Universidad de las Ciencias Informáticas

Facultad 6

Título: Base de Datos del Simulador de Sistemas Biológicos. BioSyS

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

Autores: Irais Mogena Tamayo

Michel Peña Sedeño

Tutoras: Lic.Nara Lidia Pérez Solá

Ing. Yudelkis *Abad Fuentes*

Co-Tutora: Lic.Karelia Urtate

Julio, 2008

Año 50 de la Revolución

"Si una persona es perseverante, aunque sea dura de entendimiento, se hará inteligente; y aunque sea débil se transformará en fuerte."

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 _____.

_____ _____

Irais Mogena Tamayo Michel Peña Sedeño

AGRADECIMIENTOS

Damos las gracias a Nuestro Comandante en Jefe por permitirnos participar en este hermoso programa. A todos los que de una manera u otra colaboraron con nuestra formación, a nuestros padres porque sin ellos no fuésemos quienes somos hoy, a nuestros amigos que nos han apoyado cuando más lo hemos necesitado con una sonrisa. A Adrián y Dayana, por su colaboración. A la Revolución gracias por convertirnos en jóvenes de futuro.

DEDICATORIA

A mi madre, mi tía María Elena y mi abuela, por tener fe y confiar en mí.

A mis primos Ramón y Tony, que hicieron esta carrera junto conmigo.

A mi nueva familia, gracias por existir.

A mis amigas de la universidad Elsia, Mariagne y Yelenis, por estar ahí siempre para mí.

A mi mejor amigo Lázaro Canova, por tus consejos y por llevarme siempre por el buen camino.

A mis tutores, a Nara por su apoyo incondicional, aún estando lejos y a Yudelkis que siempre estuvo ahí y nos dio el frente con todo su cariño.

A Anthony por aceptarme como soy y por todo el amor que me dio cuando más lo necesitaba.

RESUMEN

Desde la aparición del ordenador en la década de los 50 su aplicación al estudio de los seres vivos ha revolucionado la Biología, surgiendo así una nueva disciplina - la Bioinformática - en la que la aplicación del ordenador al análisis, modelado y simulación de las estructuras, ha propiciado un gran avance. Uno de los principales problemas es la no existencia en la actualidad de una aplicación repositorio para almacenar la información sobre estudios y simulaciones de sistemas biológicos. En la Universidad de las Ciencias Informáticas se realizó una base de datos que almacenaba todo lo relacionado con los estudios y simulaciones de los sistemas biológicos, para una primera versión de BioSyS, pero esta presenta algunas limitaciones que impiden el estudio de las simulaciones de sistemas biológicos pues no almacena nada relacionado con la Estimación de Parámetros, ni los Métodos Numéricos con los que se realizan las simulaciones , por lo cual nos vimos en la necesidad de rediseñar la base de datos ya existente, la cual incluirá estás dos nuevas funcionalidades. Es una base de datos dinámica, donde la información almacenada se modifica en tiempo de ejecución, permitiendo operaciones como actualización y adición de datos, además de las operaciones fundamentales de consulta.

PALABRAS CLAVE: Almacenamiento, Simulaciones, Base de datos dinámica.

Índice

Índice de Figuras:

Tabla de Ilustraciones:

INTRODUCCIÓN

Hoy en día, la Biotecnología se mueve en muchos y diversos campos. Las nuevas tecnologías para el tratamiento de la Información Genética (biochips, sistemas LIMS, bases de datos genómicas, sistemas de minería de datos, técnicas de cuantificación de la expresión génica) están revolucionando la investigación biomédica y dentro de esta, se encuentra la Biología de Sistemas; área de investigación científica que permite abordar los objetos y fenómenos como un todo.

La Biología de Sistemas es una ciencia que comenzó a desarrollarse en los años sesenta del siglo XXI, pero su institucionalización académica no se produjo hasta el año 2000. En esta, a diferencia de los métodos clásicos de estudio usados por los biólogos (que se basan en la confirmación o refutación de hipótesis en busca de nuevos resultados experimentales), la biología de sistemas emplea fundamentalmente la modelación: técnica surgida fundamentalmente del uso de modelos matemáticos que describen el comportamiento del ente en estudio. Los modelos permiten predecir el comportamiento del proceso como un sistema dinámico, generalmente tratado como una red compleja.

En la Biología de Sistemas no sólo se necesita de un mejor conocimiento de la Bioquímica, o de rápidas y potentes herramientas de cálculo masivo, sino que implica una metodología científica susceptible, aplicada no sólo por personas, sino también por las máquinas utilizadas. De esta manera, el desarrollo en modelaciones, teoría de la decisión y formas de realizar análisis, son fundamentales a la hora de investigar. Otro tema a la hora de la investigación lo constituye el desarrollo de aplicaciones y sistemas computacionales que revierten estas informaciones en soluciones. [1]

Todo esto refleja una nueva revolución en el campo de las ciencias biológicas, pero con repercusiones en otras disciplinas como la informática, las matemáticas y la física, y en particular la física de los sistemas complejos, acontecimientos que han dado lugar al surgimiento de la Bioinformática, ciencia emergente que abarca los aspectos esenciales a tener en cuenta a la hora de realizar análisis a los sistemas biológicos y muy asociada a su vez, con los procesos informáticos que garantizan el almacenamiento de datos.

La complejidad que entrañan las investigaciones en esta área han hecho necesario: el uso de computadoras y desarrollo de múltiples aplicaciones. Hasta hace poco tiempo, este desarrollo no llegaba a todos los investigadores, pues no existían software fáciles de usar, que permitieran modelar y analizar sistemas biológicos. En la actualidad se han dado grandes pasos en el análisis y diseminación de datos médicos, teniendo como reto principal ofrecer una respuesta a la avalancha de datos

procedentes de dichos análisis.

En la actualidad existen más de cien software basados en estándares que permiten modelar, simular o realizar análisis sobre modelos de sistemas biológicos, [2] estos software experimentales han permitido obtener datos genéticos a gran velocidad; así como, el desarrollo de la Internet y la World Wide Web (www) han permitido el acceso mundial a las bases de datos de información biológica.

A pesar de este actual desarrollo, aún en el mundo existen algunas limitantes, una de estas la constituye la no existencia de aplicaciones para el almacenamiento de estudios o simulaciones de sistemas biológicos. Lo que quiere decir que aunque existan grandes repositorios de modelos, si un investigador quiere saber como funcionan dicho sistema, tiene que repetir el trabajo que ya otros han realizado. [2]

En Cuba existen centros que se dedican a darle continuidad a estos estudios, donde se destaca el Centro de Investigaciones Biológicas, el Centro de Ingeniería Genética y Biotecnología (CIGB), Centro de Producción de Animales de Laboratorio (CENPALAB), Centro Nacional de Biopreparados (BIOCEN), Centro de Inmunoensayo y el Centro de Inmunología Molecular (CIM).

En estos centros se realizan simulaciones distribuidas de los sistemas biológicos modelados mediante Sistema de Ecuaciones Diferenciales (SED), para conocer la respuesta temporal del sistema, a partir de un conjunto de condiciones iníciales y una entrada de datos dada, lo que no existen son bases de datos capaces de almacenar toda esta información para así poder manipularla.

Otra problemática en estos centros, es la escasez de procesos de automatización, debido a que tienen un alto costo en el mercado internacional; y la carencia de aplicaciones, por su no comercialización o porque no cumplen con los requerimientos y necesidades de los investigadores

Por todo lo explicado anteriormente se plantea como **Problema Científico:** *¿Cómo gestionar la información generada en los estudios realizados a modelos matemáticos de sistemas biológicos con la versión BioSyS 2.0?*

Por consiguiente el **Objeto de Estudio** es el Proceso de Almacenamiento de información de sistemas Biológicos.

Y el **Campo de Acción** es una Base de Datos para almacenar información de sistemas Biológicos.

En estos momentos existe una Bases de Datos creada en la primera versión de BioSyS 1.0, la cual presentan algunas limitaciones, lo que hace que el estudio de los sistemas biológicos se haga un poco más lento, para los investigadores. Estas aplicaciones han sido diseñadas para el análisis, inserción, almacenamiento y gestión de la información.

En correspondencia con el problema planteado, el **Objetivo General** consiste en Desarrollar una Base de Datos para almacenar la información generada en los estudios realizados a los Sistemas Biológicos.

Siendo los **Objetivos específicos:**

- ✓ Realizar el análisis del módulo de Base de Datos.
- ✓ Diseñar una Base de Datos.
- ✓ Implementar el módulo de BD.
- ✓ Realizar pruebas que validen el correcto funcionamiento de la aplicación.

Para lograr cumplir con estos objetivos se trazaron algunas **Tareas**, las cuales son:

1. Revisión bibliográfica sobre sistemas biológicos.,
2. Análisis de herramientas para diseñar BD.
3. Análisis de los principales gestores de BD.
4. Levantamiento de Requisitos.
5. Diseño del diagrama de clases persistentes, diagrama Entidad-Relación y modelo de datos.
6. Diseño de una Base de Datos que almacene toda la información referente a los sistemas biológicos.
7. Implementación de los procedimientos almacenados para administrar la información almacenada.
8. Implementación de la capa de Acceso a Datos.
9. Diseño de posibles pruebas.
10. Realización de pruebas al sistema.

Aportes prácticos esperados del trabajo:

Con este trabajo, esperamos facilitarles a los investigadores una aplicación eficiente. Brindando la

posibilidad de manejar gran cantidad de datos, modificarlos, compararlos, y administrarlos. Uno de los aportes más importantes, es que permitirá realizar el análisis de las simulaciones obtenidas de forma automatizada.

El documento está estructurado en tres capítulos, abarcando todo el período de estudio, implementación y prueba.

En el Capítulo 1, se ofrecen algunos conceptos importantes acerca de la Biología de Sistemas, sus aplicaciones, además de que se describen todas las metodologías y herramientas estudiadas y utilizadas en la aplicación.

En el Capítulo 2, tratamos todo lo relacionado con el objeto de estudio y las características que deberá tener el sistema. Se encuentran los requisitos funcionales, que son las características funcionales del sistema, así como los no Funcionales, además de establecer la línea base de la arquitectura y se realizan todos los diagramas.

En el Capítulo 3 se comienza a integrar todos los módulos, para comenzar con la fase de implementación, y posteriormente se le aplicarán pruebas al sistema para comprobar su correcto funcionamiento.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1-Introducción

El volumen de información de sistemas biológicos ubicado actualmente en las bases de datos públicas y los ambiciosos proyectos masivos de estudio sobre la interacción entre proteínas, ha generado un cambio de paradigma, que dio lugar al desarrollo de la Bioinformática. Este desarrollo tecnológico alcanzado, unido al consecuente abaratamiento de información, ha propiciado el desarrollo y perfeccionamiento de técnicas de almacenamiento de datos.

La Bioinformática ofrece la posibilidad de comparar y relacionar la información genética con fines deductivos, [3] así surgen respuestas que no parecen obvias a la vista de los resultados de los experimentos. En los últimos años, se aprecia el crecimiento de una corriente de investigación y desarrollo de nuevas técnicas para la extracción del conocimiento, la minería de datos y la visualización, cuyo objetivo es acelerar los descubrimientos científicos, a partir de la reducción de los costos y el aumento del número de experimentos. [1]

1.2-Biología de Sistemas

La biología de sistemas es un área de investigación científica que se preocupa del estudio de procesos biológicos, usando un enfoque sistémico y la técnica de modelación (técnica que surge fundamentalmente del uso de modelos matemáticos). A través de los modelos, se puede analizar el comportamiento que tendrá un proceso como un sistema dinámico, generalmente, tratado como una red compleja. La fuerza que motiva el innovador enfoque de "sistemas" es la integración de la biología, la computación y la tecnología. Este enfoque permite a los científicos analizar todos los elementos en un sistema en lugar de un gen o proteína por vez. [4] (Ver anexo 1).

1.2.1-Conceptos de Biología de Sistemas (BS)

El término “biología de sistemas”, cuyos orígenes se sitúan en el siglo XIX en relación a los conocimientos sobre embriología y análisis matemáticos de redes, y luego de muchos años de estudio, han surgido innumerables conceptos de lo que es la Biología de Sistemas, alguno de ellos son:

“..La biología de sistemas implica el análisis de todos los componentes del sistema biológico, que incluye un análisis profundo de cómo se expresan los genes y sus interacciones complejas dentro de una célula, tejido o todo el organismo...” [4]

“..La biología de sistemas surge de la convergencia de dos líneas de investigación en biología molecular: una primera que se fundamenta en los trabajos seminales sobre la naturaleza del material genético, la caracterización estructural de las macromoléculas y los subsiguientes avances en tecnologías recombinantes y de alto poder de resolución; la segunda, que supone una base más alejada de la tradicional biología molecular pero no menos robusta para el concepto de biología de sistemas, tiene sus raíces en la teoría de la termodinámica de los procesos irreversibles (lejos del equilibrio), en la resolución de las rutas bioquímicas y el reconocimiento de los retrocontroles en los organismos unicelulares, así como en el creciente reconocimiento de la existencia de redes en biología. ..” [5]

Otro concepto interesante: “rama que pretende integrar diferentes niveles de información con el objetivo de comprender cómo funcionan los sistemas biológicos. Intenta crear modelos comprensibles de sistemas, mediante el estudio de las relaciones y las interacciones entre las diferentes partes de un sistema biológico” [6]

La investigación en la Biología de Sistemas, ha ido evolucionando de manera creciente. Los avances experimentales, han ido en aumento, lo cual ha transformado a esta rama de la biología, en una disciplina rica en datos, los cuales pueden simularse y así descifrar diversos mecanismos complejos. [2]

1.2.2-Simulación de Sistemas

El proceso de simulación de sistemas es la experimentación con un conjunto de hipótesis de sistemas y herramientas computacionales, con los cuáles se desea simular, es necesario realizar primeramente, un análisis previo a éste, con el fin de determinar la interacción con otros sistemas, sus restricciones, variables y sus interrelaciones y, estudiar los resultados que se esperan obtener de esta simulación.

La simulación por computadora, como el término lo indica, con esta técnica, usted diseña y construye un modelo de computadora que imita el argumento real del problema. [7]

¨La simulación es el diseñar y desarrollar un modelo computarizado de un sistema y conducirse experimentalmente con este modelo, con el objetivo de entender el comportamiento de los sistemas en el mundo real o evaluar varias estrategias con los cuales puedan operar el sistema.¨

1.3- Bases de datos

La informática ha sido de gran ayuda para los científicos, ya que se ha convertido en una herramienta indispensable en su trabajo. En este sentido, los investigadores necesitan de herramientas que le permitan almacenar e interactuar con el trabajo realizado, así como, realizar estudios posteriores con la información almacenada, lo que indujo al surgimiento y desarrollo de bases de datos.

