UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Tema: Análisis y Diseño de la nueva versión del simulador de física FISIM

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

Autor: Lisbet Alvarez Mendoza

Tutor: Yosnel Herrera Martínez

Ciudad de La Habana, Cuba
Junio 2011

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo a la facultad 4 de la Universidad de las Ciencias Informáticas, así como a dicho centro, a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente declaración de autoría a los ____ días del mes de junio del año 2011.

_____	_____
Lisbet Alvarez Mendoza	Ing. Yosnel Herrera Martínez
Firma del autor	Firma del tutor

AGRADECIMIENTOS

Agradecerles primeramente a mis padres que me dieron la vida y me encaminaron todos estos años para llegar a ser la persona que en estos momentos soy. Gracias por el apoyo que me dieron día a día, por hacer realidad todos mis sueños, por darme el cariño que me dieron y por ser las personas admirables que son, ojalá en el futuro pudiera ser aunque sea la mitad de lo que han sido. Ustedes son los mejores padres del mundo y por eso los quiero tanto.

En segundo lugar agradecerles a dos personas que han estado junto a mi toda mi vida, mis dos hermanas, ya que han sido para mí la fuerza que me levanta cada día y me da deseos de enfrentar los obstáculos de la vida. Gracias a las dos por estar siempre ahí cuando lo necesité y por haber sido las hermanas excelentes que fueron. El cariño que me han dado no tiene comparación, no hay nada en este mundo que me dé más felicidad que estar junto a ustedes, por eso siempre van a ser mis queridas tatas.

En tercer lugar agradecerle a una persona que no ha estado junto a mí toda la vida, pero que llegó hace más de un año y se quedó con mi amor, y espero que comparta conmigo lo que nos queda de existencia, mi novio. Gracias a ti por todo el apoyo que me has dado, por estar a mi lado dándole frente al futuro, por darme el amor que me has dado, por ser la persona excelente que eres. Te amo.

Muchas gracias también al resto de mi familia, no los tengo a todos juntos pero siempre han sabido dejarme claro que me quieren y me apoyan, yo tengo un lugarcito en mi corazón para cada uno de ellos.

Por último y no por eso menos importantes, agradecerles a todos mis amigos, con ustedes he pasado momentos inolvidables, he vivido experiencias únicas, he aprendido mucho. Por desgracia dejaré de ver a muchos de ustedes cuando nos graduemos, pero siempre voy a llevar el recuerdo de todos conmigo. Gracias a todos por haberme brindado su amistad y que tengan éxitos en la vida.

Del Autor.

RESUMEN

Con el transcurso de los años en Cuba la producción de software ha aumentado considerablemente lográndose el desarrollo de productos de alta calidad. En la actualidad en el sector de la educación, se ha hecho casi imprescindible el uso de las Tecnologías para la Información y las Comunicaciones para el desarrollo del proceso de enseñanza – aprendizaje, por lo que los software educativos tienen gran demanda en estos momentos. En la Universidad de las Ciencias Informáticas se está desarrollando la plataforma ZERA, esta es una plataforma educativa para la gestión del aprendizaje que tiene sus orígenes en una concepción pedagógica denominada Hiperentornos de aprendizaje, la misma contiene un grupo de productos educativos integrados a ella. FISIM es un software que se desarrolló en uno de los subsistemas de esta plataforma, este es un simulador de Física que permite realizar simulaciones de las leyes y fenómenos físicos que tienen lugar en la naturaleza.

En estos momentos este simulador posee una serie de deficiencias las cuales constituyen un problema en el seguimiento de los profesores en el proceso de enseñanza – aprendizaje de los estudiantes. Debido a las insuficiencias del simulador es que surge este trabajo en el cual se realiza una investigación para concebir una documentación de ingeniería referente al análisis y el diseño del simulador FISIM que da una solución a las necesidades actuales del mismo, la cual va a servir de guía para el futuro trabajo de los desarrolladores en la implementación de este software.

ABSTRACT

Over the years in Cuba has increased considerably and succeeded software production developing high quality products. Currently, in the educational sector has become indispensable the use of Information and Communications Technologies to development the teaching-learning process, because the educational software's are in great demand at the moment. At the University of Information Sciences is developing the ZERA platform, which is an educational platform to learning management that has its roots in a pedagogical conception called hyper environment of learning, it contains a group of educational products that are integrated into it. FISIM is software that was developed in one of the subsystems of this platform. This is a physics simulator that allows the simulation of the laws and physical phenomena that occur in nature.

At present this simulator has a number of deficiencies which is a problem in the monitoring of the teachers in the teaching-learning process of the students. Due to the inadequacies of the simulator is that this work emerged in which a research is conducted to conceive engineering documentation relating to the analysis and design of FISIM simulator that gives a solution to the current needs, which will guide the future work of developers in the implementation of this software.

ÍNDICE DE CONTENIDO

INTRODUCCIÓN

Hoy en día es significativa la influencia de las nuevas tecnologías de la información y las comunicaciones sobre numerosos sectores de la economía y la sociedad, así como la evolución de las mismas con el transcurso de los años. Los resultados de su aplicación son notables debido al amplio espectro de posibilidades que genera y por lo tanto su aceptación es un hecho confirmado e irrefutable, aunque están indiscriminadamente distribuidos y son una utopía para muchos. La esfera de la Educación ha experimentado un gran cambio con la introducción de las tecnologías en aulas y por tanto la demanda de software educativo se ha incrementado considerablemente. Por esta razón se han estado desarrollando y poniendo a disposición de la población y las escuelas de todo el mundo una gran gama de productos educativos. Un software educativo es un sistema informático destinado a apoyar el proceso de enseñanza-aprendizaje. Este tipo de sistemas ha demostrado ser tan efectivo que en la actualidad una computadora es un medio casi imprescindible en el proceso de la educación, debido a las ventajas que trae consigo el uso de las mismas en este proceso.

La Universidad de las Ciencias Informáticas (UCI) existente en nuestro país es una institución que se ha convertido en un bastión fundamental en la gestión de aplicaciones destinadas a la educación, un ejemplo de ello es la plataforma ZERA que se está desarrollando en la misma. Esta es una plataforma educativa para la gestión del aprendizaje que tiene sus orígenes en una concepción pedagógica denominada Hiperentornos de aprendizaje, propuesta y desarrollada por pedagogos y especialistas del Ministerio de Educación de Cuba. Los hiperentornos no son más que una mezcla de diferentes tipos de software educativos sustentado en tecnología hipermedia, estos constituyeron un modelo de medio de enseñanza para el apoyo al proceso de enseñanza – aprendizaje, además como fundamento teórico y conceptual del desarrollo de las colecciones cubanas de software educativo.

ZERA está concebida para ser una plataforma capaz de adaptarse a los procesos del negocio de cualquier institución o escuela. Permite la gestión de los hiperentornos de aprendizajes, la visualización de estos, y finalmente la gestión del aprendizaje. Esta además de otras funcionalidades permite un seguimiento del proceso de enseñanza y aprendizaje de los estudiantes a partir de un conjunto de herramientas como un subsistema para la Gestión de Reportes sobre la actividad académica de los estudiantes, estadísticas y trazas. La misma se divide en seis subsistemas, los cuales se encuentran estrechamente interrelacionados. Uno de los subsistemas existentes en la plataforma es el subsistema Simuladores. En el mundo existe una gama bien amplia de simuladores de Matemática, Física, Química y Biología destinados a la educación. Un simulador es un software que permite la simulación de un sistema, reproduciendo su comportamiento (1). Los simuladores son quizás las aplicaciones que más aprovechan las especificaciones de la computadora

como recurso de aprendizaje y que cada día se extiende más en áreas de la educación. El simulador permite al estudiante aprender de manera práctica, a través del descubrimiento y la construcción de situaciones hipotéticas. Este tiene la ventaja de permitirle al estudiante desarrollar la destreza mental o física a través de su uso y ponerlo en contacto con situaciones que pueden ser utilizadas de manera práctica, si son usados en trabajo colaborativo, estimulan el trabajo en equipo al exigir la discusión del tema.

En el subsistema Simuladores se desarrolló un software llamado FISIM, este es un simulador de Física el cual permite realizar simulaciones de las leyes y fenómenos físicos que tienen lugar en la naturaleza. Este simulador fue desarrollado para permitir a los docentes representar los procesos físicos con los cuales el estudiante ha de interactuar posteriormente de manera activa y consciente. Se enmarca en la creación de medios de enseñanza – aprendizaje utilizando las tecnologías, centrándose en el desarrollo de software para la simulación de leyes y procesos físicos. La versión que existe actualmente de FISIM es una aplicación de escritorio que cuenta con tres secciones, una de ellas es una hoja lógica destinada a construir y validar el modelo matemático que describe el proceso a estudiar, permitiéndoles a los estudiantes y profesores crear nuevos modelos, guardar un modelo creado o abrir modelos existentes. Otra sección es un escenario en el que se incluyen los elementos para la visualización de las simulaciones y la otra sección es un administrador de componentes el cual posee una lista de los elementos que han sido incorporados al escenario, los cuales pueden ser modificados en esta sección.

