centra generalmente en un modelo, que es algo que imita las características relevantes de la situación que se está estudiando [9]. Un modelo puede entenderse como una simplificación de la realidad que da una visión parcial de ésta, pero que es lo suficientemente simple para ser manejado en la práctica con un error aceptablemente pequeño. Sin embargo, los modelos no son completos; de lo contrario serían tan complejos como la realidad misma. [10]

Los experimentos controlados son aquellos en los que el entorno y las condiciones del sistema están bajo completo control del experimentador. El fundamental problema al crear modelos biológicos es la dificultad o imposibilidad de realizar experimentos controlados en grandes sistemas animados. El modelo entonces permite estudiar o comprender el sistema real, predecir su comportamiento y proyectar alguna modificación o intervención, que no es más que probar cambios imposibles en el caso real.[9]

Por su parte, la simulación es el proceso del diseño de un modelo de un sistema real y la realización de pruebas en ese modelo, con el propósito de comprender el comportamiento del sistema, ó bien de evaluar las distintas estrategias para la operación del sistema [5]. La simulación ayuda en la investigación al permitir una interacción con el modelo. La diferencia semántica reside en que un modelo es una representación de estructuras, mientras que una simulación infiere un proceso o interacción entre las estructuras del modelo para crear un patrón de comportamiento. [11]

1.1.3 Ventajas y desventajas

Las investigaciones mediante experimentos *in vitro* presentan un grupo de inconvenientes que se deben resolver: grandes tiempos de espera para la obtención de resultados, costos elevados en la obtención de insumos de laboratorios, tales como reactivos e instrumentos de medición, tejidos o animales. Y en el caso, de que se quiera variar algún parámetro (La concentración de una sustancia, o la sustitución de una por otra), es necesario repetir nuevamente todo el ensayo. También se enfrentan dilemas éticos respecto a la utilización o no de humanos para realizar experimentos. Incluso, existen grupos de protección de animales que se oponen a la utilización de los mismos [12]. La modelación y simulación de sistemas biológicos se convierte entonces en una potente herramienta para los biólogos ya que minimiza los costes y riesgos antes mencionados. Estas debilidades de los métodos clásicos, constituyen la fortaleza de los métodos de simulación.

En resumen, la simulación de los sistemas biológicos:

- Brinda facilidades para variar parámetros y condiciones experimentales.
- Acelera los procesos de investigación.
- Minimiza los costos de investigación y producción.
- Elimina los problemas éticos de la utilización de animales y humanos en la experimentación.
- Posibilita la aparición de nuevos modelos a partir de los existentes y probados.

Algunas desventajas que se deben considerar y valorar:

- Poca preparación o experiencia de los investigadores en la utilización de métodos de simulación.
- Carencia de software necesario para la investigación.
- Ineficiente poder de cálculo necesario para la simulación de sistemas sumamente complejos que constituyen los sistemas biológicos.

1.2 El sistema inmune, sus componentes y técnicas de modelación.

Sistema inmunológico, también llamado sistema inmune, es el sistema corporal cuya función primordial consiste en destruir los agentes patógenos que encuentra. Cualquier agente considerado extraño por un sistema inmunológico se denomina antígeno. La responsabilidad del sistema inmunológico es enorme y debe presentar una gran diversidad, con objeto de reaccionar de forma adecuada con los miles de antígenos, patógenos potenciales diferentes, que pueden invadir el cuerpo. Aún no se conocen en su totalidad los mecanismos fisiológicos complejos implicados en el sistema inmunológico, pero la investigación médica continúa desentrañándolos [13].

1.2.1 Sistema Inmune

Células

Las tres categorías de células inmunológicas son granulocitos, monocitos/macrófagos y linfocitos. Los granulocitos son las células con núcleo más abundantes en la sangre. Estas células fagocitan (ingieren) los antígenos que penetran en el cuerpo, sobre todo si estos antígenos han sido recubiertos en la sangre por inmunoglobulinas o por proteínas del sistema del complemento. Una vez ingeridos, los antígenos suelen ser destruidos por las potentes enzimas de los granulocitos.

Los monocitos constituyen un pequeño porcentaje de la totalidad de las células sanguíneas; cuando se encuentran localizados en los tejidos, fuera de la circulación sanguínea, experimentan cambios físicos y morfológicos, y reciben el nombre de macrófagos. Al igual que los granulocitos, los monocitos también ingieren sustancias extrañas, interaccionan con las inmunoglobulinas y con las proteínas del complemento, y contienen enzimas potentes dentro de su citoplasma. Sin embargo, los monocitos alteran además los antígenos, haciendo que la respuesta inmune del tercer tipo de células inmunológicas, los linfocitos, sea más fácil y más eficaz.

En algunos aspectos, los linfocitos son las células más importantes del sistema inmunológico. Existen dos tipos principales de linfocitos: los linfocitos B y los linfocitos T. Los primeros son responsables de la inmunidad humoral o serológica; es decir, los linfocitos B y sus descendientes directos, que reciben el nombre de células plasmáticas, son las células responsables de la producción de unos componentes del suero de la sangre, denominados inmunoglobulinas. Los linfocitos T son responsables de la inmunidad celular; es decir, atacan y destruyen directamente a los antígenos. Estas células también amplifican o suprimen la respuesta inmunológica global, regulando a los otros componentes del sistema inmunológico, y segregan gran variedad de citoquinas. Los linfocitos T constituyen el 70% de todos los linfocitos. Tanto los linfocitos T como los linfocitos B tienen la capacidad de recordar, desde el punto de vista bioquímico, una exposición previa a un antígeno específico, de manera que si la exposición es repetida puede producirse una destrucción más eficaz del antígeno.[14]

La respuesta inmunológica

Los componentes del sistema inmunológico actúan como un todo para desarrollar una respuesta inmunitaria eficaz. La investigación ha conseguido demostrar cómo suceden muchas de las etapas de este proceso; otras fases aún son especulativas y están siendo investigadas. Sin embargo, el proceso básico es el siguiente: cuando un antígeno patógeno, por ejemplo una bacteria, consigue superar la primera línea de defensa del cuerpo, por ejemplo la piel, se encuentra en primer lugar con los granulocitos y los monocitos, y es neutralizado en parte por anticuerpos preexistentes y por las proteínas del complemento. Después, los linfocitos y los macrófagos interaccionan en el lugar donde ha entrado la bacteria, amplificando la respuesta inmunológica; se sintetizan anticuerpos más específicos y eficaces, debido a la memoria inmunológica generada por la bacteria invasora. En los ganglios linfáticos más próximos puede tener lugar una amplificación similar de la respuesta inmunológica, así como en lugares más distantes, tales como el bazo y la médula ósea, donde también se sintetizan linfocitos.

Si todo funciona, el sistema inmunológico supera a la bacteria, de manera que la enfermedad está ya bajo control. En este momento se ponen en funcionamiento mecanismos autorreguladores supresores que detienen la respuesta inmunológica; las citoquinas tienen gran importancia en este proceso supresor. Si el sistema inmunológico no está autorregulado de una manera adecuada, se pueden originar otras enfermedades de naturaleza inmunopatológica. Una vez que el antígeno es destruido mediante esta combinación de acciones, el sistema inmunológico está preparado para responder de una manera más eficaz si el mismo tipo de microorganismo invadiera de nuevo el cuerpo. Si dicha preparación es adecuada para neutralizar totalmente a una bacteria específica antes de que ésta

produzca la enfermedad, se dice entonces que existe inmunidad frente a dicha bacteria. [14]

Proteínas

Los tres tipos de proteínas que forman parte del sistema inmunológico, y se encuentran disueltas en el suero (la porción líquida de la sangre), son las inmunoglobulinas, las citoquinas y las proteínas del complemento. Hay miles de clases diferentes de inmunoglobulinas, que reciben el nombre de anticuerpos; cada una de ellas se combina de manera exacta con un tipo específico de antígeno y contribuye a su eliminación. Esta inmensa diversidad es la característica principal del sistema inmunológico en conjunto.

Las citoquinas son compuestos solubles, responsables en gran parte de la regulación de la respuesta inmunológica. Si son segregadas por los linfocitos, reciben el nombre de linfoquinas; si son segregadas por los monocitos, se denominan monoquinas. Algunas citoquinas amplifican o incrementan una respuesta inmunológica que está en curso, otras hacen que las células proliferen, y otras pueden suprimir una respuesta inmunológica en funcionamiento. El sistema inmunológico, al igual que otros sistemas corporales, debe ser regulado de esta forma, de modo que el sistema esté activo cuando sea necesario, pero que no lo esté de una manera patológica.

Las proteínas del complemento forman una familia de compuestos que, junto con las inmunoglobulinas, actúan para propiciar una respuesta inmunológica adecuada. Una vez que un anticuerpo se une específicamente a su antígeno, las proteínas del complemento pueden unirse al complejo formado de esta forma, y facilitan que las células inmunológicas lleven a cabo la fagocitosis. [14]

1.2.2 Simulación del sistema inmune

Al constituir el sistema defensivo frente a los organismos extraños, el SI es constante objeto de investigación por parte de los científicos, de ahí que propiciar herramientas de simulación que permitan su estudio es sumamente importante.

Las Ecuaciones Diferenciales (ED) han sido utilizadas históricamente para la modelación y simulación de sistemas complejos en diferentes ciencias como la Física, Química y Biología. Pero el modelo que se logra con las mismas puede tener implícito algunas deficiencias y problemas. Si se tiene presente que el SI es sumamente complejo y dinámico, las ED pueden ser aún menos efectivas, aunque se usen

Ecuaciones Diferenciales Estocásticas (EDE).

A continuación se relacionan algunos de los inconvenientes de las ED para modelar y simular el SI:

- Solamente relacionan el comportamiento medio de un sistema.
- No pueden representar la complejidad del SI, con su enorme cantidad de células, moléculas, interacciones y procesos. (Ej. Las células tienen una historia y características individuales que influyen sobre su comportamiento).
- Los modelos de SI implican a menudo (Mientras más realista se quiera, mayor será la complejidad del sistema de ED), no linealidades que dificultan la resolución de las ecuaciones con métodos numéricos.

Sin embargo estas limitaciones podrían solucionarse modelando mediante otras técnicas de simulación como un Autómata Celular (AC) [15] que básicamente es una herramienta computacional que hace parte de la Inteligencia Artificial basada en modelos biológicos, el cual está básicamente compuesto por una estructura estática de datos y un conjunto finito de reglas que son aplicadas a cada nodo o elemento de la estructura. La potencia de los AC radica en que definiendo reglas locales logran crear complejidad en su comportamiento. [16]

1.2.3 Sistemas anteriores para la simulación del sistema Inmune

El funcionamiento exacto del SI está lejos todavía de ser conocido. Los avances en las últimas décadas han sido considerables, pero la complejidad de este sistema es insondable. Se cuenta con un grupo de conocimientos bien definidos y probados, pero existe un grupo aún mayor de interrogantes por resolver. Por tanto, la única manera de progresar es insistir en la investigación diaria, la utilización de nuevos métodos de investigación (Métodos de BS), y la elaboración de herramientas computacionales (Simuladores) que satisfagan las necesidades de los biólogos e investigadores. A continuación se describen algunos simuladores existentes para estudiar el SI.

IMMSIM Este software fue propuesto por F. Celada y P.E. Seiden en 1992; programado en APL2. En su primera versión sólo implementaba la respuesta humoral; la versión 3.0 incluye además la respuesta mediada por células. Este simulador tiene como principal desventaja que incluye solo un grupo de componentes del SI; es decir, sólo las células, interacciones y procesos que sus desarrolladores pensaron necesarios están presentes en el modelo implementado.

C-IMMSIM Software OpenSource desarrollado por Filippo Castiglione, Roma, Italia. Basado en el modelo de Celada y Seiden. Concluida la versión 6.0 en abril de 2006. Desarrollado en lenguaje C, implementa respuesta humoral y mediada por células. Hereda los problemas de ImmSim; entidades, interacciones y procesos estáticos, imposible realizar cambios a nivel de usuario. Se divide en dos funcionalidades: un ejecutable para especificar los datos de entradas que se almacenan en un fichero de texto y el simulador propiamente dicho accediéndose a este desde una interfaz poco amigable.

IMMSIM++ Software desarrollado por Steven Kleinstein en la universidad de Princeton, Estados Unidos; basado en ImmSim y C-ImmSim, programado en C++. Sólo modela entidades de la inmunidad humoral y al igual que los anteriores responde a un modelo rígido. Logra mejoras en la integración de las diferentes funcionalidades. Además permite determinar qué gráficas representar y controlar los parámetros mientras ocurre la simulación.

Como se ha podido comprobar, estos simuladores presentan una desventaja muy significativa; no es posible para los biólogos introducir y experimentar con sus propios modelos. Las células, procesos e interacciones están predefinidas, prohibiendo a los usuarios realizar cambios, sólo variar los valores de los parámetros pertinentes. El SI no tiene un funcionamiento bien definido y por tanto un único modelo no cumple con las expectativas y necesidades de los investigadores. Esto hace que las referidas herramientas de simulación del SI, carezcan de un valor práctico, aún cuando deben ser consideradas como la base para desarrollar un nuevo simulador del SI.

Por tanto surge la necesidad de desarrollar una herramienta que permita flexibilizar el estudio del SI por vía computacional, es decir, de manera genérica, que brinde al investigador la facilidad de representar su propio modelo.

1.3 Autómatas Celulares

1.3.1 Definición de los Autómatas Celulares

Los AC son redes de autómatas simples conectados localmente. Cada autómata simple produce una salida a partir de varias entradas, modificando en el proceso su estado según una función de transición. Por lo general, en un AC, el estado de una célula en una generación determinada depende única y exclusivamente de los estados de las células vecinas y de su propio estado en la generación anterior [17]. Existen dos tipos principales de vecindarios conocidos, el Vecindario Von Neumann de cinco

vecinos Figura 1.1a) y el vecindario Moore de nueve vecinos Figura 1.1 b).

Figura 1.1 Tipos de Vecindades a) Von Neumann b) Moore

Los AC son herramientas útiles para modelar cualquier sistema en el universo. Pueden considerarse como una buena alternativa a las ecuaciones diferenciales y han sido utilizados para modelar sistemas físicos, como interacciones entre partículas, formación de galaxias, cinética de sistemas moleculares y crecimiento de cristales, así como diversos sistemas biológicos a nivel celular, multicelular y poblacional. [17]

1.3.2 Estructura de los Autómatas Celulares

Se definen como componentes básicos de un AC:

- Un plano bidimensional o un espacio n-dimensional dividido en un número de subespacios homogéneos, conocidos como celdas. A todo esto se le denomina Teselación Homogénea.
- Cada celda puede estar en un estado, de un conjunto finito o numerable S de estados.
- Una Configuración C, la que consiste en asignarle un estado a cada celda del autómata.
- Una Vecindad definida para cada celda, la que consiste en un conjunto contiguo de celdas, indicando sus posiciones relativas respecto a la celda misma.
- Una Regla de Evolución, la cual define cómo debe cada celda cambiar de estado, dependiendo del estado inmediatamente anterior de su vecindad. Puede ser simétrica o no y puede incluir o no a la propia celda.
- Un Reloj Virtual de Cómputo conectado a cada celda del autómata, el cual generará "tics" o pulsos simultáneos a todas las celdas indicando que debe aplicarse la regla de evolución y de esta forma cada celda cambiará de estado. [16]

Clasificación

Stephen Wolfram ha realizado numerosas investigaciones sobre el comportamiento cualitativo de los AC. Con base en su trabajo sobre AC unidimensionales, con dos o tres estados, sobre configuraciones periódicas que se presentan en el AC, observó sus evoluciones para configuraciones iniciales aleatorias. Así, dada una regla, el AC exhibe diferentes comportamientos para diferentes condiciones iniciales.

Wolfram clasificó el comportamiento cualitativo de los AC unidimensionales. De acuerdo con esto, un AC pertenece a una de las siguientes clases:

- Clase I. La evolución lleva a una configuración estable y homogénea, es decir, todas las células terminan por llegar al mismo valor.
- Clase II. La evolución lleva a un conjunto de estructuras simples que son estables o periódicas.
- Clase III. La evolución lleva a un patrón caótico.
- Clase IV. La evolución lleva a estructuras aisladas que muestran un comportamiento complejo (es decir, ni completamente caótico, ni completamente ordenado, sino en la línea entre uno y otro, este suele ser el tipo de comportamiento más interesante que un sistema dinámico puede presentar). [17]

Los AC son útiles en la construcción de modelos donde los elementos base o componentes (actores) son de similar naturaleza y comportamiento; donde éstos se rigen por reglas parecidas y donde, en el mismo sistema real, se identifican componentes diferenciables, independientes, aislables y/o discretos.

El modelo utilizado en el desarrollo de la aplicación es un caso particular de autómata celular propuesto por Seiden y Celada [18] que tiene como características principales:

- Está formada por una matriz discreta de sitios
- Evoluciona en tiempos discretos
- El valor de cada sitio evoluciona de acuerdo a las mismas reglas probabilísticas
- Las reglas para la evolución de un sitio dependen solamente del sitio mismo
- Las entidades se mueven de un sitio a otro

Este modelo basado en AC constituye una de las alternativas más avanzadas para simular el SI, no teniendo como base de la simulación, un sistema de ED como se ha venido utilizando por décadas.

1.4 Ecuaciones Diferenciales Estocásticas

Muchos procesos en la naturaleza (Físicos, Químicos y Biológicos), han sido descritos mediante ecuaciones diferenciales(ED), que son ecuaciones en las que intervienen derivadas de una o más funciones. Se les conoce como ecuaciones de tipo Riemann-Stieltjes, y presentan sus formas propias de tratamiento, es decir, de integración y solución. También se les denominan como integrales deterministas; cada vez que se resuelva una de ellas se obtendrá la misma solución. Son idóneas para modelar sistemas determinísticos. Pero son pocos los sistemas de la naturaleza que son realmente de este tipo; por tanto, utilizar una ED de tipo Riemann-Stieltjes para modelar un proceso no determinista supone un nivel de abstracción elevado y se puede perder información útil para la investigación. Todo depende del rigor con que se quiera trabajar sobre el sistema (Ej. Un sistema biológico no da un amplio margen de error). Por las razones anteriores ha surgido una extensión de las ED, que sí posibilitan la modelación de procesos no deterministas: las ecuaciones diferenciales estocásticas (EDE).

Las EDE tienen la siguiente forma general: $x_t = x_0 + \int_0^t f(x_t,t)ds + \int_0^t g(x_t,t)dz$ donde la integral $\int_0^t f(x_t,t)ds$ es conocida como la Integral de Riemann-Stieltjes. La integral $\int_0^t g(x_t,t)dz$ no es de Riemann- Stieltjes ya que aunque z es continua no es de variación acotada. [37]

Para resolver esta ecuación diferencial se define la Integral de **g** respecto a W (Browniano) en el intervalo [a, b] para toda función g en el espacio $L^2([a,b],dt)$ de la forma $g(t) := \sum_{i=1}^{n} c_i L_{[t_{i-1},t_i]}(t)$

Teniendo que $a = t_0 < t_1 < \ldots < t_n = b$ entonces $\int_a^b g(x_t,t)dz = \sum_{i=0}^{n} c_i \left(w_{t_i} - w_{t_{i-1}}\right)$

La fundamentación anterior permitirá resolver la EDE utilizada para regular el proceso de generación de células que se lleva a cabo en la Médula Ósea. Este proceso se explicará en el siguiente capítulo.

1.5 Tendencias y tecnologías actuales para el desarrollo del software.

Para lograr el objetivo general de la investigación se hace necesario el uso de una metodología que guíe el proceso de desarrollo así como de herramientas y lenguajes que permitan el desarrollo del mismo.

1.5.1 Metodologías de desarrollo y lenguajes de modelado.

Un proceso de desarrollo de software define quién está haciendo qué, cuándo y cómo alcanzar un determinado objetivo; en la ingeniería de software el objetivo es construir un producto de software o mejorar uno existente con calidad. [20]

El alcance y complejidad de los sistemas informáticos que se desarrollan hoy en día, hace necesario el uso de una metodología de desarrollo que permita organizar y controlar los procesos de su producción y mantenimiento. Existen varias metodologías en la actualidad. Entre ellas se pueden citar SCRUM, Crystal Methodologies, Dynamic Systems Development Method, RUP, OpenUP, entre otras.

SCRUM: Desarrollada por Ken Schwaber, Jeff Sutherland y Mike Beedle. Define un marco para la gestión de proyectos, que se ha utilizado con éxito durante los últimos 10 años. Está especialmente indicada para proyectos con un rápido cambio de requisitos. Sus principales características se pueden resumir en dos. El desarrollo de software se realiza mediante iteraciones, denominadas sprints, con una duración de 30 días. El resultado de cada sprint es un incremento ejecutable que se muestra al cliente. La segunda característica importante son las reuniones a lo largo proyecto, entre ellas destaca la reunión diaria de 15 minutos del equipo de desarrollo para coordinación e integración. [39]

Crystal Methodologies: Se trata de un conjunto de metodologías para el desarrollo de software caracterizadas por estar centradas en las personas que componen el equipo y la reducción al máximo del número de artefactos producidos. Han sido desarrolladas por Alistair Cockburn. El desarrollo de software se considera un juego cooperativo de invención y comunicación, limitado por los recursos a utilizar. El equipo de desarrollo es un factor clave, por lo que se deben invertir esfuerzos en mejorar sus habilidades y destrezas, así como tener políticas de trabajo en equipo definidas. Estas políticas dependerán del tamaño del equipo, estableciéndose una clasificación por colores, por ejemplo Crystal Clear (3 a 8 miembros) y Crystal Orange (25 a 50 miembros). [39]

XP (Programación Extrema), es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. Se define como especialmente adecuada para proyectos con requisitos imprecisos y muy

cambiantes, y donde existe un alto riesgo técnico. [39]

OpenUP/Basic: es un proceso de desarrollo unificado que está basado en Rational Unified Process (RUP), desarrollado por IBM y reconocido mundialmente como uno de los procesos de desarrollo de software de mayor calidad, basándose en los principios de Adaptación, Importancia a los involucrados e interesados en los resultados del proyecto; Colaboración, Valor a la iteración; y Calidad Continua.

Esta metodología es un FrameWork de procesos de desarrollo de software de código abierto, que con el tiempo espera cubrir un amplio conjunto de necesidades en el campo del desarrollo de software. OpenUP/Basic permite un abordaje ágil al proceso de desarrollo de software, con sólo proveer un conjunto simplificado de contenidos, fundamentalmente relacionados con orientación, productos de trabajo, roles, y tareas. OpenUP/Basic es un proceso interactivo de desarrollo de software simplificado, completo y extensible. Es un proceso para pequeños equipos de desarrollo que valoran los beneficios de la colaboración y de los involucrados con el resultado del proyecto, por encima de formalidades innecesarias.

Para el ámbito académico OpenUP/Basic, da soporte a la investigación debido a su condición de ser un proyecto de código abierto (Open Source), y sirve como plataforma de enseñanza en materia de procesos de desarrollo de software. [21]

¿Por qué OpenUP?

OpenUP es una metodología ágil para pequeños equipos de desarrollo que evita formalismos innecesarios y está más centrada en el producto de la investigación. Está basada en RUP, metodología en la cual el equipo de desarrollo tiene experiencia. Además su extensibilidad propicia que la misma se adapte fácilmente a las necesidades existentes en el desarrollo del presente trabajo.

Lenguaje de modelado

El Lenguaje Unificado de Modelado (UML, por sus siglas en inglés, Unified Modeling Language), es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad. Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocios y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables.

Es importante resaltar que UML es un "lenguaje" para especificar y no para describir métodos o procesos. Se utiliza para definir un sistema de software, para detallar los artefactos en el sistema y para documentar y construir. En otras palabras, es el lenguaje en el que está descrito el modelo. Se puede aplicar en una gran variedad de formas para dar soporte a una metodología de desarrollo de software (tal como el Proceso Unificado de Rational, el OpenUP) -pero no especifica en sí mismo qué metodología o proceso usar.

UML cuenta con varios tipos de diagramas, los cuales muestran diferentes aspectos de las entidades representadas. Y en dependencia de la metodología y del software a desarrollar se implantarán unos diagramas, otros se omitirán. [22]

1.5.2 Herramientas CASE

En las últimas décadas se ha trabajado en el área de desarrollo de sistemas para encontrar técnicas que permitan incrementar la productividad y el control de calidad en cualquier proceso de elaboración de software, y hoy en día la tecnología CASE (Computer Aided Software Engineering) reemplaza al papel y al lápiz por el ordenador para transformar la actividad de desarrollar software en un proceso automatizado. Se puede definir a las Herramientas CASE como un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del Ciclo de Vida de desarrollo de un Software.

La tecnología CASE permite la automatización del desarrollo del software, contribuyendo así a elevar la productividad y la calidad de en el desarrollo de los sistemas de información. En este nuevo enfoque que persigue mejorar la calidad del software e incrementar la productividad en el proceso de desarrollo del mismo, se plantean los siguientes objetivos:

- Permitir la aplicación práctica de metodologías, lo que resulta muy difícil sin emplear herramientas.
- Facilitar la realización de prototipos y el desarrollo conjunto de aplicaciones.
- Simplificar el mantenimiento del software.
- Mejorar y estandarizar la documentación.
- Aumentar la portabilidad de las aplicaciones.
- Facilitar la reutilización de componentes de software
- Permitir un desarrollo y un refinamiento (visual) de las aplicaciones, mediante la utilización de controles gráficos (piezas de código reutilizables).