Hoy en día, según la variabilidad de los datos, pueden encontrarse bases de datos estáticas y dinámicas. Las primeras son bases de datos de sólo lectura, utilizadas primordialmente para almacenar datos históricos (estáticos) que posteriormente se pueden utilizar para estudiar el comportamiento de un conjunto de datos a través del tiempo, realizar proyecciones y tomar decisiones. En cambio, en las bases de datos dinámicas, la información almacenada se modifica en el tiempo de ejecución, permitiendo operaciones como actualización y adición de datos, además de las operaciones fundamentales de consulta.

Para la presente investigación se parte de una base de datos, ya existente, para una primera versión de BioSyS 1.0, la cual presenta algunas limitantes que imposibilitan el estudio de las simulaciones, ya que no almacena nada relacionado con la Estimación de Parámetros, ni los Métodos Numéricos utilizados para realizar las simulaciones, por tanto se desea diseñar una nueva base de datos dinámica que incluya estas dos nuevas funcionalidades y que presente una mejor gestión de la información. (Ver anexo 2)

1.4- Herramientas y Metodologías

Para lograr los objetivos propuestos**,** se debe hacer uso de la metodología y las herramientas adecuadas. En este epígrafe se explicará cada una de las tecnologías seleccionadas y la metodología utilizada.

1.4.1- Metodologías de desarrollo de software

El ingeniero informático a la hora de desarrollar una aplicación, se encuentra ante enormes disyuntivas, una de las principales es, que herramienta utilizar para lograr el producto informático deseado, indicando además qué personas deben participar en el desarrollo de las actividades y qué papel deben cumplir; definir los artefactos necesarios y además, detallar la información del producto que se obtiene como resultado de la actividad realizada. Actualmente, existen diferentes metodologías

y técnicas para el desarrollo de productos las que son analizadas y definidas en la arquitectura del proyecto, quedando como propuesta final la metodología OpenUP, a la cuál se le realizaron algunas adaptaciones debido a que no presenta el rol de desarrollador de bases de datos.

Unas de las bondades de OpenUP es la fácil adaptación de roles y artefactos, permitiéndole al desarrollador incorporar nuevas plantillas, lo que permitió adaptarla a la metodología propuesta por la Lic. Rosa María Mato García; dicha metodología es dirigida específicamente para desarrolladores de bases de datos (define los artefactos y planillas necesarias para desarrollar un buen diseño de bases de datos).

1.4.4.1- OpenUP

OpenUP/Basic 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. Permite un abordaje ágil al software en análisis con sólo proveer un conjunto simplificado de contenidos, fundamentalmente relacionados con orientación, productos de trabajo, roles, y tareas. Es un proceso interactivo de desarrollo de software simplificado, completo y extensible; 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. [8]

Aprovechando estas características se definió una especialización para el rol de desarrollador o implementador, incorporándole una nueva plantilla con los artefactos necesarios. [3]

1.4.2- Lenguaje de Modelado

1.4.2.1- Lenguaje Unificado de Modelado (UML)

Es un lenguaje gráfico para visualizar y documentar los elementos de los sistemas orientados a objetos. Permite modelar, visualizar, especificar y construir los artefactos necesarios. Al no ser un método de desarrollo, es independiente del ciclo de desarrollo que vayas a seguir, puede encajar en un tradicional ciclo en cascada, o en un evolutivo ciclo en espiral o incluso en los métodos ágiles de desarrollo, soportando tanto el modelo lógico como el físico. [4]

UML es de reciente aparición y, al ser no propietario, es usado y refinado por muchas empresas, grupos de investigadores y desarrolladores a nivel mundial. [9]

1.4.3-Herramientas CASE

Las herramientas CASE son la mejor base para el proceso de análisis y desarrollo de software. Estas constituyen 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.

Para el diseño de la base de datos se hizo necesaria la utilización de dos herramientas case para el modelado de los artefactos a obtener:

- Visual PARADIGM con la problemática de que no exporta para PostgreSQL.
- DBDesigner Fork para diseñar el modelo de datos y poder exportar para el PostgreSQL.

1.4.3.1-Visual PARADIGM

Es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño, orientados a objetos, construcción, pruebas y despliegue. Facilita la construcción de aplicaciones de calidad mejores y a un menor costo. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. (10) Proporciona abundantes tutoriales de UML, demostraciones interactivas y proyectos. Se integra con las siguientes herramientas: Java, NetBeans/IDE y Oracle, lo que permite generar objetos Java desde la base de datos, así como realizar transformaciones de diagramas de Entidad-Relación en tablas de bases de datos. [5]

1.4.3.2-DBDesigner Fork

Es un diseñador de bases de datos potente para usar en Linux, el cual no es más que un DBDesigner mejorado, mejor diseñado y más liviano que el original. [11] Su mejora consiste en un parce que permite generar scripts para el PostgreSQL.

Este programa de diseño de bases de datos, combina características y funciones profesionales de fácil uso, a fin de ofrecer un método efectivo para gestionar bases de datos. Permite además, administrar la base de datos, diseñar tablas, hacer peticiones SQL manuales y mucho más. Dispone de una interfaz profesional y de detallados manuales de uso. Es multiplataforma y facilita la creación de modelos de datos, aunque presenta algunas desventajas, ya que no permite modelar lógicamente.

Está herramienta permite crear y modificar la diferentes tablas diseñadas en el modelo de datos.

Facilita la generación de código y también genera los índices, que se encargan de asegurar la integridad relacional en el modelo de datos. Este programa permite conectarse directamente a bases de datos existentes para hacer operaciones importantes mediante el uso de clausulas como son: el FROM, WHERE, GROUP y ORDER.

Al terminar con el diseño, DBDesigner puede exportar el esquema de la base de datos a un archivo .SQL, o conectarse directamente al servidor de bases de datos y construir allí las tablas correspondientes. También puede importar un archivo .SQL de una base de datos existente. [12]

Dada su arquitectura basada en plugins, es fácilmente extensible para trabajar con diferentes servidores de base de datos. Por defecto se proporcionan dos plugins, uno para PostgreSQL y otro para MySQL. [6] La desventaja existe cuando se desea diseñar el diagrama entidad_ relación, el cual genera directamente el modelo físico sin haber pasado antes por el modelo lógico, sin validar la descripción de la estructura de la base de datos: atributos de las tablas, las relaciones existentes entre ellas, entre otras cuestiones.

1.4.4-Gestores de Bases de Datos

1.4.4.1- PostgreSQL

Se utiliza el PostgreSQL como gestor de base de datos. Siendo un sistema de administración de base de datos objeto-relacional. Es un descendiente de código abierto. Soporta SQL92 y SQL99 y ofrece muchas características modernas, tales como consultas complejas, llaves foráneas, triggers, vistas, integridad transaccional y control de concurrencia multi-versión.

A su vez, puede ser extendido por el usuario en múltiples formas; por ejemplo: agregando nuevos tipos de datos, funciones, operadores, métodos de indexación, funciones de agregación y lenguajes procedurales. Debido a su licencia libre puede ser usado, modificado y distribuido libremente de cargos para cualquier propósito, sea privado, comercial o académico. Es un gestor de base de datos rápido, seguro y fácil de usar.

PostgreSQL provee nativamente soporte para números de precisión arbitraria, texto de largo ilimitado, figuras geométricas y arreglos. [13] Adicionalmente los usuarios pueden crear sus propios tipos de datos, los que pueden ser por completo indexables gracias a la infraestructura GiST que posee. [7]

1.4.4.2- MySQL

Como gestor de bases de datos y por sus características, se utilizó el MySQL. Es un sistema de gestión de bases de datos relacional, licenciado bajo la GPL de la GNU. Su diseño multi-hilo le permite soportar una gran carga de forma muy eficiente. [14]

MySQL es un sistema de administración de bases de datos multiusuario. Ampliamente usado a nivel mundial, en el mundo del software libre, debido a su gran rapidez y facilidad de uso. [14] Es rápido y flexible a bases de datos, ya que tiene incorporado operaciones avanzadas como procedimientos almacenados y triggers. Además de ser un programa de código abierto, fácil de descargar y accesible a cualquiera usuario.

Entre sus principales características se encuentran el aprovechamiento de la potencia de sistemas multiprocesador, gracias a su implementación multi-hilo; el soporte de gran cantidad de tipos de datos para las columnas; la gestión de usuarios y contraseñas, manteniendo un muy buen nivel de seguridad en los datos y gran portabilidad entre sistemas. Una de las bondades de MySQL es que trabaja con múltiples plataformas, incluyendo AIX, BSD, FreeBSD, HP-UX, GNU/Linux, Windows XP, Windows Vista y otras versiones de Windows. [8]

1.4.5-Lenguaje de Marcado

1.4.5.1- Lenguaje de marcas extensible (XML)

XML es un lenguaje de marcado muy simple, es una manera de definir lenguajes para diferentes necesidades. Es muy similar al HTML, pero su función principal es describir datos y no mostrarlos como es el caso de HTML. Las tecnologías XML son un conjunto de módulos que ofrecen servicios útiles a las demandas más frecuentes por parte de los usuarios.

Sirve para estructurar, almacenar e intercambiar información. Además de que juega un papel fundamental en el intercambio de una gran variedad de datos. Tiene como ventajas el ser extensible, lo que quiere decir que una vez diseñado un lenguaje y puesto en práctica, igual es posible extenderlo con la adición de nuevas etiquetas de manera que los antiguos consumidores de la vieja versión todavía puedan entender el nuevo formato. Es sencillo de entender y procesar, y facilita la compatibilidad entre aplicaciones. [9]

1.4.6-Lenguaje de Programación

1.4.6.1-Lenguaje Java

Java es un lenguaje de Programación Orientado a Objetos (POO) que permite organizar el código en entidades como: clases compuestas de datos y funciones, y a través de la característica de la herencia organizar las clases en jerarquías. [15]

La segunda característica es la independencia de plataforma, significando que programas escritos en el lenguaje pueden ejecutarse igualmente en cualquier tipo de hardware, lo que constituye una ventaja significativa para los desarrolladores de software. Fue diseñado para crear software altamente fiable; para ello proporciona numerosas pruebas en compilación y en tiempo de ejecución. Java ofrece toda la funcionalidad de un lenguaje potente, pero sin las características menos usadas y más confusas de éstos. Reduce en un 50% los errores más comunes de programación y elimina la definición de tipos, así como los macros.

Se beneficia todo lo posible de la tecnología orientada a objetos; no intentando conectar todos los módulos que comprendan una aplicación hasta el tiempo de ejecución. Simplifica el uso de protocolos nuevos o actualizados. Si su sistema ejecuta una aplicación Java sobre la red y encuentra una pieza de la aplicación que no sabe manejar, Java es capaz de traer automáticamente cualquiera de esas piezas que el sistema necesita para funcionar. Su seguridad se basa en la implementación de barreras de seguridad y en el sistema de ejecución en tiempo real. [10]

1.4.6.1- Lenguaje de Consulta Estructurado (SQL)

SQL es un potente lenguaje, para organizar, administrar y consultar datos almacenados en una computadora. Una de sus características es el manejo del álgebra y el cálculo relacional, permitiendo lanzar consultas con el fin de recuperar -de una forma sencilla- información de interés de una base de datos, así como también hacer cambios sobre la misma.

Más específicamente SQL esta definido en torno al modelo de bases de datos relacionales, ventaja que lo impone como el sistema de mayor aceptación. Algunas de estas son: [11]

- Marco teórico sólido, fundamentado en el álgebra relacional
- Simplicidad de conceptos (modelo de base de datos: tablas=líneas x columnas)
- Definición de vínculos en la consulta, brindando gran flexibilidad

- Fácil y rápido aprendizaje
- Arquitectura cliente-servidor
- Integración con cualquier lenguaje de programación
- Estandarización

1.4.7- Herramienta de desarrollo

1.4.7.1- NetBeans

Constituye el ambiente idóneo para el desarrollo de aplicaciones de escritorio usando Java y un entorno de desarrollo integrado (IDE). Facilita todas las herramientas necesarias para crear aplicaciones de escritorio profesional, corriendo en muchas plataformas incluso Windows, Linux, el Mac OS X y Solaris. Posee un consumo más bajo de memoria y sensibilidad mientras se trabaja con proyectos grandes.

NetBeans es un producto de código abierto, con todos los beneficios del software disponible en forma gratuita, el cual ha sido examinado por una comunidad de desarrolladores. Este enfoque de bienes comunes creativos ha permitido una mayor capacidad de uso, con cada nueva versión, y ha proporcionado a los desarrolladores mayor flexibilidad, al modificar el IDE, si así lo desean. [16]

En cuanto al desarrollo de bases de datos permite correr las consultas SQL desde él mismo, revisa la sintaxis inmediatamente, y su depurador de errores (debugger) es excelente; además de tener un buscador de ocurrencias en tiempo real. La ingeniería delantera e inversa permite al diseñador de bases de datos diseñar aplicaciones usando el UML, generando el código desde Java, desde el modelo de UML o actualizando el modelo de cambios hechos en el código fuente. [12]

1.4.7.2- Hibernate

Hibernate es una herramienta de mapeo objeto-relacional para la plataforma Java que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación, a través de archivos declarativos (XML) que permiten establecer dichas relaciones. Tiene como objetivo facilitar la persistencia de objetos Java en bases de datos relacionales y al mismo tiempo la consulta de estas bases de datos para obtener objetos.

Busca solucionar el problema de la diferencia entre los dos modelos usados hoy en día para organizar y manipular datos: El usado en la memoria de la computadora (orientación a objetos) y el usado en las

bases de datos (modelo relacional). Para lograr esto, permite al desarrollador detallar cómo es su modelo de datos, qué relaciones existen y qué forma tienen. Con esta información le permite a la aplicación manipular los datos de la base operando sobre objetos, con todas las características de la programación orientada a objetos.

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

Tiene la funcionalidad de crear el diseño de la base de datos a partir de la información disponible, además de que presenta una Interfaz de Programación de Aplicaciones (API) para construir las consultas, permite diseñar objetos persistentes que podrán incluir polimorfismo, relaciones entre clases, colecciones, y un gran número de tipos de datos. De una manera muy rápida y optimizada permite generar diseños de bases de datos en cualquiera de los entornos soportados. [13]

1.4.7.3- Generador de Datos EMS 2005 para MySQL

El generador de Datos EMS 2005 para MySQL es una potente herramienta para generar datos aleatorios a las tablas de la bases de datos. Permite definir tablas y campos para la generación de datos, establecer rangos de valores, obtener listas de valores de consultas SQL y muchas otras características para generar los datos de los ensayos de una forma simple y directa.

Se encuentra verdaderamente ligado al lenguaje SQL, ofreciéndole la capacidad de sincronizar bases de datos creadas en base al lenguaje anteriormente nombrado. Cuenta con un funcionamiento altamente comprensible y dinámico para quien ya conoce y sabe manipular el lenguaje SQL.