En estos momentos la aplicación presenta una serie de deficiencias, una de ellas es que actualmente cualquier usuario puede ejecutarla y trabajar con la misma una vez que se haya descargado en una máquina, ya que esta no cuenta con un sistema de autenticación. Otra es la complejidad del modelo que se construye en la hoja lógica, pues este tiene una estructura definida, compuesta por una sección de fórmulas y una sección de datos, por lo que si el estudiante o profesor no sabe la estructura correcta del modelo lo podría introducir con errores, provocando que estos no pueda realizar la simulación correctamente. Por otra parte las simulaciones que se realizan o cargan no se encuentran agrupadas por temas, o sea no se tiene definido cuándo una simulación es sobre un fenómeno electromagnético, uno termodinámico o de otro tipo, esto produce un déficit en cuanto a las funcionalidades y los elementos necesarios para la realización de una simulación específica, pues cuando se realiza o carga una simulación en la barra de herramientas no aparecen los elementos necesarios para realizar una simulación atendiendo el tipo de esta, en lugar de eso aparecen un grupo de elementos generales, imposibilitando que el usuario pueda realizar funcionalidades específicas de una simulación. Todas estas insuficiencias constituyen una deficiencia en el seguimiento de los profesores en el proceso de enseñanza – aprendizaje de los alumnos.

Como consecuencia se plantea el siguiente **problema a resolver:** ¿Cómo mejorar la estructura de las simulaciones del simulador FISIM en función del beneficio del proceso enseñanza – aprendizaje de los usuarios que interactúan con el mismo?

Para dar solución al problema que queremos resolver se plantea como **objetivo general de la investigación** desarrollar el análisis y diseño del simulador FISIM de forma tal que traiga beneficios en el proceso enseñanza – aprendizaje posibilitando su posterior implementación.

Los **objetivos específicos** que permitirán dar cumplimiento al objetivo general son los siguientes:

- Realizar un estudio del estado del arte referente a los simuladores.
- Definir los nuevos requerimientos funcionales y no funcionales del simulador FISIM.
- Analizar y diseñar una propuesta de solución que permita crear nuevas funcionalidades para el simulador FISIM.

Como **objeto de estudio** de esta investigación se tienen los procesos asociados al desarrollo de simuladores físicos.

Nuestro **campo de acción** lo constituye el proceso de modelado del simulador FISIM existente en la plataforma ZERA.

Nuestra **idea a defender** es la que se expone a continuación: Si se realiza el análisis y diseño de un simulador físico que posea una mayor organización de los elementos y secciones que pone a disposición de los usuarios, entonces se podrá en un futuro desarrollar el mismo, logrando que este incida de una forma dinámica e interactiva en el proceso de enseñanza – aprendizaje de la asignatura Física de los estudiantes.

Para dar cumplimiento a los objetivos de la investigación se plantean las siguientes **tareas de investigación:**

- Realización de un análisis bibliográfico para conformar la base teórica metodológica de la investigación.
- Realización de una investigación referente a simuladores físicos existentes en Cuba y otros países teniendo en cuenta las necesidades del simulador FISIM.
- Investigación de las tendencias y las tecnologías actuales más utilizadas en el desarrollo de simuladores físicos.
- Identificación de nuevos requerimientos.

- Descripción de las nuevas funcionalidades del sistema.
- Realización del análisis de una propuesta de solución.
- Realización del diseño de una propuesta de solución.

Métodos científicos:

Todos los métodos científicos utilizados para dar cumplimiento a los objetivos y tareas del trabajo de diploma son categorizados como métodos teóricos:

- **Método dialéctico:** Se evidencia en la realización de todas las tareas, pues permite abordar el objeto de investigación considerándolo en su desarrollo y teniendo en cuenta la interrelación entre todos sus componentes. Busca las contradicciones existentes y explica los cambios cualitativos que se producen en el sistema y dan paso a un nuevo objeto. Como parte del método dialéctico se utilizarán la inducción y la deducción, también el análisis y la síntesis como procedimientos para comprender la esencia del fenómeno que se investiga.
- **Método histórico – lógico:** Se manifiesta en el momento de estudiar las metodologías, herramientas y lenguajes de modelado, así como en la investigación del modelado de simuladores. Permite abordar el objeto desde el punto de vista diacrónico y sincrónico. Permite, además, comprender los fundamentos teóricos que sustenta la propuesta e identificar la pertinencia de cada uno de ellos para la realización del análisis y del diseño.
- **Método sistémico:** Lo podemos ver principalmente en la realización del estudio del arte de las principales metodologías de desarrollo que permiten realizar una correcta captura de requisitos. Permite abordar todos los componentes del diseño en su interrelación, y establecer relaciones de subordinación y coordinación entre los mismos.
- **Método de la modelación:** Es empleado para especificar el enfoque del sistema, el cual permite crear abstracciones con el objetivo de explicar la realidad. Esto se evidencia con el uso del lenguaje de modelado UML, el que permitirá reflejar la estructura, relaciones internas y características de la solución a través de diagramas.
- **Método empírico de la observación:** Es utilizado para verificar las deficiencias del simulador actual a partir de indicadores preestablecidos, específicamente a la hora de analizar la solución del FISMAT existente para definir los problemas que presenta y las mejoras que necesita.

Estructuración del contenido:

El documento está conformado por tres capítulos, los cuales se encuentran estructurados de la siguiente manera:

- **Capítulo 1. Fundamentación Teórica:** Se aborda el estudio del arte del tema que se investiga. También las metodologías de desarrollo de software más conocidas actualmente a nivel mundial, entre las que se realiza un estudio comparativo con el objetivo de escoger una de ellas. Además se proponen las herramientas a utilizar para el modelado.

- **Capítulo 2. Características del Sistema:** Se elabora una descripción de los procesos que son objeto de automatización que están presentes en el simulado. Además se especifican los requisitos funcionales y no funcionales del simulador y se definen también las descripciones de los casos de uso del sistema.

- **Capítulo 3. Análisis y Diseño del Sistema:** Está constituido por los diagramas de clases del análisis y los diagramas de clases del diseño del simulador, así como los diagramas de interacción correspondientes a cada caso de uso.

1. CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

La simulación es un método muy usado actualmente en numerosos campos, es utilizada para representar un fenómeno mediante otro que lo hace mucho más simple y entendible. No solamente se enfoca en el modelado de sistemas de alto nivel, pues ha evolucionado hasta el punto de que se centra más en la creación de prototipos de software, consiguiendo un ahorro de tiempo y dinero a las empresas en su desarrollo. Los principales sistemas informáticos que hacen uso de esta técnica son los simuladores, los cuales son muy diversos y utilizados en los procesos de enseñanza en dependencia de la clasificación de los mismos.

Este capítulo tiene como principal objetivo situar al lector en el tema del trabajo, abordándose en él conceptos asociados con el contenido del mismo. También se hace referencia a algunos simuladores existentes en nuestro país o en el resto del mundo, brindando sus características generales. Además se dan a conocer algunas metodologías y herramientas para el desarrollo de los simuladores.

1.1. Conceptos asociados

Numerosos son los autores que se han dedicado a darle a conocer al mundo los conceptos de Simulador, Plataforma Educativa, Software Educativo e Hiperentorno de Aprendizaje, cada uno desde su punto de vista y aplicado sus conocimientos a la rama en la que se especializan. Durante nuestra investigación se consultaron algunos de ellos para tomar aquellos conceptos que se aplicaran más al tema de la misma, por lo que se decidió tomar los que se explican en los subepígrafes 1.1.1, 1.1.2, 1.1.3 y 1.1.4, basándonos en la similitud de los mismos con lo que se quiere transmitir a los lectores del presente trabajo.

1.1.1. Simulador

Un simulador es un sistema de software que imita tanto el comportamiento de un sistema del mundo real, como los procesos de entrada que manejan o controlan el sistema simulado. Las simulaciones pueden usarse para obtener conocimiento acerca de sistemas existentes, para predecir su comportamiento y para propósitos de enseñanza. (1)

Una de las características claves de la simulación es la habilidad de modelar el comportamiento del sistema considerando el progreso del tiempo. Las características inherentes a las aplicaciones que simulan algún sistema deben ser consideradas desde la etapa de análisis para reflejar los requerimientos asociados durante la etapa de diseño del sistema. Los simuladores constituyen una poderosa herramienta para lograr el cumplimiento de diversos objetivos de enseñanza en varias disciplinas. (1)

A pesar de que la simulación como técnica de enseñanza - aprendizaje surge antes de la aparición de la primera computadora electrónica y que su valor es reconocido no sólo en el campo del software educativo, la simulación por computadoras ofrece grandes ventajas en el proceso docente pues a través de la misma se puede lograr imitar, reproducir, replicar, con un alto grado de similitud, procesos, objetos y fenómenos del mundo real, algunos de los cuales no podrían ser presentados a los alumnos y otros, mucho menos permitirían interactuar con ellos.