Rational Rose

Es una de las más poderosas herramientas de modelado visual para el análisis y diseño de sistemas basados en objetos. Se utiliza para modelar un sistema antes de proceder a construirlo. Cubre todo el ciclo de vida de un proyecto. Entre sus principales características se encuentran las siguientes:

- Soporta UML 2.0
- Permite generar código a partir de modelos de los siguientes lenguajes de programación: Ada, ANSI C++, C++, CORBA, Java/J2EE, Visual C++ y Visual Basic.
- Soporte a modelos de análisis, ANSI C++, Rose J y Visual C++ según el documento.
- Los componentes del modelo se pueden controlar independientemente, lo que permite una gestión y un uso de modelos más granular.
- Soporte para compilación de las construcciones más habituales de Java 1.5.
- Generación de código en lenguaje Ada, ANSI C++, C++, CORBA, Java y Visual Basic, con funciones configurables de sincronización entre los modelos y el código.
- Complemento de modelado Web que incluye funciones de visualización, modelado y herramientas para desarrollar aplicaciones Web.
- Modelado en UML para diseñar bases de datos, que integra los requisitos de datos y aplicaciones mediante diseños lógicos y analíticos.
- Creación de definiciones de tipo de documento DTD en XML.
- Integración con otras herramientas de desarrollo de IBM Rational.
- Integración con cualquier sistema de control de versiones compatible con SCC, como IBM Rational ClearCase.
- Posibilidad de publicar en la Web modelos e informes para mejorar la comunicación entre los miembros del equipo.[23]

Visual Paradigm

Visual Paradigm es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software. Entre sus características principales se encuentran:

- Soporta hasta la fecha UML 2.1 y BPMN (Business Process Modeling Notation)
- Permite realizar ingeniería tanto directa como inversa en más de 10 lenguajes tales como: Java, C++, CORBA IDL, PHP, XML Schema, Ada y Python, entre otros.

- La herramienta es colaborativa, es decir, soporta múltiples usuarios trabajando sobre el mismo proyecto.
- Genera la documentación del proyecto automáticamente en varios formatos como Web o PDF, y permite control de versiones.
- Soporta la sincronización entre el código en Java y los modelos permitiendo trabajar en cualquiera de las dos partes.
- Permite importar o exportar archivos de otras herramientas CASE como Rational Rose.
- Es multiplataforma. [24]

¿Por qué Visual Paradigm?

Rational Rose es una herramienta potente y muy usada en el mundo pero no se puede ejecutar sobre el Sistema Operativo Linux, plataforma sobre la que se trabaja en la presente investigación. Visual Paradigm por su parte es multiplataforma, se integra con NetBeans, características que hacen seleccionarla como herramienta CASE para el desarrollo de la investigación.

1.5.3 Lenguaje de programación

C++

El C++ es un lenguaje de programación, diseñado a mediados de los años 1980, por Bjarne Stroustrup, como extensión del lenguaje de programación C. Se puede decir que C++ es un lenguaje que abarca tres paradigmas de la programación: la programación estructurada, la programación genérica y la programación orientada a objetos.

Las principales características del C++ son las facilidades que proporciona para la programación orientada a objetos y para el uso de plantillas o programación genérica (templates). Además posee una serie de propiedades difíciles de encontrar en otros lenguajes de alto nivel:

- Posibilidad de redefinir los operadores (sobrecarga de operadores)
- Identificación de tipos en tiempo de ejecución (RTTI)

C++ está considerado por muchos como el lenguaje más potente, debido a que permite trabajar tanto a alto como a bajo nivel, sin embargo es a su vez uno de los que menos automatismos trae (obliga a hacerlo casi todo manualmente al igual que C, como la liberación de memoria dinámica), lo que

"dificulta" mucho su aprendizaje. [25]

Algunos de los compiladores más conocidos:

- Borland C++.
- Microsoft Visual C++.
- G++

Java

Java es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems a principios de los años 1990. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel como punteros. [26]

El lenguaje Java se creó con cinco objetivos principales:

- Usar la metodología de la programación orientada a objetos.
- Permitir la ejecución de un mismo programa en múltiples sistemas operativos.
- Incluir por defecto soporte para trabajo en red.
- Diseñarse para ejecutar código en sistemas remotos de forma segura.
- Ser fácil de usar y tomar lo mejor de otros lenguajes orientados a objetos, como C++.

Java cuenta entre sus potencialidades que es orientado a objetos, es independiente de la plataforma, posee un recolector de basura propio y es muy robusto y flexible.

¿Por qué Java?

Se han visto un grupo de características referentes a C++ y Java. Se utilizará el segundo, por ejecutarse independiente de la plataforma, ser sumamente flexible y eliminar la complejidad y responsabilidades que supone programar con C++.

1.5.4 Entorno integrado de desarrollo (IDE)

NetBeans

NetBeans se refiere a una plataforma para el desarrollo de aplicaciones de escritorio usando Java y a un

Entorno Integrado de Desarrollo (IDE) desarrollado usando la Plataforma NetBeans. Es un proyecto de código abierto de gran éxito con una gran base de usuarios, una comunidad en constante crecimiento, y con cerca de 100 socios en todo el mundo. Sun Microsystems fundó el proyecto de código abierto NetBeans en junio 2000 y continúa siendo el patrocinador principal de los proyectos.

Esta plataforma es una base modular y extensible usada como una estructura de integración para crear aplicaciones de escritorio grandes. Empresas independientes asociadas, especializadas en desarrollo de software, proporcionan extensiones adicionales que se integran fácilmente en la plataforma y que pueden también utilizarse para desarrollar sus propias herramientas y soluciones. [27]

La plataforma ofrece servicios comunes a las aplicaciones de escritorio, permitiéndole al desarrollador enfocarse en la lógica específica de su aplicación. Entre las características de la plataforma están:

- Administración de las interfaces de usuario (ej. menús y barras de herramientas)
- Administración de las configuraciones del usuario
- Administración del almacenamiento (guardando y cargando cualquier tipo de dato)
- Administración de ventanas
- Framework basado en asistentes (diálogo paso a paso).

Eclipse

Eclipse es una plataforma de software de código abierto independiente de la plataforma para desarrollar lo que el proyecto llama "Aplicaciones de Cliente Enriquecido", opuesto a las aplicaciones "Cliente-liviano" basadas en navegadores.

Este mecanismo de módulos es una plataforma ligera para componentes de software. Adicionalmente al permitirle a Eclipse extenderse usando otros lenguajes de programación como son C/C++ y Python, permite a Eclipse trabajar con lenguajes para procesado de texto como LaTeX, aplicaciones en red como Telnet y Sistema de gestión de base de datos. La arquitectura plugin permite escribir cualquier extensión deseada en el ambiente, como sería Gestión de la configuración. Esta herramienta provee soporte para Java y CVS en el SDK de Eclipse.

El IDE Eclipse cuenta entre sus principales potencialidades con:

- Editor de texto.

- Resaltado de sintaxis.
- Compilación en tiempo real.
- Pruebas unitarias con JUnit.
- Control de versiones con CVS
- Asistentes (wizards): para creación de proyectos, clases, tests, entre otros.
- Refactorización.

Asimismo, a través de plugins libremente disponibles es posible añadir:

- Control de versiones con Subversion.
- Integración con Hibernate. [28]

¿Por qué NetBeans?

Aunque son dos herramientas equilibradas, se determinó utilizar NetBeans porque se puede ejecutar con menos recursos computacionales sobre todo si el software en desarrollo presenta gran cantidad de componentes visuales como es el caso del presente trabajo.

Conclusiones

En este capítulo se definieron conceptos importantes relacionados con el objeto de estudio, se describieron algunas herramientas que simulan el SI y se presentaron las principales características de las metodologías y herramientas. Se decidió utilizar EDE y AC debido a su gran vinculación con los procesos dinámicos y complejos como los que tienen lugar en el SI. Finalmente se determinó aplicar la metodología de desarrollo OpenUP; usar el lenguaje de modelado UML sobre la herramienta CASE Visual Paradigm; lenguaje de programación Java y NetBeans como entorno de desarrollo.

Capítulo 2
Características del Sistema

Introducción

En este capítulo se propone una solución que da respuesta al problema planteado. En una primera parte se explica el modelo sobre el cual se basa el sistema a construir. Luego se plantean los requerimientos funcionales y no funcionales, así como el modelo de casos de uso del sistema y se realiza la descripción de los mismos.

2.1 Modelo BitString

Dentro del modelo general de simulación se usa el modelo de BitString. El mismo consiste en codificar las proteínas o receptores con una cadena binaria de forma que constituyan los posibles enlaces por el cual interactúan dos entes biológicos. Este modelo consta de dos parámetros fundamentales, en primer lugar el NBitString que significa la longitud de la cadena, en el caso de la Figura 2.1 es 16. En segundo lugar minmatch que consiste en la mínima cantidad de bits que deben machear[1] entre dos cadenas para que ocurra un proceso determinado.

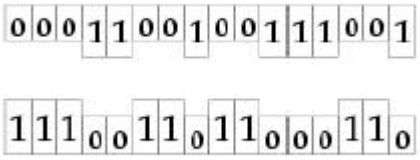


Figura 2.1 Representación gráfica del modelo BitString

2.2 Compartimentos del modelo.

En el capítulo anterior se mencionaron algunas herramientas que simulan el SI. Las mismas (IMSIMM, C-IMMSIM, IMMSIM++) siguen el modelo descrito por Seiden y Celada; todos con la limitante de tener entidades pre-definidas. En este capítulo se propone un modelo basado en el de Seiden y Celada [3] ero con algunas características propias que posibiliten la flexibilidad del sistema.

[1] Dos bits machean si tienen valores opuestos.

El espacio de simulación se compone por celdas (Nodos Linfáticos) y entidades. Como entidades se entienden aquellos componentes del SI que intervienen en la respuesta inmunitaria hacia agentes extraños que atacan al organismo, es decir células y moléculas que intervienen en los procesos que permiten al sistema cambiar de estados y ejecutar acciones.

Los tres órganos fundamentales que componen el Sistema Inmune son los Nodos Linfáticos, el Timo y la Médula Ósea (Figura 2.2). A continuación se describen detalladamente el rol que van a desempeñar en el modelo J-ImmSim.

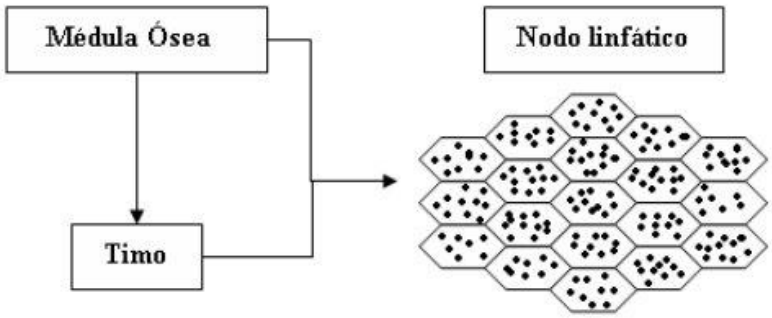


Figura 2.2 Órganos fundamentales del Sistema Inmune

Médula Ósea.

Es el órgano donde se generan todas las células en un estado determinado, que intervendrán posteriormente en los procesos e interacciones que se llevan a cabo en el sistema. Algunas de ellas maduran en la propia médula y otras lo hacen en el timo. Este proceso de nacimiento en la médula se rige por la siguiente EDE: $\mathrm{dx}(t) = n(\bar{x} - x(t)) + \sigma d\varepsilon(t),\ x(0) = 0$

En nuestro caso significa: $\frac{dx_i(t)}{dt} = \frac{\ln 2}{\tau_i}\left(x_i(0) - x_i(t)\right) + \frac{d\varepsilon(t)}{dt}$

Este proceso es importante en la regulación, ya que garantiza que cuando el número de células es mayor que el valor inicial el promedio de muerte aumenta y $\frac{dx_i}{dt} < 0$, por el contrario, si el número de células es menor que el valor inicial, el promedio de nacimiento aumenta y $\frac{dx_i}{dt} > 0$

Timo.

Es el órgano donde las células que no maduraron en la Médula Ósea entran en un proceso de maduración formado por una selección positiva y una negativa.

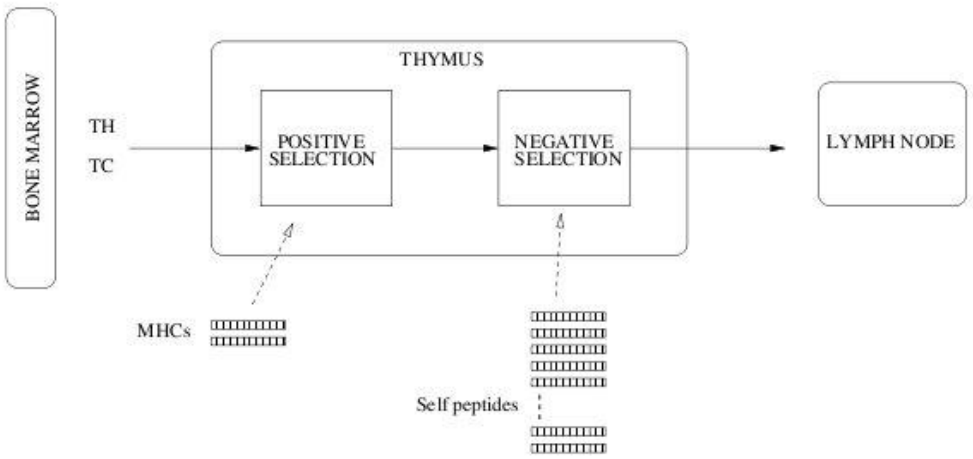


Figura 2.3Proceso de selección positiva y negativa llevado a cabo en el Timo

La primera consiste en que cada célula tiene que reaccionar con un mínimo de afinidad con alguna de las demás células. Esto garantiza que la célula sea útil, es decir, que posteriormente pueda interactuar con otra entidad.

La selección negativa consiste en que las células no deben reaccionar con lo propio del organismo, o sea, deben ser tolerantes, de lo contrario provocaría enfermedades autoinmunes en los organismos. Supongamos una célula X con un receptor TCR, la probabilidad de que esta célula pase la selección negativa está dada por la siguiente fórmula, donde ThymEff es un parámetro que significa la eficiencia del Timo.

$$Pr = \left[\prod_{j,k}(1 - A_{jk} \times B_{jk})(1 - C_{jk} \times D_{jk})\right]^{ThymEff}$$

Donde:

$$A_{jk} = Affinity\left(TCR, left\left(MHC_j\right) + right(peptide_k)\right)$$

$$B_{jk} = match\left(MHC_j, peptide_k\right)^{PresentStr}$$

$$C_{jk} = Affinity\left(TCR, left\left(MHC_j\right) + left(peptide_k)\right)$$

$$D_{jk} = match\left(MHC_j, peptide_k\right)^{PresentStr}$$

Affinity: es la afinidad entre dos cadenas binarias.
left: es la parte izquierda de la cadena binaria.
right: es la parte derecha de la cadena binaria.
match: es la cantidad de bit que machean entre dos cadenas binarias.
PresentStr: Parámetro introducido por el usuario.
MHC y peptide: son moléculas entradas por el usuario, representan lo propio del organismo. [29]

Nodo Linfático.

Constituye el espacio de simulación. Dentro de él encontraremos un conjunto de células y moléculas que participarán en los procesos e interacciones del sistema que tienen lugar dentro del mismo.

El espacio de simulación funcionará como un autómata celular probabilístico (ACP) teniendo lugar un proceso muy importante conocido como Difusión que tiene como objetivo mantener en equilibrio las concentraciones de células y moléculas en las celdas.

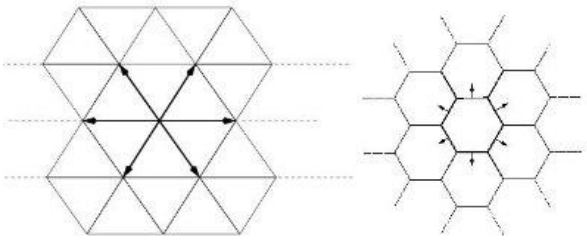


Figura 2.4 Vecindad del autómata utilizado en el proceso de Difusión.

La probabilidad de difundir hacia un vecino X, escogido aleatoriamente, para cada célula en cada paso de tiempo es $1 - dx$, donde $dx = \begin{cases} N_x/\bar{M} \; si \leq 1 \\ 1 \quad Otro \; caso \end{cases}$

Donde Mes el número máximo de células en el volumen representado por cada celda y Nx es el número total de células (de todos los tipos) en cada celda. [29]

2.3 Entidades del Sistema

Las entidades del sistema representan la unidad básica de información y comportamiento. En las herramientas estudiadas que simulan el SI, la forma en que son tratadas las entidades (atributos y valores rígidos de los mismos), constituyen la principal deficiencia en cuanto a flexibilidad, ya que imposibilita a los investigadores realizar cambios en la estructura y función de una determinada entidad. El modelo a proponer concibe células y moléculas. Las células tienen un grupo de características bien definidas entre los que sobresalen los receptores que poseen. Las moléculas por su parte se dividen en anticuerpos, antígenos (virus o bacterias) y moléculas simples (citoquinas e inmunoglobulinas).

Célula

Una célula contiene un conjunto de características que la distinguen como son:

- Edad.
- Promedio de Vida.
- Órgano de Maduración.
- Tiempo de Duplicación.
- Receptores (De dos tipos: genéricos[1] y no genéricos).

Moléculas

Las moléculas complejas (antígenos y anticuerpos) son manejadas en el sistema como objetos con sus respectivas características y otras simples que son tratadas por cantidades.

[1] Receptor que no importa su valor, solo su existencia

Antígenos

Todos los agentes extraños que penetran en el organismo y pueden provocar daños al mismo, dígase virus, bacterias, parásitos, entre otros. Presentan un grupo de características que los diferencian de las demás moléculas

- Peptides (Cadenas binarias Internas).
- Epitopes(Cadenas binarias Externas)
- probabilidad de machear con MHCI[1]
- probabilidad de machear con MHCII [2]

Anticuerpos

Constituyen biomoléculas que se pueden encontrar solubles en los líquidos del cuerpo (sangre, linfa) o adheridos a células para contribuir en las defensas del organismo contra antígenos. Presenta un receptor como característica principal.

Moléculas Simples (Inmunoglobulinas y Citoquinas)

Estas moléculas regulan algunos procesos e interacciones. Por tanto, lo que interesa es saber su concentración, no la composición que puedan presentar (Composición química o eléctrica). Pueden inhibir o estimular cierto proceso estocásticamente, es decir, no se puede afirmar categóricamente que al existir una determinada concentración de una molécula X siempre tendremos el mismo efecto. Se define un umbral (H); con este y la concentración real (Co) que existe en el nodo calcularemos la probabilidad de inhibir o estimular (según el caso) de la siguiente manera:

Pa = Co/H

Si Pa >= 1 Inhibe o Estimula

Si 0 <=Pa <1 se compara con un número aleatorio.

[1] Complejo de Histocompatibilidad de Clase I
[2] Complejo de Histocompatibilidad de Clase II

2.4 Procesos

Se identifican como procesos todas aquellas acciones que hacen las células o se realizan sobre ellas con un objetivo específico. Están estrechamente relacionados con los receptores pues para que una célula entre en un proceso debe tener un receptor asociado al mismo.

Entre los principales se definen:

- Ingestión de Antígenos.
- Secreción de Anticuerpos.
- Producción de Células.
- Maduración de Células.
- División Clonal.
- Muerte de Células.
- Degradación de Moléculas Simples.

Interacciones

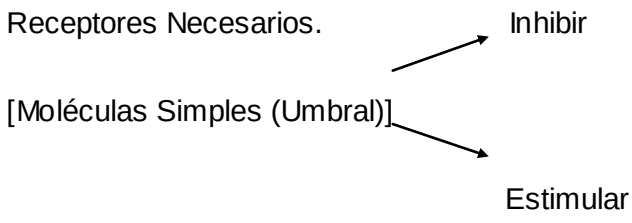
Ocurren localmente, es decir, dentro de un nodo linfático; las entidades de un nodo X no interactúan con las entidades de un nodo Y. Pueden ser de tres tipos principales: interacciones Célula-Célula, Célula-Antígeno e interacciones Antígeno-Anticuerpo.

Se ha definido como patrón de interacción aquellas acciones a realizarse cuando se encuentran dos entes biológicos bajo ciertas condiciones definidas por el biólogo.

Patrones de Interacción

Célula/Célula

Condiciones: Células Involucradas.



Acciones Célula:

Actualizar Receptores (Diferenciarse)
Morir
Duplicarse
Secretar Anticuerpos
Formar Complejo Celular
Convertirse en Célula de Memoria

Acciones Molécula:

Secretar, Consumir y Degradar

Anticuerpo/Antígeno

Condición: Anticuerpo involucrado.
Antígeno involucrado.

Acciones: Formar InmunoComplejo.
Eliminar Antígeno y Anticuerpo.

Célula/Antígeno

Condición: Célula involucrada.
Receptor involucrado.
Antígeno involucrado.

Acciones: Actualizar Receptores (Diferenciarse)
Morir
Duplicarse
Secretar Anticuerpos
Convertirse en Célula de Memoria
Eliminar o no el Antígeno

2.5 Requerimientos del software

Los requerimientos del software constituyen las capacidades o condiciones que el sistema debe cumplir o alcanzar [41]. A continuación se exponen los requisitos funcionales y no funcionales.

Requisitos Funcionales

R.1 Adicionar Célula

R.2 Adicionar Molécula Simple

R.3 Adicionar Antígeno

R.4 Adicionar Anticuerpo

R.5 Adicionar Complejo Celular.

R.6 Adicionar InmunoComplejo.

R.7 Eliminar Ente Biológico.

- R.7.1 Eliminar Célula.
- R.7.2 Eliminar Molécula Simple.
- R.7.3 Eliminar Antígeno
- R.7.4 Eliminar Anticuerpo.
- R.7.5 Eliminar Complejo Celular.
- R.7.6 Eliminar InmunoComplejo.

R.8 Adicionar Interacción Célula-Célula

R.9 Adicionar Interacción Célula-Antígeno

R.10 Adicionar Interacción Antígeno-Anticuerpo

R.11 Eliminar Interacción

- R.11.1 Eliminar interacción Célula-Célula.
- R.11.2 Eliminar interacción Célula-Antígeno.
- R.11.3 Eliminar interacción Antígeno-Anticuerpo.

R.12 Gestionar Tratamiento

- R.12.1 Adicionar Tratamiento.
- R.12.2 Eliminar Tratamiento.

R.13 Gestionar Modelo

- R.13.1 Crear Modelo.
- R.13.2 Importar Modelo.
- R.13.3 Exportar Modelo.

R.14 Gestionar Etiqueta

R.14.1 Definir Etiqueta.

R.14.2 Eliminar Etiqueta.

R.15 Gestionar Receptor

R.15.1 Adicionar Receptor.

R.15.2 Eliminar Receptor.

R.16 Editar Preferencias del simulador.

R.17 Simular el Sistema Inmune

Requisitos No Funcionales

De Software

Se debe disponer de la máquina virtual de Java versión 1.5 o superior, como sistema operativo Linux y como Asistente Matemático Matlab versión 7.0 o superior.

De Hardware

Se requiere disponer de una computadora personal con al menos 512 MB de memoria RAM y procesador de 3.0 GHz.

Restricciones en el diseño y la implementación

Se debe usar Java como lenguaje de programación. Como entorno integrado de desarrollo (IDE por sus siglas en inglés) se utilizará Netbeans 5.5; UML como lenguaje de modelado y como herramienta CASE para el modelado de los artefactos Visual Paradigm.

Políticos y Culturales

La aplicación utilizará el español como idioma. Algún cambio que se quiera hacer, será a través de la dirección del proyecto que desarrolla esta investigación.

Apariencia o interfaz externa.

La interfaz de manera general debe ser sencilla permitiendo al usuario ejecutar opciones del menú a través de combinaciones de teclas.

2.6 Diagrama de Casos de Uso del Sistema

Los diagramas de casos de uso describen las relaciones y las dependencias entre un grupo de casos de uso y los actores participantes en el proceso. Los diagramas de casos de uso sirven para facilitar la comunicación con los futuros usuarios del sistema, y con el cliente, y resultan especialmente útiles para determinar las características necesarias que tendrá el sistema. En otras palabras, los diagramas de casos de uso describen qué es lo que debe hacer el sistema, pero no cómo [30]. A continuación se muestra lo que debe hacer J-ImmSim

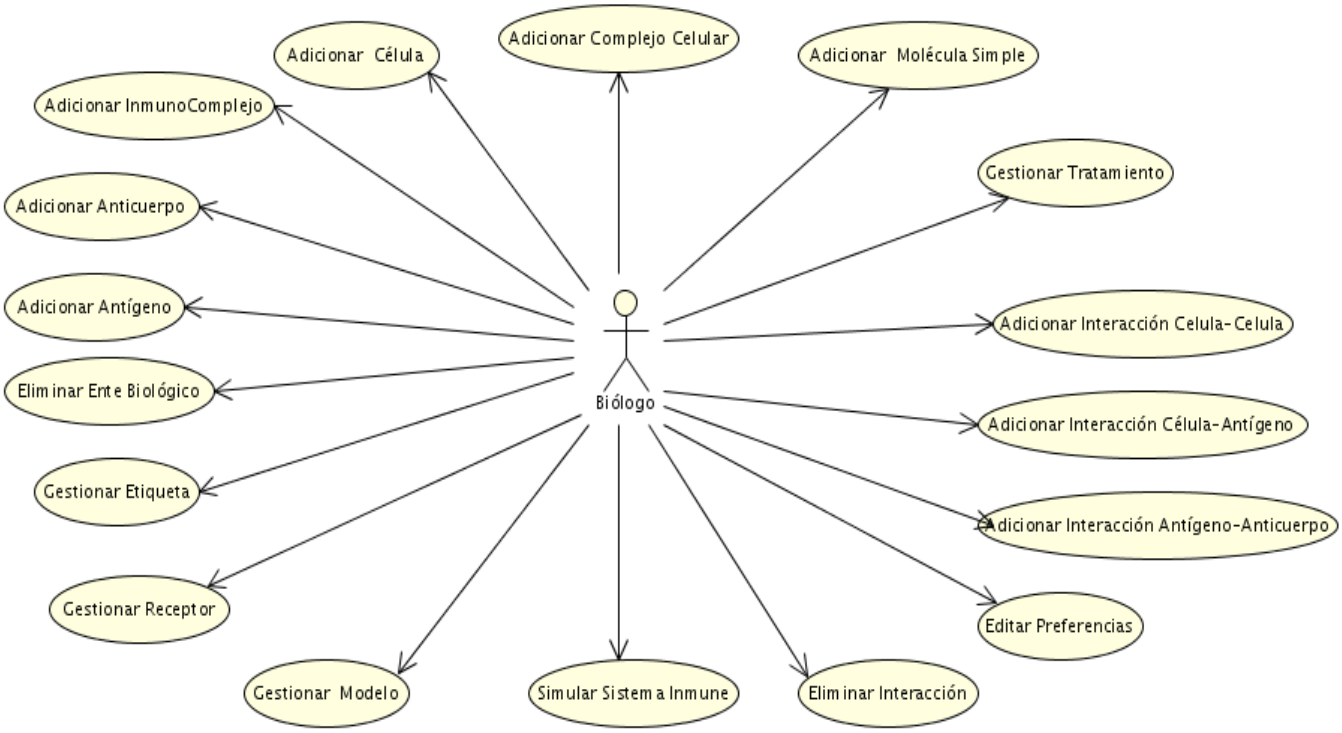


Figura 2.5 Diagrama de Casos de Uso del Sistema

2.7 Descripción Textual de los Casos de Uso del Sistema

Las descripciones de casos de uso son reseñas textuales del caso de uso. Normalmente tienen el formato de una nota o un documento relacionado de alguna manera con el caso de uso, y explica los procesos o actividades que tienen lugar en el caso de uso [30]. A continuación se representa la descripción textual del caso de uso Simular Sistema Inmune. Las demás descripciones textuales aparecen resumidas pero pueden ser vistas en su totalidad en el Expediente de Proyecto.