El objetivo de uso es realizar pruebas a la base de datos, para así, validar los requisitos funcionales y el correcto funcionamiento del diseño implementado. [14]

1.4.7.4- EMS MySQL Manager

Constituye una potentísima herramienta para gestionar bases de datos en MySQL. Permite un control total y absoluto sobre la misma, permitiendo realizar acciones que otros programas similares son incapaces de prestar. Ofrece versiones tanto para Windows como para Linux. Entre sus características principales están el soporte completo para las versiones entre la 3.23 y la 5.02, rápido manejo de base de datos y navegación, manejo rápido de todos los objetos MySQL, herramientas avanzadas para el

manejo de datos, potentes herramientas de seguridad y gran capacidad para la exportación e importación de bases de datos.

Su interfaz gráfica de usuario permite crear y editar todos los objetos de una base de datos de la manera más simple y sencilla, ejecutar scripts SQL, manejar usuarios y administrar sus privilegios, construir consultas SQL visualmente, extraer e imprimir metadatos, importar y exportar datos, ver y editar imágenes (Campos BLOB), y muchos otros servicios que harán que el trabajo con un servidor MySQL sea una labor verdaderamente fácil. [17]

Permite la manipulación de bases de datos: mediante consultas que permiten verificar el resultado y el tiempo de ejecución de las mismas. [15]

1.5- Sistema Operativo

Ubuntu es un sistema operativo de código abierto desarrollado en torno al kernel de Linux. (18) Se basa principalmente en que el software debe ser gratuito, los usuarios deben poder usar el software en su lengua materna y deben poder hacerlo independientemente de sus limitaciones; además, de las facilidades que brinda para modificar el software del modo que se crea más conveniente.

Basado en Debian, Ubuntu pretende crear una distribución que proporcione un sistema GNU/Linux actualizado y coherente para la informática de escritorio y servidores. Incluye una cuidadosa selección de los paquetes de Debian, para incluir solo aplicaciones importantes y de alta calidad; y mantiene un poderoso sistema de gestión de paquetes que permite instalar y desinstalar programas de una forma fácil y limpia, a diferencia de la mayoría de las distribuciones, que vienen con una enorme cantidad de software que puede o no ser de utilidad. [16]

1.6- Patrón DAO

El objetivo principal del uso del Patrón **O**bjeto de **A**cceso a **D**atos (DAO) es permitir obtener los datos o ser capaz de guardarlos en algún sitio, sin que los investigadores conozcan donde está almacenada esa información o dónde se guardan los mismos.

Este patrón es uno de los patrones de diseño estándar del entorno de desarrollo de Java (J2EE) y una solución al problema del diferencial de impedancia entre un programa de aplicación orientado a objetos y una base de datos relacional. Es adaptable prácticamente a cualquier lenguaje de programación. (Ver anexo 3)

Contiene una clase DAO factoría, una clase interface y una clase concreta que implementa la interface además de objetos de transferencia de datos. Tiene como propósito principal, en pocas palabras, abstraer y encapsular todos los accesos a la fuente de datos; con esto se obtiene un bajo nivel de acoplamiento entre clases, reduciendo la complejidad a la hora de realizar cambios; y aislando las conexiones a la fuente de datos en una capa fácilmente identificable. [17]

1.7- Administrador de Base de Datos

1.7.1- PhpMyAdmin

Programa de libre distribución en PHP (lenguaje de programación para páginas Web). Es una herramienta muy completa que permite acceder a todas las funciones típicas de la base de datos MySQL a través de una interfaz web [19] que presenta un diseño sencillo, atractivo y muy intuitivo. Posee ventajas que lo identifican con respecto a otros de su tipo, ya que es totalmente gratuito (código abierto) y muy completo.

Cuando se hace referencia al PhpMyAdmin, se hace referencia a un conjunto de archivos escritos en PHP que se copian en un directorio de servidor web, de modo que, cuando se accede a estos archivos, muestre las páginas donde se puede encontrar las bases de datos a las que se desea acceder en el servidor y todas sus tablas. La herramienta facilita la creación de tablas, inserción de datos en las tablas existentes, navegar por los registros de las tablas, editarlos y borrarlos, incluso ejecutar sentencias SQL y hacer una salva de la base de datos. [18]

1.7.2- PhpPgAdmin

PhpPgAdmin es una nueva versión de un cliente web de bases de datos en PostgreSQL que necesita para funcionar un servidor web implementado en PHP. Tiene la funcionalidad básica de servir de soporte a procedimientos almacenados, triggers y vistas.

Posibilita administrar varios servidores, soportando múltiples versiones de PostgreSQL, administración de usuarios, grupos, bases de datos, esquemas, etc. Facilita la manipulación de datos, exportando datos a diferentes formatos, importando sentencias SQL y mucho más. Importa SQL scripts, copias de datos y archivos XML. Fácil de instalar y configurar. [19]

1.8- Conclusiones

Para el desarrollo de la aplicación, la metodología empleada fue OpenUP, la cuál se encuentra en fase de experimentación aún, aunque se conoce su fácil adaptación a proyectos pequeños de tres ó seis personas, además de que es aplicable para proyectos de poca duración. El lenguaje a utilizar fue Java, por ser multiplataforma, y la herramienta más factible en este caso fue el NetBeans como IDE, e Hibernate como framework. Como herramienta de modelado de artefactos se empleó el Visual Paradigm y para el diseño del modelo de datos el DBDesigner Fork; como gestor de base de datos, se utilizó el MySQL y el PostgreSQL, respectivamente, por ser sistemas de administración de bases de datos, fuertes, seguros, y además, multiplataforma. Para hacerle pruebas a la base de datos se utilizó el Generador de Datos EMS 2005 para MySQL y el EMS MySQL Manager.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

2.1- Introducción

En el presente capítulo, se realizará una descripción de las características que el sistema debe tener, detallándose las funcionalidades requeridas por el usuario. Se definen los actores del sistema, junto con sus funciones y se hará un levantamiento de requisitos del sistema.

2.2- Estrategia de Integración

En el proyecto BioSyS se realizó un profundo estudio, el cuál arrojó, las tablas necesarias para cada módulo, identificándose nueve posibles clases persistentes y sus posibles atributos. Con esta información se diseñó el diagrama de clases persistentes, el modelo entidad relación y se realizó la descripción de cada uno de sus atributos, lo que permitió realizar la integración de los módulos y así generar la base de datos.

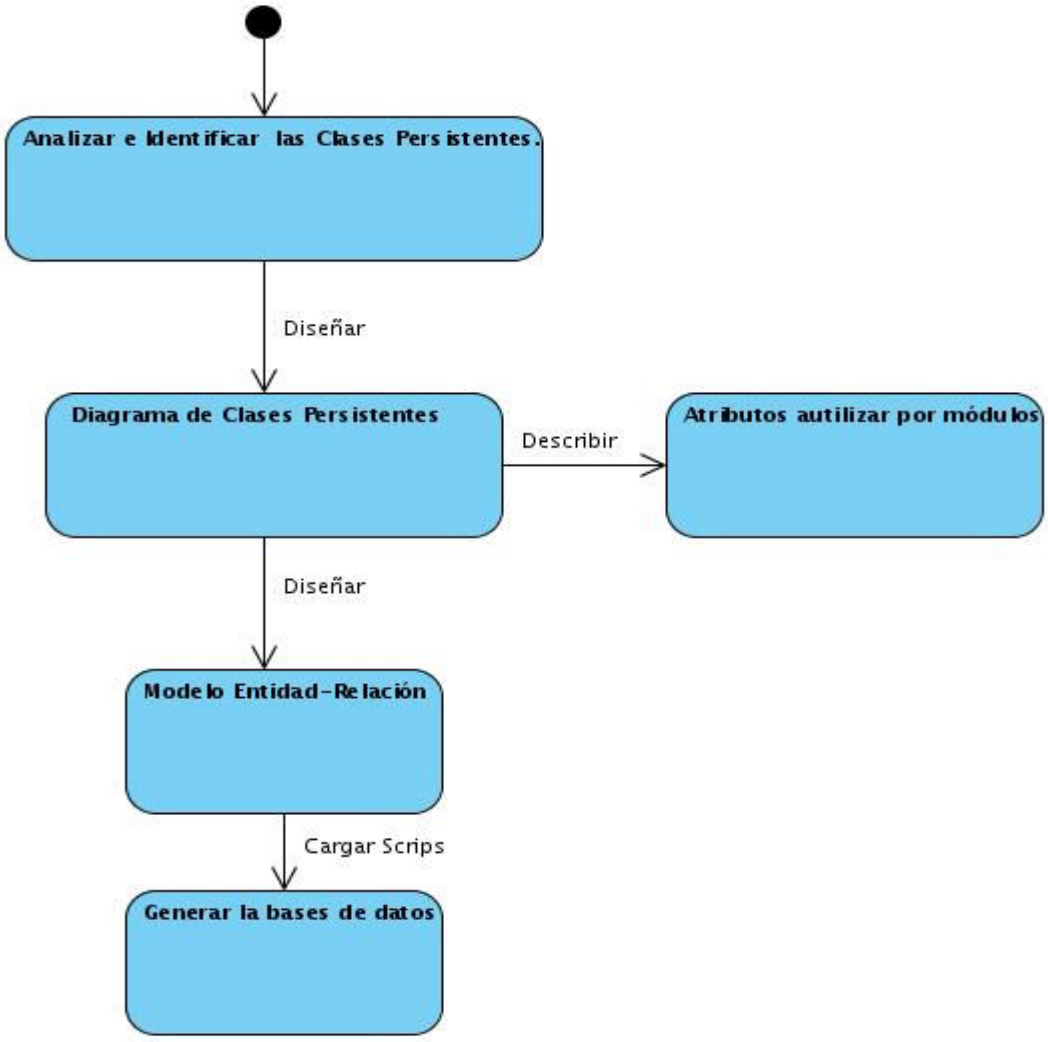


Fig. 1 Estrategia de Integración

2.3- Descripción de la Arquitectura

La base de datos propuesta contiene nueve tablas, donde se recoge la información de los modelos matemáticos, las simulaciones, los métodos numéricos y el juego de datos utilizado en cada simulación. En la base de datos se almacenan además, los resultados de las simulaciones realizadas por el investigador, quien va a ser el único actor del sistema.

La Arquitectura en Capas, propuesta específicamente es la de tres niveles. La aplicación se encuentra dividida en tres capas lógicas distintas, (definidas según las características del patrón DAO) cada una de ellas con un grupo de interfaces perfectamente definidas:

La primera capa se denomina capa de presentación y normalmente consiste en una interfaz gráfica amigable al usuario (PhpMyAdmin, PhpPgAdmin).

La capa intermedia, o capa de empresa, que consiste en la aplicación o lógica de empresa, básicamente el código al que recurre la capa de presentación para recuperar los datos deseados. En esta capa el patrón DAO define por cada clase persistente una clase interfaz y a su vez, una clase concreta, donde se implementan los métodos definidos, permitiendo obtener los datos mediante el uso de objetos, los que van a estar definidos en la clase factory, que es la encargada de establecer la conexión con la capa de acceso a datos.

La capa de datos, que contiene los datos necesarios para la aplicación, almacenados en la base de datos y además incluye las consultas y procedimientos almacenados.

La separación existente entre la capa lógica de la aplicación (interfaz amigable al usuario) y la capa intermedia (patrón DAO) añade una enorme flexibilidad al diseño de la aplicación, donde pueden construirse y desplegarse múltiples interfaces de usuario sin cambiar en absoluto la lógica de la aplicación siempre que esté presente una interfaz claramente definida en la capa de presentación.

2.4- Actor del Sistema

Nombre	**Descripción**
Rol Investigador	Persona que interactúa directamente con la aplicación, se encarga de insertar los parámetros y las condiciones iníciales que va a utilizar para simular, además de

almacenar los resultados, para realizar posteriormente análisis y búsquedas.

2.5- Levantamiento de Requisitos

Para elaborar el software, es necesario inicialmente comprender los requisitos funcionales, que se traducen en las necesidades y expectativas de los usuarios, lo que constituye un aspecto fundamental para que la gestión de proyecto sea eficiente y se obtenga como resultado un producto entregado en el tiempo establecido, y con la calidad requerida. Dicho trabajo se encuentra en su etapa inicial, y aunque la metodología propuesta por la arquitectura no cuenta entre sus roles con el desarrollador de bases de datos, la nueva planilla insertada propone como actividades primarias a realizar: el levantamiento de requisitos, o funcionalidades que el sistema deberá cumplir.

2.5.1- Requisitos Funcionales

Son capacidades o condiciones que el sistema debe cumplir. Los requisitos funcionales permiten expresar una especificación más detallada de las responsabilidades del sistema que se propone. Ellos permiten determinar, de una manera clara, lo que debe hacer el sistema. Y se mantienen invariables sin importar con que propiedades o cualidades se relacionen.

Después de un estudio minucioso de los módulos de BioSyS se proponen como Requisitos Funcionales del módulo de base de datos:

1. Gestionar información de los sistemas.

    1.1 Insertar sistemas.

    1.2 Eliminar sistemas.

    1.3 Modificar sistemas.

2. Gestionar información de las simulaciones.

    2.1 Insertar simulación.

    2.2 Eliminar simulación.

2.3 Modificar simulación.

3. Gestionar los valores.

3.1 Insertar valores.

3.2 Eliminar valores.

2.3 Modificar valores.

4. Gestionar las condiciones iníciales.

4.1 Insertar condiciones iníciales

4.2 Eliminar condiciones iníciales

2.3 Modificar condiciones iníciales

5. Gestionar los modelos matemáticos.

5.1 Insertar modelos matemáticos.

5.2 Eliminar modelos matemáticos.

5.3 Modificar modelos matemáticos.

6. Gestionar Parámetros.

6.1 Insertar parámetros.

6.2 Eliminar parámetros.

6.3 Modificar parámetros.

7. Listar los Métodos Numéricos.

8. Mostrar Resultados.

2.5.2- Requisitos no Funcionales

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable, por ejemplo, pudiera desearse que el sistema responda dentro de un intervalo de tiempo especificado o que obtenga los resultados de los cálculos con un nivel de precisión dado.

- Requisitos de Software

El sistema operativo a utilizar deberá ser Linux, de acuerdo a las peticiones del cliente. La Comunicación Cliente-Servidor será a una velocidad constante de 10/100 Mbps El lenguaje de programación será Java, y como herramienta CASE utilizaremos el Visual Paradigm. Y como gestor de bases de datos el MySQL y el PostgreSQL.

- Requisitos de Hardware

Los servidores de bases de datos a utilizar deberán ser Pentium IV, los cuales deben contar con 512 MB de memoria RAM como mínimo, aunque lo ideal seria 1 GB. La capacidad del disco duro del servidor deberá ser de 40 GB como mínimo, debido a la cantidad de información a almacenar. Las computadoras clientes tendrán como mínimo 128 MB de memoria RAM.

- Requerimientos de apariencia o interfaz externa

El sistema deberá presentar una interfaz externa que modele la acción actor-sistema, la cuál deberá ser sencilla y fácil de comprender.