Tipos de Simuladores:

Los simuladores pueden ser divididos en cuatro grupos o categorías las cuales son (2):

- **Simuladores físicos:** Son aquellos en los que un objeto físico es presentado a través de la pantalla para que el estudiante pueda usarlo o aprender de él. Aunque las simulaciones físicas son muy usuales, generalmente estas juegan un papel secundario y existen para su empleo en simulaciones procedimentales.

- **Simuladores procedimentales:** Más que ser usados para que el estudiante aprenda sobre un objeto físico, proporcionan una vía para que el alumno adquiera los conocimientos y habilidades necesarias para aprender a usarlo.

- **Simuladores situacionales:** Son empleados para reflejar las actitudes y el comportamiento del ser humano ante diferentes situaciones y explorar los efectos de diferentes tratamientos de una situación y les posibilitan al estudiante jugar diferentes papeles en la propia simulación por lo que en casi toda simulación situacional el alumno es una parte integrante de la misma y juega uno de los papeles principales.

- **Simuladores de procesos:** Son sistemas en los que el estudiante, a diferencia de las simulaciones situacionales, no juega un papel activo, ni constantemente manipula como en las simulaciones físicas o procedimentales sino que el alumno solamente se limita a seleccionar diferentes valores para cada uno de los parámetros que se contemplan en la simulación y luego observa cómo ocurre el proceso.

1.1.2. Plataforma Educativa

Son herramientas que permiten interactuar con varios usuarios a la vez con fines pedagógicos, usadas para organizar e implantar cursos en línea o actividades educativas en general. Es un Software que permite a un profesor, crear un espacio en la web donde sea capaz de colgar todos los materiales que crea son de

interés para los usuarios que recibirán el curso, enlazar tantos otros, incluir foros, wikis, recibir tareas de sus alumnos, desarrollar cuestionarios, promover debates, chats, obtener estadísticas de evaluación y uso (3). Es aquella que permite transmitir conocimientos y adquirir habilidades de forma personalizada a través de Internet. También se puede definir como un espacio al cual se ha adaptado una aplicación TIC que vincula los procesos de enseñanza-aprendizaje a un modelo pedagógico a través de un entorno virtual.

En conclusión, las plataformas educativas son un sistema informático localizado en un sitio web que suele ser de acceso restringido con el fin de identificar el perfil del usuario. Este sistema informático habilita un espacio de trabajo compartido por alumnos y profesores en el cual se intercambian documentos y actividades en el proceso enseñanza-aprendizaje.

1.1.3. Software Educativo

Según Jaime Sánchez (1999) define el concepto genérico de Software Educativo como cualquier programa computacional cuyas características estructurales y funcionales sirvan de apoyo al proceso de enseñar, aprender y administrar (4). Un ejemplo más restringido de Software Educativo lo define como aquel material de aprendizaje especialmente diseñado para ser utilizado por una computadora en los procesos de enseñar y aprender. Programas para ordenadores creados con la finalidad específica de ser utilizados como medio didáctico, es decir para facilitar los procesos de enseñanza y aprendizaje.

1.1.4. Hiperentorno de Aprendizaje

Un hiperentorno de aprendizaje se define como una modalidad informática que se sustenta en la tecnología hipermedia y en el que están presentes un conjunto de elementos representativos de diversas tipologías de software educativo.

1.2. Sistemas similares

En el mundo existen varios simuladores físicos destinados a diversos niveles de escolaridad. Teniendo en cuenta las necesidades del simulador FISIM se realizó un estudio de otros simuladores físicos existentes para buscar las funcionalidades que estos tenían que pudieran agregársele al mismo. Los sistemas estudiados fueron MyPhysicsLab, PHET, PHUN, Interactive Physics y Cocodile Phisics. A continuación el los subepígrafes 1.2.1, 1.2.2, 1.2.3, 1.2.4 y 1.2.5 se muestran las características de los mismos.

1.2.1. MyPhysicsLab

Fue desarrollado desde 2001. Está dirigido a todos los niveles de educación donde se imparte la asignatura de física. Se distribuye como un Applet de Java. Es código abierto y descargable. Se puede cambiar parámetros como la constante elástica. Estas simulaciones físicas forman parte de una colección de más de 24 fenómenos físicos independientes.

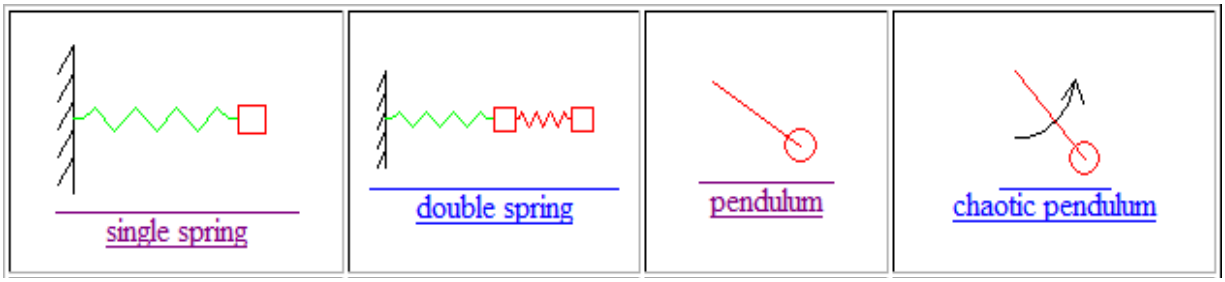


Figura 1: Simulador MyPhysicsLab

1.2.2. PHET

Es un interesante conjunto de simuladores didácticos e interactivos diseñados para enseñar los conceptos básicos de diferentes fenómenos físicos. Con Phet se puede experimentar con la gravedad, con tiros parabólicos, con señales de radio y efectos electromagnéticos, construir sencillos circuitos eléctricos, representar ecuaciones en gráficas, experimentar con señales láser, entre otras posibilidades. Cada simulador de Phet incluye los controles necesarios para configurar los parámetros básicos del fenómeno que estudia. Todos los simuladores están desarrollados en el lenguaje de programación Java. Para ponerlos en marcha sólo se necesita un navegador y la máquina virtual de Java correspondiente. (5)

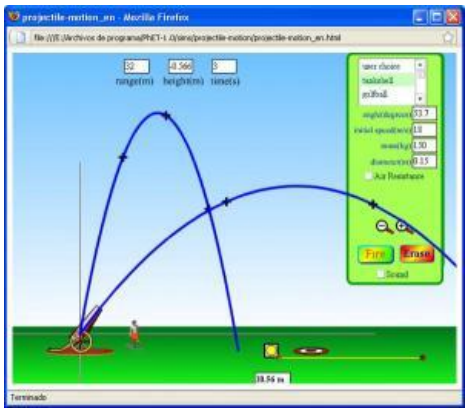


Figura 2: Simulador PHET

1.2.3. PHUN

Creado por Emil Ernerfeldt, un estudiante sueco y se llama Phun (juego de palabras, divertido). Incluye gravedad, fluidos y muchas más opciones para crear pequeños mundos físicos virtuales. Es un programa que se puede descargar libremente para usos no comerciales. Funciona en Windows y Linux, próximamente se podrá utilizar también con Mac. (6)

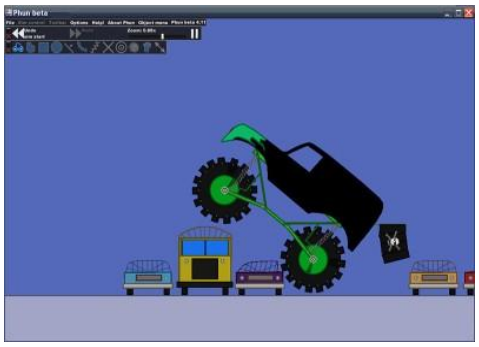


Figura 3: Simulador PHUN

1.2.4. Interactive Physics

Interactive Physics es como una pizarra electrónica, donde se ven imágenes y gráficos que cobran vida, facilitando a los alumnos el entendimiento de los fenómenos físicos. Con este programa el profesor puede simular sus experimentos de física en la computadora. Por ejemplo se pueden variar los parámetros de fricción, gravedad, tiempo, fuerza, velocidad inicial, etc. y ver las variables en forma instantáneas. Se pueden demostrar principios básicos de la física, como así también, explorar nuevas situaciones. Es una de las herramientas ideales para la creación de experiencias educativas y experimentos.

Con las herramientas provistas se pueden abordar estudios de estabilidad, caída libre, velocidad con respecto a distintos sistemas de referencia, aceleración, primera y segunda ley de Newton, distinción entre masa y peso, cinemática y dinámica rotacional, colisiones, trabajo y potencia, etc. También se pueden crear nuevas experiencias definiendo los entornos y las condiciones iniciales.