Caso de Uso	Simular Sistema Inmune
Actores	Biólogo (Inicia el CU)
Propósito	Simular las reacciones del sistema inmune bajo determinadas condiciones.
Descripción	El caso de uso inicia cuando el biólogo desea simular una respuesta inmunitaria y selecciona la opción simular e introduce las cantidades de células, moléculas y demás entes biológicos y el tiempo de simulación. El caso de uso finaliza cuando el sistema luego de haber realizado los cálculos muestra una gráfica de población con respecto al tiempo y guarda en fichero los resultados de las simulaciones.
Prioridad:	Crítico
Prototipo Interfaz:	Anexo 1.1
Precondiciones:	Debe existir un modelo creado.
Referencias:	R.17

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1.) El caso de uso se inicia cuando el biólogo selecciona la opción Simular en el menú Simular.	2.) El sistema muestra la interfaz para introducir los datos de la simulación.
3.) El biólogo introduce los datos e indica Aceptar.	4.) El sistema valida los datos entrados por el biólogo, realiza los cálculos pertinentes y muestra una gráfica de poblaciones con respecto al tiempo.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
3.1) El biólogo indica la opción Cerrar	3.2) El sistema muestra la interfaz principal
	4.1) Si hay errores muestra un mensaje de notificación.

Poscondiciones	Queda la información de la simulación guardada en un fichero.

Caso de Uso	Gestionar Modelo
Actores	Biólogo (Inicia el CU)
Propósito	Crear, exportar e importar el modelo con las entidades introducidas por el biólogo.
Descripción	El caso de uso se inicia cuando el biólogo selecciona la opción de crear, importar o exportar un modelo biológico. El caso de uso finaliza cuando se incorpora un modelo al sistema (creado o importado) o cuando el existente es guardado en una dirección introducida por el biólogo.
Prioridad:	Crítico
Prototipo Interfaz:	Anexo 1.2

Caso de Uso	Adicionar Célula
Actores	Biólogo (Inicia el CU)
Propósito	Crear célula.
Descripción	El caso de uso inicia cuando el biólogo selecciona la opción de crear una nueva célula del menú Entidades e introduce los datos correspondientes a dicha célula. El caso de uso finaliza cuando el sistema luego de haber validado los datos introducidos los guarda en un fichero.
Prioridad:	Secundario
Prototipo Interfaz:	Anexo 1.3

Caso de Uso	Adicionar Complejo Celular
Actores	Biólogo (Inicia el CU)
Propósito	Crear Complejo Celular.
Descripción	El caso de uso inicia cuando el biólogo selecciona la opción de crear un nuevo Complejo Celular del menú Entidades e introduce los datos correspondientes a dicho complejo. El caso de uso finaliza cuando el sistema luego de haber validado los datos introducidos por el biólogo los guarda en un fichero.
Prioridad:	Secundario
Prototipo Interfaz:	Anexo 1.4

Caso de Uso	Adicionar InmunoComplejo
Actores	Biólogo (Inicia el CU)
Propósito	Crear InmunoComplejo.
Descripción	El caso de uso inicia cuando el biólogo selecciona la opción de crear un nuevo InmunoComplejo del menú Entidades e introduce los datos correspondientes a dicho ente biológico. El caso de uso finaliza cuando el sistema luego de haber validado los datos introducidos por el biólogo los guarda en un fichero.
Prioridad:	Secundario
Prototipo Interfaz:	Anexo 1.5

Caso de Uso	Adicionar Molécula Simple
Actores	Biólogo (Inicia el CU)
Propósito	Crear Molécula Simple.
Descripción	El caso de uso inicia cuando el biólogo selecciona la opción de crear una nueva molécula simple del menú Entidades e introduce los datos correspondientes a dicha molécula. El caso de uso finaliza cuando el sistema luego de haber validado los datos introducidos por el biólogo los guarda en un fichero.
Prioridad:	Secundario
Prototipo Interfaz:	Anexo 1.6

Caso de Uso	Adicionar Antígeno
Actores	Biólogo (Inicia el CU)
Propósito	Crear un antígeno.
Descripción	El caso de uso inicia cuando el biólogo selecciona la opción de crear un nuevo antígeno del menú Entidades e introduce los datos correspondientes a dicho antígeno. El caso de uso finaliza cuando el sistema luego de haber validado los datos introducidos por el biólogo los guarda en un fichero.
Prioridad:	Secundario
Prototipo Interfaz:	Anexo 1.7

Caso de Uso	Adicionar Anticuerpo
Actores	Biólogo (Inicia el CU)
Propósito	Crear un anticuerpo.
Descripción	El caso de uso inicia cuando el biólogo selecciona la opción de crear un nuevo anticuerpo del menú Entidades e introduce los datos correspondientes a dicho anticuerpo. El caso de uso finaliza cuando el sistema luego de haber validado los datos introducidos por el biólogo los guarda en un fichero.
Prioridad:	Secundario
Prototipo Interfaz:	Anexo 1.8

Caso de Uso	Eliminar Ente Biológico.
Actores	Biólogo (Inicia el CU)
Propósito	Eliminar entes biológicos del sistema (Células, Moléculas, Anticuerpos, Antígenos, Complejos Celulares e InmunoComplejos)
Descripción	El caso de uso inicia cuando el biólogo selecciona la opción de eliminar ente del menú Entidades y selecciona el ente biológico a eliminar. El caso de uso finaliza cuando el sistema luego de haber comprobado la independencia del ente seleccionado, lo borra del fichero.
Prioridad:	Secundario
Prototipo Interfaz:	Anexo 1.9

Caso de Uso	Adicionar Interacción Célula-Célula
Actores	Biólogo (Inicia el CU)
Propósito	Crear una interacción Célula-Célula.
Descripción	El caso de uso se inicia cuando el biólogo selecciona la opción de crear una nueva interacción Célula-Célula. Selecciona las precondiciones (células iniciales y receptores involucrados) y acciones a realizar (Duplicarse, Morir, etc.). El caso de uso finaliza cuando el sistema valida todos los parámetros entrados y guarda en un fichero dicha interacción.
Prioridad:	Secundario
Prototipo Interfaz:	Anexo 1.10

Caso de Uso	Adicionar Interacción Célula-Antígeno
Actores	Biólogo (Inicia el CU)
Propósito	Crear una interacción Célula-Antígeno.
Descripción	El caso de uso se inicia cuando el biólogo selecciona la opción de crear una nueva interacción Célula-Antígeno. Selecciona las precondiciones (célula inicial, su receptor involucrado y el antígeno) y acciones a realizar (Duplicarse, Morir, Secretar Anticuerpos, etc.). El caso de uso finaliza cuando el sistema valida todos los parámetros entrados y guarda en un fichero dicha interacción.
Prioridad:	Secundario
Prototipo Interfaz:	Anexo 1.11

Caso de Uso	Adicionar Interacción Antígeno-Anticuerpo
Actores	Biólogo (Inicia el CU)
Propósito	Crear una interacción Antígeno-Anticuerpo.
Descripción	El caso de uso se inicia cuando el biólogo selecciona la opción de crear una nueva interacción Antígeno-Anticuerpo. Selecciona las precondiciones (antígeno y anticuerpo involucrado) y acciones a realizar (eliminar antígeno o secretar inmunocomplejos). El caso de uso finaliza cuando el sistema valida todos los parámetros entrados y guarda en un fichero dicha interacción.
Prioridad:	Secundario
Prototipo Interfaz:	Anexo 1.12

Caso de Uso	Eliminar Interacción.
Actores	Biólogo (Inicia el CU)
Propósito	Eliminar una interacción.
Descripción	El caso de uso inicia cuando el biólogo selecciona la opción de Eliminar del menú Interacciones y selecciona la interacción a eliminar. El caso de uso finaliza cuando el sistema borra la interacción del fichero.
Prioridad:	Secundario
Prototipo Interfaz:	Anexo 1.13

Caso de Uso	Gestionar Tratamiento
Actores	Biólogo (Inicia el CU)
Propósito	Crear o eliminar Tratamiento.
Descripción	El caso de uso se inicia cuando el biólogo selecciona la opción de adicionar un nuevo tratamiento o eliminarlo. El caso de uso finaliza cuando el sistema luego de validar los datos introducidos por el biólogo actualiza el fichero correspondiente a los tratamientos.
Prioridad:	Secundario
Prototipo Interfaz:	Anexo 1.14

Caso de Uso	Gestionar Etiqueta.
Actores	Biólogo (Inicia el CU).
Propósito	Definir y eliminar etiqueta.
Descripción	El caso de uso se inicia cuando el biólogo selecciona la opción de definir etiqueta o eliminarla del menú Entidades. El biólogo selecciona la célula y el receptor y especifica el nombre de la etiqueta que va a adicionar en el primer caso y selecciona la etiqueta a eliminar en el segundo. El caso de uso finaliza cuando es actualizada la información del fichero correspondiente a etiquetas.
Prioridad:	Secundario
Prototipo Interfaz:	Anexo 1.15

Caso de Uso	Gestionar Receptor.
Actores	Biólogo (Inicia el CU)
Propósito	Adicionar y Eliminar Receptor
Descripción	El caso de uso se inicia cuando el biólogo selecciona la opción de Adicionar o Eliminar Receptor. El caso de uso finaliza cuando el sistema luego de validar los datos introducidos por el biólogo actualiza el fichero correspondiente a los receptores.
Prioridad:	Secundario
Prototipo Interfaz:	Anexo 1.16

Caso de Uso	Editar Preferencias.
Actores	Biólogo (Inicia el CU)
Propósito	Modificar la Preferencias del Sistema
Descripción	El caso de uso se inicia cuando el biólogo selecciona la opción Editar Preferencias del menú Simular. El biólogo modifica la configuración del sistema. El caso de uso finaliza cuando el sistema luego de validar los datos introducidos por el biólogo actualiza el fichero correspondiente a las preferencias.
Prioridad:	Secundario
Prototipo Interfaz:	Anexo 1.17

Conclusiones

En el capítulo que concluye se mostraron las características principales del sistema. Se describió el modelo base a implementar en J-ImmSim el cual es más flexible que los utilizados en los simuladores anteriores. Se presentaron los requisitos funcionales y no funcionales del mismo así como el diagrama de casos de uso del sistema con sus respectivas descripciones textuales.

Capítulo 3
Diseño del Sistema

Introducción

En este capítulo se presenta el diseño del sistema, el estilo arquitectónico y los patrones de diseño por los que se rige J-ImmSim. Se muestran los Diagramas de Clases y de Secuencia de los casos de uso críticos del sistema así como el Modelo de Despliegue.

3.1 Estilos Arquitectónicos y Patrones de Diseño

3.1.1 Estilos Arquitectónicos

¿Qué es la Arquitectura del Software? En un sentido amplio se puede decir que la Arquitectura del Software es el diseño de más alto nivel de la estructura de un sistema, programa o aplicación y tiene la responsabilidad de:

- Definir los módulos principales
- Definir las responsabilidades que tendrá cada uno de estos módulos
- Definir la interacción que existirá entre dichos módulos:
- Control y flujo de datos
- Secuenciación de la información
- Protocolos de interacción y comunicación
- Ubicación en el hardware

La Arquitectura del Software provee una visión abstracta de alto nivel, posponiendo el detalle de cada uno de los módulos definidos a pasos posteriores del diseño. [31]

Arquitectura de J-ImmSim

J-ImmSim es un software atípico. Muchas de sus clases se crean o cambian de especificación en tiempo de ejecución. Un ejemplo aclara lo anterior; cada entidad (célula, antígeno, anticuerpo, entre otras), que se incorpora al modelo representa una clase java que debe ser generada y compilada, para después

poder ser usada mediante el API Reflection (java.lang.reflect) [32]. El biólogo ejecuta el simulador, elige la opción de agregar una nueva entidad, especifica las características de la misma y entonces se genera una nueva clase java que puede ser usada por el propio simulador.

En el proceso descrito anteriormente se identifican dos funcionalidades generales de J-ImmSim. Primero, la edición o construcción de un modelo por el biólogo (Generación de entidades, interacciones y otros). Segundo, la simulación o utilización de ese modelo generado. Estas dos funcionalidades diferentes y relacionadas entre sí exigen que un grupo de paquetes y clases tengan la responsabilidad de generar el modelo, y otro grupo sea responsable de la simulación. En este último se incluyen las clases que han sido generadas en el proceso de construcción del modelo. Esto implica un alto grado de complejidad por lo que se requiere de una gran coordinación. Para ello son necesarias reiteradas consultas y comprobaciones del simulador, incluidos diferentes ficheros de configuración.

En el diseño de sistemas informáticos actual se suele usar las arquitecturas multinivel o programación por capas. En dichas arquitecturas a cada nivel se le confía una misión simple, lo que permite el diseño de arquitecturas escalables

La Capa de Presentación o también denominada Capa de Usuario presenta el sistema al usuario, le comunica la información y captura la información que éste introduce. Esta capa se comunica únicamente con la capa de negocio. También es conocida como interfaz grafica y debe tener la característica de ser entendible y fácil de usar. La Capa de Negocio es donde residen los programas que se ejecutan, se reciben las peticiones del usuario y se envían las respuestas tras el proceso. Se denomina Capa de Negocio o Lógica del Negocio porque es aquí donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, y con la capa de datos para almacenar o recuperar los mismos. La tercera capa es la Capa de Datos, encargada del almacenamiento y acceso a los mismos. Esta capa recibe y responde solicitudes de la Capa de Negocio.

La ventaja principal de este patrón 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. Una Arquitectura en Tres Capas ha sido elegida como la solución más factible para el desarrollo de J-ImmSim (Ver Figura 3.1). [40]

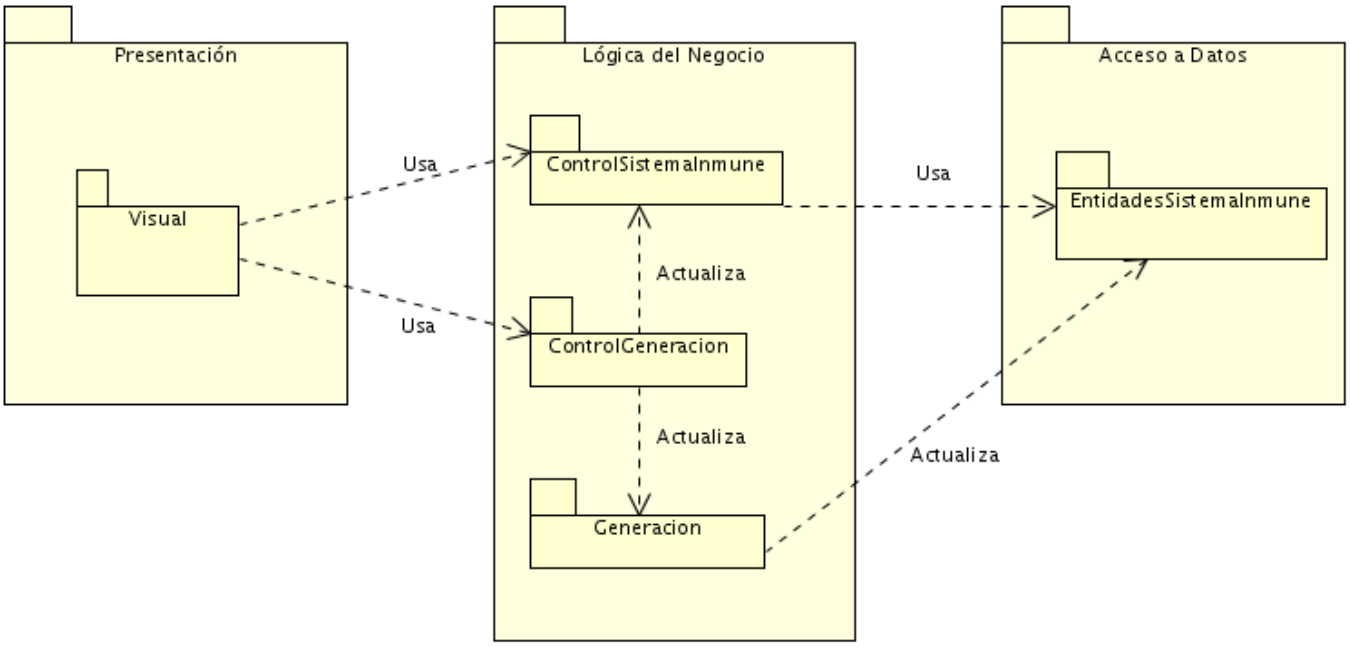


Figura 3.1Patrón Arquitectónico de J-ImmSim

La primera capa incluye todas las interfaces visuales del sistema, tanto para crear el modelo como para simular el mismo, integradas en el paquete Visual. La segunda capa (Lógica del Negocio), incluye tres paquetes: ControlSistemaInmune, Generación y ControlGeneracion. Este último paquete se encarga de coordinar las acciones de las clases que generan nuevas clases (Ej. generación de una célula), y a su vez es responsable de realizar algunos cambios necesarios en las clases que se encuentran en el paquete ControlSistemaInmune, quien se encarga de coordinar o controlar las acciones implicadas en la simulación (Caso de uso Simular). La tercera capa (Acceso a Datos) incluye el paquete EntidadesSistemaInmune conformado por clases que contienen la información correspondiente a las células, moléculas, entre otras. Resumiendo, la capa de presentación (Capa 1) utiliza los paquetes de la capa lógica del negocio (ControlSistemaInmune, Generación y ControlGeneracion). La capa 2 utiliza los paquetes de la Capa 3(EntidadesSistemaInmune) y al mismo tiempo el paquete ControlGeneracion realiza cambios en algunas clases del paquete ControlSistemaInmune. La Capa 3 es de persistencia. El paquete Generacion escribe en EntidadesSistemaInmune y además accede al nivel inferior y último donde se encuentra el paquete de Configuración.

1.2 Patrones de Diseño

GRASP (Patrones para Asignar Responsabilidades)

En la terminología de objetos, el patrón es una descripción de un problema y su solución, que recibe un nombre y que puede emplearse en otros contextos.

A continuación se mencionan los cinco patrones GRASP más utilizados:

- Experto
- Creador
- Alta Cohesión
- Bajo Acoplamiento
- Controlador [33]

Patrones de Diseño en J-ImmSim

A continuación se presentan tres ejemplos de utilización de patrones de diseños en J-ImmSim; Patrón Creador (PCr), Patrón Alta Cohesión (PAC) y Patrón Controlador (PCo). El PCr especifica bajo qué circunstancias una clase A es responsable de crear objetos de una clase B.

Inicialmente, en el simulador J-ImmSim se deben crear las entidades (Células y Moléculas), necesarias para comenzar la simulación (Interacciones y Procesos). ¿Pero quién es responsable de crear las entidades? Estas últimas se encuentran en los Nodos Linfáticos y es en estos donde las mismas interactúan entre sí y sufren un grupo de procesos. Por tanto, podemos decir que la clase NodoLinfatico contiene objetos de las clases Celula, MoleculaSimple, Anticuerpo, entre otras. Es por tanto NodoLinfatico la clase responsable de crear los objetos que contiene.

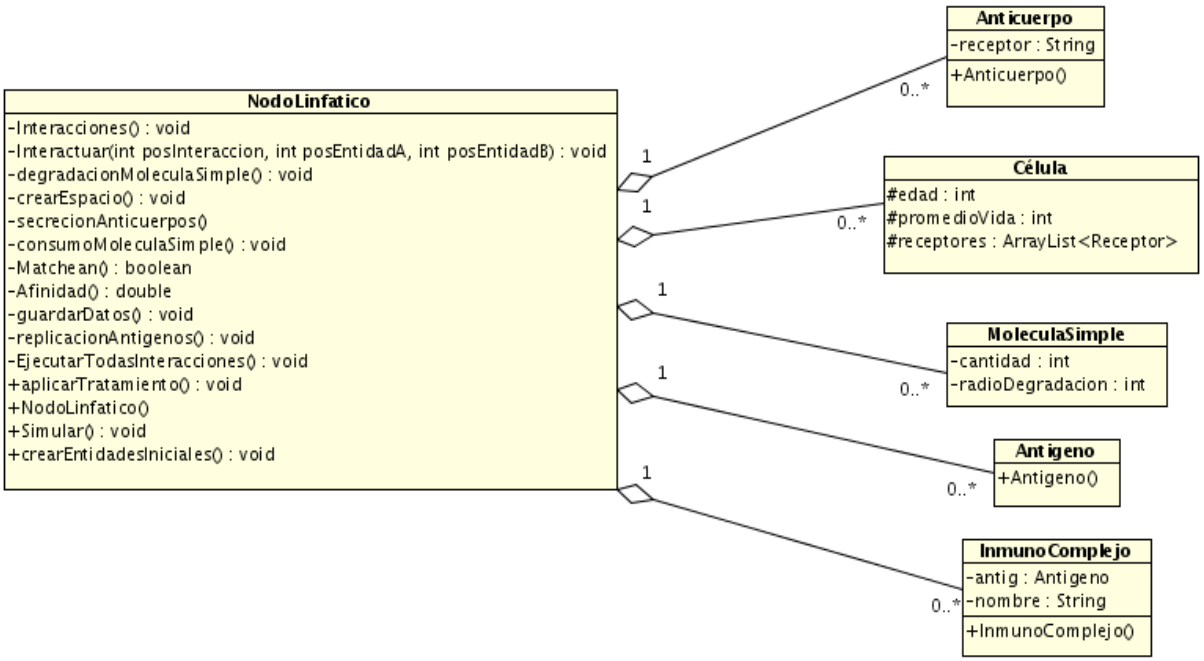


Figura 3.2 Ejemplo Patrón Creador

El PAC describe cuán relacionadas y enfocadas deben estar las responsabilidades de una clase. Como se ha explicado, J-ImmSim tiene la funcionalidad de agregar una nueva célula que implica la generación y compilación de una nueva clase. Pero, ¿qué clase o clases deben realizar esta función? Se ha dado solución a este problema mediante dos clases: CC_Celula y CE_Celula. La primera es una controladora que coordina parte del trabajo: rectifica que el nombre de la célula (Que es también el nombre de la clase que se generará) cumpla con la especificación de la expresión regular "[A-Z] ([a-zA-Z]| [0-9])*" y además compruebe que la célula a adicionar no exista. Después de hacer esto, entonces pasa un mensaje a CE_Celula y esta genera la nueva clase y la compila.

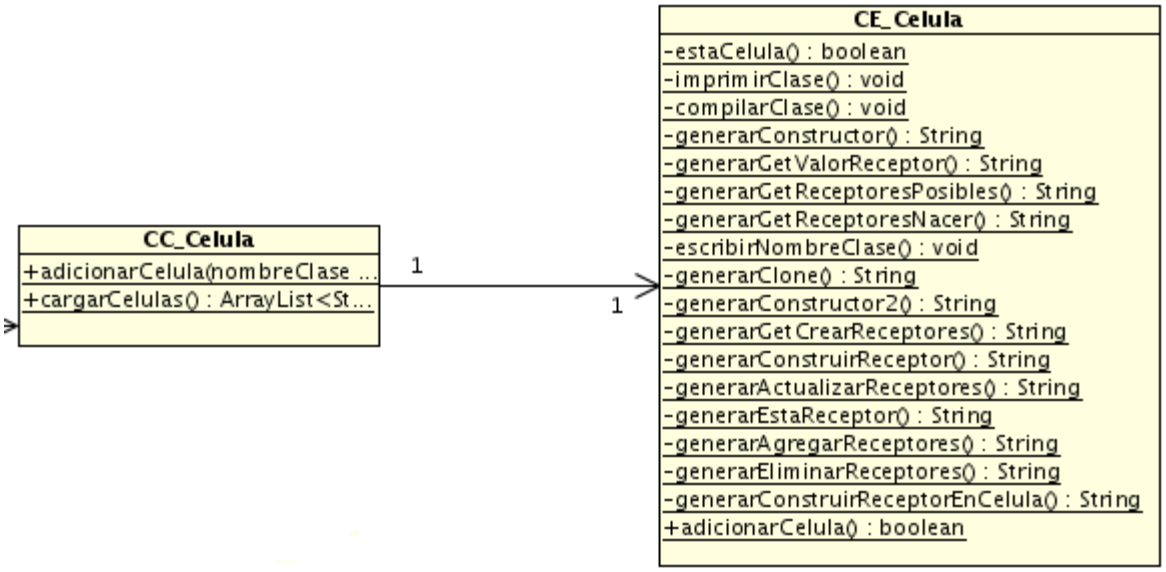


Figura 3.3 Ejemplo Patrón Alta Cohesión

Si queremos realizar un cambio en la especificación del nombre de la célula, bastaría con realizar el cambio pertinente en la clase controladora CC_Celula. En caso de necesitar hacer algún cambio en la nueva clase que se generará, bastaría con modificar la clase CE_Celula. Se garantiza así que las clases no se ocupen de un trabajo excesivo y el diseño se comprenda con más facilidad.