- Requerimientos de Usabilidad

La aplicación dará la posibilidad de almacenar los resultados obtenidos a partir de la modelación y simulación de sistemas biológicos, mediante interfaces que permitan el intercambio de datos de manera fácil para el usuario.

- Requerimientos de Seguridad

Confidencialidad**:** La información manejada por el sistema esta protegida de acceso no autorizado y divulgación.

Disponibilidad**:** Significa que los usuarios autorizados se les garantizará el acceso a la información y que los dispositivos o mecanismos utilizados para lograr la seguridad no ocultarán o retrasarán a los usuarios para obtener los datos deseados en un momento dado.

2.6- Casos de uso del sistema

1. Gestionar información de los sistemas.
2. Gestionar información de las simulaciones.
3. Gestionar los valores.
4. Gestionar las condiciones iníciales.
5. Gestionar los modelos matemáticos.
6. Gestionar parámetros.
7. Listar los métodos numéricos.
8. Mostrar Resultados.

2.7- Descripción de las entidades

Nombre: Sistema	
Tipo de clase: entidad	
Atributo:	**Tipo:**
Id_Sistema	int
Nombre_Sistema	text
SBML	text
CantPob	int
Para cada responsabilidad:	
Nombre:	Insertar Sistema
Descripción:	Se llenan los campos necesarios para insertar un nuevo sistema a la base de datos.
Nombre:	Modificar Sistemas
Descripción:	Para modificar un sistema, se le pasa como parámetro el identificador del Sistema, con el cual se busca el Sistema a modificar
Nombre:	Eliminar Sistema
Descripción:	Para eliminar un sistema de la base de datos, se le pasa como parámetro el identificador del Sistema, con el cual se puede eliminar el Sistema.

Nombre: Modelo_Matematico	
Tipo de clase: entidad	
Atributo	**Tipo**
Id_Modelo	int
Sistema_Id_sistema	int
Nombre_Modelo	text
Matlab_Model	text
MathML	text
Java_Model	text
Editor	text
Para cada responsabilidad:	
Nombre:	Insertar Modelo Matemático
Descripción:	Se llenan los campos necesarios para insertar un nuevo Modelo Matemático a la base de datos
Nombre:	Modificar Modelo Matemático
Descripción:	Para modificar un Modelo Matemático, se le pasa como parámetro el identificador del Modelo Matemático, con el cual se busca el modelo a modificar
Nombre:	Eliminar Modelo Matemático
Descripción:	Para eliminar un Modelo Matemático de la base de datos, se le pasa como parámetro el identificador del Modelo Matemático, con el cual se puede eliminar.

Nombre: Simulacion	
Tipo de clase: entidad	
Atributo	**Tipo**
Id_Simulacion	text
Modelo_Matematico_Id_Modelo	int
Metodo_Numerico_idMetodoNum	int
Para cada responsabilidad:	

Nombre:	Insertar Simulación
Descripción:	Se llenan los campos necesarios para insertar una nueva simulación a la base de datos
Nombre:	Modificar Simulación
Descripción:	Para modificar una Simulación, se le pasa como parámetro el identificador de la Simulación, con el cual se busca y se modifica.
Nombre:	Eliminar Simulación
Descripción:	Para eliminar una Simulación de la base de datos, se le pasa como parámetro el identificador y luego se elimina de la base de datos.

Nombre: Condiciones_Iniciales	
Tipo de clase: entidad	
Atributo	**Tipo**
Id_Cond_Iniciales	int
Modelo_Matematico_Id_Modelo	int
Simbolo_CIniciales	text
Posicion_CIniciales	int
Para cada responsabilidad:	
Nombre:	Insertar condición inicial
Descripción:	Se llenan los campos necesarios para insertar una nueva condición inicial a la base de datos
Nombre:	Modificar condición inicial
Descripción:	Para modificar una condición inicial, se le pasa como parámetro el identificador con el cual se busca la condición inicial a modificar.
Nombre:	Eliminar condición inicial
Descripción:	Para eliminar una condición inicial de la base de datos, se le pasa como parámetro el identificador de la condición inicial, con la cual se puede eliminar.

Nombre: Metodo_Numerico
Tipo de clase: entidad

Atributo	**Tipo**
idMetodoNum	int
programa	text
algoritmo	text
TAbsoluto	double
TRelativo	double

Nombre: Parametros	
Tipo de clase: entidad	
Atributo	**Tipo**
Id_Parametros	int
Modelo_Matematico_Id_Modelo	int
Simbolo_Parametros	text
Posicion_Parametros	int
Para cada responsabilidad:	
Nombre:	Insertar Parametros
Descripción:	Se llenan los campos necesarios para insertar un nuevo Parametros a la base de datos
Nombre:	Modificar Parametros
Descripción:	Para modificar un parámetro, se le pasa el identificador con el cual se busca y se modifica.
Nombre:	Eliminar Parametros
Descripción:	Para eliminar un parámetro de la base de datos, se le pasa el identificador y se elimina.

Nombre: Valores_Condiciones_Iniciales	
Tipo de clase: entidad	
Atributo	**Tipo**
Id_Valor	int

Condiciones_Iniciales_Id_Cond_Iniciales	int
Valor	double
Para cada responsabilidad:	
Nombre:	Insertar Condiciones_Iniciales
Descripción:	Se llenan los campos necesarios para insertar las Condiciones Iníciales
Nombre:	Modificar Condiciones_Iniciales
Descripción:	Para modificar una condición inicial, se le pasa el identificador con el cual se busca y se modifica.
Nombre:	Eliminar Condiciones_Iniciales
Descripción:	Para eliminar una condición inicial de la base de datos, se le pasa el identificador y se elimina.

Nombre: Valores_Parametros	
Tipo de clase: entidad	
Atributo	**Tipo**
Id_Valor	int
Parametros_Id_Parametros	int
Valor	double
Para cada responsabilidad:	
Nombre:	Insertar Valores de Parametros
Descripción:	Se llenan los campos necesarios para insertar un nuevo valor a la base de datos
Nombre:	Modificar Valores de Parametros
Descripción:	Para modificar un Parametros, se le pasa como el identificador con el cual se busca y se modifica.
Nombre:	Eliminar Valores de Parametros
Descripción:	Para eliminar el Valor de un Parametros de la base de datos, se le pasa el identificador y se elimina.

Nombre: Resultados	
Tipo de clase: entidad	
Atributo	**Tipo**
Id_Resultado	int
Id_Poblacion	int
Valor	double
Tiempo	double
Simulacion_Id_Simulacion	text
Para cada responsabilidad:	
Nombre:	Mostrar Resultado()
Descripción:	El actor escoge la opción mostrar resultado, donde puede realizar búsquedas pasándole el identificador del resultado.