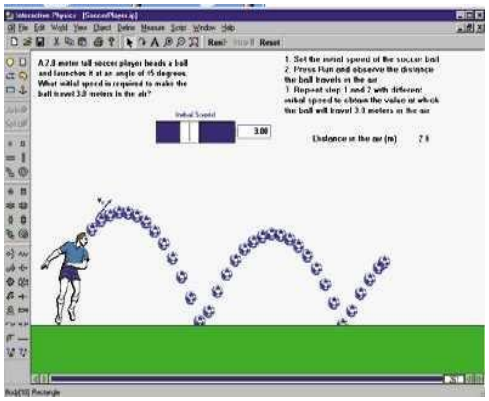


Figure 4: Simulador Interactive Physics

1.2.5. Cocodile Phisics

Fue desarrollado por Cocodrileen en el año 1994. Es una aplicación de escritorio, multiplataforma (Windows/Linux), se encuentra disponible en inglés y español, y permite realizar experimentos físicos de: electricidad, ondas, movimiento, fuerza y óptica. No tiene integración a otros sistemas como MOODLE u otras tecnologías e-learning.

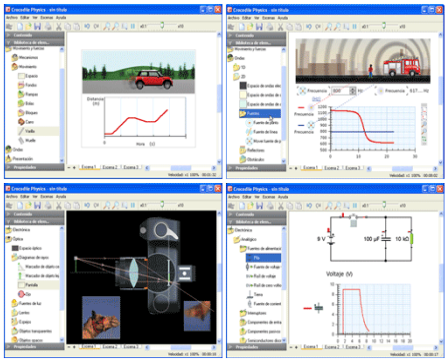


Figure 5: Simulador Cocodile Phisics

1.2.6. Aportes de los sistemas similares

Encontrar funcionalidades necesarias para FISIM no es una tarea fácil, luego de indagar en las características de los simuladores anteriormente mencionados se decidió no utilizar ninguna de las funcionalidades que poseen los sistemas MyPhysicsLab, PHUN e Interactive Physics, pues las mismas no nos brindaban ventajas para nuestro software. Por otra parte se tomaron algunas de las que poseen los sistemas PHET y Cocodile Phisics.

En el caso de PHET tomamos lo referente los experimentos con la gravedad y con tiros parabólicos, aplicando las características de estas funcionalidades a las simulaciones de mecánica. Por otra parte de Cocodile Phisics se tomó lo referente a los experimentos de óptica, los cuales son de ayuda para las simulaciones de óptica que se desean generar en FISIM, así como algunas de las características de ese sistema por ser una aplicación de escritorio como nuestro software.

1.3. Metodologías de Desarrollo de Software

1.3.1. Rational Unified Process (RUP)

Es una metodología pensada para la Ingeniería de Software que va más allá del análisis y diseño orientado a objetos, para proporcionar un conjunto de técnicas que soportan el ciclo completo de desarrollo de software. Además de que unifica completamente a un equipo de desarrollo y optimiza la productividad de cada uno de los miembros del mismo, ayuda a los líderes de proyecto a incrementar su experiencia en el

desarrollo. Los verdaderos aspectos definitorios del proceso unificado se resumen en tres frases claves, dirigido por casos de uso, centrado en la arquitectura, e iterativo e incremental (7).

Iterativo e incremental: Donde cada fase se desarrolla en iteraciones, de forma tal que se pueda dividir en pequeños proyectos mejorando su comprensión y desarrollo. RUP propone que cada fase se desarrolle en iteraciones. Una iteración involucra actividades de todos los flujos de trabajo, aunque desarrolla fundamentalmente unos más que otros.

Dirigido por casos de uso: Los casos de uso reflejan lo que los usuarios futuros necesitan y desean, lo cual se capta cuando se modela el negocio y se representa a través de los requerimientos. A partir de aquí los casos de uso guían el proceso de desarrollo ya que los modelos que se obtienen, como resultado de los diferentes flujos de trabajo, representan la realización de los casos de uso.

Centrado en la arquitectura: La arquitectura muestra la visión común del sistema completo en la que el equipo de proyecto y los usuarios deben estar de acuerdo, por lo que describe los elementos del modelo que son más importantes para su construcción, los cimientos del sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo económicamente.

Fases (7)**:**

- **Inicio:** Alcanzar un acuerdo entre todos los interesados respecto a los objetivos del ciclo de vida para el proyecto, generar el ámbito de trabajo, el caso de negocio, síntesis de arquitectura posible y el alcance del proyecto.
- **Elaboración:** Establecimiento de la línea base para la arquitectura del sistema y proporcionar una base estable para el diseño y el esfuerzo de implementación de la siguiente fase, mitigando la mayoría de los riesgos tecnológicos.
- **Construcción:** Completar el desarrollo del sistema basado en la línea base de la arquitectura. En otras palabras lograr la funcionalidad operativa del software.
- **Transición:** Garantizar que el software esté listo para entregarlo a los usuarios, y lograr la aprobación cuanto antes para liberar el producto al mercado.

Flujos de Ingeniería (7)**:**

- **Modelo del negocio:** Describe los procesos de negocio, identifica quiénes participan y las actividades que requieren automatización.
- **Requerimientos:** Define qué es lo que el sistema debe hacer, para lo cual se identifican las funcionalidades requeridas y las restricciones que se imponen.

- **Análisis y Diseño:** Describe cómo el sistema será realizado a partir de las funcionalidades previstas y las restricciones impuestas (requerimientos), indica con precisión lo que se debe programar.
- **Implementación:** Define cómo se organizan las clases y objetos en componentes, cuáles nodos se utilizarán y la ubicación en ellos de los componentes y la estructura de capas de la aplicación.
- **Pruebas:** Busca los defectos del software a lo largo del ciclo de vida.
- **Despliegue:** Produce el release del producto y realiza las actividades (empaque, instalación, asistencia a usuarios, etc.) para entregarle el software a los usuarios finales.

Flujos de apoyo (7)**:**

- **Administración del proyecto:** Involucra actividades con las que se busca producir un producto que satisfaga las necesidades del cliente.
- **Administración de configuración y cambios:** Describe cómo controlar los elementos producidos por todos los integrantes del equipo de proyecto en cuanto a: utilización y actualización concurrente de elementos, control de versiones, etc.
- **Ambiente:** Contiene actividades que describen los procesos y herramientas que soportarán el equipo de trabajo del proyecto así como el procedimiento para implementar el proceso en una organización.

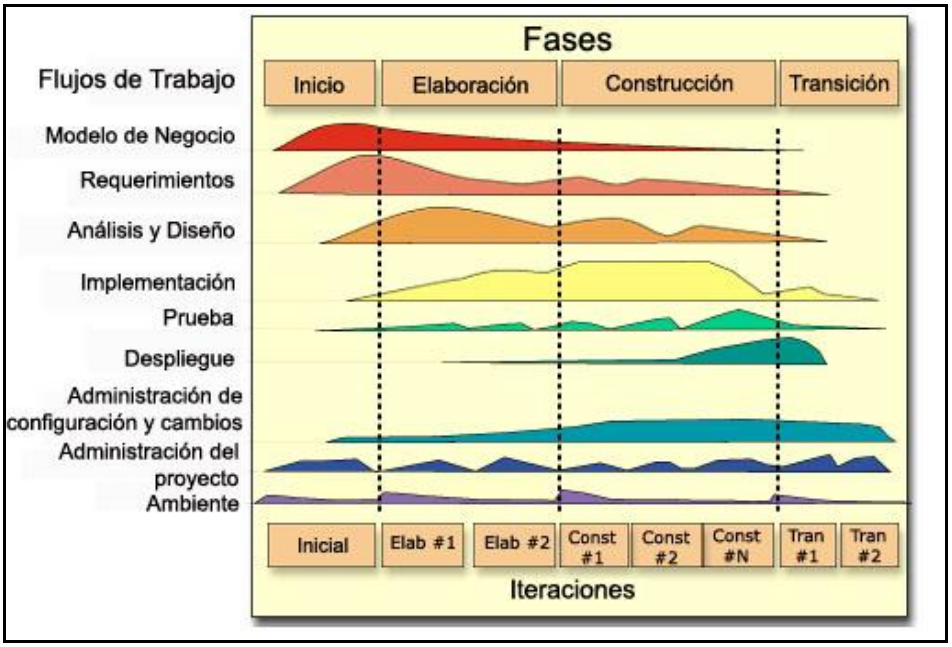


Figura 6: Fases e Iteraciones de la Metodología RUP

1.3.2. Programación Extrema (XP)

La Programación Extrema (XP) o Extreme Programming es otra de las metodologías de desarrollo de software que existen en la actualidad. Mientras que RUP intenta reducir la complejidad del software por medio de estructura y la preparación de las tareas pendientes en función de los objetivos de la fase y actividad actual, XP, como toda metodología ágil, lo intenta por medio de un trabajo orientado directamente al objetivo, basado en las relaciones interpersonales y la velocidad de reacción (8).

Extreme Programming se basa en la retroalimentación 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.