El PCo se encarga de asignar la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase. La clase SistemaInmune contiene los nodos y estos a su vez las entidades. Por tanto, podemos pensar en esta clase como el "sistema global" que se encuentra más cercano a la capa de aplicación; cuando se ejecuta el CU Simular Sistema Inmune debe ser la clase SistemaInmune la que recibe la delegación por parte de la clase visual y coordina las acciones a seguir para dar cumplimiento al caso de uso. Conceptualmente es el Sistema Inmune lo que se quiere simular.

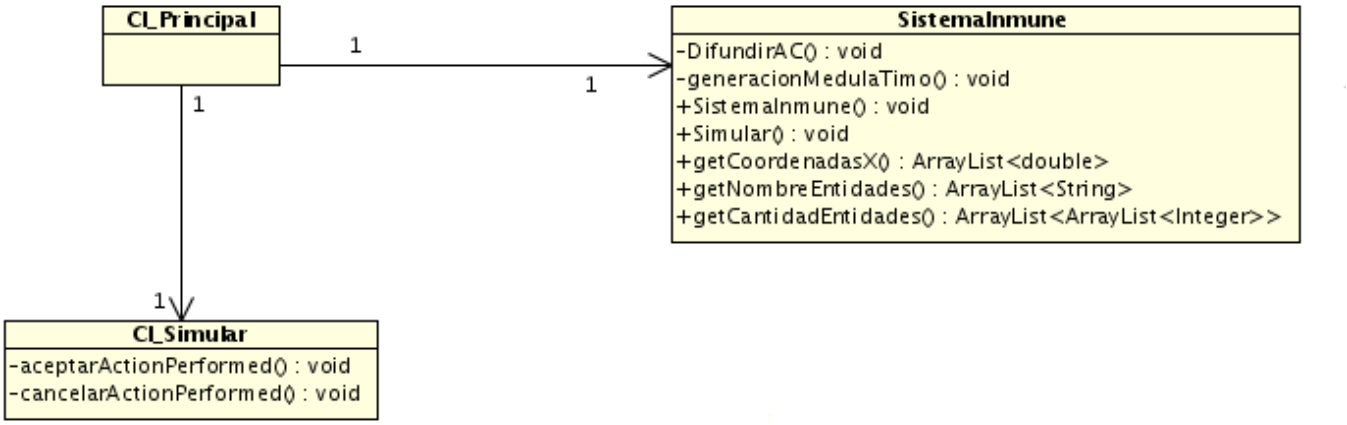


Figura 3.4 Ejemplo Patrón Controlador

3.2 Diagramas de Clases

Los diagramas de clases muestran las diferentes clases que componen un sistema y cómo se relacionan unas con otras. Son diagramas estáticos porque muestran las clases, junto con sus métodos y atributos, así como las relaciones estáticas entre ellas, pero no muestran los métodos mediante los que se invocan entre ellas [30]. A continuación se muestran los diagramas de clases correspondientes a los casos de usos críticos Simular Sistema Inmune y Gestionar Modelo. Además se muestra el diagrama de clases del caso de uso secundario Adicionar Célula, los demás se encuentran en el Expediente de Proyecto.

En el diagrama de clases correspondiente al caso de uso Simular Sistema Inmune (Figura 3.2) la clase CI_Principal recibe el evento del usuario para ejecutar el caso de uso, luego crea una instancia de la clase CI_Simular donde el usuario introduce los datos necesarios. Esta clase crea un objeto de la clase SistemaInmune encargada de controlar todos los procesos que se llevan a cabo en este caso de uso. La misma está relacionada con al menos una instancia de las clase NodoLinfatico donde se crean todas las entidades y se ejecutan las interacciones. Se utiliza la relación de composición entre la clase SistemaInmune y NodoLinfático porque existe una relación fuerte entre las mismas, o sea, no se concibe un Sistema Inmune que no cuente con al menos un Nodo Linfático. Por otra parte las relaciones de agregación existentes entre la clase NodoLinfatico y las entidades Celula, Molecula, Antigeno, etcétera son relaciones débiles debido a que no es obligatorio que exista alguno de estos entes biológicos en el Nodo Linfático.

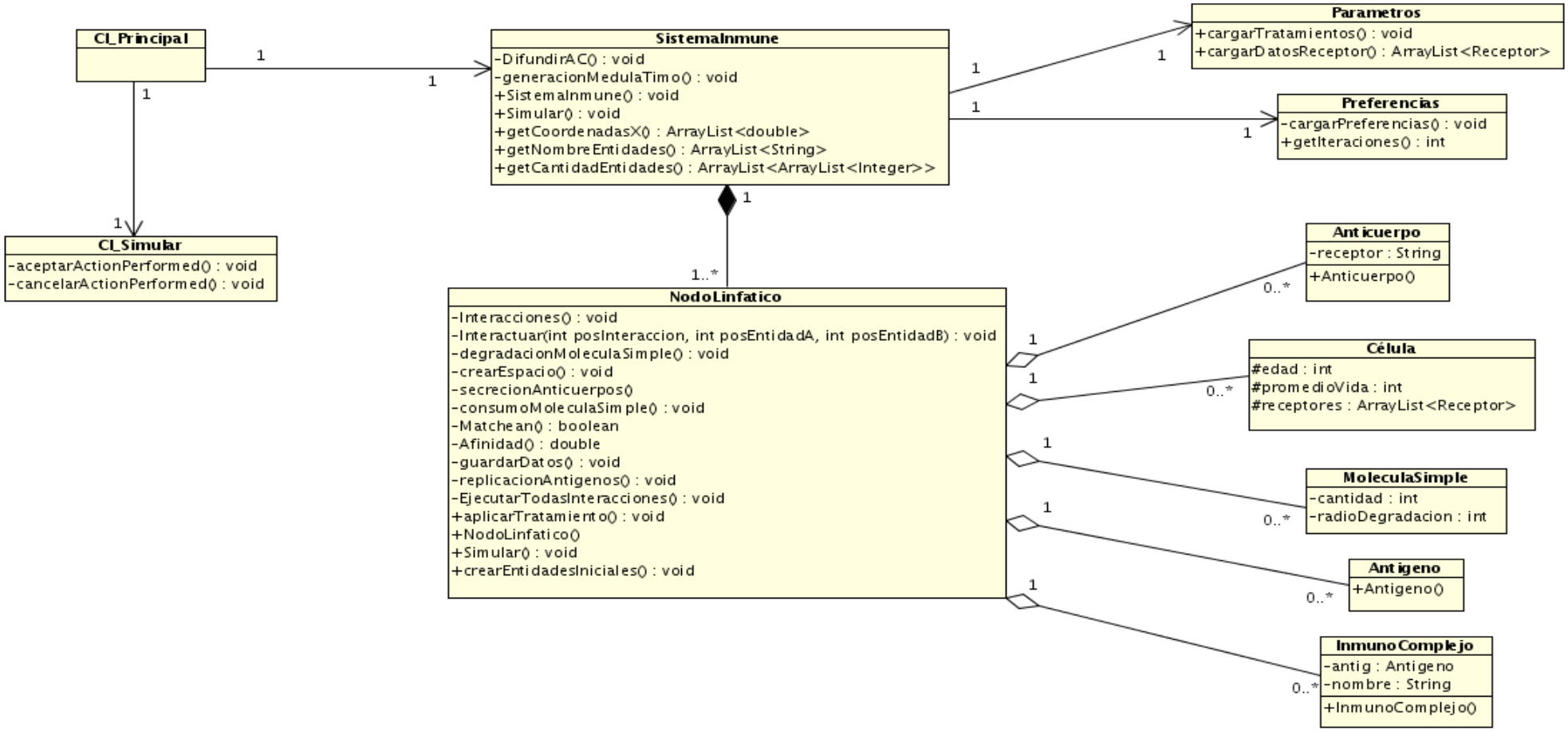


Figura 3.5 Diagrama de Clases del Caso de Uso Simular

El diagrama de clases del caso de uso Gestionar Modelo (Figura 3.3) está compuesto por las clases CI_Principal, CI_NuevoModelo y GestionarModelo. La primera recibe el evento invocado por el usuario, la segunda es creada en caso del escenario NuevoModelo para que el biólogo introduzca el nombre del mismo. Por último la clase GestionarModelo tiene la responsabilidad de controlar y ejecutar los tres escenarios del caso de uso en cuestión.

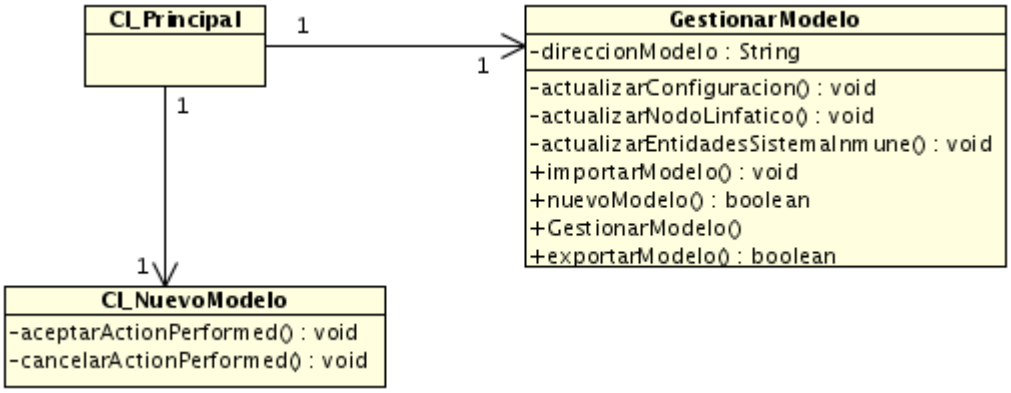


Figura 3.6 Diagrama de Clases del Caso de Uso Gestionar Modelo

El diagrama de clases del caso de uso Adicionar Célula (Figura 3.4). La CI_Principal recibe el evento invocado por el usuario, posteriormente construye una instancia de la CI_AdicionarCelula donde el usuario tiene la posibilidad de agregar un nuevo receptor donde juegan su rol las clases CC_Receptor y CE_Receptor. Luego cuando el usuario ha entrado los datos correspondientes a la célula la CC_Celula controla que los datos sean válidos y que no haya sido creada anteriormente y le comunica a la CE_Celula que conforme la célula, es decir la clase java y compile la misma.

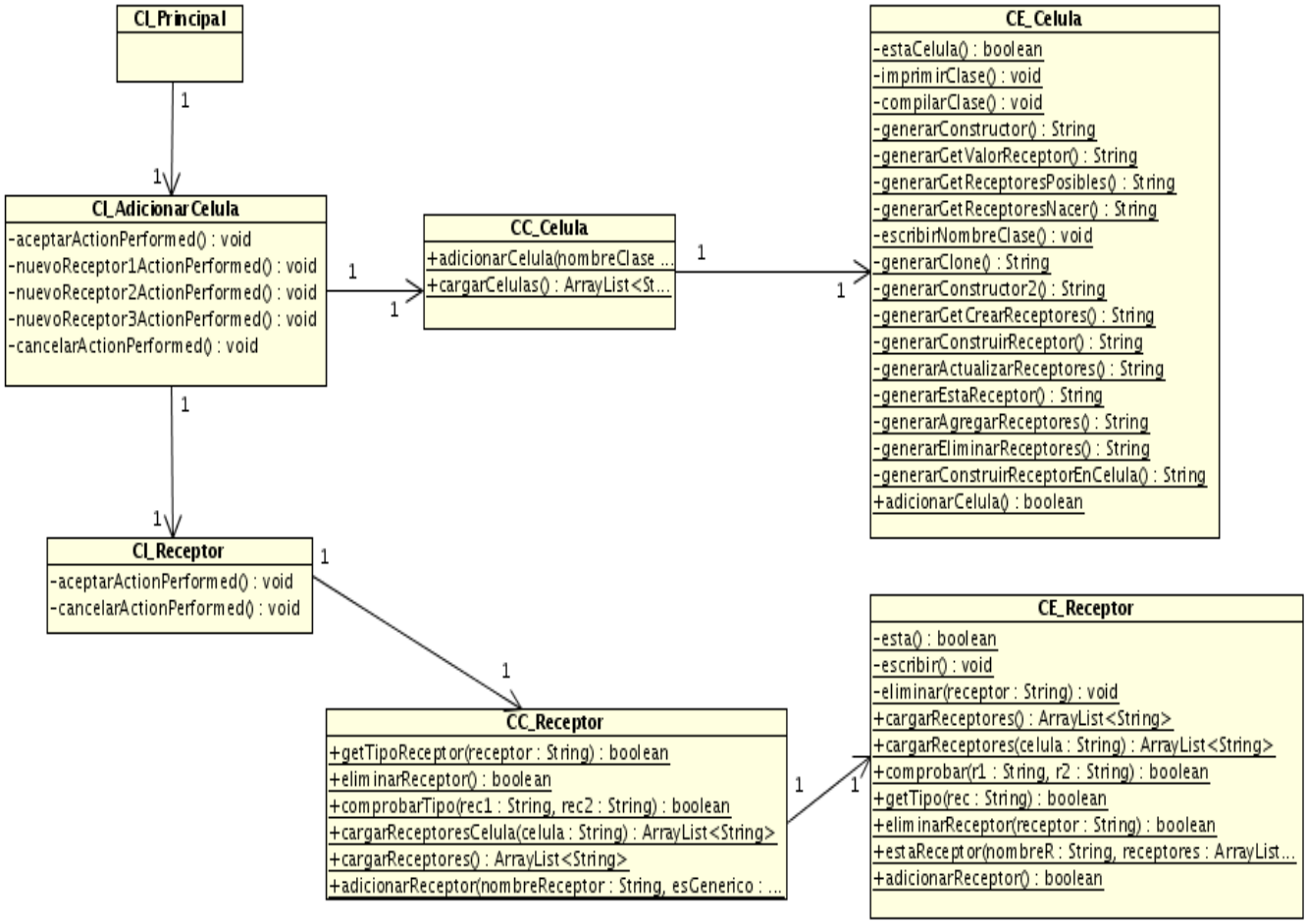
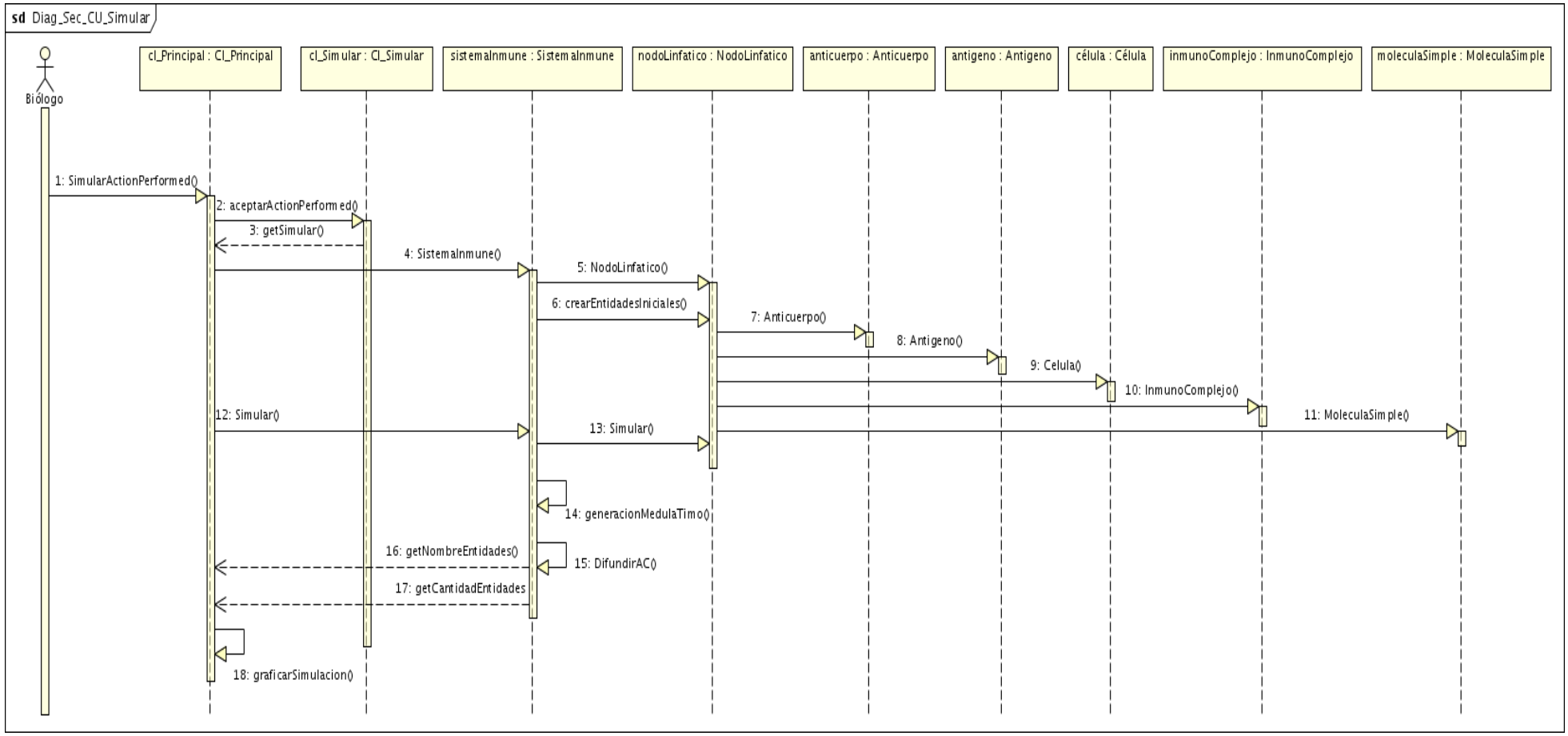


Figura 3.7 Diagrama de Clases del Caso de Uso Adicionar Célula

3.3 Diagramas de Secuencia.

Los diagramas de secuencia son una forma de diagrama de interacción que muestran los objetos como líneas de vida a lo largo de la página y con sus interacciones en el tiempo representadas como mensajes dibujados como flechas desde la línea de vida origen hasta la línea de vida destino. Los diagramas de secuencia son buenos para mostrar qué objetos se comunican con qué otros objetos y qué mensajes disparan esas comunicaciones. Los diagramas de secuencia no están pensados para mostrar lógicas de procedimientos complejos. [34]



3.8 Diagrama de Secuencia del Caso de Uso Simular

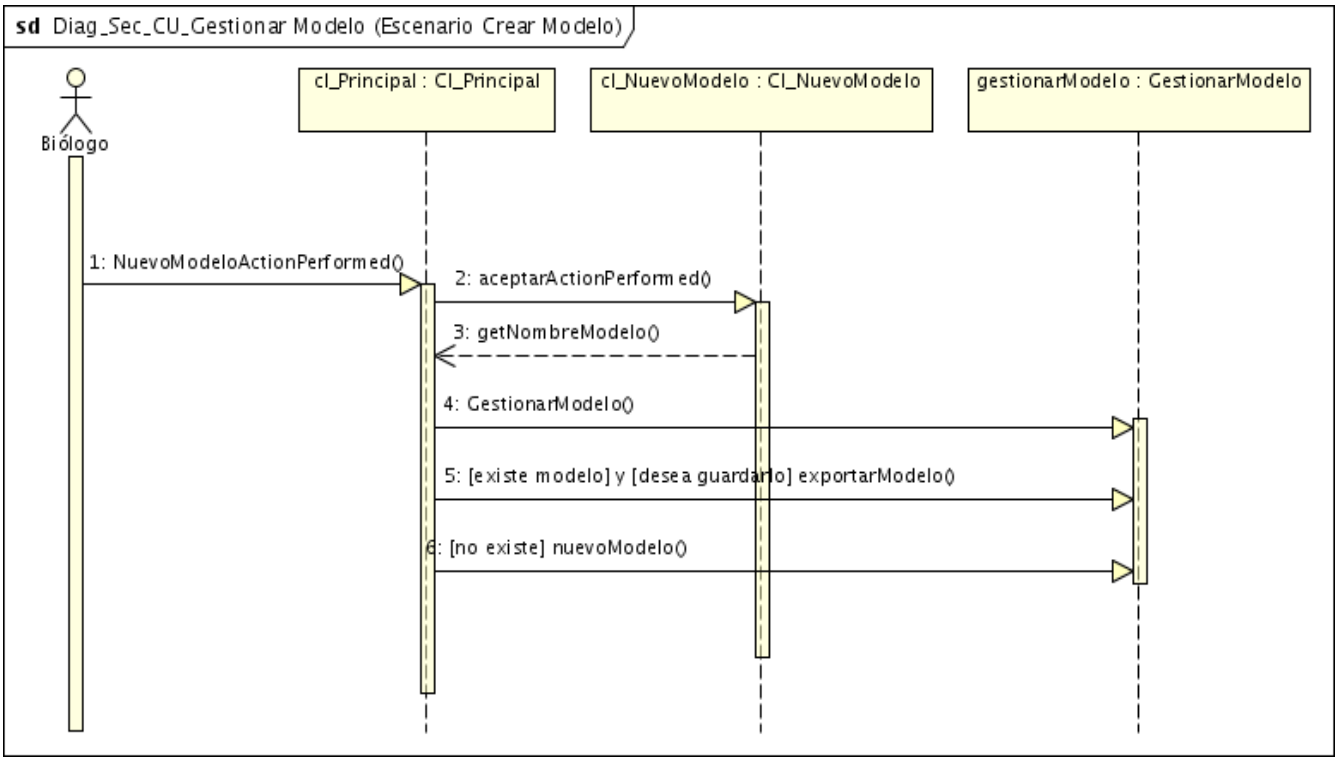


Figura 3.9 Diagrama de Secuencia del Caso de Uso Gestionar Modelo (Escenario Crear Modelo)

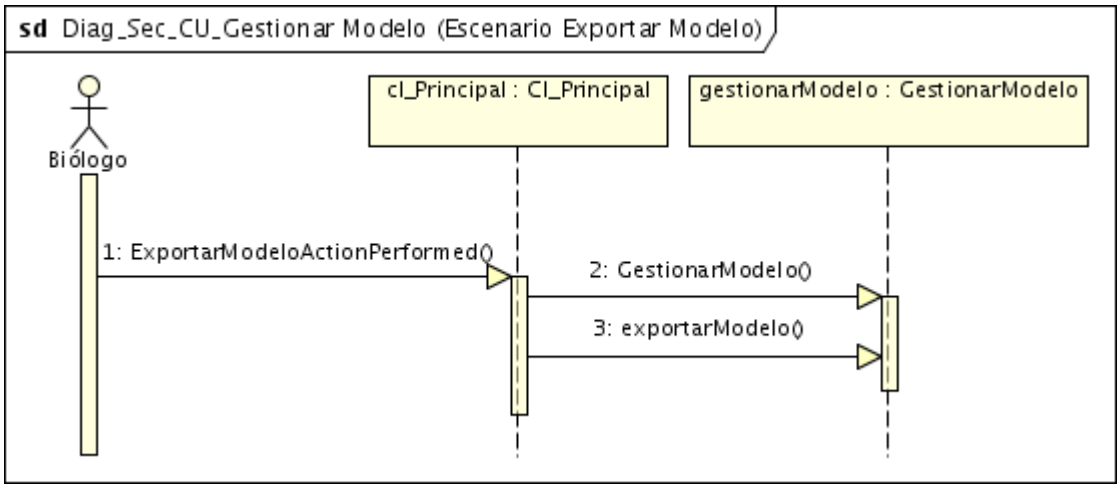


Figura 3.10 Diagrama de Secuencia del Caso de Uso Gestionar Modelo (Escenario Exportar Modelo)

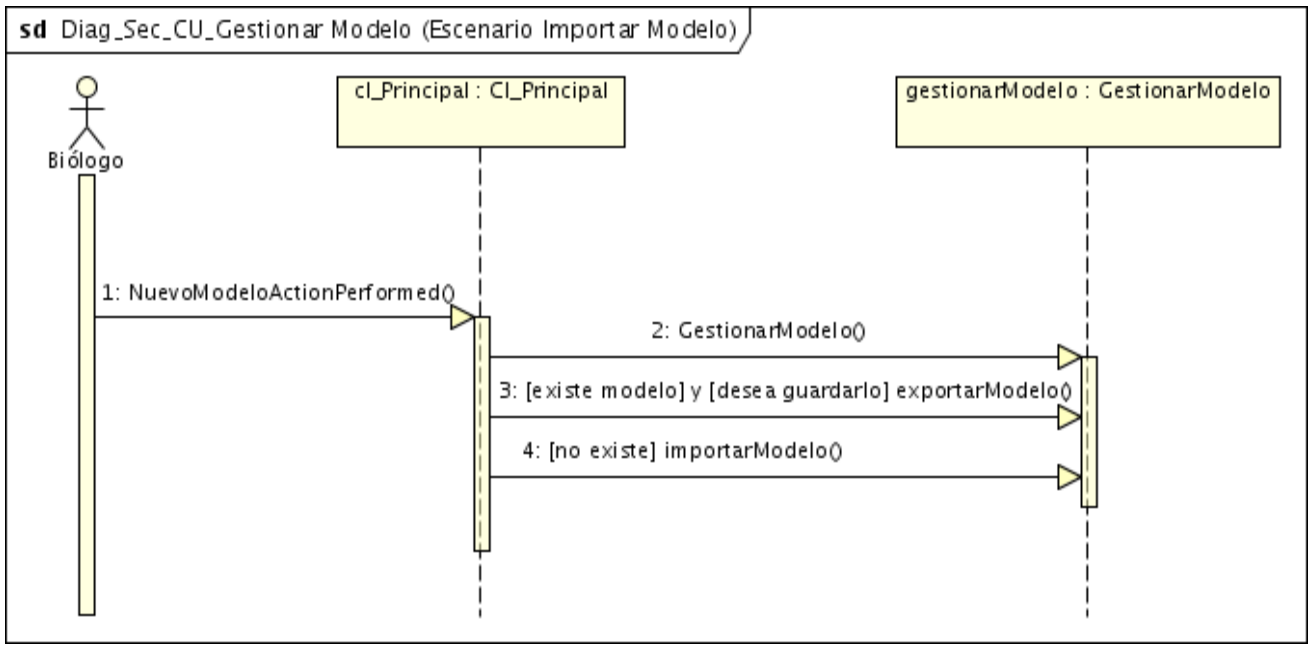


Figura 3.11 Diagrama de Secuencia del Caso de Uso Gestionar Modelo (Escenario Importar Modelo)