2.8- Diagrama de clases persistentes

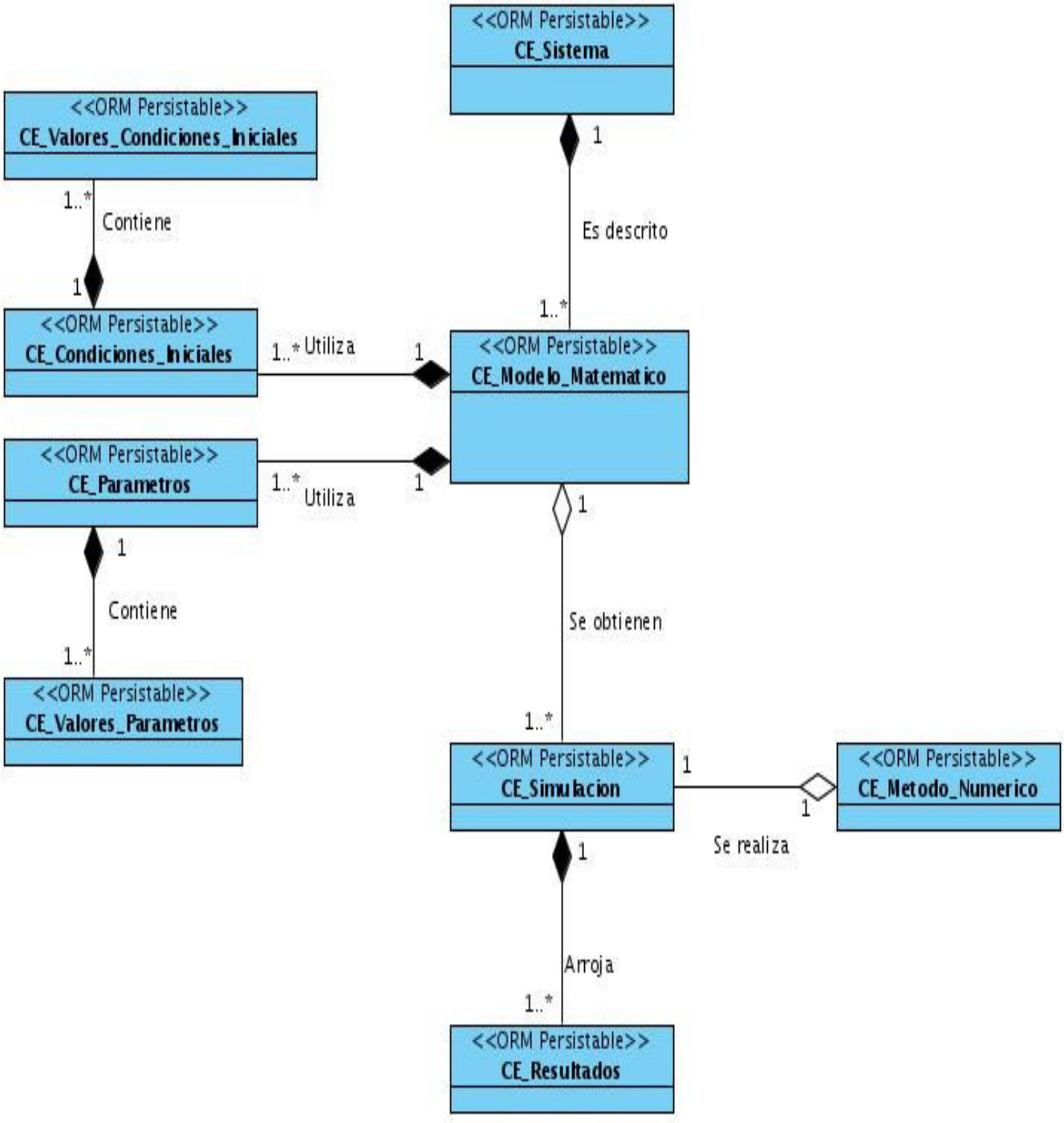


Fig. 2 Diagrama de Clases Persistentes

2.9- Modelo Entidad Relación

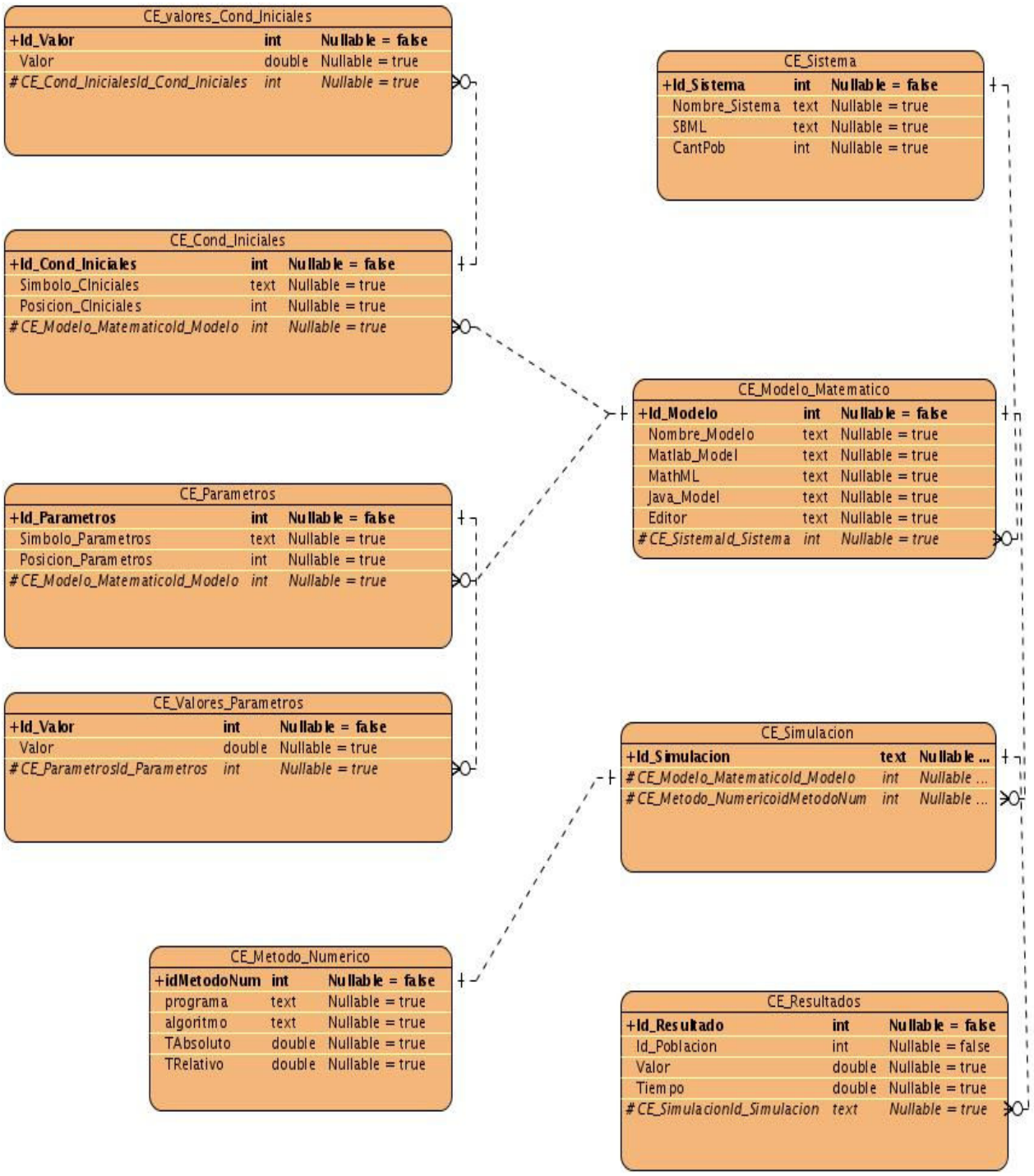


Fig. 3 Modelo Entidad-Relación

2.10- Modelo físico de Datos

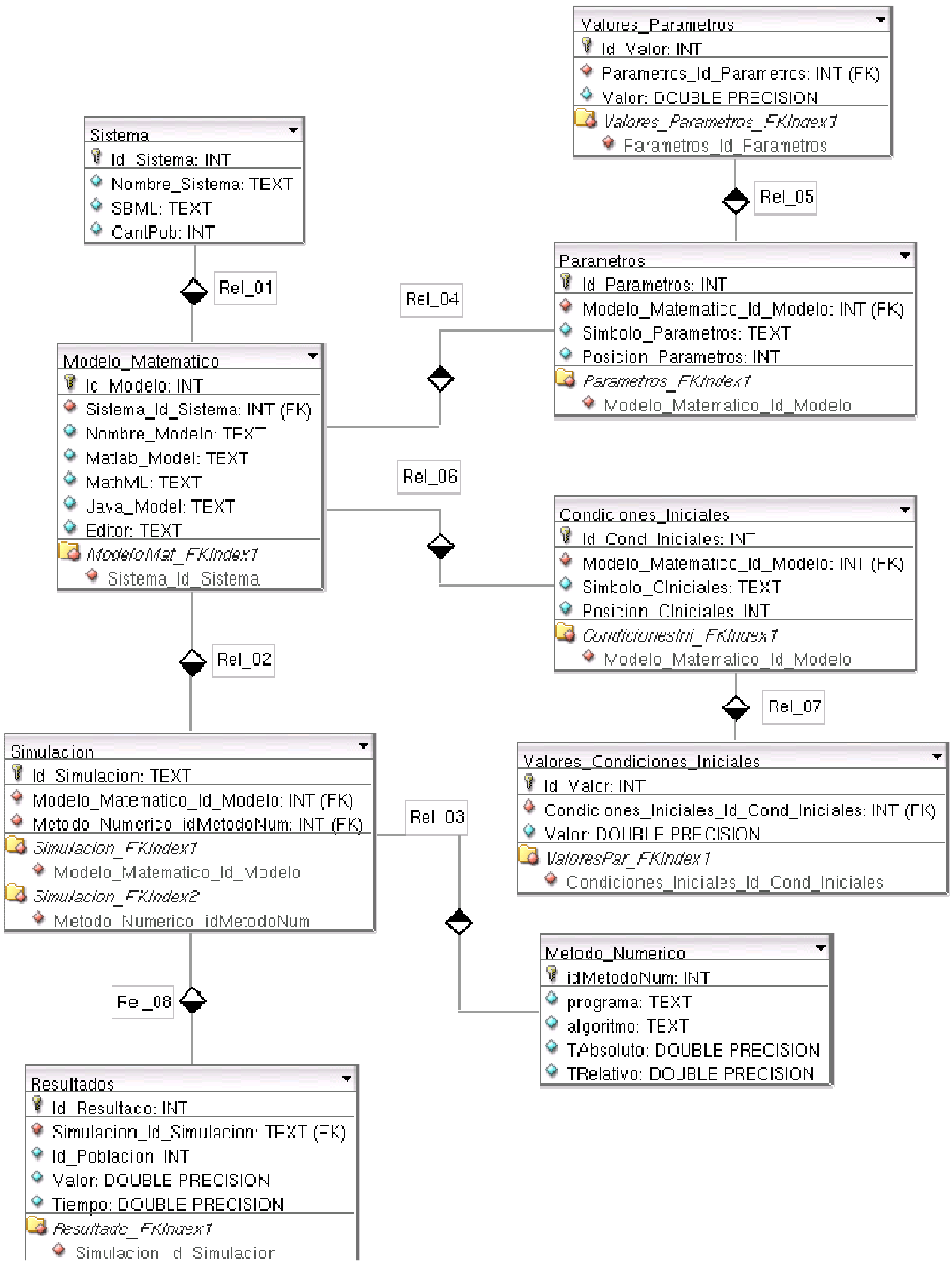


Fig. 4 Modelo Físico de Datos

2.11- Descripción de las tablas

La *tabla sistema* almacena los datos de los sistemas biológicos: los cuales se basan en poblaciones, compuestos químicos, y sus interacciones; las relaciones entre los operadores dentro de las poblaciones, y los estudios a diferentes niveles de complejidad. (2) Como llave primaria de la tabla se encuentra el identificador del sistema, el que se encarga de identificar cada sistema utilizado para realizar una simulación. Los demás atributos describen el sistema que se esta utilizando: el nombre del sistema (nombre _sistema), la cantidad de poblaciones o cantidad de elementos por los que esta compuesto el sistema (CantPob), y un Lenguaje de Marcado de Sistemas Biológicos creado para describir las simulaciones en la biología de sistemas (SBML) (Ver anexo 4). Con este SBML se genera un modelo matemático que se modifica mediante el Editor de Ecuaciones. Cada sistema puede ser descrito mediante diferentes modelos matemáticos.

La *tabla modelo_matematico* describe matemáticamente el comportamiento del sistema biológico. Los modelos están compuestos por arreglos de condiciones iníciales y parámetros. (2) Su llave primaria es un identificador del modelo, además del nombre del mismo, y las herramientas con las que se va a simular, las que pueden ser MatLab, o Java, o MathML (Ver anexo 5).

Una vez que se tienen los modelos matemáticos se puede comenzar a simular. Las simulaciones pueden ser simples o múltiples. (2) La base de datos almacena las simulaciones realizadas en la *tabla simulacion,* teniendo como atributo primario el identificador de la misma, el que es generado por la concatenación de los valores proveniente de las variables y los parámetros utilizados mediante underscored, permitiendo posteriormente conocer el juego de datos con los que se realizó la simulación en dicho momento. Además se tienen los identificadores de método numérico y modelo matemático con el objetivo de conocer con cuál método numérico y modelo matemático se simuló. Aquí es importante destacar que esta disyuntiva podía haberse resuelto insertando el identificador de método numérico en la tabla modelo matemático más no es una alternativa fiable en la solución por problema de migración de llaves.

En la tabla *metodos_numericos* existe un identificador, un asistente matemático o programa (Matlab y Java) y los algoritmos con los que realizará dicha simulación, señalando además los tiempos absolutos y relativos.

Para realizar una simulación el usuario debe insertar los *parámetros* y las *condiciones iníciales* que

quiere utilizar, ambas tablas definidas por identificadores, símbolos y posiciones.

Para obtener los resultados de las simulaciones realizadas en un tiempo determinado, a cada parámetro y condición inicial se le asignarán uno o varios valores numéricos, los cuales se encuentran almacenados en la *tabla valores_variables*.y *valores_parametros,* específicamente en el atributo valor.

2.12-Conclusiones

En este capítulo se realizó un análisis de la situación problémica, donde se comprobó la necesidad de crear una nueva base de datos, la cual incluyera nuevas funcionalidades y permitiera una mejor gestión de la información. Se realizó una extracción de los requisitos del sistema a implementar, así como de los casos de uso significativos. Además de una descripción de cada clase persistente, lo cual trajo como resultado el diseño del modelo entidad-relación y el modelo de datos físicos de la base de datos.

CAPITULO 3: IMPLEMENTACIÓN Y PRUEBA

3.1- Introducción

En este capítulo se abordan los artefactos referentes al flujo de trabajo de implementación y de pruebas, documentando la Validación teórica y funcional de la base de dato, las descripciones de las principales consultas desarrolladas, la utilización del Patrón DAO y el NetBeans, herramienta que se utilizó para generar la capa de Acceso a Datos.

3.2- Diagrama de Despliegue

Es un tipo de diagrama del Lenguaje Unificado de Modelado que se utiliza para modelar el hardware utilizado en las implementaciones de sistemas y las relaciones entre sus componentes, hardware y software, es decir, la configuración de los elementos de procesamiento en tiempo de ejecución y los componentes software (proceso y objetos que se ejecutan en ellos). El diagrama de despliegue es un grafo de nodos unidos por conexiones de comunicación.

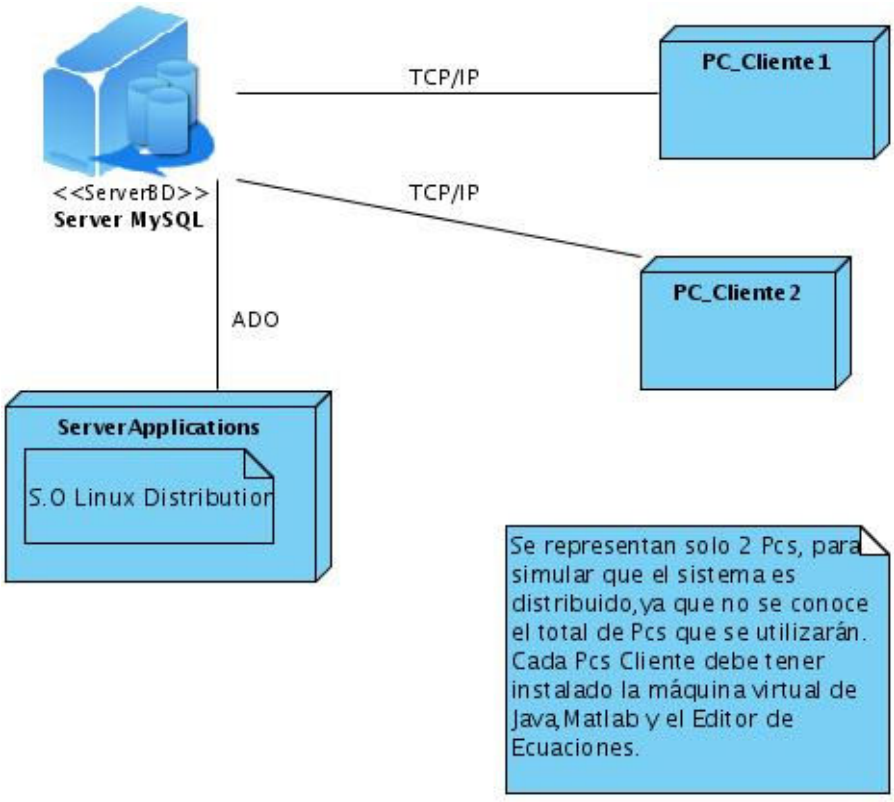


Fig. 5 Diagrama de Despliegue

3.3-Diagrama de Componentes

Representa la separación de un sistema de software en componentes físicos (por ejemplo código fuente, binario y ejecutable) y muestra las dependencias entre estos componentes. Se utilizan para modelar la vista estática de un sistema. Los elementos de modelado dentro de un diagrama de componentes serán componentes y paquetes. En cuanto a los componentes, sólo aparecen tipos de componentes, incluyendo los requisitos y las restricciones impuestas por los lenguajes de programación, y los paquetes representan la división física del sistema.

Estereotipos Utilizados:

- Ejecutable
- Biblioteca
- Entidad
- Archivo

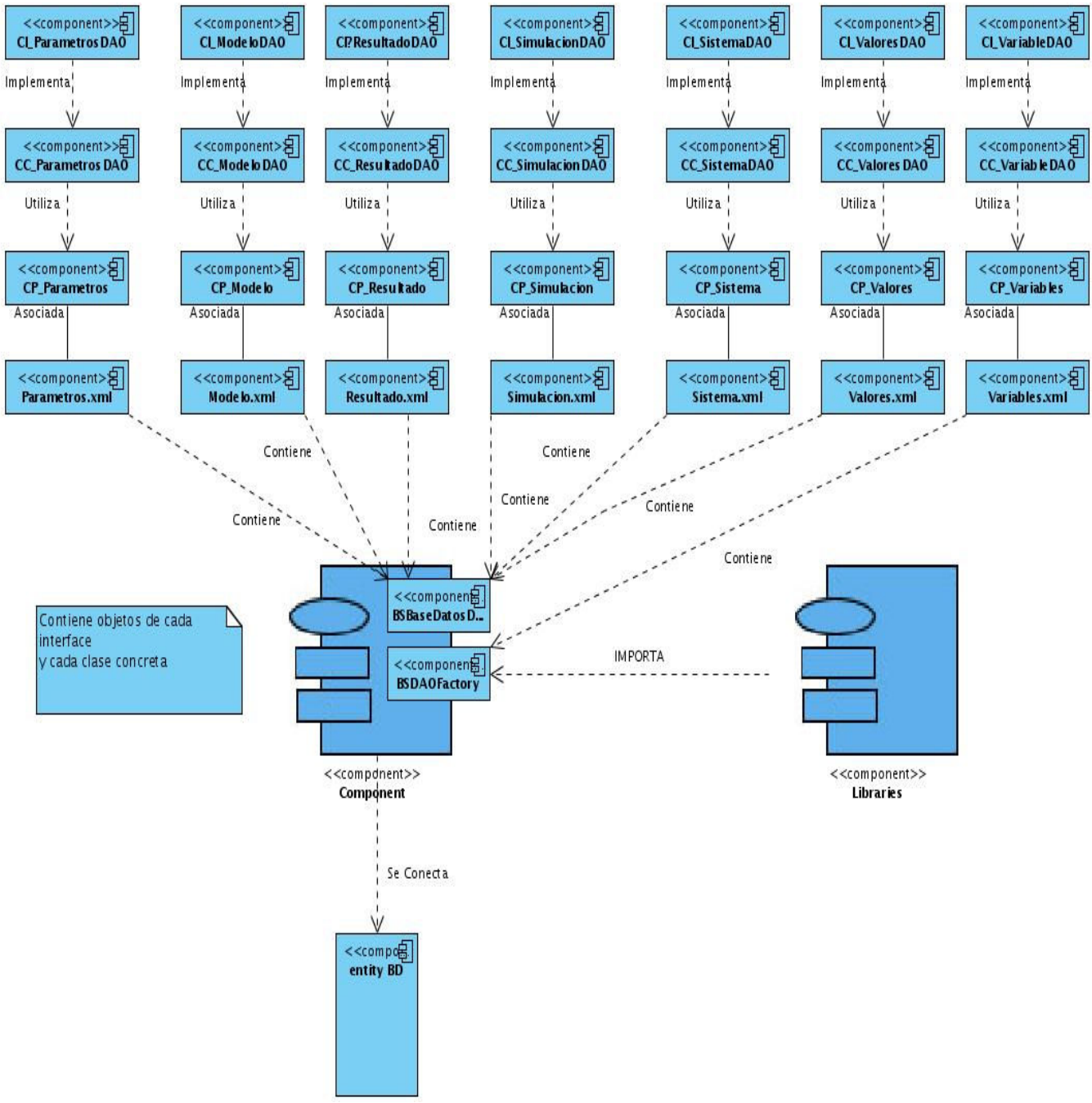


Fig. 6 Diagrama de Componentes

3.4- Validación teórica del diseño

3.4.1- Integridad de datos

El término *integridad de datos* se refiere a la corrección y completitud de los datos en una base de datos. Por esto hay que tener en cuenta que la entrada de datos tenga formatos válidos, así como la modificación de las tablas (a través de las sentencias INSERT, UPDATE, DELETE), ya que los cambios pueden perderse debido a un error del sistema o a un fallo en el suministro de energía. Una de las funciones importantes de un sistema manejador de bases de datos (DBMS) relacional es mantener la integridad y seguridad de los datos.

Primeramente se verifica la integridad referencial, aspecto importante que garantiza que los datos que referencian a otros (claves foráneas) sean correctos. La integridad referencial hace que el sistema gestor de la base de datos garantice la no existencia de claves foráneas: valores que no estén en la tabla principal.