El ciclo de vida ideal de XP consta de seis fases: Exploración, Planificación de la Entrega (Release), Iteraciones, Producción, Mantenimiento y Muerte del Proyecto (9). Los objetivos de XP son muy simples: la satisfacción del cliente. Esta metodología trata de dar al cliente el software que él necesita y cuando lo necesita. Por tanto, debemos responder muy rápido a las necesidades del cliente, incluso cuando los cambios sean al final de ciclo de la programación. El segundo objetivo es potenciar al máximo el trabajo en grupo. Tanto los jefes de proyecto, los clientes y desarrolladores, son parte del equipo y están involucrados en el desarrollo del software (10). Entre sus principales características tenemos que:

La metodología se basa en (7):

- Pruebas Unitarias: Se basa en las pruebas realizadas a los principales procesos, de tal manera que, adelantándose en algo hacia el futuro, se puedan hacer pruebas de las fallas que pudieran ocurrir. Es como si se adelanta en la obtención de los posibles errores.
- Re-fabricación: Se basa en la reutilización de código, para lo cual se crean patrones o modelos estándares, siendo más flexible al cambio.
- Programación en pares: Una particularidad de esta metodología es que propone la programación en pares, la cual consiste en que dos desarrolladores participen en un proyecto en una misma estación de trabajo. Cada miembro lleva a cabo la acción que el otro no está haciendo en ese momento. Es como el chofer y el copiloto: mientras uno conduce, el otro consulta el mapa.

Lo fundamental en este tipo de metodología es (7):

- La comunicación, entre los usuarios y los desarrolladores
- La simplicidad, al desarrollar y codificar los módulos del sistema
- La retroalimentación, concreta y frecuente del equipo de desarrollo, el cliente y los usuarios finales.

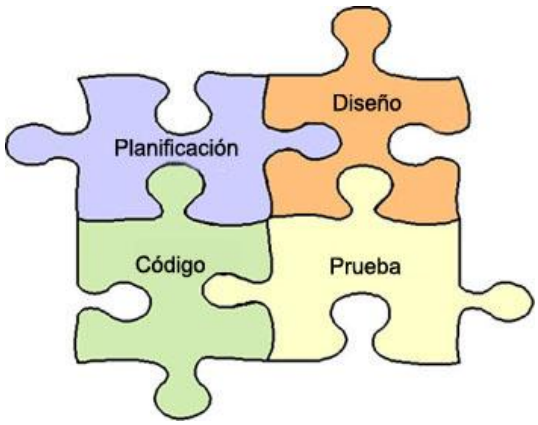


Figura 7: Metodología XP

1.4. Lenguajes de Modelado

1.4.1. Lenguaje de Modelado Unificado (UML)

Las ramas más antiguas de la ingeniería han encontrado útil, desde hace mucho tiempo, representar los diseños mediante dibujos. Desde los inicios del software, los programadores han encapsulado sus conceptos en diversos tipos de dibujos o, más ampliamente, de modelos (11).

Los modelos de ingeniería tienen como propósito ayudar a resolver un problema complejo, comunicar ideas acerca de un problema o solución y guiar la implementación.

Para que un modelo sea eficaz debe satisfacer las siguientes características (11):

- Abstracto: Debe enfatizar los elementos importantes y ocultar los irrelevantes.
- Comprensible: Debe ser fácil de entender para los observadores.
- Preciso: Representar de forma fiel el sistema que modela.
- Predictivo: Se puede usar para deducir conclusiones sobre el sistema que modela.
- Barato: Mucho más barato y sencillo de construir que el sistema que modela.

El Lenguaje Unificado de Modelado (UML) es un lenguaje estándar de modelado para software – un lenguaje para la visualización, especificación, construcción y documentación de los artefactos de sistemas en los que el software juega un papel importante. Básicamente, UML permite a los desarrolladores visualizar los resultados de su trabajo en esquemas o diagramas estandarizados (11).

UML es un lenguaje de modelado de propósito general que pueden usar todos los modeladores. No tiene propietario y está basado en el común acuerdo de gran parte de la comunidad informática (12).

Es importante destacar que UML es un lenguaje de modelado, no un método o un proceso, se emplea para definir, detallar y documentar los artefactos de un sistema de software. En la última versión se adicionaron

diversas novedades que resuelven carencias desde el punto de vista práctico fundamentalmente. Entre los diagramas que propone UML para modelar un sistema se encuentran (12):

- Diagramas de estructura (clases, componentes, objetos, despliegue, paquetes).
- Diagramas de comportamiento (actividades, casos de uso, estado).
- Diagramas de interacción (secuencia, comunicación).

1.5. Herramientas CASE

CASE es una sigla, que corresponde a las iniciales de: Computer Aided Software Engineering; y en su traducción al Español significa Ingeniería de Software Asistida por Computación.

El concepto de CASE es muy amplio; y una buena definición genérica, que pueda abarcar esa amplitud de conceptos, sería la de considerar a la Ingeniería de Software Asistida por Computación (CASE), como la aplicación de métodos y técnicas a través de las cuales se hacen útiles a las personas comprender las capacidades de las computadoras, por medio de programas, de procedimientos y su respectiva documentación. Estas herramientas permiten organizar y manejar la información de un proyecto informático. Permitiéndoles a los participantes de un proyecto, que los sistemas (especialmente los complejos), se tornen más flexibles, más comprensibles y además mejorar la comunicación entre los participantes (13).

A continuación en los subepígrafes 1.5.1 y 1.5.2 se describen los principales componentes de las herramientas CASE más utilizadas y sus funcionalidades las cuales son Visual Paradigm y Rational Rose.

1.5.1. Visual Paradigm

El Visual Paradigm es una suite completa de herramientas CASE que da soporte al modelado visual con UML 2.0 ofreciendo distintas perspectivas del sistema. Independiente de la plataforma y dotada de una buena cantidad de productos o módulos para facilitar el trabajo durante la confección de un software así como garantizar la calidad del producto final. Además es importante destacar que tiene versiones con licencia libre para el uso de la misma (14).

Posee entre sus principales características las siguientes (14):

- Es profesional: brinda la posibilidad de crear un conjunto bastante amplio de artefactos utilizados con mucha frecuencia durante la confección de un Software. Todos estos, cumpliendo con el Standard UML 2.0.
- Es amigable: puede ser utilizado en varios idiomas, sus componentes se encuentran relacionados, por lo que se hace muy fácil la creación de cualquier tipo de diagrama, ya que cada componente

utilizado en el diagrama que se esté creando, sugiere nuevos posibles componentes a utilizar, por lo que ya no es necesario localizarlos en la barra donde pueden aparecer un número grande de componentes.

- Brinda un número considerable de estereotipos a utilizar, lo que permite un mayor entendimiento de los diagramas.
- Facilidades para redactar especificaciones de casos de uso: es posible crear plantillas para las especificaciones de casos de uso y describirlos, por lo que no se necesita de una herramienta externa como editor de texto.
- Generación de código e ingeniería inversa: brinda la posibilidad de generar código a partir de los diagramas, para plataformas como .Net, Java y PHP, así cómo obtener diagramas a partir del código.
- Integración con distintos Ambientes de Desarrollo Integrados (IDE): se integra fácilmente con varios IDEs, entre ellos el de Visual Studio y el Eclipse.
- Interoperabilidad con otras aplicaciones: brinda la posibilidad de intercambiar información mediante la importación y exportación de ficheros con aplicaciones como por ejemplo Visio y Rational Rose. Además permite importar y exportar XML y XMI.
- Generación de código ORM: permite generar a partir de un Diagrama de Entidad Relación una Base de Datos Relacional y el código necesario para acceder a esta base de datos utilizando Java, PHP, C# o Enterprise Object Framework.
- Generación de documentación: brinda la posibilidad de documentar todo el trabajo sin necesidad de utilizar herramientas externas.
- Disponibilidad en múltiples plataformas: Microsoft Windows (98, 2000, XP, o Vista), Linux, Mac OS X, Solaris o Java.

1.5.2. Rational Rose

Rational Rose es una herramienta de diseño de software destinada a modelado visual. Proporciona un lenguaje común de modelado para el equipo que facilita la creación de software de calidad más rápidamente.

Dos rasgos importantes del Rational Rose son su habilidad para ofrecer desarrollo iterativo e ingeniería bidireccional. El Rational Rose permite a los diseñadores aprovecharse del desarrollo iterativo, porque la nueva aplicación puede ser creada por etapas con la salida de una iteración como entrada de la próxima. Además Rose permite la generación de código a partir de un diseño en UML en lenguajes como C++,

VisualBasic, Java, Ada, genera IDL's para aplicaciones CORBA. Soporta realizar ingeniería inversa por lo que se puede obtener un diseño a partir del código de un programa. Está disponible en la plataforma Windows: en MicrosoftWindows NT 4.0, Windows 95, o Windows 98; y la licencia es exclusivamente propietaria (14).