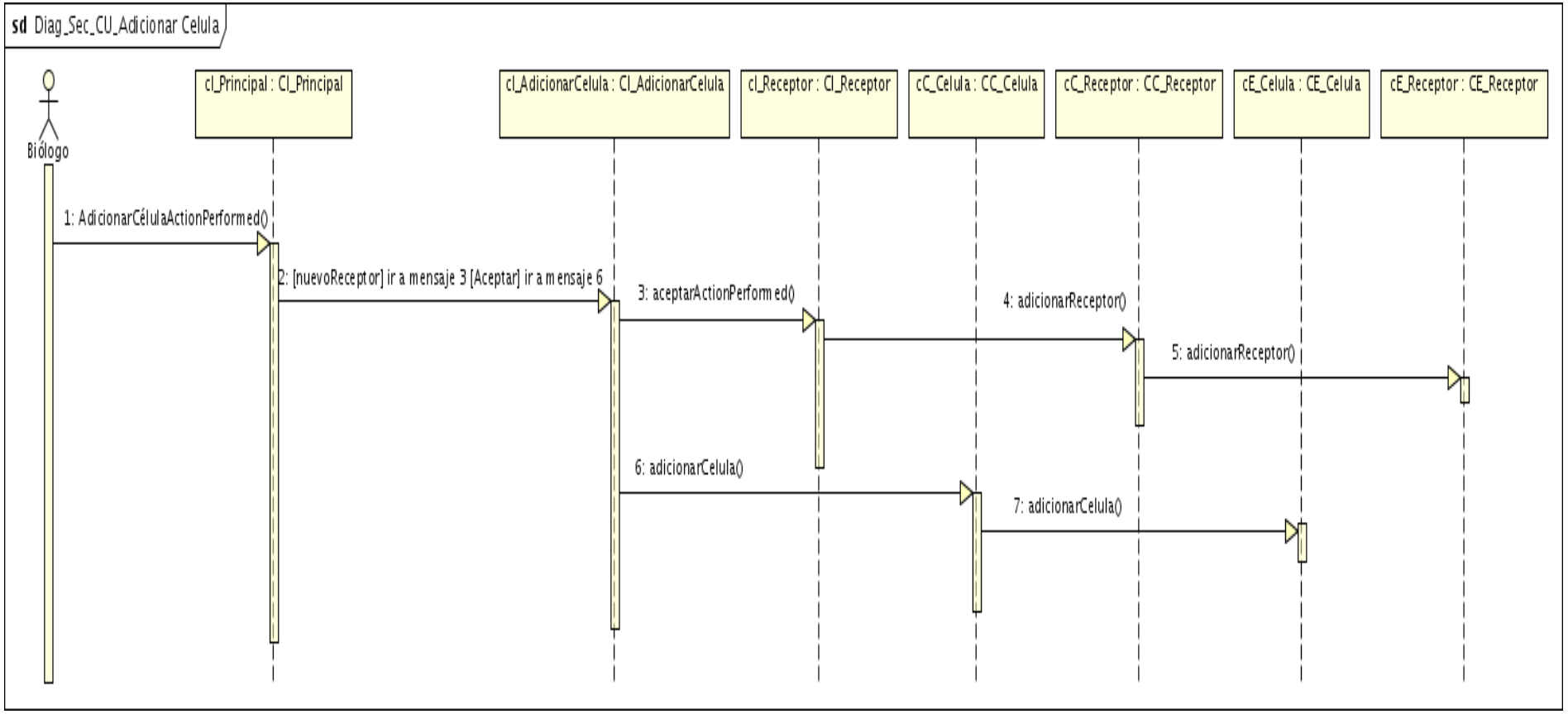


Figura 3.12 Diagrama de Secuencia del Caso de Uso Adicionar Célula

3.4 Modelo de Despliegue

Los diagramas de despliegue muestran las relaciones físicas entre los componentes hardware y software en el sistema final, es decir, la configuración de los elementos de procesamiento en tiempo de ejecución y los componentes software (procesos y objetos que se ejecutan en ellos) [35]. En el caso de J-ImmSim el diagrama de despliegue está compuesto por un nodo que representa una computadora cliente la cuál debe tener como características físicas principales al menos 512 MB de Memoria RAM y procesador de 3.0 GHz. Es necesario para un buen funcionamiento del sistema que la computadora tenga como Sistema Operativo Linux y que tenga instalada la máquina virtual de Java 1.5, Matlab 7.0 o versiones superiores.

Conclusiones

En el capítulo que concluye se describieron el estilo arquitectónico y los patrones de diseño utilizados en el desarrollo de J-ImmSim. Posteriormente se mostraron los diagramas de clases y de secuencia de los casos de usos principales así como el modelo de despliegue.

Capítulo 4
Implementación del Sistema

Introducción

En este capítulo se tratará lo concerniente a la implementación del sistema y la validación del mismo. Inicialmente se mostrará el diagrama de componentes así como el código fuente de algunas de las clases principales y las principales pantallas de la aplicación. Más adelante se mostrarán los resultados de la validación realizada al software mediante un modelo tomado de la literatura.

4.1 Diagrama de componentes

Un diagrama de componentes muestra un conjunto de componentes y sus relaciones. Los diagramas de componentes se utilizan para describir la vista de implementación estática de un sistema y están estrechamente relacionados con los diagramas de clases, ya que un componente normalmente se corresponde con una o más clases, interfaces o colaboraciones [36]. En la figura 4.1 se muestra el diagrama de componentes correspondiente a J-ImmSim.

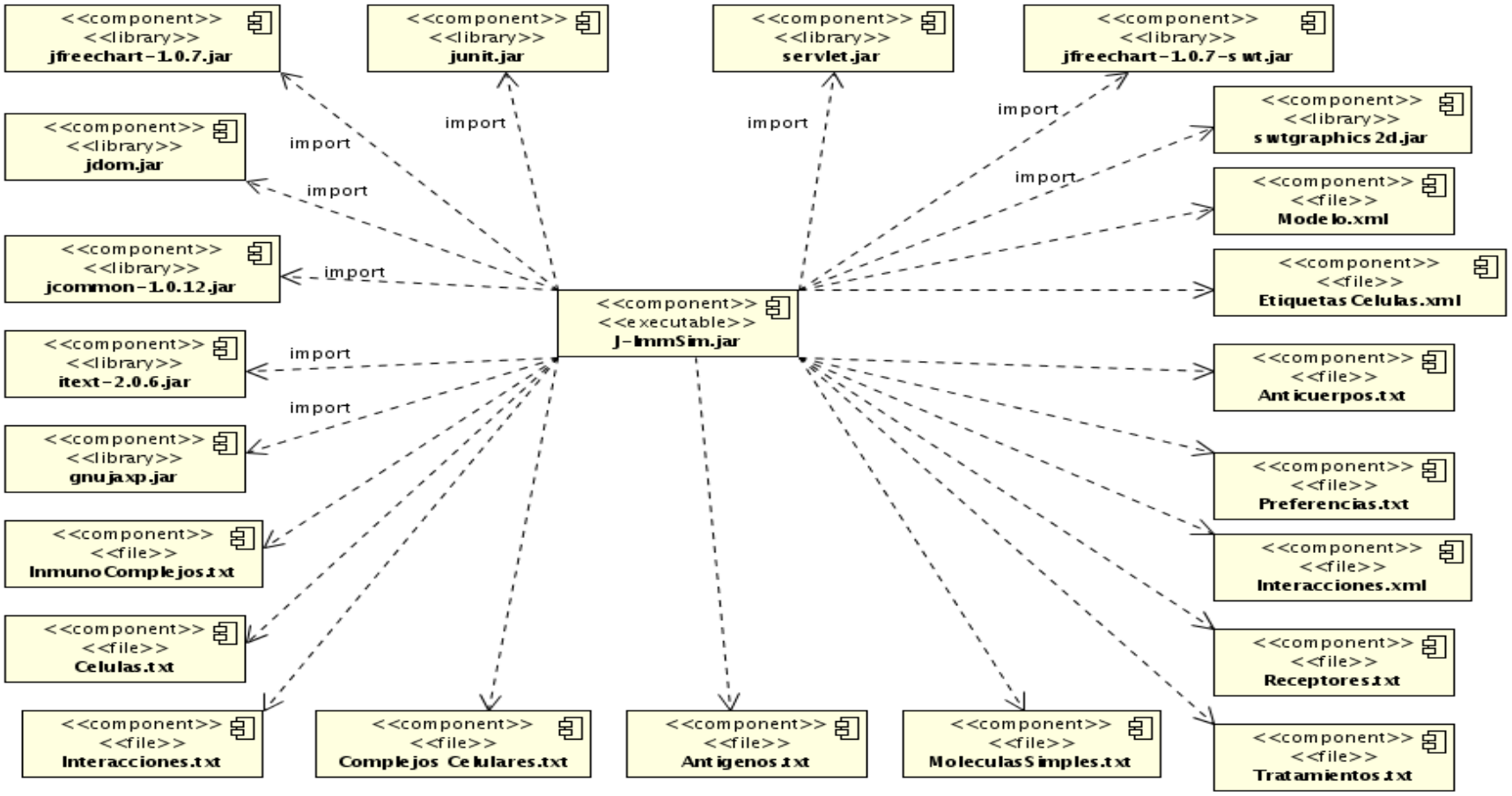


Figura 4.1 Diagrama de Componentes de J-ImmSim

4.2 Código fuente de las principales clases.

A continuación se muestran las clases Célula y Receptor, las cuales son de gran importancia en el sistema ya que constituyen dos de los entes biológicos más usados en las interacciones

Clase Celula	
Atributo	**Descripción**
protected double edad	Representa el tiempo que ha vivido la célula
protected double promedioVida	Representa el tiempo promedio que puede vivir la célula (similar a la esperanza de vida)
protected ArrayList<Receptor> receptores	Contiene los receptores presentes en la célula
Método	**Descripción**
public Celula(double pvida)	Crea la célula pasando como parámetro el promedio de vida
public Celula(int edad, ArrayList<Receptor> receptores, double pvida)	Crea la célula pasando como parámetro la edad, los receptores que presenta y el promedio de vida
public Celula(int edad, double pvida)	Crea la célula pasando como parámetro la edad y el promedio de vida
public void setMemoria(double porCiento)	Eleva el promedio de vida de la célula en el porciento indicado.
public boolean getEtiqueta(ArrayList<String> receps)	Retorna verdadero si los receptores introducidos por parámetros forman parte de la célula.
public abstract void envejecer()	Aumenta la edad de la célula. Se implementa en las clases hijas.
public abstract boolean estaReceptor(String receptor)	Comprueba la existencia del receptor introducido por parámetro en la célula
public abstract boolean estaDisponible()	Comprueba si la célula está disponible para interactuar. Se redefine en la clase hija

```
package EntidadesSistemaInmune.Genericas;

import EntidadesSistemaInmune.Genericas.Receptor;
import java.util.ArrayList;

public abstract  class Celula
{
     protected  double edad;
     protected double promedioVida;
     protected  ArrayList<Receptor> receptores;

   /** Creates a new instance of cell */
    public Celula(double pvida)
     {
       this.edad =0;
       this.receptores  = new ArrayList<Receptor>();
       this.promedioVida = pvida;
    }
    public Celula(int edad, ArrayList<Receptor> receptores, double pvida)
     {
       this.edad = edad;
       this.receptores = receptores;
       this.promedioVida = pvida;
     }
     public abstract String getNombre();
     public Celula(int edad, double pvida)
   {
      this.edad = edad;
      this.promedioVida = pvida;
   }
 public  double getPromedioVida()
 {
    return promedioVida;
```

```
 }
public abstract boolean estaReceptor(String receptor);
public double getEdad()
{
  return this.edad;
}
public void setEdad(int nuevaEdad)
{
   this.edad = nuevaEdad;
 }
public void setMemoria(double porCiento)
{
  promedioVida = promedioVida + (promedioVida* porCiento/ 100);
}
public abstract  void envejecer();
public  ArrayList<Receptor> getReceptores()
{
   return this.receptores;
}
public boolean getEtiqueta(ArrayList<String> receps)
{
   for(int i = 0; i < receps.size(); i++)
      if(!estaReceptor(receps.get(i)))
            return false;
       return true;
}
public abstract  boolean estaDisponible();

}
```

Clase Receptor	
Atributo	**Descripción**
private String nombre	Nombre del receptor
private boolean esGenerico	Verdadero si es genérico
private boolean esPredefinido	Verdadero si es predefinido por el usuario
private boolean enCelula	Verdadero si toma valor en la célula
String valor	Guarda el valor del receptor (BitString)
Método	**Descripción**
public Receptor(String nombre, boolean esGenerico, String valor)	Crea un receptor con el nombre, el valor y si es genérico.
public Receptor(String nombre, boolean esGenerico, boolean esPredefinido, boolean enCelula, String valor)	Crea un receptor con el nombre, el valor y si es genérico, predefinido o si toma valor en la célula.

```
package EntidadesSistemaInmune.Genericas;
import Excepciones.ErrorGenericoNoValor;

public class Receptor
{
   private String nombre;
   private boolean esGenerico;
   private boolean esPredefinido;
   private boolean enCelula;
   String valor;

   /** Creates a new instance of Receptor */
   public Receptor(String nombre, boolean esGenerico, boolean esPredefinido,
                   boolean enCelula, String valor)
 {
      this.nombre = nombre;
      this.esGenerico = esGenerico;
      this.esPredefinido = esPredefinido;
      this.enCelula = enCelula;
```

```
        this.valor = valor;
    }
 public Receptor(String nombre, boolean esGenerico, String valor)
{
   this.nombre = nombre;
   this.esGenerico = esGenerico;
   this.valor = valor;
}
public String getNombre()
{
  return this.nombre;
 }
 public boolean getEsGenerico()
{
  return this.esGenerico;
}
public boolean isEnCelula()
{
   return enCelula;
}
public boolean isEsPredefinido()
{
    return esPredefinido;
}
public String getValor()
{
   return valor;}
public void setValor(String nuevoValor)
{
   this.valor = nuevoValor;
}
}
```

4.3 Validación de J-ImmSim

Variar parámetros en un experimento *in vitro* no resulta muy práctico y eficiente, sobre todo en relación con el tiempo de la investigación; esta es la fortaleza que puede tener un simulador. Existen diferentes formas de validar un modelo, una de ellas es tomar datos de experimentos conocidos (propios o de otros investigadores) y probar el mismo. Otra opción es partir de un modelo ya probado y demostrar la similitud de los resultados, esta variante es la que se ha elegido y se presenta a continuación.

Modelo a probar

La validación de J-ImmSim se basa en el modelo de Seiden y Celada (Franco Celada, del Hospital Metropolitano de Enfermedades Articulares de Nueva York, y Philip E. Seiden, del Centro de Investigaciones Thomas J. Watson de IBM) para la respuesta humoral [38] y se evalúa variando un grupo de parámetros, principalmente las poblaciones iniciales de los diferentes entes biológicos. La figura 4.2 muestra gráficamente los entes biológicos e interacciones existentes.

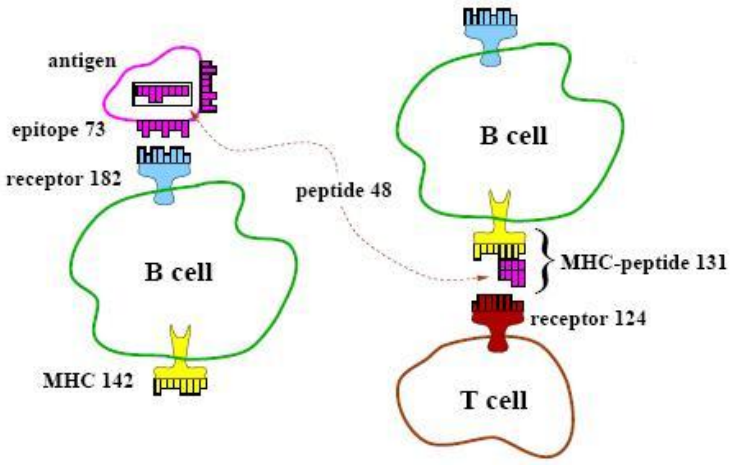


Figura 4.2 Modelo de Validación

La Célula B interactúa con el Antígeno (BCR[1] y epítope respectivamente) y se diferencia expresando el receptor MHC péptido. La Célula T interactúa con una Célula B que presenta el receptor MHC-péptido y en este proceso se duplican ambas células, las células B se diferencian en células de memoria y generan anticuerpos quienes son los encargados de eliminar los antígenos.

[1] Receptor de la Célula B

En este sistema un paso de tiempo equivale a 8 horas. La prueba consiste en introducir 500 Células B y T respectivamente, 150 entes de vacunas (antígeno inactivo sin capacidad de replicarse, solo le brindan al Sistema Inmune la información de los epítopes del antígeno que será introducido más tarde). El SI debe tomar la información brindada por la vacuna y combatir el antígeno, este proceso se conoce como Respuesta Primaria, al cabo de 132 horas se vuelve a introducir la misma cantidad del antígeno pero esta vez con una tasa de replicación de 1.2, lo que conlleva a una Respuesta Secundaria del SI.

Por no contar con una suficiente capacidad de cálculo, el modelo original se ha escalado a la mitad, es decir, la cantidad de entes biológicos y el tiempo de simulación se redujeron a un 50%. A continuación se muestra una tabla con los datos de 10 simulaciones realizadas (Ver Anexo 2).

TEARP = cantidad antígenos eliminados (150)/TRP
TEARS = cantidad antígenos eliminados (150)/TRS

# S	CB	CT	CBMa	Antic1	TRP	TRS	TEARP	TEARS
1	500	606	85	2855	165	68	0.909	2.205
2	655	676	24	3143	210	63	0.714	2.38
3	715	762	17	3487	302	56	0.496	2.67
4	656	655	47	3188	231	112	0.649	1.339
5	572	612	44	2952	296	63	0.506	2.38
6	637	690	37	3184	176	64	0.852	2.343
7	512	606	75	2836	184	96	0.815	1.563
8	567	640	45	2798	208	72	0.721	2.083
9	552	588	48	2750	352	50	0.426	3.0
10	537	591	26	2695	208	64	0.721	2.343
Promedio	590	643	45	2989	233	70	***0.68***	***2.231***

Leyenda

S: Número Simulación
CB: CelulaB
CT: CelulaT
CBMa: CelulaBMacrófago
Antic1: Anticuerpo1
Vac: Vacuna
Antig1: Antigeno1
TRP: Tiempo Respuesta Primaria
TRS: Tiempo Respuesta Secundaria
TEARP: Tasa de Eliminación de Antígenos (Respuesta Primaria)
TEARS: Tasa de Eliminación de Antígenos (Respuesta Secundaria).

En la figura 4.3 se muestra un gráfico de barras con la Tasa de Eliminación de Antígenos en la Respuesta Primaria (RP) y Secundaria (RS) respectivamente de cada una de las 10 simulaciones y el promedio general de las dos respuestas. Como se puede apreciar la RS representa el 328% de la RP ante un antígeno activo, es decir, es 228% más rápida que la RP demostrando la validez de las simulaciones realizadas en J-ImmSim.

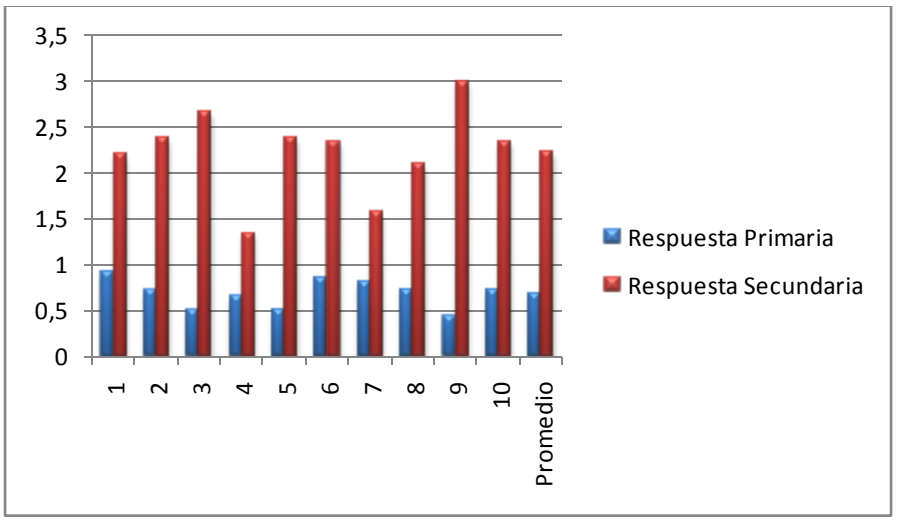


Figura 4.3 Respuesta Primaria y Secundaria de J-ImmSim

Conclusiones

En el capítulo que finaliza se mostró el Diagrama de Componentes y el código fuente de dos de las clases principales del software J-ImmSim. En una segunda parte se validó el sistema teniendo como punto de partida un modelo biológico tomado de la literatura corroborando que el software simula la respuesta primaria y secundaria con resultados semejantes a los referenciados en la literatura.

Conclusiones Generales

- Se definió un modelo base para la simulación del Sistema Inmune utilizando como fundamentación matemática Ecuaciones Diferenciales Estocásticas y como técnica de Inteligencia Artificial Autómatas Celulares.
- Se realizó el diseño del simulador que fue utilizado en la implementación del mismo. Entre las funcionalidades y novedades de J-ImmSim en su versión 1.0 se pueden mencionar las siguientes:
  - ✓ Definición del modelo biológico y simulación del mismo integrado en un único software.
  - ✓ Introducción de entes biológicos definidos por el biólogo.
  - ✓ Definición de interacciones no predefinidas siguiendo tres patrones: Célula-Célula, Célula-Antígeno y Anticuerpo-Antígeno.
  - ✓ Definición de la escala de tiempo por parte del biólogo.
  - ✓ Introducción de aleatoriedad al ejecutar las interacciones entre los diferentes entes biológicos.
- Se validó la herramienta J-ImmSim mediante un modelo tomado de la literatura.

Recomendaciones

1. Definir e implementar otros patrones de interacciones que permitan otros comportamientos entre los entes biológicos.
2. Implementar la versión distribuida del sistema (J-ImmSim 2.0).
3. Implementar un módulo de Análisis de las simulaciones.

Referencias Bibliográficas

[1] SHANNON, ROBERT E. *"System Simulation. The Art and Science"*. PrenticeHall, Englewood Cliffs, NY, 1975. [Consultado: 27 noviembre 2007]

[2] INSTITUTO NACIONAL DEL CANCER. *"Terapias biológicas del cáncer: preguntas y respuestas"* [Consultado: 11 diciembre 2007; Disponible en: http://www.cancer.gov/espanol/cancer/hojas-informativas/terapias-biologicas-respuestas]

[3] SEIDEN, P.E. & CELADA, F. "*A model for simulating cognate recognition and response in the immune system".* J. theor. Biol. 158: 329-357. 1992.

[4] PELACHO AJA, ANA MARÍA. *"El libro electrónico del cultivo in Vitro" [*Consultado: 12 febrero 2008; Disponible en: http://www.etsea2.udl.es/invitro/]

[5] BALLÚ, ANNA. "*Biología de Sistemas*". [Consultado: 5 diciembre 2008; Disponible en: http://www.uvic.cat/eps/dept/biologiasistemes/es/inici.html]

[6] OPENWETWARE PROJECT. "*Biología Sintética*"[Consultado: 23 noviembre 2007; Disponible en: http://openwetware.org/wiki/Biolog%C3%ADa_Sint%C3%A9tica:Preguntas]

[7] ADEREM, ALAM. *"Premises underlying systems biology research"* [Consultado: 18 noviembre 2007; Disponible en: http://www.systemsbiology.org/Systems_Biology_in_Depth/Premise_of_Systems_Biology]

[8] TORRES, NESTOR V. *"Caos en Sistemas Biológicos" [Consultado: 21 noviembre 2007; Disponible en:* http://webpages.ull.es/users/imarrero/sctm04/modulo2/8/ntorres.pdf]

[9] FIUNER. "*Modelización de Sistemas Biológicos, Bioingeniería I".* Año: 2007 [Consultado: 27 noviembre 2007]

[10] FARBIARZ, JORGE. *"Complejidad, Caos y Sistemas Biológicos"* [Consultado: 25 noviembre 2007; Disponible en:http://encolombia.com/medicina/academedicina/m-02JFarbiarz.htm]

[11] PLANCHART MÁRQUEZ, ORLANDO. *"La Modelación Matemática: alternativa didáctica en la enseñanza de precálculo".* [Consultado: 28 noviembre 2007; Disponible en: http://cremc.ponce.inter.edu/1raedicion/modelacion.htm]

[12] INTERNATIONAL VEGETARIAN UNION. "*La vivisección y disección de animales: una atrocidad ética y científica" [Consultado: 1 diciembre 2007; Disponible en:* http://www.ivu.org/ave/vivisec.html]

[13] CENTRO DE CONOCIMIENTO PRONAT. "*Glosario de Términos Naturistas y Farmacéuticos"* [Consultado: 13 diciembre 2007; Disponible en: http://www.pronat.com.mx/Glosario.htm]

[14] *"Sistema inmunológico"* Enciclopedia Microsoft® Encarta® Online. Año: 2007 [Consultado: 3 diciembre 2007; Disponible en: http://es.encarta.msn.com/encyclopedia_761575681/Sistema_inmunol%C3%B3gico.html]

[15] DUFFY, KATHLEEN. "*Los autómatas celulares y la tapicería cósmica"* [Consultado: 15 diciembre 2007; Disponible en: http://iieh.net/articulos/articulo0001.php]

[16] GONZÁLEZ VARGAS, LUIS FERNANDO. *"Una Introducción a los Autómatas Celulares"* [Consultado: 23 Enero, 2007, Disponible en: http://yupana.autonoma.edu.co/publicaciones/yupana/005/autocelular/Automatas.html]

[17] RED Científica. "*Autómatas Celulares"* [Consultado: 15 diciembre 2007; Disponible en: http://www.redcientifica.com/gaia/ac/auto_c.htm]

[18] GARCIA, JOSE A. *"Modelación Matemática de la hipermutación somática y sus implicaciones biológicas".* [Consultado: 12 febrero 2008; Disponible en: www.ci.ulsa.mx/~jgarcia/proyec/tcr.pdf]

[19] UNIVERSIDAD DE VALENCIA. *"Procesos Estocásticos".* [Consultado: 12 febrero 2008; Disponible en: www.uv.es/olmos/mod_din_estocastico.pdf]

[20] JACOBSON, IVAR; BOOCH, GRADY Y RUMBAUGH, JAMES. *"El proceso unificado de software".* Primera edición. Pearson Educación, S.A. Año:2000.