Garantizando algunos Tipos de Restricciones de Integridad:

- *Chequeo de Validez*: Cuando se crea una tabla cada columna tiene un tipo de dato y el DBMS asegura que solamente los datos del tipo especificado sean ingresados en la tabla. En caso contrario se emitirá una llamada de aviso.
- *Integridad de entidad*: Establece que la clave primaria de una tabla debe tener un valor único para cada fila especificándose en la sentencia CREATE TABLE, además de que será autoincremento. El DBMS comprueba automáticamente la unicidad del valor de la clave primaria con cada sentencia INSERT Y UPDATE. Un intento de insertar o actualizar una fila con un valor de la clave primaria ya existente, fallará. Todos los atributos primarios de las tablas nunca serán nulos.
- Se implementa una *cláusula CHECK* que obliga a que el sistema compruebe que se cumple la expresión (una expresión booleana cualquiera en SQL) dentro del contexto en que se haya colocado la cláusula CHECK.

3.4.2-Normalización de la base de datos

Las bases de datos relacionales pasan por un proceso al que se le conoce como normalización, garantizando herramientas que eviten la duplicidad de registros, así como, decidir a qué entidad

pertenece cada atributo, a través de campos claves o llaves. La integridad referencial y la normalización de una base de datos, son procesos necesarios para que una base de datos sea utilizada de manera óptima.

La base de datos se encuentra en Tercera Forma Normal, primeramente porque todas sus tablas están en segunda forma normal (2NF) y ningún atributo no-primario de la tabla es dependiente transitivamente de una clave candidata.

A pesar de lo anteriormente expuesto, una base de datos normalizada impide las dependencias funcionales de los datos para que el proceso de actualización de la base de datos sea fácil y eficiente. Sin embargo, la realización de consultas puede requerir la combinación de varias tablas para unir la información. A medida que el número de tablas combinadas crece, el tiempo de ejecución de la consulta aumenta considerablemente. Por este motivo, el uso de una base de datos normalizada no es siempre la mejor alternativa. Una base de datos con la medida justa de desnormalización reduce el número de tablas que deben combinarse sin dificultar en exceso el proceso de actualización. Suele ser la solución más acertada en estos casos. [20]

3.4.3- Análisis de la Redundancia de la Información

Una base de datos normalizada logra evitar la aparición de información redundante. Cuyo objetivo principal es lograr una redundancia nula; no obstante, en algunos casos se hace necesaria la aparición de redundancias. En el diseño de BioSyS para la nueva versión, se obtuvieron nueve tablas, evitando la aparición de redundancia de información, la ocupación de memoria no óptima, y un costo elevado de mantenimiento.

3.4.4- Análisis de la seguridad de la base de datos

PhpMyAdmin y el PhpPgAdmin se han convertido en herramientas muy utilizadas entre los desarrolladores y webmasters debido a su poder y simplicidad. Permiten acceder a todas las funciones típicas de la base de datos a través de una interfaz web muy intuitiva, facilitando la:

- ✓ Creación de bases de datos
- ✓ Creación de usuarios y permisos
- ✓ Creación de tablas y campos
- ✓ Exportación y copias de seguridad

- ✓ Importación de archivos

Luego de la instalación se pueden “Añadir privilegios a la base de datos” mediante las casillas de marcar (checkbox), relativos a los permisos que debe tener cada usuario sobre la base de datos, y en función de estos permisos ejecutar unas u otras consultas.

Para una mayor seguridad se utilizaron algunos comandos, entre ellos el comando GRANT, el cual permite conceder los privilegios de acceso necesario a los objetos de la base de datos, estos privilegios pueden ser cualquier combinación de seleccionar, insertar, anular o modificar.

Afortunadamente, SQL proporciona el comando *REVOQUE*, cuya función es quitar los permisos previamente concedidos, su sintaxis es muy similar a la del comando *GRANT* la única diferencia es que con este se especifica el final de los permisos otorgados.

3.5- Pruebas de Validación

La validación de un modelo se puede definir como la demostración de su exactitud para una aplicación concreta. En este sentido, la exactitud es la ausencia de errores sistemáticos y aleatorios: se conocen habitualmente como fidelidad y precisión respectivamente.

Tienen como finalidad concentrar toda la información disponible sobre criterios técnicos de validación con el fin de obtener resultados que permitan realizar comparaciones mediante criterios de validación, y se deberán diseñar pruebas que tengan la mayor probabilidad de encontrar el mayor número de errores con la mínima cantidad de esfuerzo y tiempo.

3.5.1-Validación de códigos informáticos

Un requisito previo esencial son las buenas prácticas de programación (es decir, modular y documentar plenamente). Los puntos específicos que requieren atención son los posibles efectos de la precisión de la máquina y los factores informáticos específicos en la obtención del modelo. Los informes sobre errores internos del programa informático son fuentes importantes de información, así como la evaluación del producto intermedio y sistemático. Para la construcción del software existen una serie de paradigmas, dichos paradigmas tienen como objetivo la construcción de un producto de buena calidad.

3.5.2-Validación Funcional

Es la verificación del modelo frente a observaciones obtenidas de manera independiente. La evaluación ideal consiste en obtener los datos pertinentes de las diferentes consultas y realizar una comparación estadística de los resultados obtenidos. [20] Para esto se requiere una información más detallada que la disponible habitualmente. Dichos datos no pueden validar el modelo de entidad-relación pero su respuesta ante las consultas como tal, si.

3.5.3-Generación de Código

Los generadores profesionales de hoy en día generan código para entidades, y tecnologías partiendo de un modelo predeterminado (típicamente de una estructura de tablas de base de datos). Brinda la posibilidad de desde un modelo, realizar cambios, de forma fácil: generar nuevamente los artefactos, por un cambio en el modelo.

Para la aplicación de esta prueba se hizo uso del EMS MySQL Manager 2005, programa que permite realizar pruebas de validación de consultas, comprobando también si las sintaxis de dichas consultas es correcta. Se obtiene un tiempo de ejecución, parámetro muy útil a la hora de comprobar que la búsqueda realizada en un tiempo es óptima.

Las principales consultas utilizadas son la de búsqueda dado algunos atributos, para realizarle pruebas a la base de datos seleccionamos las más utilizadas.

1. ”Obtener los resultados de todos los valores y tiempo de las tablas sistema, modelo_matematico y simulacion en donde el identificador del sistema sea igual a 1”

SELECT resultados.Valor , resultados.Tiempo FROM sistema INNER JOIN modelo_matematico ON (sistema.Id_Sistema = modelo_matematico.sistema_Id_Sistema) INNER JOIN simulacion ON (modelo_matematico.Id_Modelo = simulacion.Id_Modelo) INNER JOIN resultados ON (simulacion.Id_Simulacion = resultados.Id_Simulacion) WHERE sistema.Id_Sistema = 1

Se obtuvo como resultado final, 3 columnas con 30 filas, para un tiempo de 2422 ms.

2. ”Obtener los identificadores de las simulaciones en donde el programa usado en el método numérico sea Matlab y el nombre del modelo sea "il2model1“

SELECT simulacion.Id_Simulacion FROM modelo_matematico INNER JOIN simulacion ON (modelo_matematico.Id_Modelo = simulacion.Id_Modelo) INNER JOIN metodo_numerico ON (simulacion.idMetodoNum = metodo_numerico.idMetodoNum) WHERE metodo_numerico.programa = "Matlab" AND modelo_matematico.Nombre_Modelo = "il2model1"

Arrojando un resultado de 7 columnas con 9 filas en un tiempo de 219 ms.

3. ”Mostrar los nombres de los modelos y los valores de parámetros en donde el nombre del sistema sea “Interleuquina”

SELECT modelo_matematico.Nombre_Modelo, valores_parametros.Valor FROM sistema INNER JOIN modelo_matematico ON (sistema.Id_Sistema = modelo_matematico.sistema_Id_Sistema) INNER JOIN parametros ON (modelo_matematico.Id_Modelo = parametros.modelo_matematico_Id_Modelo) INNER JOIN valores_parametros ON (parametros.Id_Parametros = valores_parametros.Id_Parametros) WHERE sistema.Nombre_Sistema = "Interleuquina"

El resultado obtenido fue de 2 columnas con 30 filas en un tiempo de 531 ms.

4. ”Mostrar los símbolos de las condiciones iníciales en donde el identificador del valor de la tabla valores_variables sea igual a 1 y el identificador del modelo_matematico igual a 2”

SELECT cond_iniciales.Simbolo_Condiciones FROM modelo_matematico INNER JOIN cond_iniciales ON (modelo_matematico.Id_Modelo = cond_iniciales.modelo_matematico_Id_Modelo) INNER JOIN valores_variables ON (cond_iniciales.Id_Cond_Iniciales = valores_variables.Id_Cond_Iniciales)

WHERE valores_variables.Id_Valor = 1 AND modelo_matematico.Id_Modelo = 2

Se ejecuto en 125 ms de tiempo, y no se mostró ningún valor, ya que no existe ningún identificador igual 2.

5. ”Mostrar los simbolos de las condiciones iníciales en donde el identificador del valor de la tabla valores_variables sea igual a 1 y el nombre del modelo_matematico “dy2_eval”

SELECT cond_iniciales.Simbolo_Condiciones FROM modelo_matematico INNER JOIN cond_iniciales ON (modelo_matematico.Id_Modelo= cond_iniciales.modelo_matematico_Id_Modelo) INNER JOIN valores_variables ON

(cond_iniciales.Id_Cond_Iniciales = valores_variables.Id_Cond_Iniciales) WHERE valores_variables.Id_Valor = 1 AND modelo_matematico.Nombre_Modelo = "dy2_eval"

Dicha consulta se demoro 47 ms y no se mostró ningún valor, ya que no existe ningún identificador

6. ”Mostrar los identificadores de las simulaciones en donde el programa usado sea Matlab y el identificador del modelo_matematico sea igual a 2”

   SELECT simulacion.Id_Simulacion FROM simulacion INNER JOIN metodo_numerico ON (simulacion.idMetodoNum = metodo_numerico.idMetodoNum) INNER JOIN modelo_matematico ON (simulacion.Id_Modelo = modelo_matematico.Id_Modelo) WHERE metodo_numerico.programa = "Matlab" AND modelo_matematico.Id_Modelo = 2

El EMS MySQL manager 2005 demora 265 ms, no se obtuvo ningún resultado, ya que no hay Id_Modelo igual a 2.

7. ”Mostrar los identificadores de las simulaciones en donde el identificador del modelo sea igual a 3 y el identificador del sistema sea igual a 1”

SELECT simulacion.Id_Simulacion FROM modelo_matematico NNER JOIN simulacion ON (modelo_matematico.Id_Modelo=simulacion.Id_Modelo) INNER JOIN sistema ON (modelo_matematico.sistema_Id_Sistema=sistema.Id_Sistema)WHERE modelo_matematico.Id_Modelo = 3 AND sistema.Id_Sistema = 1

Los resultados se obtuvieron en 109 ms con 0 filas y 0 columnas.

8. ”Mostrar los identificadores de las simulaciones en donde el nombre del modelo "dy2_eval" y el nombre del sistema descrito sea "PresaDepredador"

   SELECT simulacion.Id_Simulacion FROM simulacion INNER JOIN modelo_matematico ON (simulacion.Id_Modelo=modelo_matematico.Id_Modelo) INNER JOIN sistema ON (modelo_matematico.sistema_Id_Sistema=sistema.Id_Sistema)WHERE modelo_matematico.Nombre_Modelo= "dy2_eval" AND sistema.Nombre_Sistema= "PresaDepredador"

   El EMS MySQL manager 2005 demora 219 ms en realizar dicha consulta, y mostro 2 columnas

con 20 filas.

Prueba	Criterio	Filas Resultado	Col Resultado	Tiempo(ms)
1	Id_Sistema	30	3	2422
2	Nombre_Modelo, Matlab_Model	9	7	219
3	Nombre_Sistema	30	2	531
4	Valor, Id_Modelo	0	0	125
5	Valor, Nombre_Modelo	0	0	47
6	Matlab_Model, Id_Modelo	0	0	265
7	Id_Modelo, Id_Sistema	0	0	128
8	Nombre_Modelo, Nombre_Sistema	20	2	219

Luego de analizar los resultados obtenidos y teniendo en cuenta el tiempo de ejecución de cada consulta, se concluye que es más factible realizar las búsquedas a partir de los identificadores de cada tabla, teniendo en consideración el número de tablas donde se realizarán las búsquedas.

3.5.3.1-Gráficas de Resultados

Los resultados se graficaron, y las imágenes son:

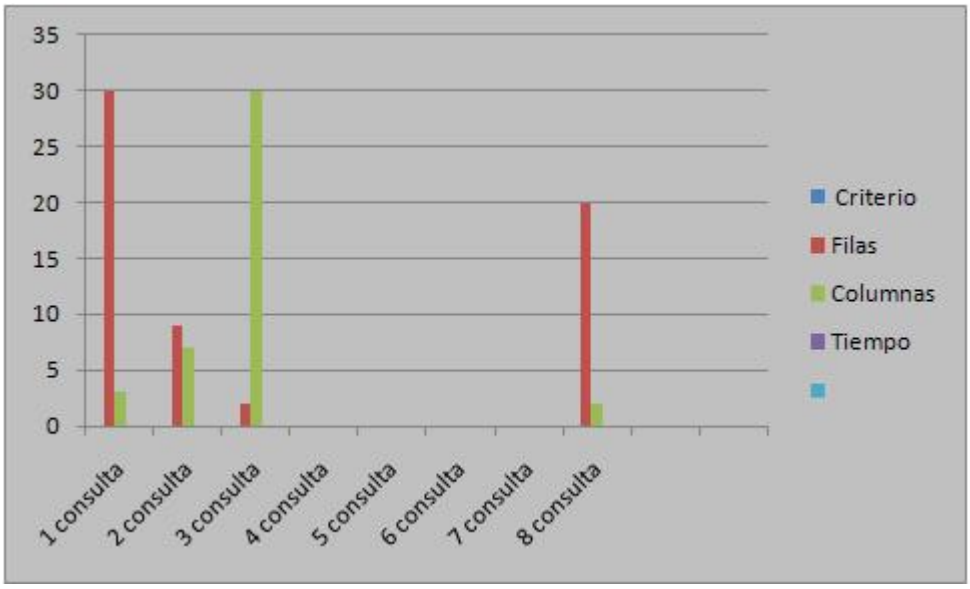


Ilustración 1 Gráfica de Resultados

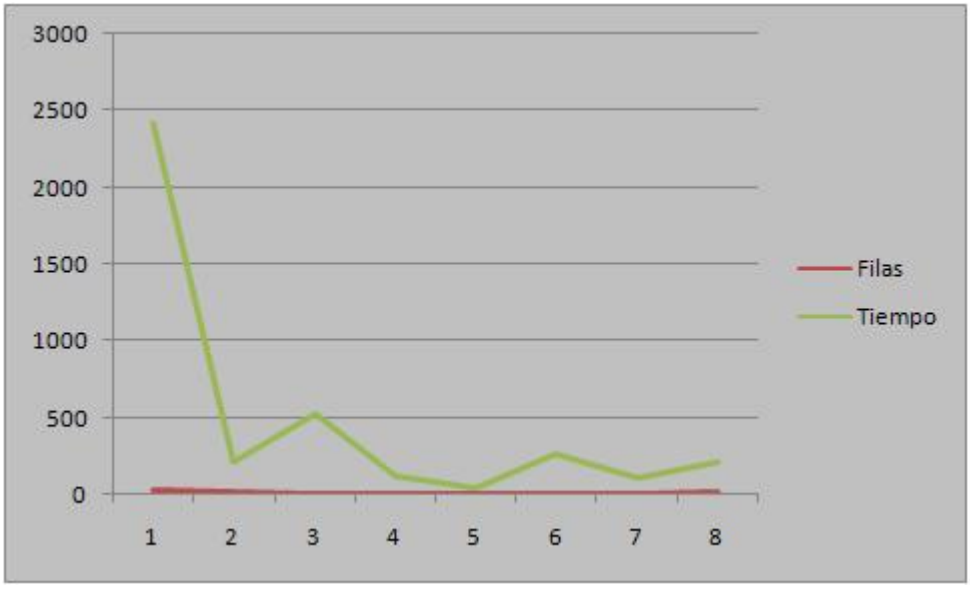


Ilustración 2 Diagrama de Tiempo

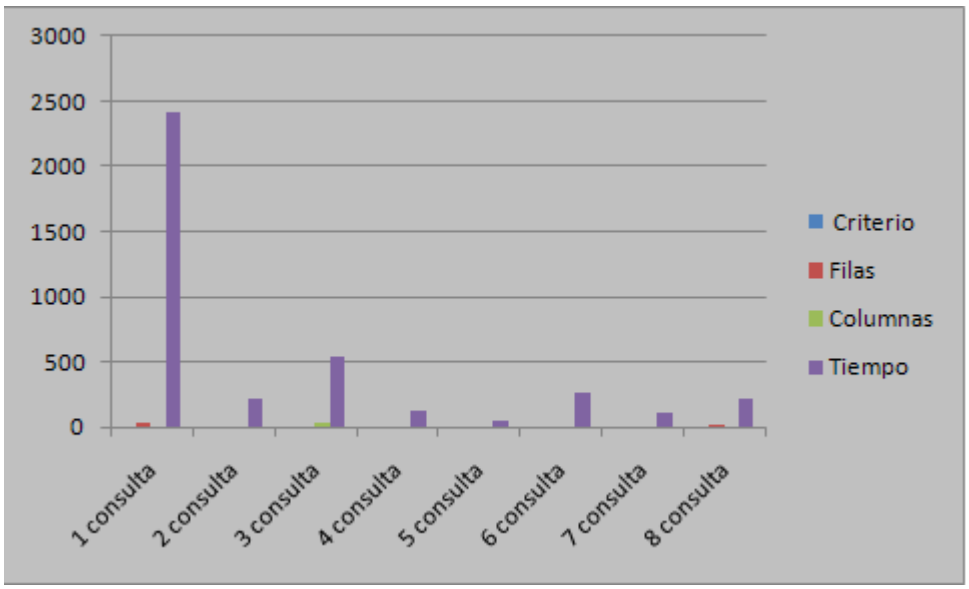


Ilustración 3 Gráfica Resultado

Las gráficas muestran el proceso de pruebas realizado. La primera gráfica señala la variación de los resultados haciendo énfasis en las filas y columnas. La segunda muestra la variación del tiempo con respecto a las filas y la tercera señala el papel que juega el criterio a la hora de obtener los resultados. Evidenciándose el tiempo de ejecución que varia de acuerdo a la concatenación de tablas en las que debe realizar el proceso de búsqueda, así como la cantidad de atributos que debe comparar.

3.5.4-Prueba de Carga Intensiva

Para el llenado de la base de datos se utilizó la herramienta *MySQL Data Generator*, la cual permite la generación de datos para una o varias tablas a la vez. Se generaron para las pruebas a la base de datos un volumen de 100 valores random generando 900 valores para las nueve tablas en 0:00:02 segundos sin errores. Luego se insertaron en cada tabla 10000 valores random generando 74300 valores en 0:00:06 segundos. Por ultimo se insertaron 100 000 valores random en 0:00:20 segundos con 15700 errores. La base de datos tiene actualmente 35839 resultados.

Datos a Insertar	Datos Insertados	Tiempo	Errores
100	900	0:00:02	0
10000	74300	0:00:06	0
100 000	91250	0:00:20	15700

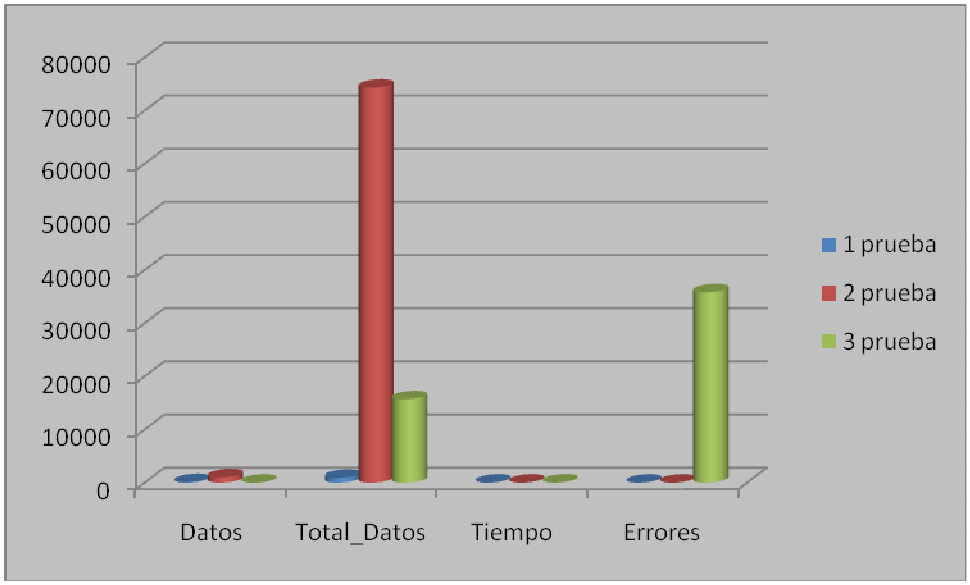


Ilustración 4 Gráfica de Prueba de Carga Intensiva

Ilustración 5 Gráfica de Errores

Las gráficas muestran los valores de las pruebas de carga intensiva, para una mejor visión del impacto de la simultaneidad de datos soportados. La primera ilustración muestra como se comportaron las métricas de datos insertados respecto al tiempo; y la segunda ilustra el margen de errores obtenidos en cada carga de datos, el que puede ser gradual en dependencia de la cantidad de datos a procesar.

3.6-Conclusiones:

En este capítulo se diseñó el diagrama de despliegue y de componentes, los cuales describen la arquitectura lógica de componentes, y con la ayuda del NetBeans e Hibernate se implementó la capa de acceso a datos, y así permitir el acceso a los datos almacenados en la base de datos. Se diseñaron y realizaron pruebas que validaron el correcto funcionamiento de la aplicación.

CONCLUSIONES

Con la fase de implementación y prueba se finalizó el desarrollo de la nueva base de datos para la nueva versión de BioSyS. De esta forma se concluye que:

- ✓ Se realizó el análisis y diseño de la base de datos.
- ✓ Se creó la nueva versión de la base de datos.
- ✓ Se implementó la capa de acceso a datos.
- ✓ Se realizó la validación del sistema.

RECOMENDACIONES

- Actualmente el proceso de simulación realizado es un poco superficial, solamente se analizan los valores de tiempo absoluto y relativo, no analizando así el comportamiento de estos durante el proceso de simulación, por lo que se recomienda Insertar intervalos de tiempo que reflejen su comportamiento durante toda la trayectoria del sistema analizado.
- Aplicar pruebas de conexiones a la base de datos en un período de tiempo más prolongado.
- Incorporar una funcionalidad que permita almacenar información en ficheros y posteriormente estudiarlos.

REFERENCIAS BIBLIOGRÁFICAS

1. **Mulet, Yanet Alonso Delgado Yunet González.** *Software para la Simulación de Sistemas Biológicos: Módulo de Simulación y Análisis.* La Habana : s.n., 2007.

2. **Lemus, Lic. Noel Moreno.** *BioSyS: Software para la simulación y análisis de sistemas.* La Habana : s.n., 2007.

3. **Solórzano, Ligeya Perezleo.** Impacto de la Bioinformática en las ciencias biomédicas . [En línea] Ligeya Perezleo Solórzano. [Citado el: 17 de octubre de 2007.] http://bvs.sld.cu/revistas/aci/vol11_4_03/aci07403.htm.

4. Biología de sistemas. [En línea] 2004. [Citado el: 19 de octubre de 2007.] http://bvs.isciii.es/bibgen/Actividades/curso_virtual/Ftes_informacion/Biologiasistemas.htm.

5. LA BIOTECNOLOGÍA EN EL ESPEJO. [En línea] [Citado el: 22 de octubre de 2007.] http://www.institutoroche.es/editorial2.php?op=biotecnologia&id=29.

6. Un diagrama derivado de la biología de sistemas. [En línea] [Citado el: 22 de octubre de 2007.] http://www.proz.com/kudoz/english_to_spanish/biology_tech_chemmicro_/2130827-a_systems_biology_derived_picture.html.

7. EL CONCEPTO BÁSICO DE SIMULACIÓN POR COMPUTADORA. [En línea] [Citado el: 23 de octubre de 2007.] http://www.dei.uc.edu.py/tai99/introsimulacion/simucon.htm..

8. Impacto de la Bioinformática en las ciencias biomédicas . [En línea] [Citado el: 24 de octubre de 2007.] http://bvs.sld.cu/revistas/aci/vol11_4_03/aci07403.htm.

9. ¿Qué es OpenUp / Basic ? [En línea] [Citado el: 1 de noviembre de 2007.] http://www.cendesi.com/site/es/articles.php?lng=es&pg=11.

10. Paradigma visual para UML (Plataforma Java) (Visual Paradigm for UML [Java Platform]) 6.0 . [En línea] [Citado el: 5 de noviembre de 2007.] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%5Bcuenta_de_Plataforma_de_Java_14715_p/.

11. DB Designer Fork: Una herramienta para todo desarrollador. [En línea] [Citado el: 5 de noviembre de 2007.] http://slackdhabyx.wordpress.com/2007/10/12/db-designer-fork-una-herramienta-para-todo-desarrollador/.

12. FUNDAMENTOS DE BASE DE DATOS . [En línea] [Citado el: 7 de noviembre de 2007.] http://jeenrike.blogspot.com/2006/07/4-dbdesigner.html.

13. PostgreSQL, BBDD libre. [En línea] [Citado el: 7 de noviembre de 2007.] http://www.deambulando.com/2007/01/15/postgre/.

14. BASE DE DATOS. [En línea] [Citado el: 7 de noviembre de 2007.] http://www.wizhosting.com/faq/faq.php?print=true&cat_name=BASE%20DE%20DATOS&category_id=5

15. Programación en el lenguaje Java . [En línea] [Citado el: 12 de noviembre de 2007.] http://www.sc.ehu.es/sbweb/fisica/cursoJava/Intro.htm.

16. Conozca el nuevo NetBeans. [En línea] [Citado el: 13 de noviembre de 2007.] http://www.sun.com/emrkt/innercircle/newsletter/latam/0207latam_feature.html.

17. **Eduardo.** Clientes gráficos para MySQL. [En línea] [Citado el: 12 de noviembre de 2007.] http://www.mysql-hispano.org/page.php?id=21&pag=10.

18. Ubuntu: GNU/Linux para seres humanos . [En línea] [Citado el: 12 de noviembre de 2007.] http://doc.ubuntu-es.org/Filosof%C3%ADa_Ubuntu.

19. Curso de PHP. [En línea] [Citado el: 12 de noviembre de 2007.] http://www.adrformacion.com/cursos/php/leccion1/tutorial2.html.

20. Validación de los modelos de relación dosis-respuesta. [En línea] [Citado el: 15 de enero de 2008.] http://www.fao.org/docrep/009/y4666s/y4666s0c.htm.

BIBLIOGRAFÍA:

1. Acceso a fuentes de información genómica y herramientas ...(Consultado el 8 de noviembre)Disponible en:http://bvs.isciii.es/bibgen/Actividades/curso_virtual/Ftes_informacion/Biologiasistemas.htm
2. Biología de Sistemas: Doctorado del Departamento de Bioquímica de Sistemas (Consultado el: 8 de noviembre) Disponible en: http://www.uvic.cat/eps/dept/biologiasistemes/es/inici.html#professors.
3. Características de OpenUP (Consultado el miércoles 12 de diciembre) Disponible en: http://10.34.20.5:5800/OpenUP/
4. Características de UML (Consultado el miércoles 12 de diciembre)Disponible en: http://www.abcdatos.com/tutoriales/tutorial/l7157.html
5. Características de Visual Paradigm (Consultado el miércoles 12 de diciembre) Disponible en:http://www.versioncero.com/noticia/210/visual-paradigm-for-uml.
6. Características de DBDesigner (Consultado el miércoles 12 de diciembre)Disponible en: http://www.fileheaven.com/descargar/dbdesigner-4/25815.htm
7. Características de PostgreSQL (Consultado el martes 12 de febrero)Disponible en: http://www.sobl.org/traducciones/practical-postgres/node12.html
8. Características de MySQL (Consultado el martes 12 de febrero)Disponible en: http://www.webestilo.com/mysql/intro.phtml
9. Características de XML (Consultado el martes 12 de febrero)Disponible en: http://www.dcc.uchile.cl/~rbaeza/inf/xml.html
10. Características de Java (Consultado el 21 de mayo)Disponible en: http://www.gnu.org/copyleft/fdl.html
11. Características del lenguaje SQL (Consultado el viernes 11 de junio)Disponible en: http://www.desarrolloweb.com/directorio/bases_de_datos/lenguaje_sql/
12. Características de NetBeans (Consultado el sábado 31 de mayo)Disponible en: http://www.netbeans.org/index_es.html
13. Características de Hibernate(Consultado el sábado 31 de mayo)Disponible en: http://mundogeek.net/archivos/2007/01/27/hibernate/
14. Características del EMS Data Generator for MySQL (Consultado el lunes 9 de junio)Disponible en:http://www.topshareware.com/EMS-Data-Generator-for-MySQL-download-44297.htm
15. Características del MySQL Manager (Consultado el lunes 9 de junio)Disponible en: http://www.beupdated.net/es/Business/Databases-Tools/EMS-MySQL-Manager-Professional-

for-Windows-14029.html

16. Características de Ubuntu (Consultado el martes 12 de febrero)Disponible en: http://www.guadalinex.org/guadapedia/index.php/Ubuntu
17. Características del Patrón DAO (Consultado el martes 13 de mayo)Disponible en: http://weblogs.javahispano.org/hip/
18. Características del PhpMyAdmin (Consultado el martes 13 de mayo)Disponible en: http://www.desarrolloweb.com/articulos/844.php
19. Características del PhpPgAdmin (Consultado el martes 13 de mayo)Disponible en: http://www.vivaphp.com.ar/soft/phppgadmin-3-5.html
20. Normalización de una base de datos (Consultado el sábado 5 de abril)Disponible en: http://www.trucostecnicos.com/trucos/ver.php?id_art=278