Proporciona una herramienta de modelado visual para capturar y compartir los requerimientos del negocio y el seguimiento de ellos a medida que cambian a lo largo del proceso. Está basado en modelos de desarrollo para Java y entornos de aplicación J2EE, arquitectos y desarrolladores de software de apoyo. Permite el desarrollo de aplicaciones de software, modelado de datos, servicios de diseño web, modelado de negocios, y la extensión de aplicaciones heredadas (15). La desventaja que tiene es que s una herramienta propietaria.

1.6. Metodología, Herramientas y Lenguaje de Modelado a utilizar

Luego de un análisis de una forma más detallada, se puede decir que para suplir la falta de requisitos, casos de uso, y demás herramientas, XP utiliza historias de usuarios, estas no son más que las necesidades escritas por los usuarios con la ayuda de los diseñadores, que quieren ser satisfechas con el sistema. La gestión de requisitos es extremadamente simple, básicamente consiste en trabajar estrechamente con el cliente, haciendo pequeñas iteraciones que no son más que pequeñas entregas cada dos semanas, donde no existe más documentación que el código en sí. Cada versión contiene las modificaciones necesarias según el cliente vaya retroalimentando el sistema, por eso es necesaria la disponibilidad del cliente durante todo el desarrollo. No existe documentación del proyecto, esto constituye la mayor debilidad de este modelo, lo que más se acerca a la documentación son las historias de usuario, pero al concluir el proyecto se descartan. Además la ausencia de documentación hace que el mantenimiento del sistema realizado por otro equipo de trabajo sea muy engorroso y difícil.

Por su parte RUP cuenta con una amplia documentación generada durante todo su ciclo de vida, constituida por varios artefactos de gran importancia par un mejor entendimiento del negocio, cada fase se desarrolla en iteraciones, de forma tal que se pueda dividir en pequeños proyectos mejorando su comprensión y desarrollo y no necesariamente tiene que estar el cliente presente en el desarrollo del sistema, ni formar parte del equipo de trabajo. Es una metodología que define claramente las actividades que se realizan por roles y la creación de artefactos es de vital importancia en la construcción del producto.

La metodología XP es muy utilizada pero requiere una fuerte vinculación del cliente con el equipo de desarrollo, lo cual es difícil de lograr, ya que nuestro cliente no se encuentra cerca del equipo de desarrollo y no existe un diálogo constante con este. Esa es una de las razones por las que la dirección del proyecto

decidió utilizar en el desarrollo del trabajo la metodología RUP, además de que brinda una mejor posibilidad de realizar una captura de requisitos que concuerde más con las expectativas del cliente y del producto (9) y es una de las más utilizadas y difundidas en la actualidad, por su flexibilidad, capacidad de adaptación para los distintos tipos de proyectos y robustez. Hay que señalar también que.

La selección de una herramienta Case no es una tarea simple, pues no existe una herramienta mejor con respecto a otra, ya que todas son vulnerables a producir fallas. Se decidió utilizar como herramienta CASE para el modelado de la aplicación el Visual Paradigm debido, fundamentalmente, a que es multiplataforma y tiene licencia de uso libre, lo que permitirá transferir los modelos obtenidos al cliente, al finalizar el desarrollo del sistema, también porque soporta todos los diagramas que son necesarios generar durante el ciclo de vida de RUP.

Son varios los lenguajes de modelado que pueden ser utilizados para modelar un sistema, pero se decidió para el presente trabajo utilizar UML porque es un lenguaje de modelado de propósito general que pueden usar todos los modeladores. No tiene propietario y está basado en el común acuerdo de gran parte de la comunidad informática (12). Además se acopla bien a la metodología seleccionada pues propone un grupo de diagramas para modelar el sistema que concuerdan con los que son necesarios generar durante el ciclo de vida de la misma.

1.7. Conclusiones

En este capítulo se ha realizado un estudio sobre algunos simuladores utilizados en la enseñanza de la Física en las escuelas que poseen características similares a las que tiene FISIM, seleccionando algunas funcionalidades de los mismos para agregarlas a nuestro sistema. Por otra parte a partir del estudio que se realizó sobre las principales metodologías y herramientas utilizadas para el desarrollo de software y teniendo en cuenta las características del sistema a desarrollar, se decidió utilizar como metodología de desarrollo de software RUP, la herramienta Case Visual Paradigm para el modelado de los diagramas del sistema y UML como lenguaje de modelado, persiguiendo con todo esto facilitar el futuro desarrollo del simulador FISIM.

2. CAPÍTULO 2. CARACTERÍSTICAS DEL SISTEMA

El resultado final de un proyecto de software es un producto que toma forma a lo largo del desarrollo del proyecto. La calidad del producto final, está estrechamente ligada a la calidad del proceso de desarrollo de software, entre otros aspectos (16). El diseño del sistema es uno de los tantos aspectos que tienen gran importancia dentro del desarrollo de un software.

Para en un futuro poder realizar el diseño de un sistema informático es necesario la generación de un grupo de artefactos que describan las principales características del sistema a desarrollar. Debido a ello en este capítulo se aborda el funcionamiento general del sistema mediante un modelo de dominio, el que va a tener su base en los principales conceptos manejados y las relaciones existentes entre ellos. También se identifican los principales requisitos funcionales y no funcionales que el sistema debe cumplir. Además se describen los actores y los casos de uso del sistema, así como sus relaciones, lo cual se va a reflejar en un diagrama de casos de uso realizado para mejorar la comprensión del sistema.

2.1. Modelo de Dominio

Según RUP un modelo representa una forma de contemplar el sistema que se modela, según el punto de vista que se elabora. El modelado del negocio es una técnica para comprender los procesos de negocio de la organización (11). Si los procesos están claramente definidos y no se van a introducir cambios entonces se justifica la realización de un modelo de negocio. Si se determina que no es necesario un modelo completo del negocio como es el caso, se va realizar un modelo de dominio.

El Modelo de Dominio o Modelo Conceptual es una representación visual de los conceptos u objetos del mundo real que son significativos para un problema o área de interés. Representa clases conceptuales del dominio del problema, conceptos del mundo real, no de los componentes de software. En este modelo no se muestra comportamiento. Las clases conceptuales pueden tener atributos pero no métodos (17). Su principal objetivo es ayudar a los implicados en el desarrollo del producto a utilizar un vocabulario común que posibilite una mejor comprensión entre ellos, para poder entender y describir las clases más importantes dentro del contexto donde se ubica el sistema. Además permite y facilita el levantamiento de requisitos así como definir los procesos y roles que sean más significativos.

2.1.1. Análisis de los conceptos del Dominio.

Usuario: Persona con conocimientos de Física y Matemática que desea simular un fenómeno físico, puede ser un estudiante o un profesor.

Simulación: Imita tanto el comportamiento de un sistema del mundo real, como los procesos de entrada que manejan o controlan el sistema simulado. Puede usarse para obtener conocimiento acerca de sistemas existentes, para predecir su comportamiento y con propósitos de enseñanza. Está compuesta por un modelo matemático y un grupo de elementos. Las simulaciones que se pueden realizar son simulación de mecánica, simulación de óptica, simulación de termodinámica o simulación de circuito.

Simulación de mecánica: Permite modelar e imitar el comportamiento de un fenómeno o proceso mecánico definido.

Simulación de óptica: Permite modelar e imitar el comportamiento de un fenómeno o proceso de óptica definido.

Simulación de termodinámica: Permite modelar e imitar el comportamiento de un fenómeno o proceso termodinámico definido.

Simulación de circuito: Permite modelar e imitar el comportamiento de un fenómeno o proceso de circuito definido.

Modelo Matemático: Describe el fenómeno o proceso a estudiar. Tiene una estructura definida compuesta por una sección de fórmulas y una sección de datos, en la sección de datos se define de uno a tres casos a estudiar.

Elemento: Son todos los componentes que se pueden adicionar al escenario, los mismos pueden ser elementos generales, elementos de circuito, elementos de termodinámica, elementos de mecánica y elementos de óptica.

Elementos generales: Componentes que pueden ser utilizados en cualquier tipo de simulación, los mismos pueden ser un vector, un punto, una etiqueta, un cursor, una imagen o un medidor.

Vector: Elemento general que permite representar vectorialmente el comportamiento de alguna magnitud.

Punto: Elemento general que permite representar un componente con determinado comportamiento.

Etiqueta: Elemento general que permite mostrar cierta información estática o tomada a partir de la evaluación de una variable.

Cursor: Elemento general que permite modificar el valor de una variable sin la necesidad de hacerlo en el modelo.

Imagen: Las imágenes pueden ser usadas de manera diversa y permiten un gran número de opciones a configurar.

Medidor: Elemento general que brinda la posibilidad de representar visualmente el cambio de estado de una magnitud determinada en función de parámetros definidos.

Elementos de circuito: Componentes que pueden ser utilizados en simulaciones de Circuitos, los mismos pueden ser un cable conductor, un interruptor, una bombilla, un amperímetro, un voltímetro, una resistencia.