[21] CENTRO DE DESARROLLO DE SISTEMAS DE INFORMACION *"OpenUp*" [Consultado: 12 diciembre 2007; Disponible en: http://www.cendesi.com/site/es/articles.php?lng=es&pg=11]

[22] UNIVERSIDAD AUTÓNOMA DE MADRID *"Clases y Herencia"* [Consultado: 12 diciembre 2007; Disponible en: http://tangow.ii.uam.es/mfreire/poo_0506/p2/index.html]

[23] GSINNOVA. *"Rational Rose Enterprise".* [Consultado: 11 diciembre 2007; Disponible en: http://64.233.169.104/search?cache:gxSciUl7NMEJ:www.rational.com.ar/herramientas/roseenterprise.html +caracter%C3%ADsticas+rational+rose&hl=es&ct=clnk&cd=7&gl=cu]

[24] VISUAL PARADIGM FOR UML. *"Features"* [Consultado: 12 diciembre 2007; Disponible en: http://www.visual-paradigm.com/product/vpuml/features/]

[25] UNIVERSIDAD DE MALAGA. *"POO"* [Consultado: 6 marzo 2008; Disponible en: http://juanfc.lcc.uma.es/EDU/PM/2.POO-Presentacion.pdf]

[26] MANUAL DE JAVA. "*Características de Java*". [Consultado: 3 febrero 2007; Disponible en: http://www.manual-java.com/manualjava/caracteristicas-java.html]

[27] ARIKAH. *"NetBeans".* [Consultado: 12 enero 2008; Disponible en: *http://www.arikah.net/enciclopedia-espanola/Netbeans]*

[28] BERMEJO SANZ, LAURA *"Eclipse como IDE".* [Consultado: 15 enero 2008; Disponible en: http://kybele.escet.urjc.es/documentos/HC/Exposiciones/EclipseIDE.pdf]

[29] CASTIGLIONE, FILIPPO. *"C-ImmSim v.6*". Año: 2006. Número de Páginas: 71.

[30] KDE Documentation. "*Elementos de UML".* [Consultado: 15 enero 2008; Disponible en: http://docs.kde.org/kde3/es/kdesdk/umbrello/uml-elements.html]

[31] MICROSOFT. "*Arquitectura de software".* [Consultado: 1 abril 2008; Disponible en: http://www.microsoft.com/spanish/msdn/arquitectura/roadmap_arq/arquitectura_soft.mspx#EEB]

[32] "*Introducción al API Reflection (Reflexión) de Java".* [Consultado: 10 diciembre 2007; Disponible en: http://www.javahispano.org/contenidos/es/introduccrion_al_api_reflection_reflexrion_de_ja/]

[33] LARMAN, CRAIG: "*UML y Patrones. Introducción al análisis y diseño orientado a objetos*". Año: 1999. México. Prentice-Hall Hispanoamérica.

[34] SPARX SYSTEMS "*Diagrama de Secuencia UML 2*". [Consultado: 4 abril 2008; Disponible en: http://www.sparxsystems.com.ar/tutorial/uml_tutorial2.html]

[35] FERNANDEZ VILAS, ANA *"Diagrama de Despliegue".* [Consultado: 4 abril 2008; Disponible en: http://www-gris.det.uvigo.es/~avilas/UML/node50.html]

[36] FERNANDEZ VILAS, ANA *"Diagrama de Componentes".* [Consultado: 4 abril 2008; Disponible en: http://www-gris.det.uvigo.es/~avilas/UML/node22.html]

[37] MARTÍNEZ H, ELISEO. *"El Movimiento Browniano".* [Consultado: 23 enero 2008; Disponible en: http://www.uantof.cl/facultades/csbasicas/Matematicas/academicos/emartinez/dinamica/Brown/brown.html]

[38] SOCIEDAD ITALIANA DE FÍSICA *"Modeling Evolution and Inmune System by Cellular Automata"*

[39] CANOS, JOSE H y LETELIER, PATRICIO. "*Metodologías Ágiles en el Desarrollo de Software"* [Consultado: 12 junio 2008; Disponible en: http://www.willydev.net/descargas/prev/TodoAgil.Pdf]

[40] UNIVERSIDAD SIMÓN BOLIVAR *"Arquitectura de capas"* [Consultado: 14 junio 2008; Disponible en: http://www.ldc.usb.ve/~teruel/ci3715/clases/arqCapas.html]

[41] IEEE *"Modelo FURPS+"* [Consultado: 14 junio 2008; Disponible en: http://www.ewh.ieee.org/r9/guadalajara/boletin/marzo02/modelofurps.htm]

Bibliografía

- ADEREM, ALAM. *"Premises underlying systems biology research"* [Disponible en: http://www.systemsbiology.org/Systems_Biology_in_Depth/Premise_of_Systems_Biology]
- ARIKAH. *"NetBeans".* [Disponible en: *http://www.arikah.net/enciclopedia-espanola/Netbeans]*
- BALLÚ, ANNA. "*Biología de Sistemas*". [Disponible en: http://www.uvic.cat/eps/dept/biologiasistemes/es/inici.html]
- BERMEJO SANZ, LAURA *"Eclipse como IDE".* [Disponible en: http://kybele.escet.urjc.es/documentos/HC/Exposiciones/EclipseIDE.pdf]
- CANOS, JOSE H y LETELIER, PATRICIO. "*Metodologías Ágiles en el Desarrollo de Software"* [Disponible en: http://www.willydev.net/descargas/prev/TodoAgil.Pdf]
- CASTIGLIONE, FILIPPO. *"C-ImmSim v.6*". Año: 2006.
- CENTRO DE CONOCIMIENTO PRONAT. "*Glosario de Términos Naturistas y Farmacéuticos"* [Disponible en: http://www.pronat.com.mx/Glosario.htm]
- CENTRO DE DESARROLLO DE SISTEMAS DE INFORMACION *"OpenUp*" [Disponible en: http://www.cendesi.com/site/es/articles.php?lng=es&pg=11]
- Centro de Tecnología Informática Universidad de Navarra [Disponible en: http://www.unav.es/cti/manuales/Java/indice.html]
- CRAIG LARMAN: "*UML y Patrones. Introducción al análisis y diseño orientado a objetos*". [Disponible en: http://bibliodoc.uci.cu/pdf/reg00061.pdf]
- DUFFY, KATHLEEN. "*Los autómatas celulares y la tapicería cósmica"* [Disponible en: http://iieh.net/articulos/articulo0001.php]
- FARBIARZ, JORGE. *"Complejidad, Caos y Sistemas Biológicos"* [Disponible en: http://encolombia.com/medicina/academedicina/m-02JFarbiarz.htm]
- *FERNANDEZ* VILAS, ANA *"Diagrama de Componentes".* [Disponible en: http://www-gris.det.uvigo.es/~avilas/UML/node22.html]
- *FERNANDEZ* VILAS, ANA *"Diagrama de Despliegue".* [Disponible en: http://www-gris.det.uvigo.es/~avilas/UML/node50.html]
- FIUNER. "*Modelización de Sistemas Biológicos, Bioingeniería I".* Año: 2007
- GARCIA, JOSE A. *"Modelación Matemática de la hipermutación somática y sus implicaciones biológicas".* [Disponible en: www.ci.ulsa.mx/~jgarcia/proyec/tcr.pdf]

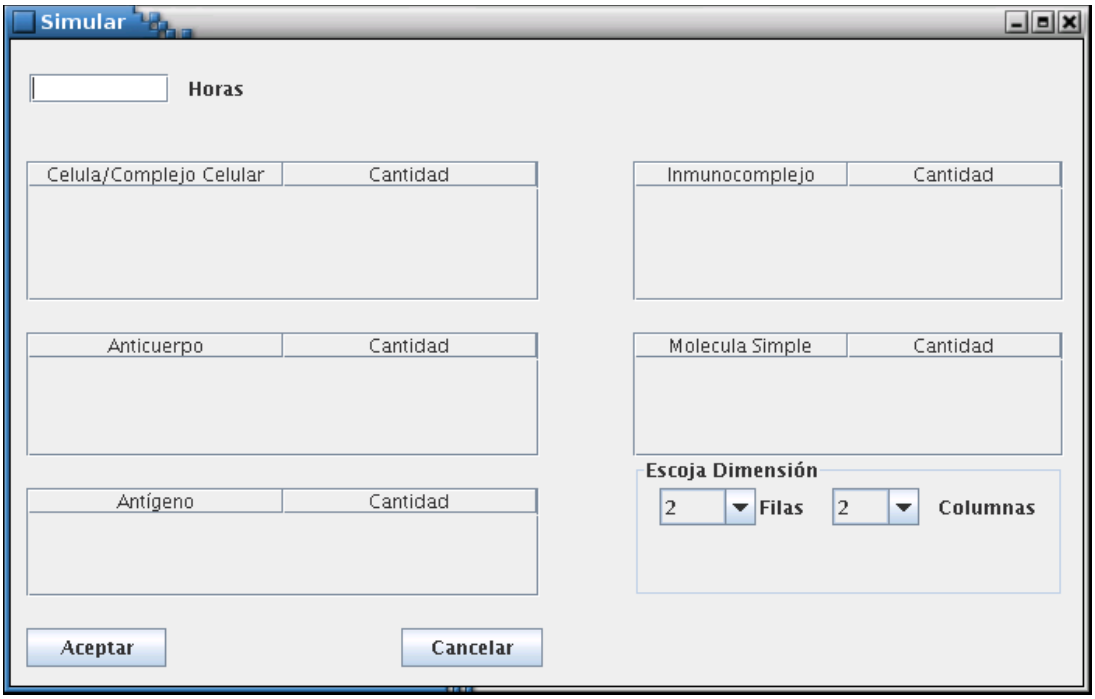
- GONZÁLEZ VARGAS, LUIS FERNANDO. *"Una Introducción a los Autómatas Celulares"* [Disponible en: http://yupana.autonoma.edu.co/publicaciones/yupana/005/autocelular/Automatas.html]
- GSINNOVA. *"Rational Rose Enterprise".* [Disponible en: http://64.233.169.104/search?cache:gxSciUl7NMEJ:www.rational.com.ar/herramientas/roseenterprise.html+caracter%C3%ADsticas+rational+rose&hl=es&ct=clnk&cd=7&gl=cu]
- IEEE *"Modelo FURPS+"* [Disponible en: http://www.ewh.ieee.org/r9/guadalajara/boletin/marzo02/modelofurps.htm]
- INSTITUTO NACIONAL DEL CANCER. *"Terapias biológicas del cáncer: preguntas y respuestas"* [Disponible en: http://www.cancer.gov/espanol/cancer/hojas-informativas/terapias-biologicas-respuestas]
- INTERNATIONAL VEGETARIAN UNION. "*La vivisección y disección de animales: una atrocidad ética y científica*" [Disponible en: http://www.ivu.org/ave/vivisec.html]
- "*Introducción al API Reflection (Reflexión) de Java".* [Disponible en: http://www.javahispano.org/contenidos/es/introduccrion_al_api_reflection_reflexrion_de_ja/]
- JACOBSON, IVAR; BOOCH, GRADY Y RUMBAUGH, JAMES. *"El proceso unificado de software".* Primera edición. Pearson Educación, S.A. Año:2000.
- Java en Castellano [Disponible en: http://www.programacion.net/java/]
- KDE Documentation. "*Elementos de UML".* [Disponible en: http://docs.kde.org/kde3/es/kdesdk/umbrello/uml-elements.html]
- Kleinstein, Steven "*ImmSim++"* [Disponible en: http://www.cs.princeton.edu/immsim/download.html]
- Laboratorio de Química Computacional. *"Autómatas Celulares"* [Disponible en: *http://www.luventicus.org/laboratorio/AutomatasCelulares/index.html]*
- MANUAL DE JAVA. "*Características de Java*". [Disponible en: http://www.manual-java.com/manualjava/caracteristicas-java.html]
- MICROSOFT. "*Arquitectura de software".* [Disponible en: http://www.microsoft.com/spanish/msdn/arquitectura/roadmap_arq/arquitectura_soft.mspx#EEB
- OPENWETWARE PROJECT. "*Biología Sintética*" [Disponible en: http://openwetware.org/wiki/Biolog%C3%ADa_Sint%C3%A9tica:Preguntas]

- PELACHO AJA, ANA MARÍA. *"El libro electrónico del cultivo in Vitro"* [Disponible en: http://www.etsea2.udl.es/invitro/]
- Planchart MÁRQUEZ, ORLANDO. *"La Modelación Matemática: alternativa didáctica en la enseñanza de precálculo".* [Disponible en: http://cremc.ponce.inter.edu/1raedicion/modelacion.htm]
- RED Científica. "*Autómatas Celulares"* [Disponible en: http://www.redcientifica.com/gaia/ac/auto_c.htm]
- RUBI CAPACETI, JORGE *"Realidad Virtual"* [Disponible en: http://www.innovatecno.com/Automata.php]
- SEIDEN, P.E. & CELADA, F. "*A model for simulating cognate recognition and response in the immune system".* J. theor. Biol. 158: 329-357. 1992.
- SHANNON, ROBERT E. *"System Simulation. The Art and Science"*. PrenticeHall, Englewood Cliffs, NY, 1975.
- Sistema *inmunológico* Enciclopedia Microsoft® Encarta® Online. Año: 2007 [Disponible en: http://es.encarta.msn.com/encyclopedia_761575681/Sistema_inmunol%C3%B3gico.html]
- SOCIEDAD ITALIANA DE FÍSICA *"Modeling Evolution and Inmune System by Cellular Automata"*
- SPARX *SYSTEMS* "*Diagrama de Secuencia UML 2*". [Disponible en: http://www.sparxsystems.com.ar/tutorial/uml_tutorial2.html]
- *Teoría de autómata celular* [Disponible en: http://delta.cs.cinvestav.mx/~mcintosh/comun/tesismaestria/rene/tesisReneHtml/node14.html]
- TORRES, NESTOR V. *"Caos en Sistemas Biológicos" [Disponible en:* http://webpages.ull.es/users/imarrero/sctm04/modulo2/8/ntorres.pdf]
- UNIVERSIDAD AUTÓNOMA DE MADRID *"Clases y Herencia"* [Disponible en: http://tangow.ii.uam.es/mfreire/poo_0506/p2/index.html]
- UNIVERSIDAD DE MALAGA. *"POO"* [Disponible en: http://juanfc.lcc.uma.es/EDU/PM/2.POO-Presentacion.pdf]
- UNIVERSIDAD DE VALENCIA. *"Procesos Estocásticos".* [Disponible en: www.uv.es/olmos/mod_din_estocastico.pdf]
- UNIVERSIDAD SIMÓN BOLIVAR *"Arquitectura de capas"* [Disponible en: http://www.ldc.usb.ve/~teruel/ci3715/clases/arqCapas.html]

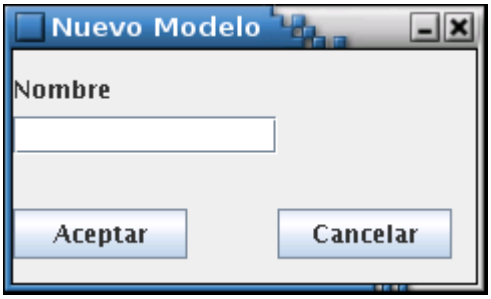
- Visionero, Visual Paradigm for UML. 2008. [Disponible en: http://www.versioncero.com/noticia/210/visual-paradigm-for-uml]
- Visual Paradigm [Disponible en: http://www.visual-paradigm.com/product/dbva]
- *VISUAL* PARADIGM FOR UML. *"Features"* [Disponible en: http://www.visual-paradigm.com/product/vpuml/features/]

Anexos

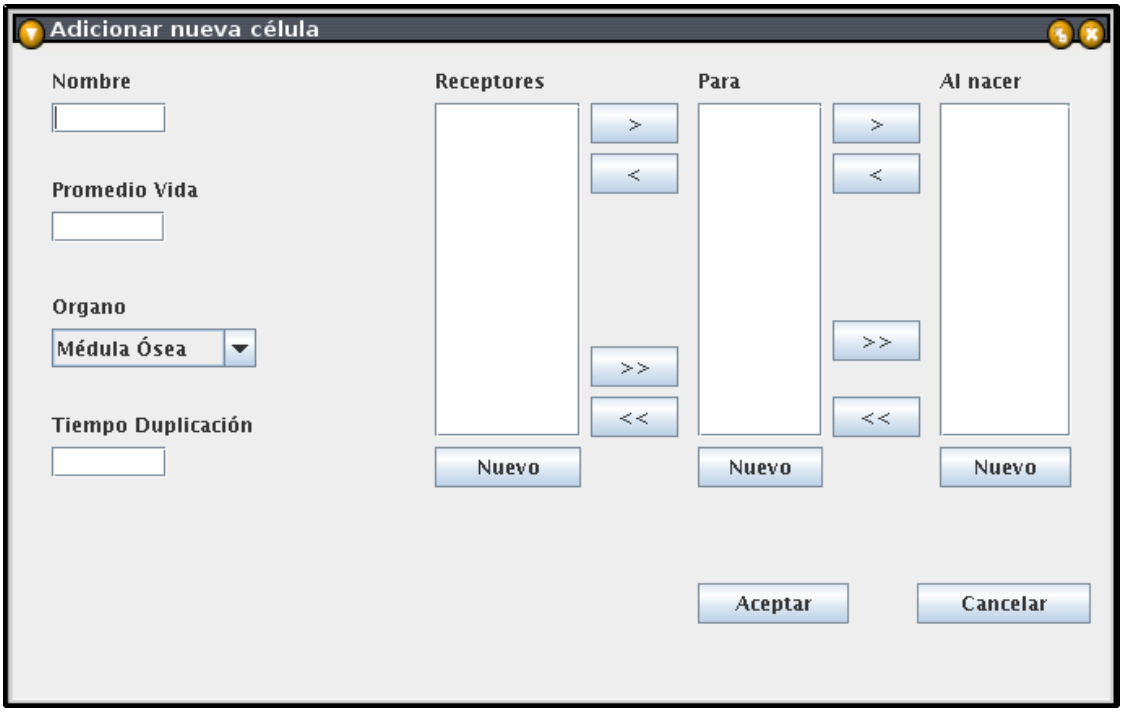
Anexo 1 Prototipos de Interfaz No Funcionales.



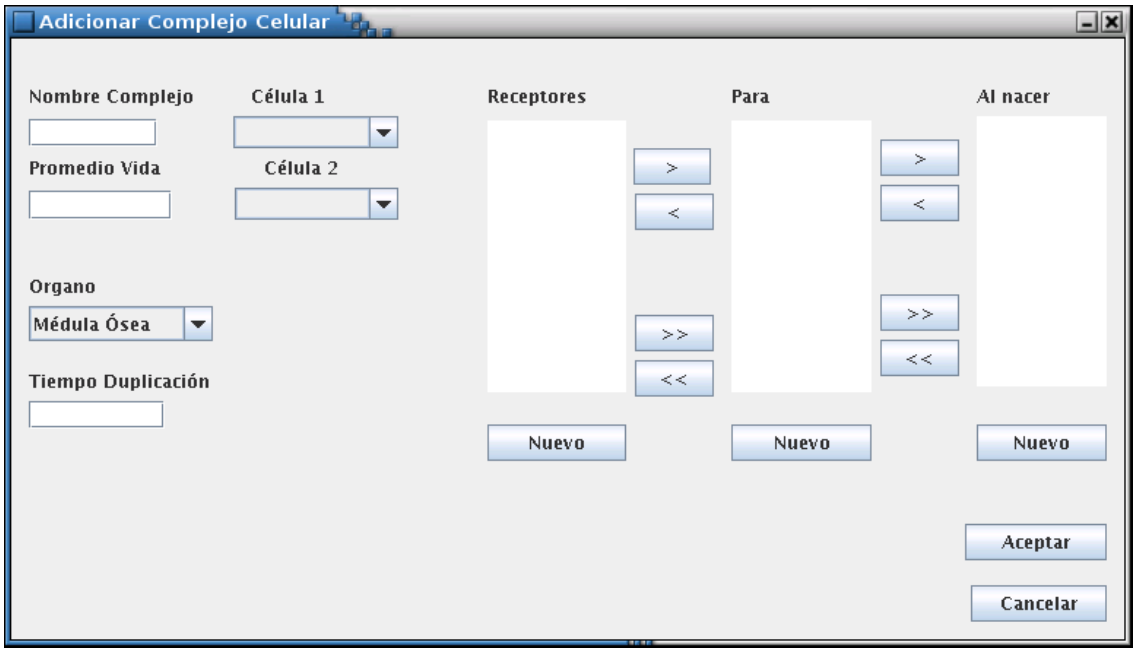
Anexo 1.1 Prototipo de Interfaz Caso de Uso Simular Sistema Inmune



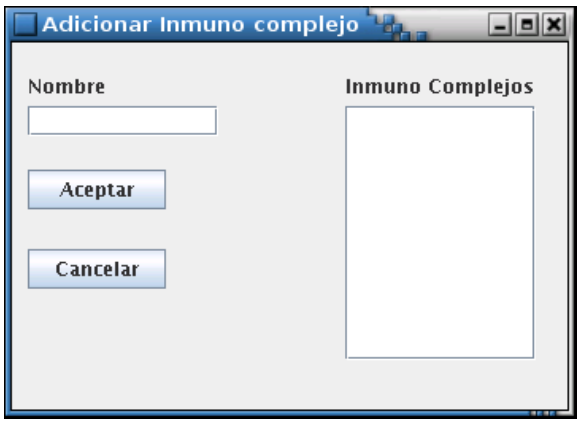
Anexo 1.2 Prototipo de Interfaz Caso de Uso Gestionar Modelo



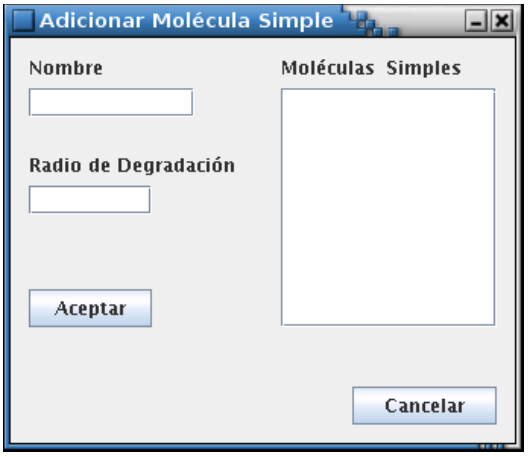
Anexo 1.3 Prototipo de Interfaz Caso de Uso Adicionar Célula



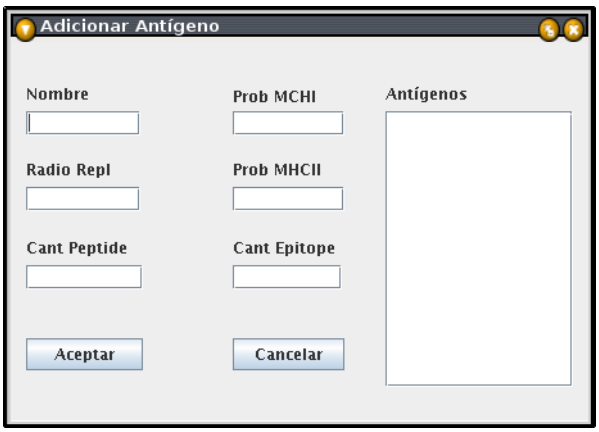
Anexo 1.4 Prototipo de Interfaz Caso de Uso Adicionar Complejo Celular



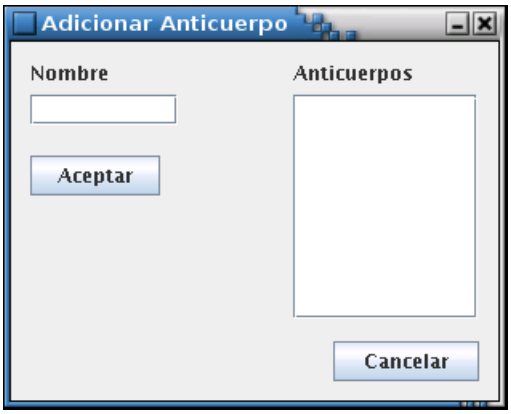
Anexo 1.5 Prototipo de Interfaz Caso de Uso Adicionar InmunoComplejo