GLOSARIO TÉRMINOS

CENPALAB: Centro de Ingeniería Genética y Biotecnología

CIGB: Centro de Ingeniería Genética y Biotecnología

BIOCEN: Centro Nacional de Biopreparados

CIM: Centro de Inmunología Molecular

SED: Sistema de Ecuaciones Diferenciales

UML: Lenguaje Unificado de Modelado

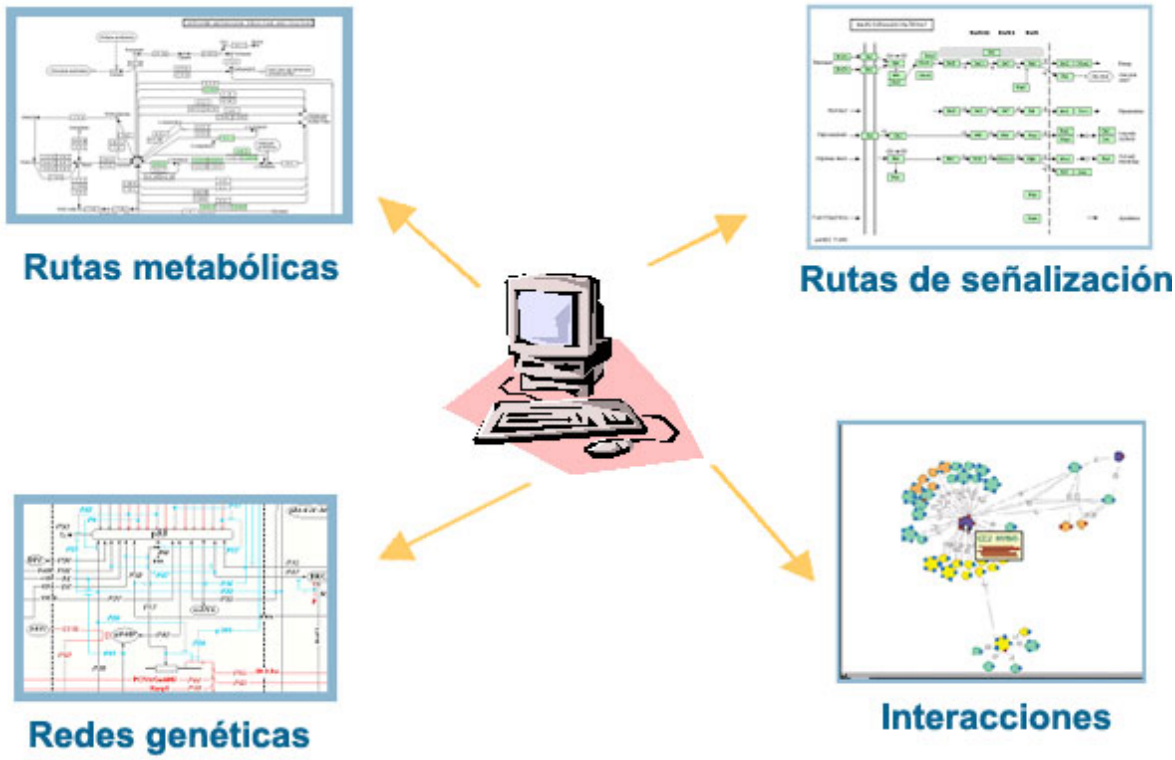
POO: Programación Orientada a Objetos

DAO: Objeto de Acceso a Datos

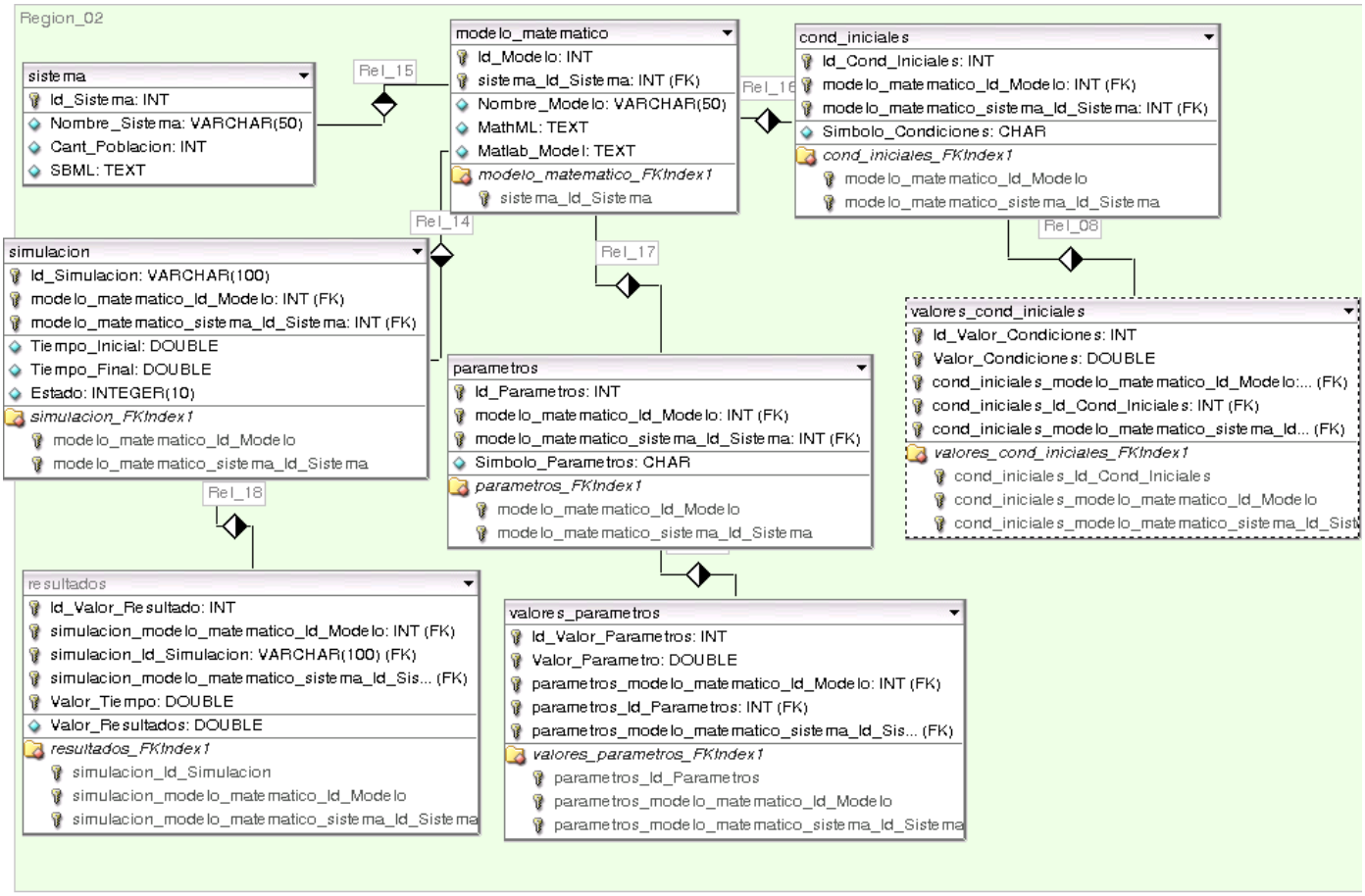
SBML: Lenguaje de marcado de sistemas biológicos.

DBMS: Sistema manejador de bases de datos

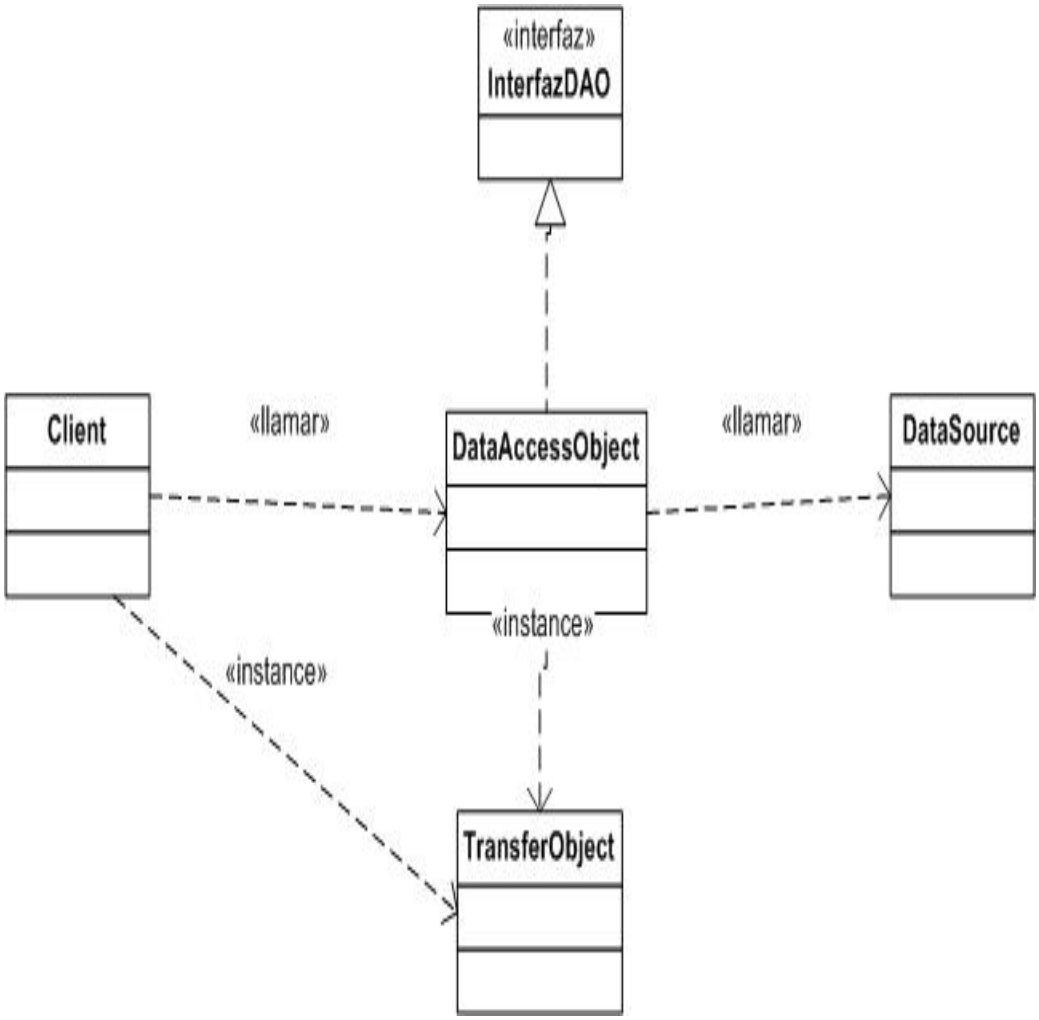
ANEXOS



Concepto de Biología de Sistemas [Anexo 1]

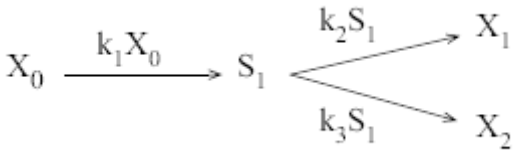


Diseño de BD BioSyS 1.0 [Anexo 2]



Patrón DAO [Anexo 3]

Archivo SBML




```
<?xml version="1.0" encoding="UTF-8"?>
<sbml xmlns="http://www.sbml.org/sbml/level2" level="2" version="1">
<model id="Branch">
<notes>
<body xmlns="http://www.w3.org/1999/xhtml">
<p>Simple branch system.</p>
<p>The reaction looks like this:</p>
<p>reaction-1: X0 -> S1; k1*X0;</p>
<p>reaction-2: S1 -> X1; k2*S1;</p>
<p>reaction-3: S1 -> X2; k3*S1;</p>
</body>
</notes>
<listOfCompartments>
<compartment id="compartmentOne" size="1"/>
</listOfCompartments>
<listOfSpecies>
<species id="S1" initialConcentration="0" compartment="compartmentOne"
boundaryCondition="false"/>
<species id="X0" initialConcentration="0" compartment="compartmentOne"
boundaryCondition="true"/>
<species id="X1" initialConcentration="0" compartment="compartmentOne"
boundaryCondition="true"/>
<species id="X2" initialConcentration="0" compartment="compartmentOne"
boundaryCondition="true"/>
</listOfSpecies>
<listOfReactions>
<reaction id="reaction_1" reversible="false">
<listOfReactants>
<speciesReference species="X0"/>
</listOfReactants>
<listOfProducts>
<speciesReference species="S1"/>
</listOfProducts>
<kineticLaw>
<math xmlns="http://www.w3.org/1998/Math/MathML">
<apply>
<times/>
<ci> k1 </ci>
<ci> X0 </ci>
```

```
</apply>
</math>
<listOfParameters>
<parameter id="k1" value="0"/>
</listOfParameters>
</kineticLaw>
</reaction>
<reaction id="reaction_2" reversible="false">
<listOfReactants>
<speciesReference species="S1"/>
</listOfReactants>
<listOfProducts>
33
<speciesReference species="X1"/>
</listOfProducts>
<kineticLaw>
<math xmlns="http://www.w3.org/1998/Math/MathML">
<apply>
<times/>
<ci> k2 </ci>
<ci> S1 </ci>
</apply>
</math>
<listOfParameters>
<parameter id="k2" value="0"/>
</listOfParameters>
</kineticLaw>
</reaction>
<reaction id="reaction_3" reversible="false">
<listOfReactants>
<speciesReference species="S1"/>
</listOfReactants>
<listOfProducts>
<speciesReference species="X2"/>
</listOfProducts>
<kineticLaw>
<math xmlns="http://www.w3.org/1998/Math/MathML">
<apply>
<times/>
<ci> k3 </ci>
<ci> S1 </ci>
</apply>
</math>
<listOfParameters>
<parameter id="k3" value="0"/>
</listOfParameters>
```

```
</kineticLaw>
</reaction>
</listOfReactions>
</model>
</sbml>
```

Archivo SBML [Anexo 4]

```
<math>
<apply>
<power/>
<apply>
  <plus/>
  <ci>x</ci>
 <ci>y</ci>
</apply>
</apply>
</math>
```

x + y MathML [Anexo 5]