Elementos de termodinámica: Componentes que pueden ser utilizados en simulaciones de Termodinámica, los mismos pueden ser un recipiente, un foco, un tubo, un compresor o una máquina.

Elementos de mecánica: Componentes que pueden ser utilizados en simulaciones de Mecánica, los mismos pueden ser un péndulo, una polea, un objeto, un resorte o una superficie.

Elementos de óptica: Componentes que pueden ser utilizados en simulaciones de Óptica, los mismos pueden ser un espejo, una lámina, un medio, un rayo luminoso, un objeto, una pizarra.

Gráfica: Representación de datos, generalmente numéricos, mediante líneas superficies o símbolos, para ver la relación que guardan entre sí plasmados en coordenadas cartesianas. Permite representar dinámicamente una variable en función otra.

2.1.2. Diagrama del Modelo de Dominio

A continuación se muestra el diagrama del modelo de dominio correspondiente al software FISIM:

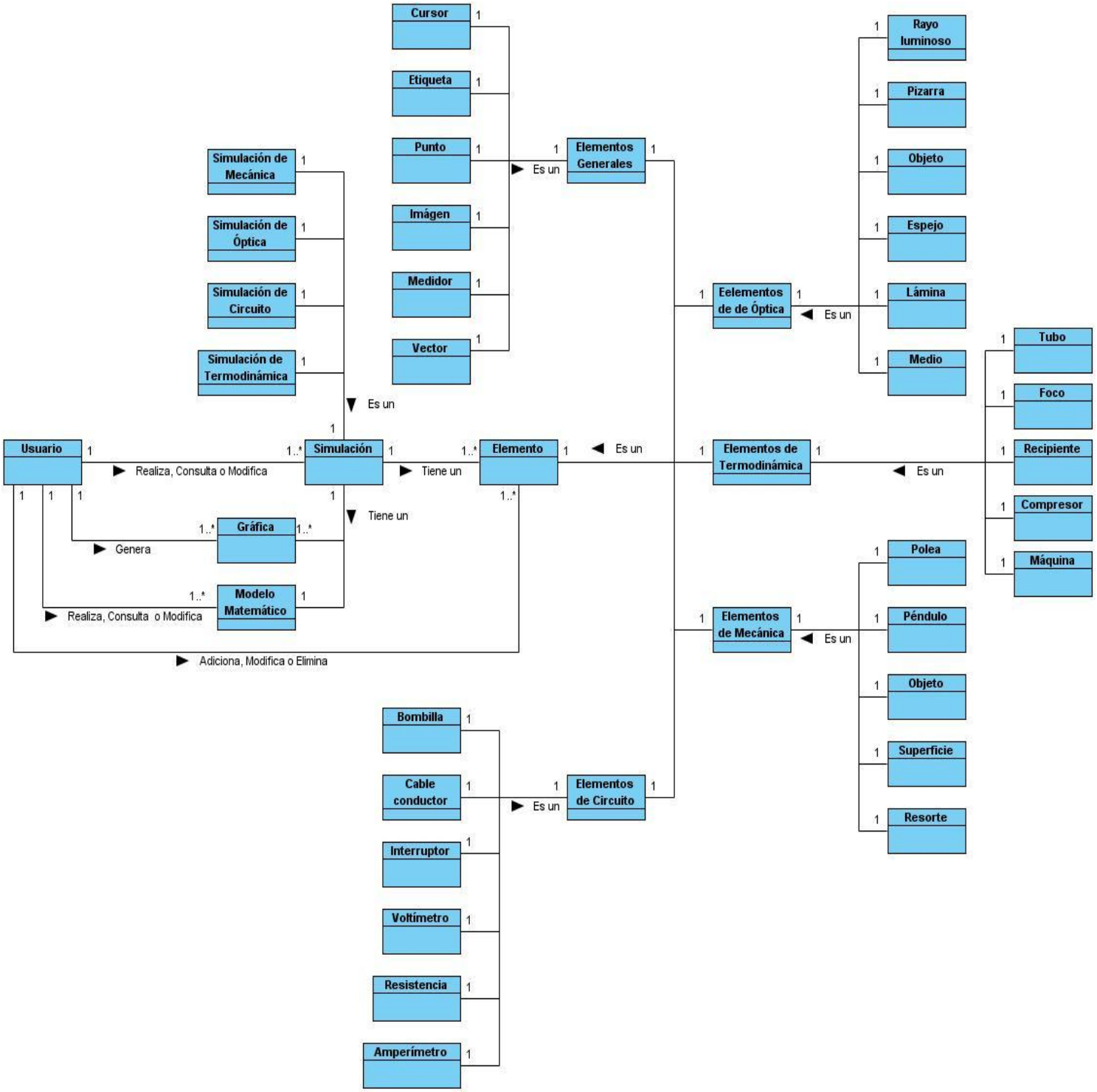


Figura 8. Representación del Modelo de Dominio

2.2. Descripción del sistema propuesto

El usuario se autentica en la aplicación, luego realiza, consulta o modifica una simulación. Si va a realizar una simulación debe seleccionar el tipo de simulación que desea efectuar. Después de seleccionado el tipo de simulación a desarrollar este procede a realizar el modelo matemático que va a describir el proceso a simular en la hoja lógica. Una vez confeccionado el modelo matemático el usuario puede añadir los elementos necesarios para la ejecución de la simulación, así como un grupo de elementos generales que van a responder al comportamiento de alguna magnitud definida en el modelo. También el mismo puede generar gráficas que muestren el comportamiento en los ejes X y Y de las variables definidas en el modelo para esa simulación, así como realizar modificaciones a los elementos adicionados o eliminarlos en el administrador de componentes.

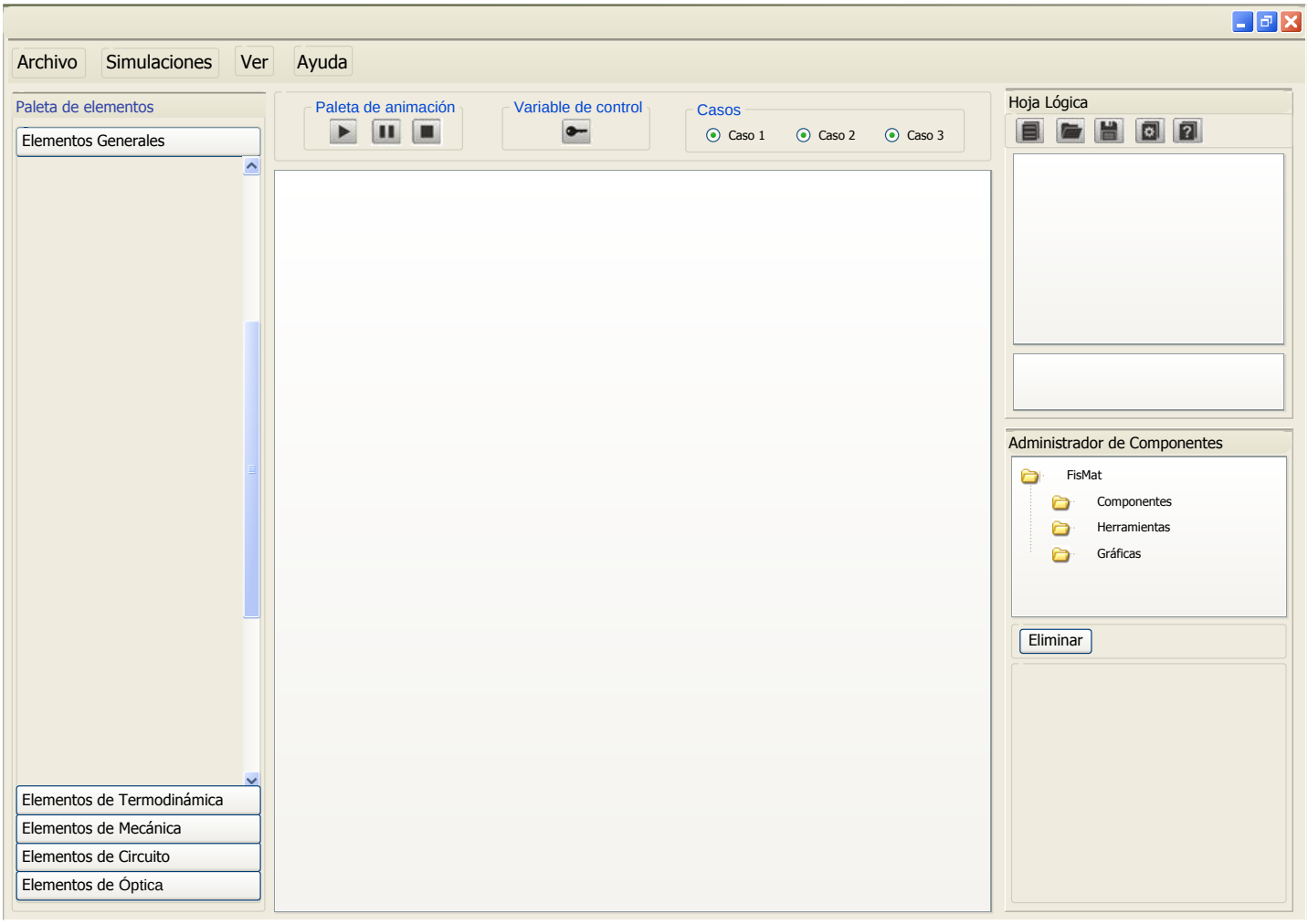


Figura 9. Propuesta de Interfaz Principal de FISIM

2.3. Requerimientos del software

Cualquier metodología de Análisis y Diseño para el desarrollo de sistemas tiene como punto de partida la captura de requisitos, obtenidos por los analistas en interacción con los usuarios, que más tarde serán analizados y plasmados en herramientas propias de cada metodología de manera que cubran las expectativas de los usuarios y que se ajusten a las tendencias actuales de desarrollo de aplicaciones (16).