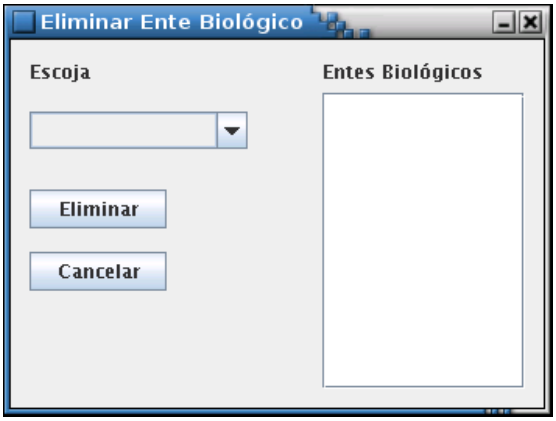
Anexo 1.6 Prototipo de Interfaz Caso de Uso Adicionar Molécula Simple



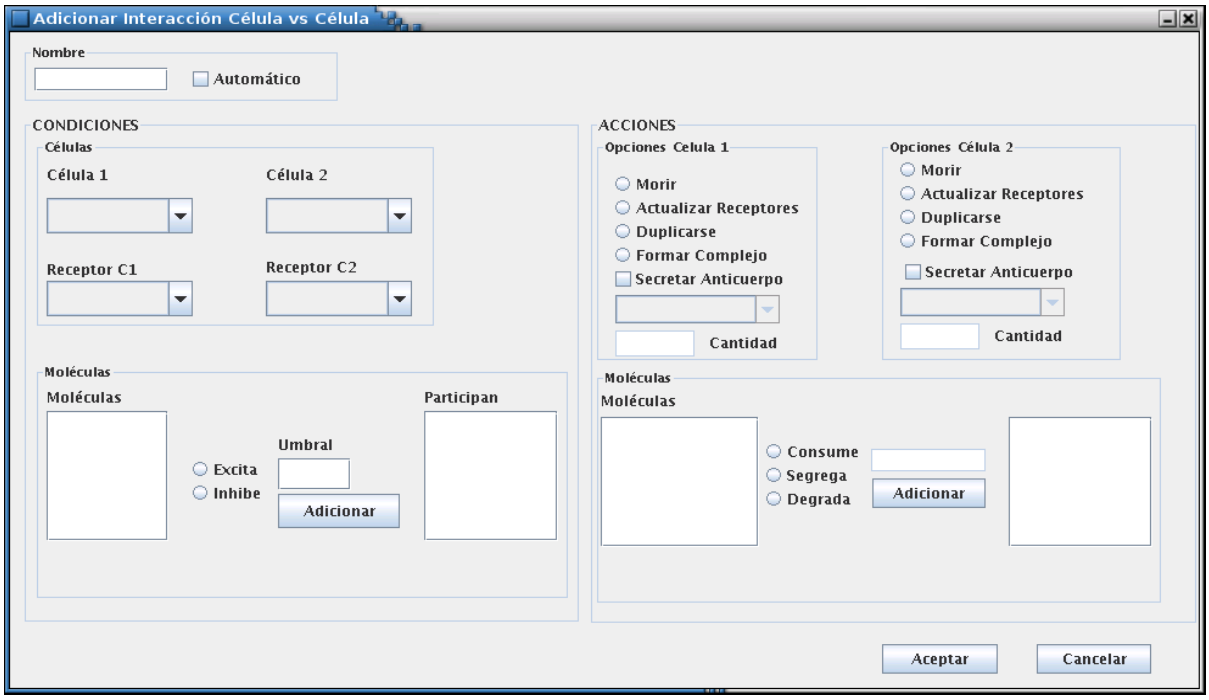
Anexo 1.7 Prototipo de Interfaz Caso de Uso Adicionar Antígeno



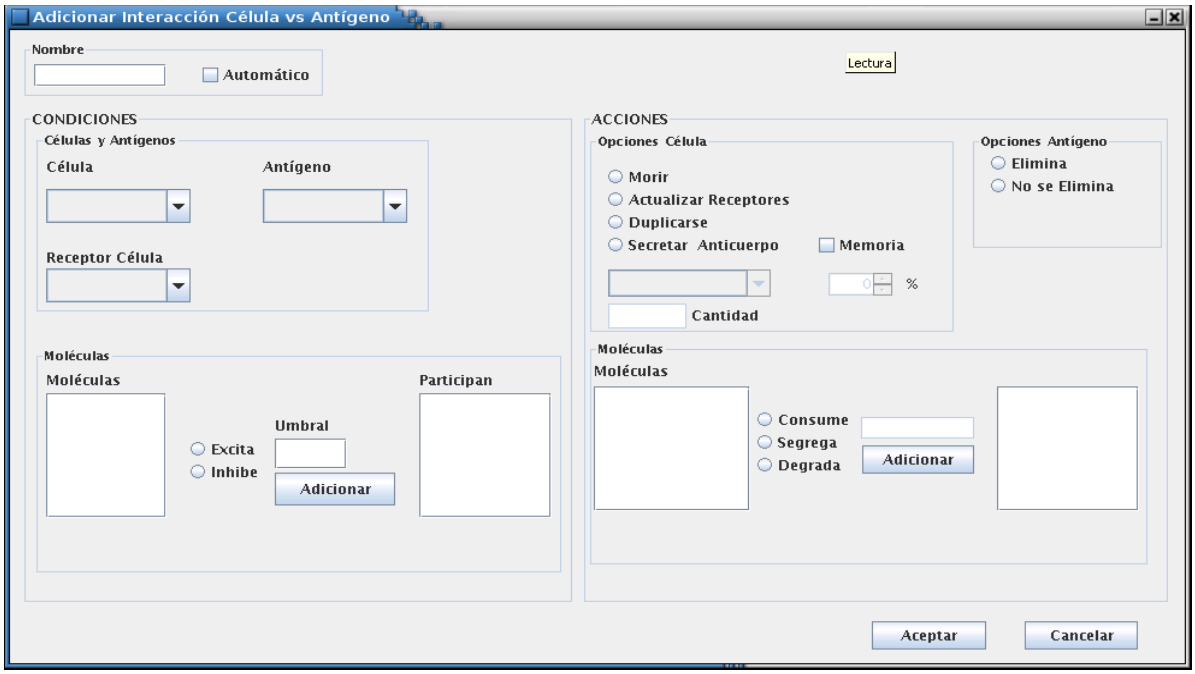
Anexo 1.8 Prototipo de Interfaz Caso de Uso Adicionar Anticuerpo



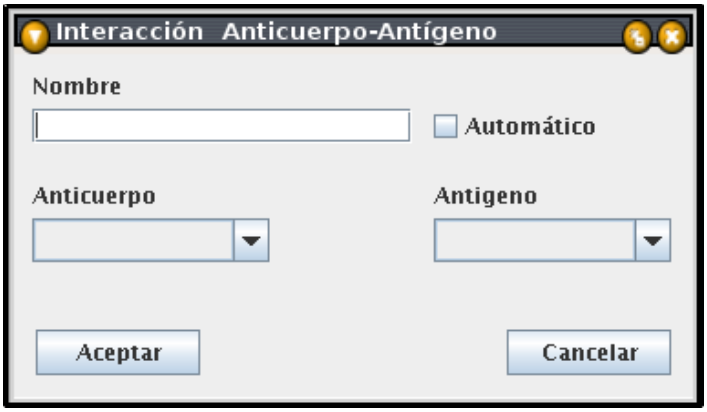
Anexo 1.9 Prototipo de Interfaz Caso de Uso Adicionar Anticuerpo



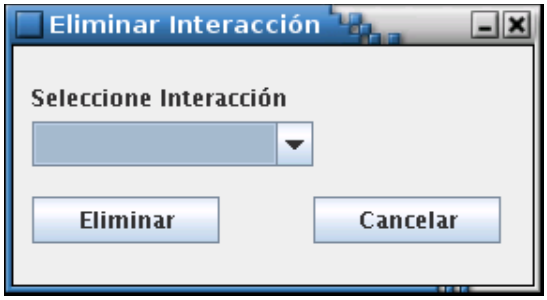
Anexo 1.10 Prototipo de Interfaz Caso de Uso Adicionar Interacción Célula-Célula



Anexo 1.11 Prototipo de Interfaz Caso de Uso Adicionar Interacción Célula-Antígeno

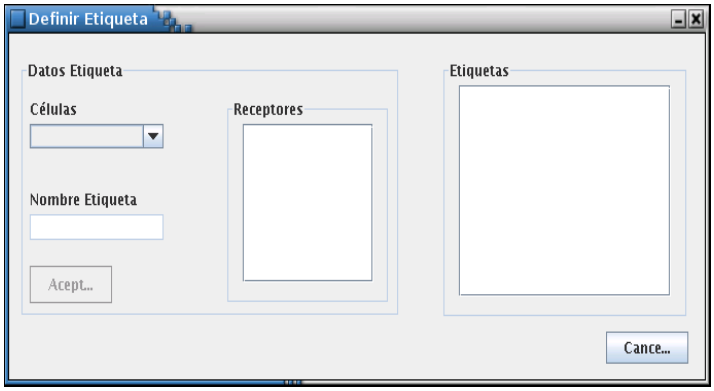


Anexo 1.12 Prototipo de Interfaz Caso de Uso Adicionar Interacción Anticuerpo-Antígeno

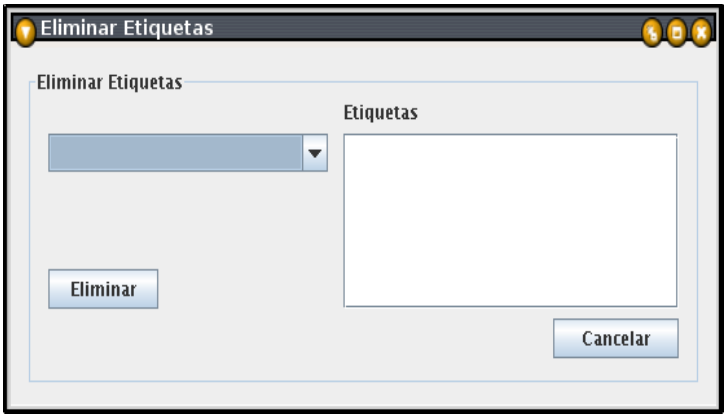


Anexo 1.13 Prototipo de Interfaz Caso de Uso Eliminar Interacción

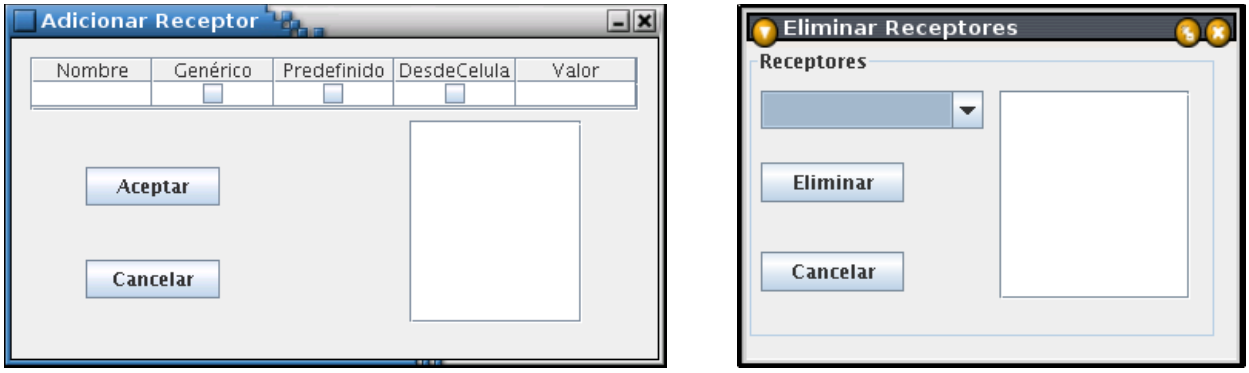
Anexo 1.14 Prototipo de Interfaz Caso de Uso Gestionar Tratamiento



Anexo 1.15 Prototipo de Interfaz Caso de Uso Gestionar Etiqueta (Escenario Definir Etiqueta)



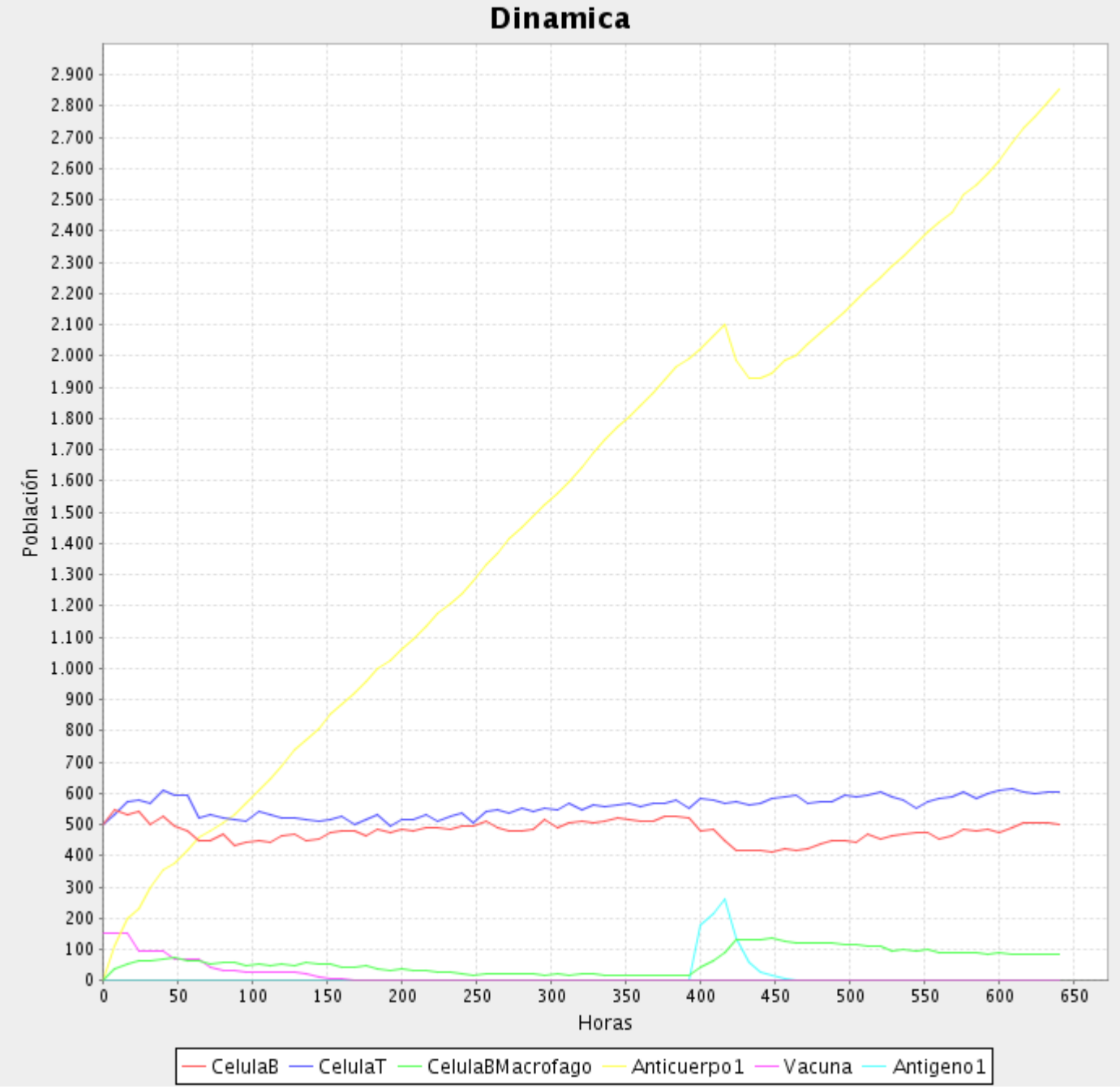
Anexo 1.15 Prototipo de Interfaz Caso de Uso Gestionar Etiqueta (Escenario Eliminar Etiqueta)



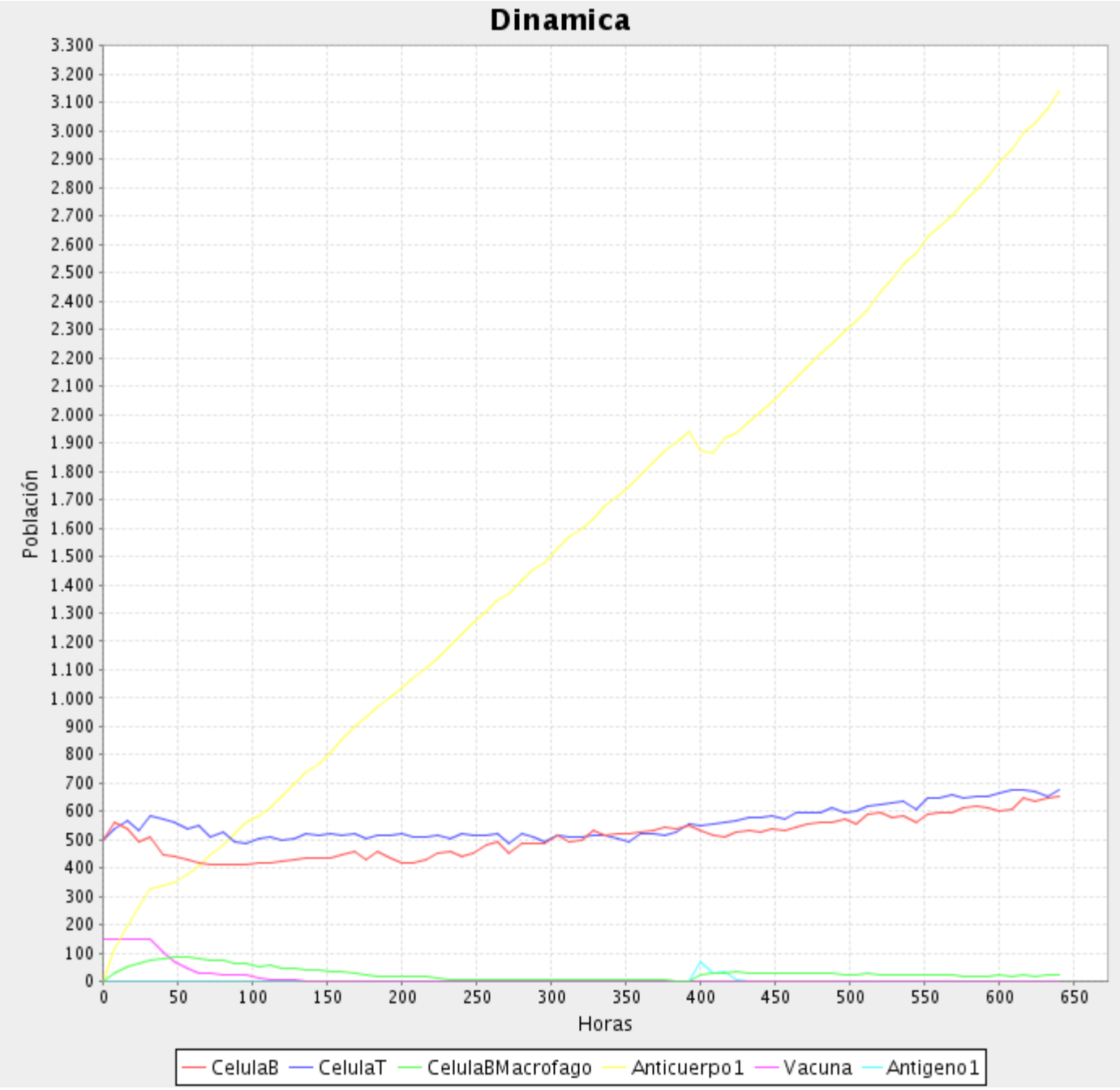
Anexo 1.16 Prototipo de Interfaz Caso de Uso Gestionar Etiqueta

Anexo Prototipo de Interfaz 1.17 Caso de Uso Editar Preferencias

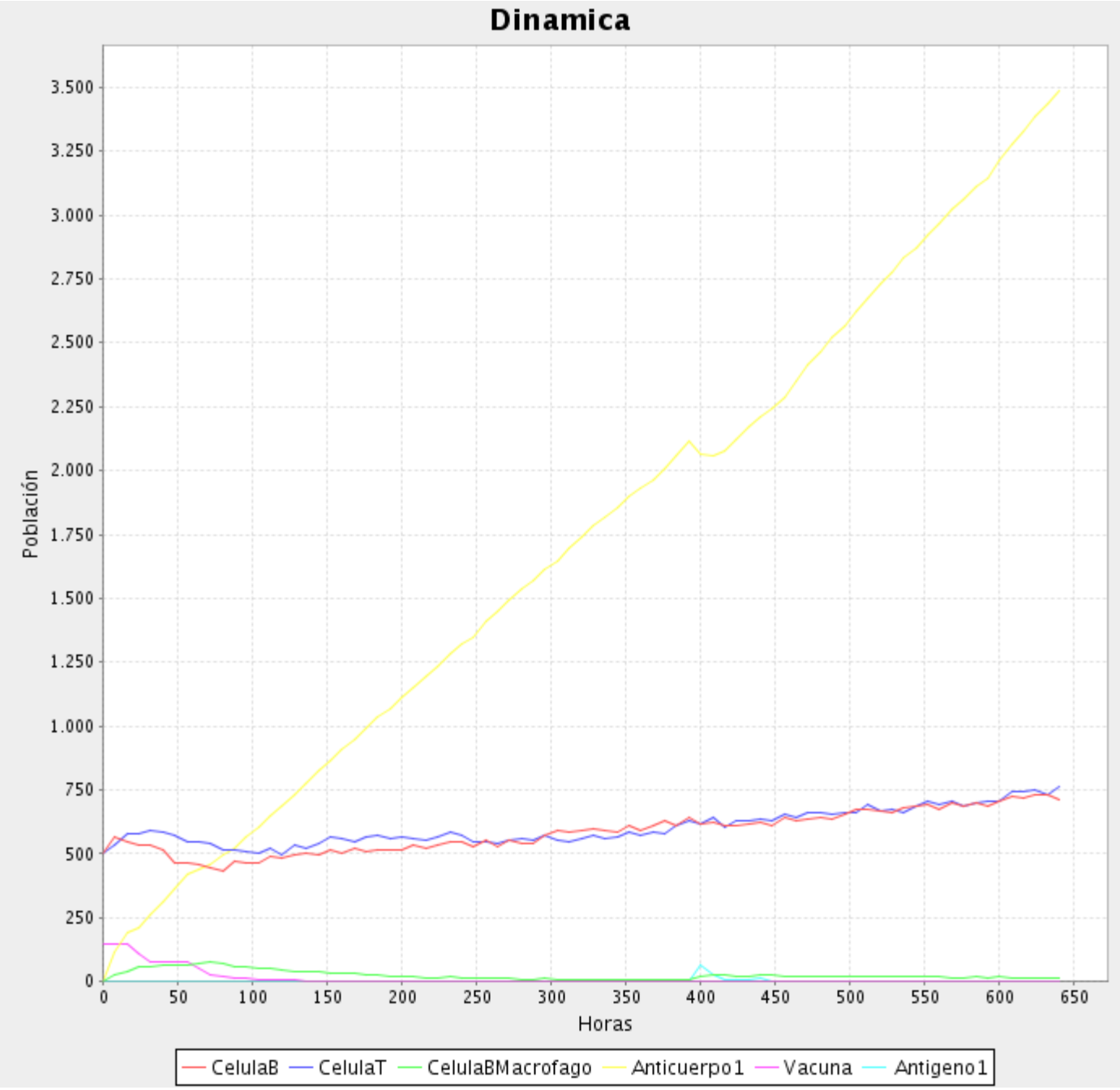
Anexo 2 Gráficas de las 10 simulaciones realizadas para validar J-ImmSim.



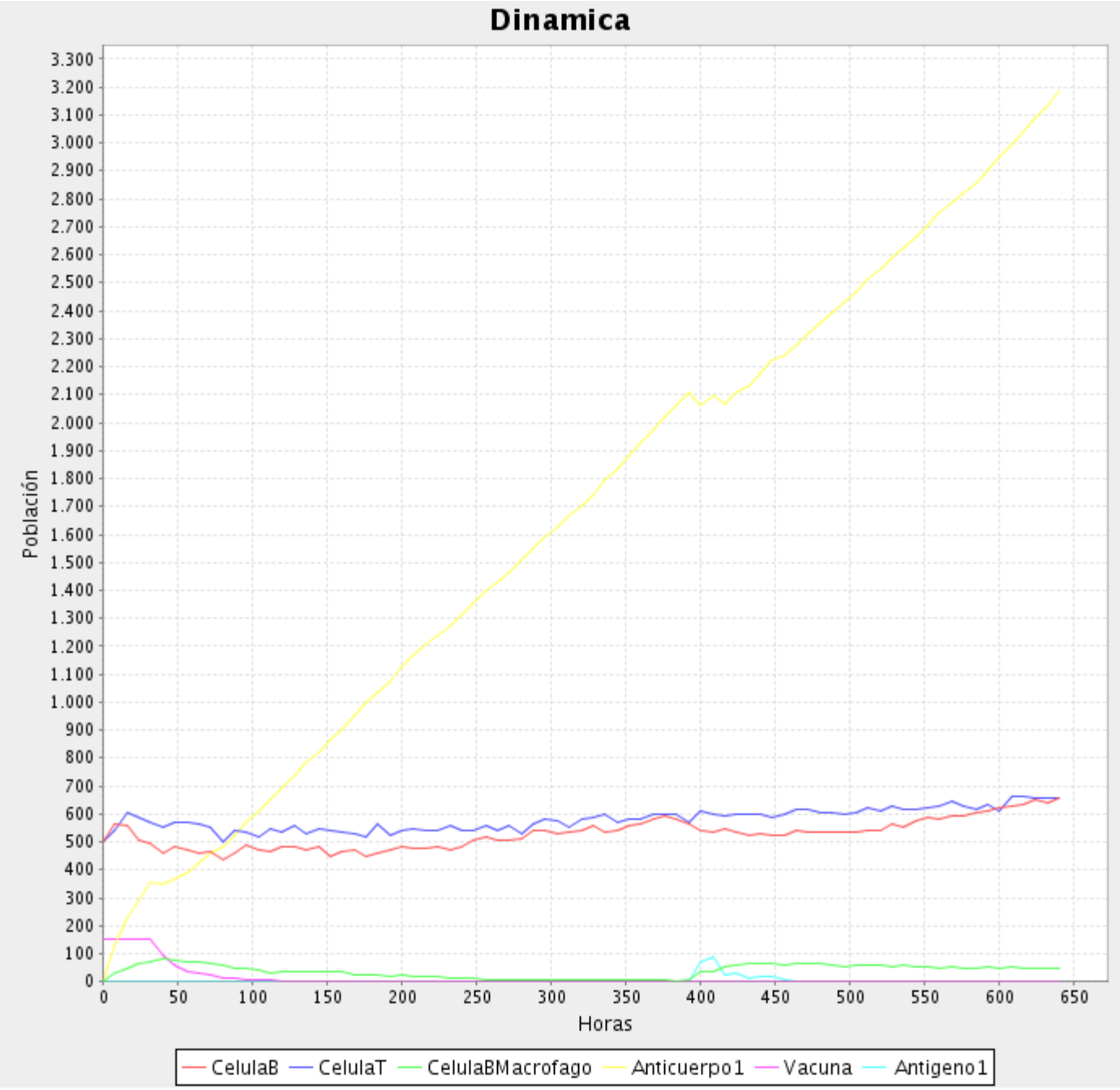
Anexo 2.1 Gráfica Simulación 1



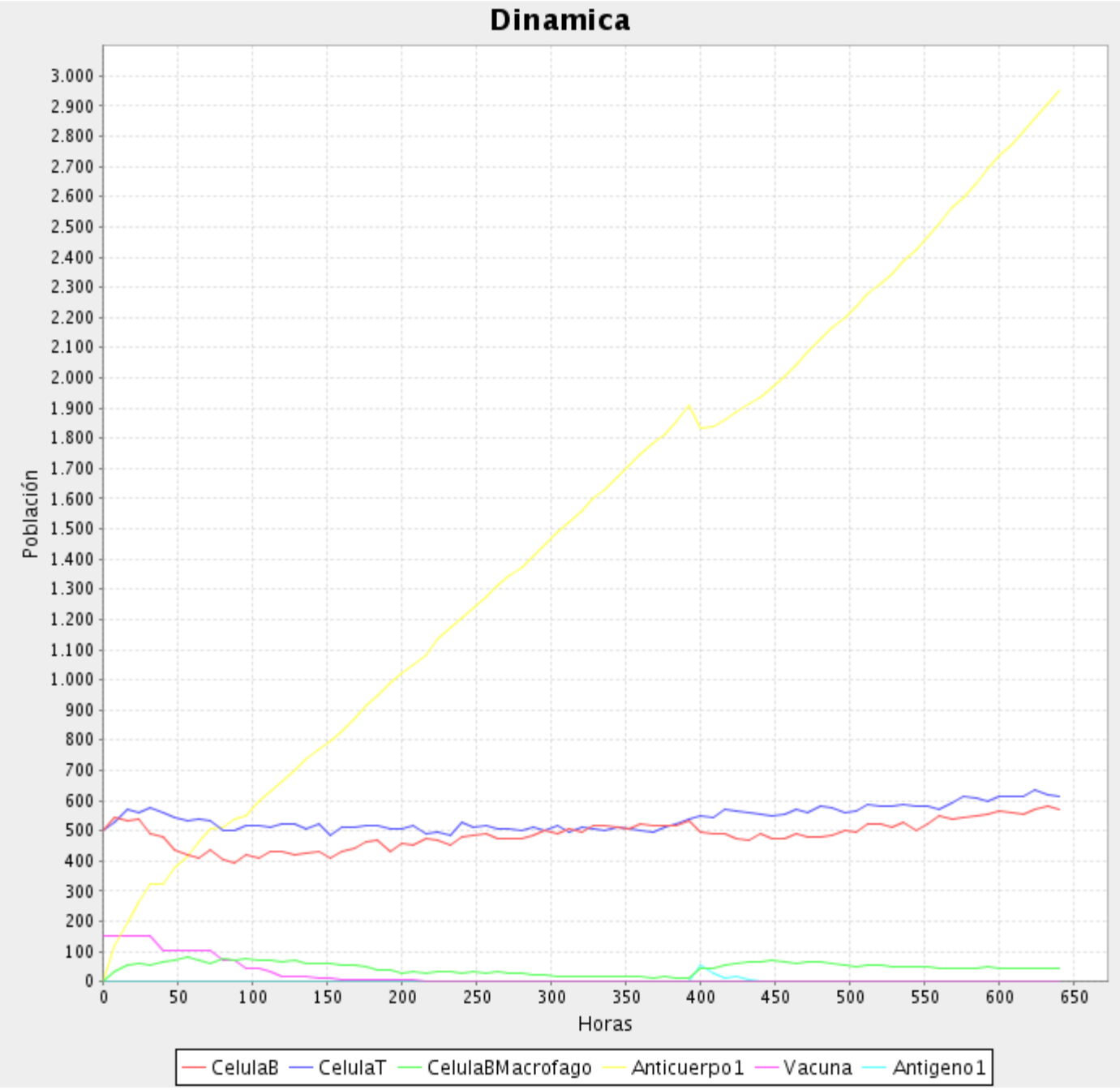
Anexo 2.2 Gráfica Simulación 2



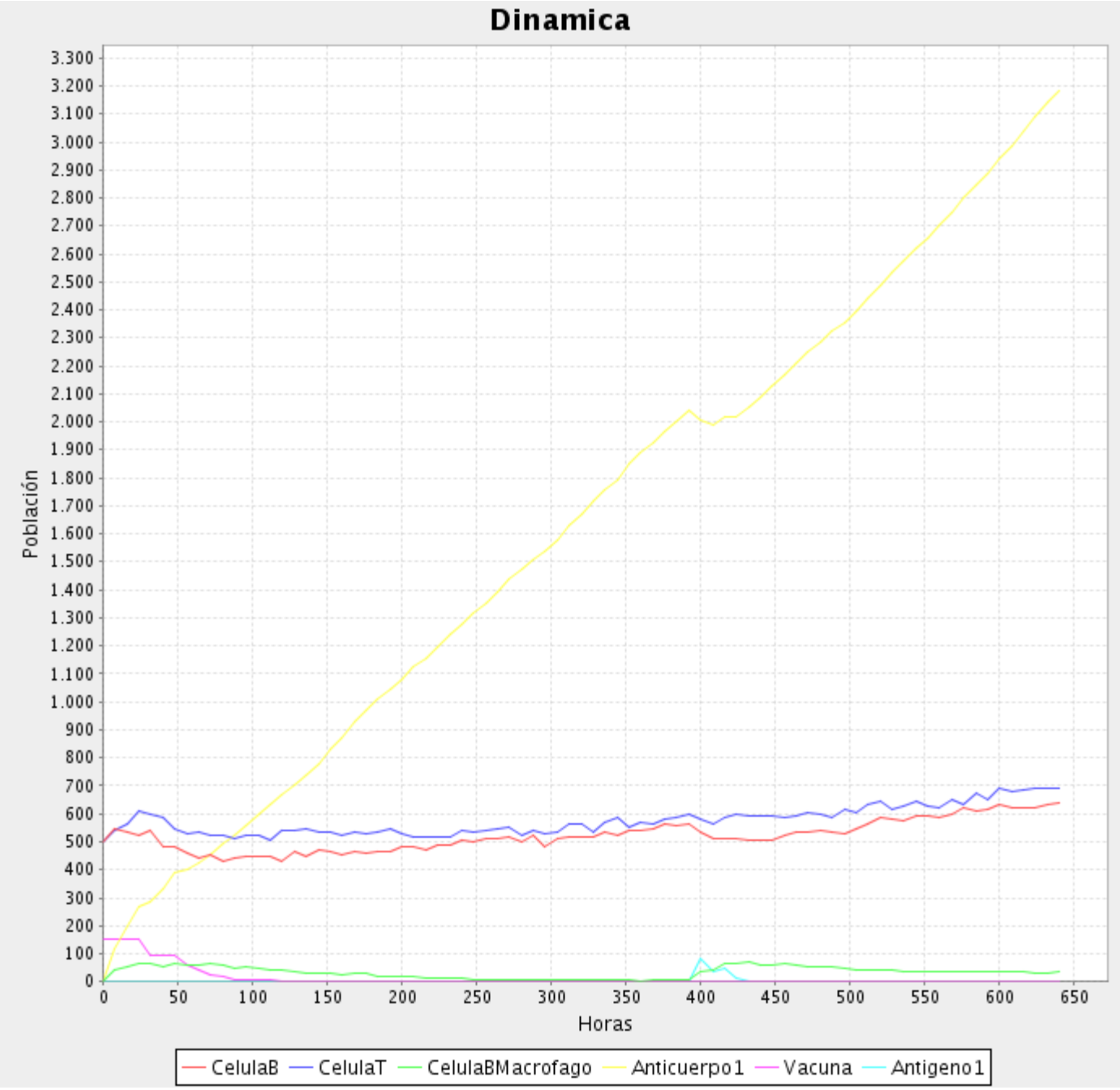
Anexo 2.3 Gráfica Simulación 3



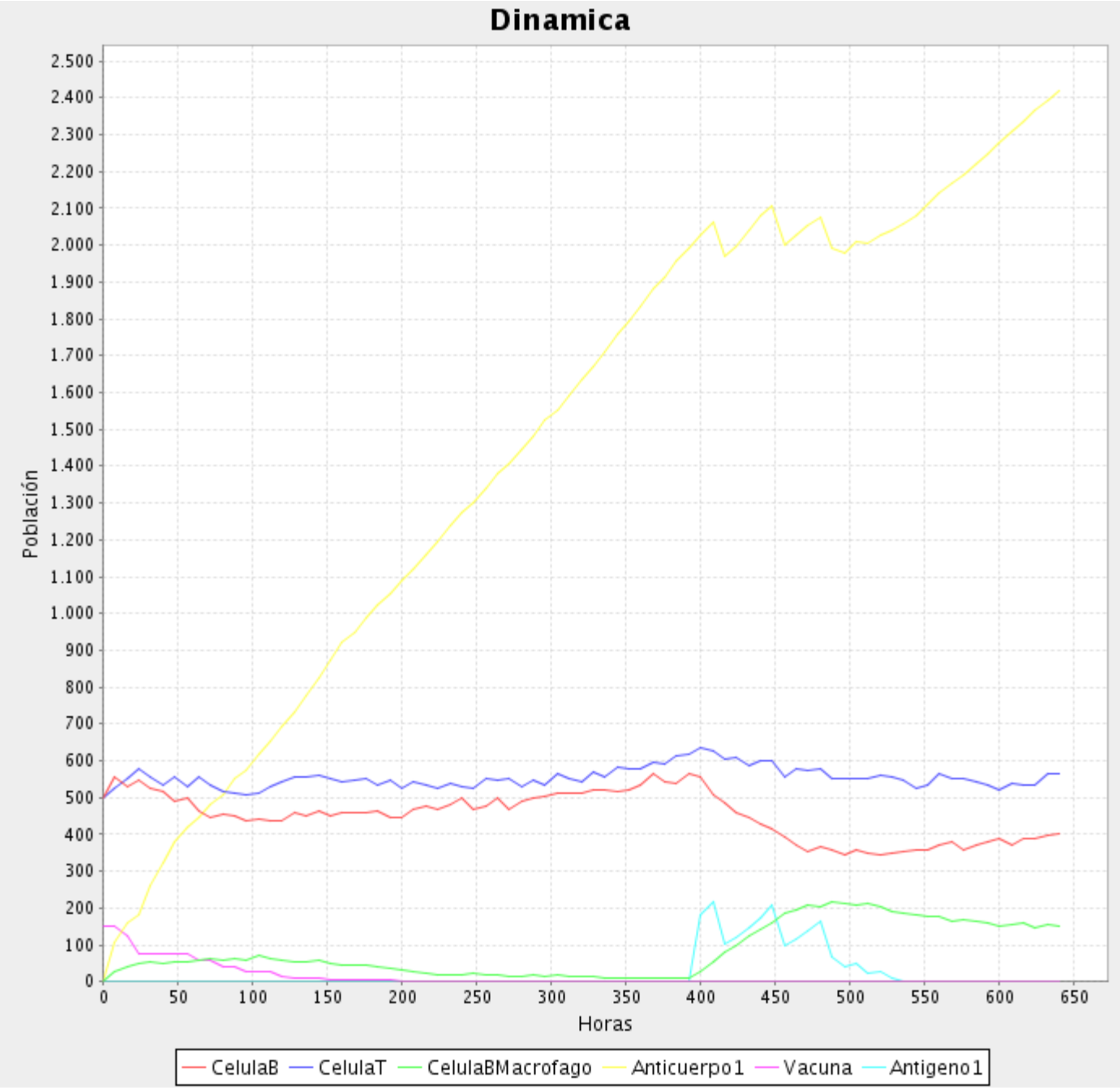
Anexo 2.4 Gráfica Simulación 4



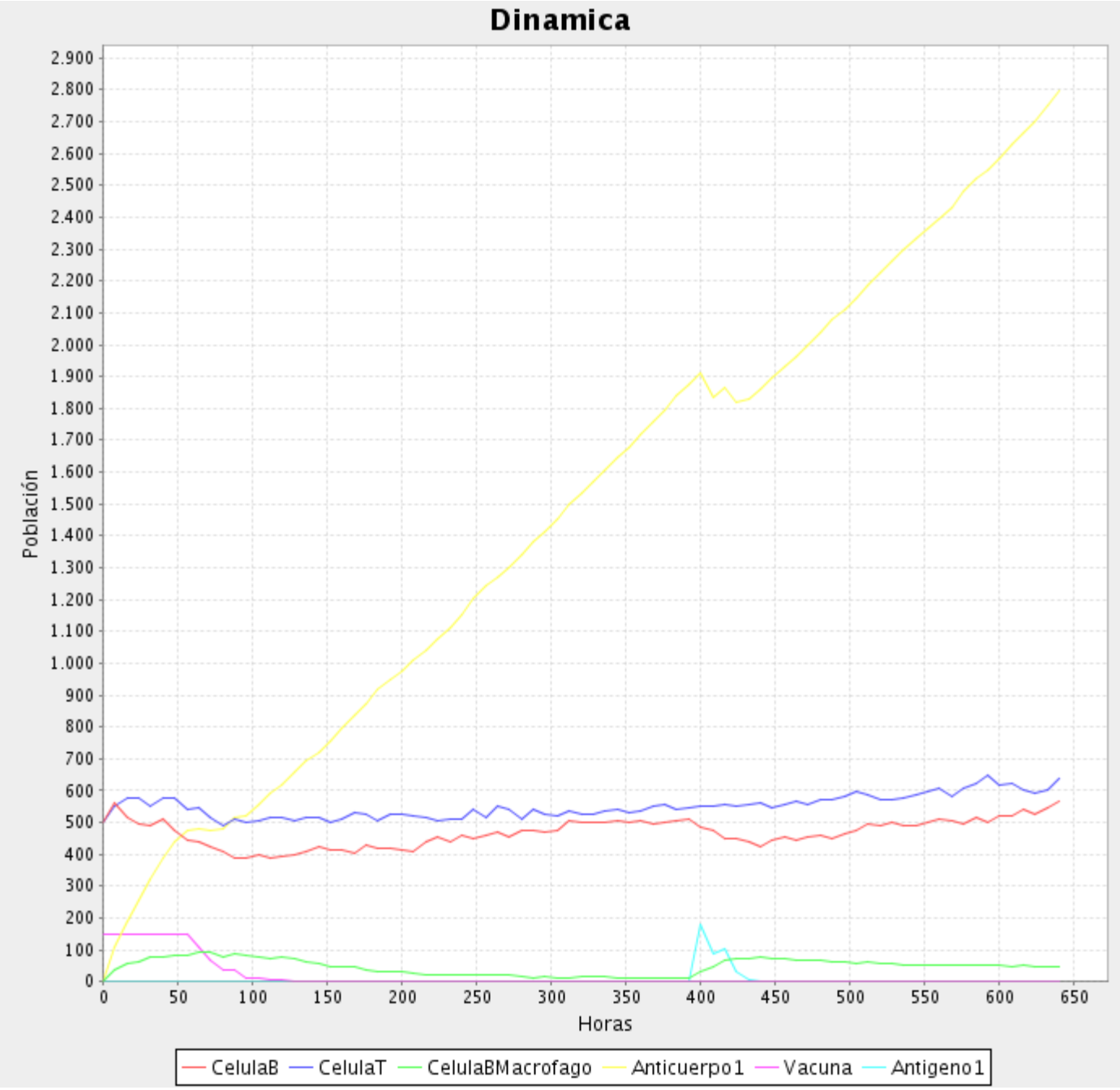
Anexo 2.5 Gráfica Simulación 5



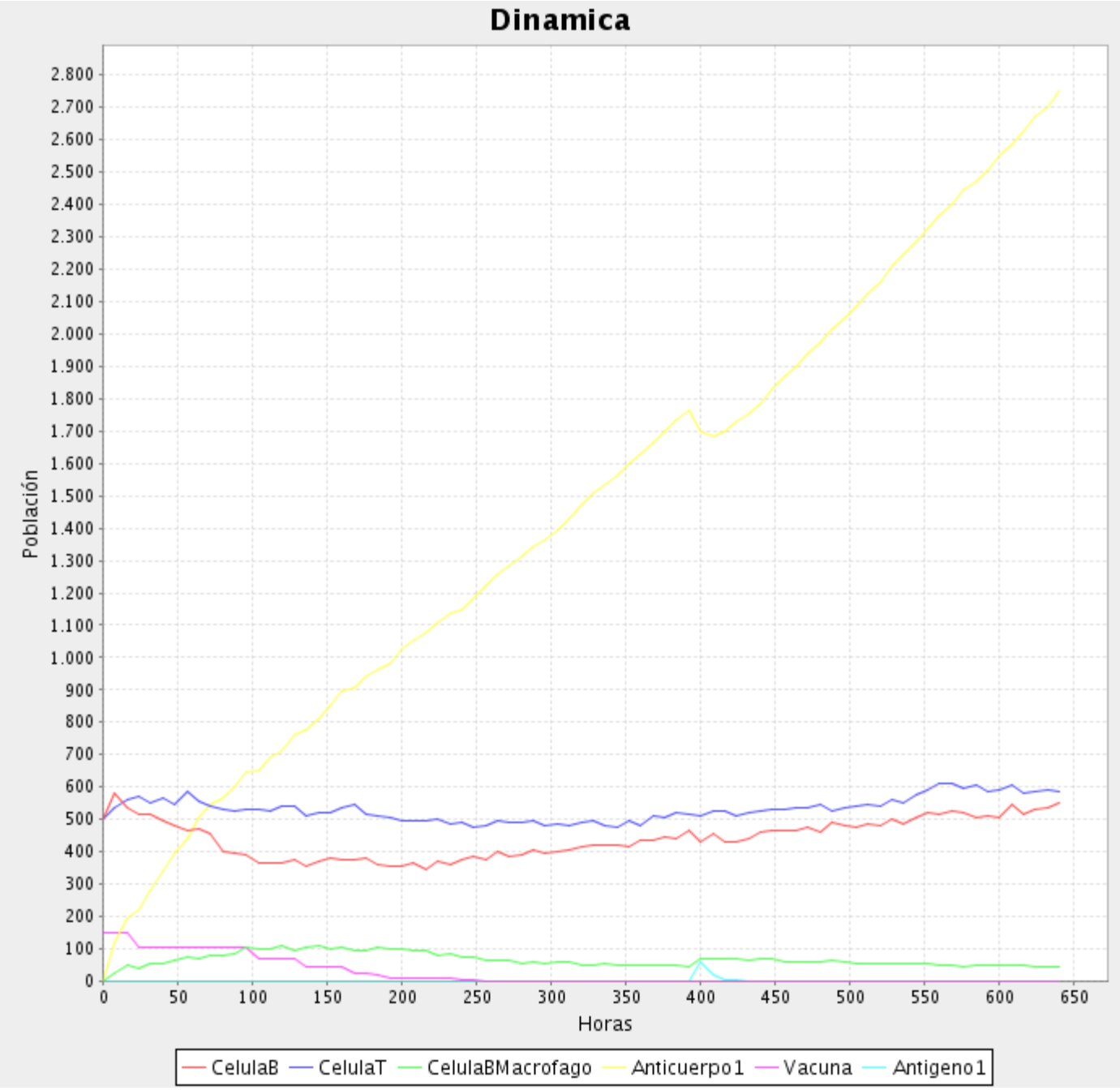
Anexo 2.6 Gráfica Simulación 6



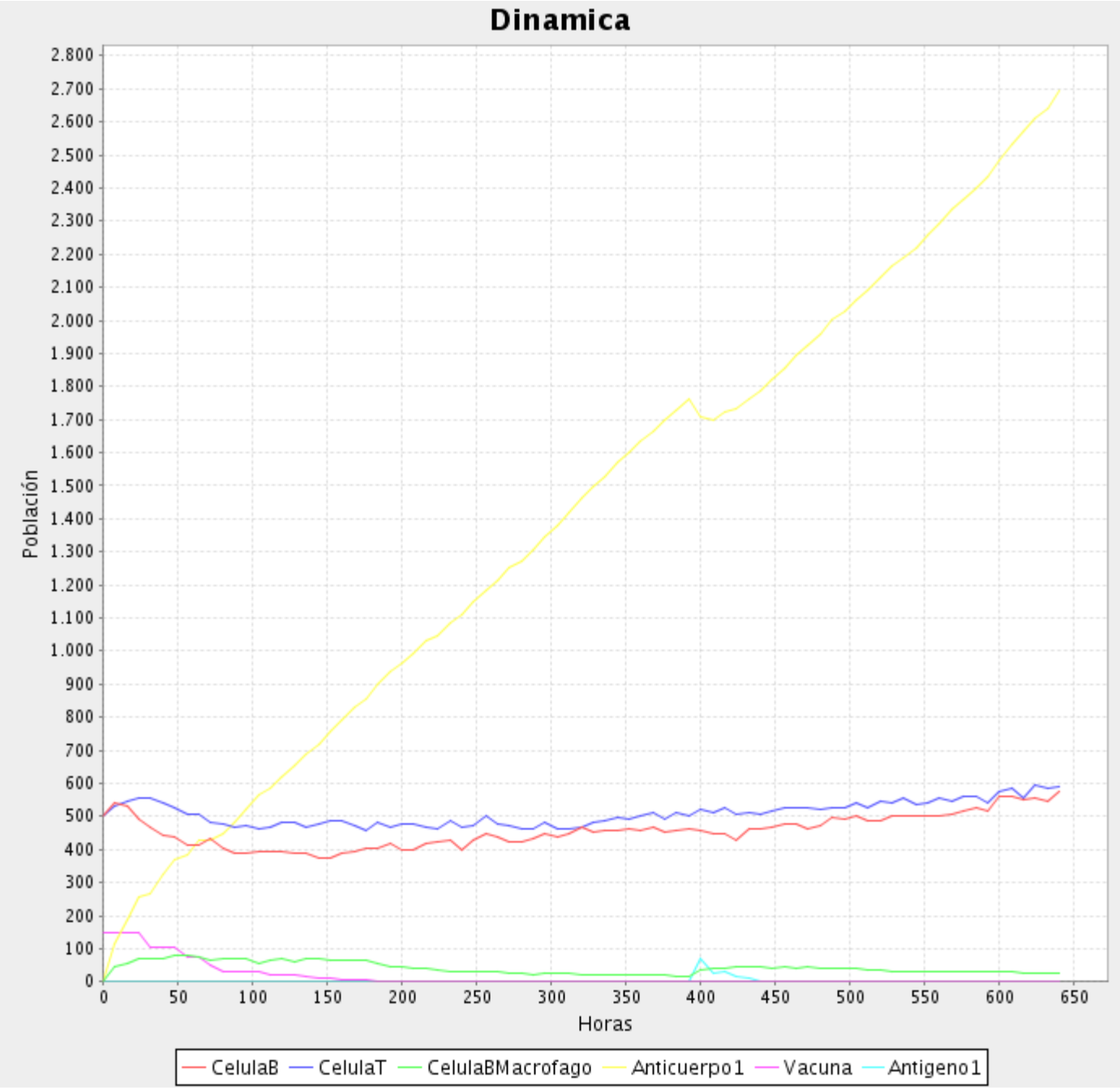
Anexo 2.7 Gráfica Simulación 7



Anexo 2.8 Gráfica Simulación 8



Anexo 2.9 Gráfica Simulación 9



Anexo 2.10 Gráfica Simulación 10

Glosario de Términos

Bazo: Órgano del tamaño de un puño, que se encuentra en la cavidad abdominal.

Célula de Memoria: Célula que ha sido estimulada y posee una vida más prolongada que una célula que no ha sido estimulada.

Epítope: Porción específica de un antígeno macromolecular a la que se une un anticuerpo.

Fagocitosis: Ingestión por parte de una célula de otra célula de menor tamaño, fragmentos u otras partículas.

Médula Ósea: Tejido suave y esponjoso que se encuentra en las cavidades óseas. Su función es producir células.

Péptido: Secuencia de aminoácidos que forma parte de los antígenos

Plugin: Palabra inglesa para referirse a pequeños paquetes de software que añaden nuevas funcionalidades a un juego o programa

Respuesta Inmune: Es la reacción natural del cuerpo contra objetos u organismos extraños, tales como virus, bacterias, etc.

Sistema Inmune: Es una red compleja y vital de células y órganos que protegen al cuerpo de la invasión de agentes extraños al organismo.