Un requerimiento es una condición o necesidad de un usuario para resolver un problema o alcanzar un objetivo. Es una condición o capacidad que debe estar presente en un sistema o componentes de sistema para satisfacer un contrato, estándar, especificación u otro documento formal.

La obtención de requerimientos es un paso muy importante para el posterior desarrollo de las siguientes etapas, pues un error en estas fases iniciales puede dar al traste con un sistema que no cumpla las expectativas de los usuarios y difícilmente aporte valor agregado al negocio para el que debe ser concebido (16).

Todos sabemos que los requerimientos de software se dividen en requerimientos funcionales y requerimientos no funcionales. Los requerimientos funcionales definen las funciones que el sistema será capaz de realizar y describen las transformaciones que el sistema realiza sobre las entradas para producir salidas. Los requerimientos no funcionales tienen que ver con características que de una u otra forma puedan limitar el sistema, como por ejemplo, el rendimiento, interfaces de usuario, fiabilidad, mantenimiento, seguridad, portabilidad, estándares, etc.

Teniendo en cuenta las características y necesidades del simulador FISIM se identificaron los requerimientos que se exponen en los puntos 2.3.1. Requerimientos Funcionales y 2.3.2. Requerimientos no Funcionales.

2.3.1. Requerimientos Funcionales

RF 1: Iniciar sesión de trabajo por autenticación.

RF 2: Registrar inicio de sesión:

- **RF 2.1:** Registrar fecha en la que se conectó.
- **RF 2.2:** Registrar hora en la que se conectó.
- **RF 2.3:** Registrar nombre de la Persona.
- **RF 2.4:** Registrar IP de la máquina que ejecuta la acción.

RF 3: Mantener informado al usuario del resultado de las operaciones.

RF 4: Cerrar la interfaz de autenticación después de tres intentos.

RF 5: Validar la integridad de los datos introducidos por el usuario.

RF 6: Mostrar escritorio de trabajo del usuario autenticado.

RF 7: Gestionar una simulación:

- **RF 7.1:** Crear una simulación determinada.
- **RF 7.2:** Abrir una simulación previamente elaborada.
- **RF 7.3:** Modificar una simulación que se haya creado o abierto.

RF 8: Guardar una simulación previamente elaborada.

RF 9: Guardar los cambios realizados a una simulación.

RF 10: Ejecutar una simulación para mostrar el comportamiento de sus elementos durante la misma.

RF 11: Pausar una simulación que se esté ejecutando.

RF 12: Poner en el inicio una simulación que se esté ejecutando.

RF 13: Gestionar un modelo matemático:

- **RF 13.1:** Crear un modelo matemático que describa un proceso o fenómeno físico.
- **RF 13.2:** Abrir un modelo matemático previamente guardado.
- **RF 13.3:** Modificar un modelo matemático anteriormente creado o abierto.
- **RF 13.4:** Interpretar un modelo matemático.

RF 14: Guardar un modelo matemático creado.

RF 15: Guardar los cambios realizados a un modelo matemático.

RF 16: Mostrar los errores que contenga el modelo matemático definido.

RF 17: Medir durante la simulación, el valor de las variables definidas en el modelo matemático.

RF 18: Gestionar un elemento.

- **RF 18.1:** Incluir un elemento al escenario.
- **RF 18.2:** Mostrar datos de un elemento incluido en el escenario.
- **RF 18.3:** Modificar datos de un elemento incluido en el escenario.
- **RF 18.4:** Eliminar un elemento existente en el escenario.

RF 19: Crear enlace entre dos puntos.

RF 20: Mover un objeto previamente adicionado a cualquier posición dentro del escenario.

RF 21: Listar todos los elementos ubicados en el escenario.

RF 22: Seleccionar un objeto en el administrador de componentes.

RF 23: Mostrar en el escenario la posición (x; y) de un elemento seleccionado.

RF 24: Adicionar al escenario sin importar el tipo de simulación cualquiera de los siguientes elementos: Cursores, Etiquetas, Puntos, Vectores, Imágenes, Gráficas y Medidores.

RF 25: Mostrar de manera puntual o vectorial el comportamiento de las variables definidas.

RF 26: Definir el comportamiento de cada elemento según las variables del modelo matemático definido en el intérprete.

RF 26.1: Definir la posición en el eje x y/o en el eje y de un elemento general.

RF 26.2: Definir las dimensiones de un elemento general.

RF 26.3: Definir la visibilidad de un elemento general.

RF 27: Mantener informado al usuario del resultado de las operaciones.

RF 28: Adicionar al escenario si la simulación a desarrollar es de Termodinámica cualquiera de los siguientes elementos: Recipiente, Foco, Tubo, Compresor, y Máquina.

RF 29: Validar si los elementos a adicionar corresponden a la tipología de simulación en cuestión.

RF 30: Adicionar al escenario si la simulación a desarrollar es de Óptica cualquiera de los siguientes elementos: Espejo, Lámina de Vidrio, Lente, Medio, Rayo Luminoso, Objeto, Pizarra y Lupa.

RF 31: Adicionar al escenario si la simulación a desarrollar es de Circuito cualquiera de los siguientes elementos: Cable Conductor, Interruptor, Pila, Batería, Bombilla, Amperímetro, Voltímetro, Condensador, Resistencia, Fuente de Corriente Alterna o Motor.

RF 32: Adicionar al escenario si la simulación a desarrollar es de Mecánica cualquiera de los siguientes elementos: Péndulo, Polea, Masa, Bloque de Madera, Pendiente, Carrito, Cañón, Resorte o Superficie.

RF 33: Graficar el comportamiento de una variable durante la simulación.

RF 34: Mostrar la ayuda del simulador.

RF 35: Listar los contenidos a consultar.

RF 36: Seleccionar un contenido existente en la ayuda del simulador.

2.3.2. Requerimientos No Funcionale

Software:

- Las computadoras deben tener instalado **el navegador web Mozilla Firefox, Internet Explorer, Chrome, Opera o Safari.**
- Las computadoras deben tener instalado el NetBeans y la Máquina Virtual de Java.

Hardware:

- Las computadoras locales que brindarán el servicio del sistema al cliente no deberán presentar potencias menores a las brindadas por una Pentium 4 con al menos 512 MB de RAM y 1 GB de espacio en disco duro.

Interfaz, Usabilidad y Rendimiento:

- Interfaz amigable para usuarios finales. Su funcionamiento deberá ser intuitivo, y requerir de información mínima.
- Claridad y buena organización de la información, permitiendo la interpretación correcta e inequívoca de la información.
- Ejecución de acciones de una manera rápida, minimizando los pasos a dar en cada proceso.
- La interfaz de la plataforma se personaliza de acuerdo al usuario que inicia sesión. Se carga la identidad de la escuela a la que pertenece.

Configuración, Conectividad y Seguridad:

- Contar con el motor Java 6, que también podrá ser descargado de la plataforma, en la misma ubicación que la aplicación del laboratorio de física.
- La aplicación requiere validación de uso por el servidor local o central. Si no obtiene la validación, no puede iniciar.
- La validación permite al servidor determinar, los datos que enviará a la aplicación, dependiendo del período escolar en que está inscrito el usuario. Es decir, el usuario recibe solo el contenido y material necesario para realizar las prácticas del nivel educativo al que pertenece.
- La aplicación requiere conexión web todo el tiempo que esté activa, pues llama del servidor local o central, elementos dinámicos que la conforman.
- El esqueleto que queda almacenado en cada PC no es funcional sin la conexión y validación previamente mencionadas.

Funcionalidad:

- Reducir al mínimo el tiempo en que carga la aplicación.

Legales, Derecho de Autor y otros:

- Una vez terminado el simulador debe ser sometido a una evaluación y certificación por parte del cliente del producto.
- Creación de un documento en el que quede plasmado la autoría de las personas que desarrollaron el simulador.

2.4. Modelo de Casos de Uso del Sistema

Un caso de uso es una descripción de la secuencia de interacciones que se producen entre un actor y el sistema, cuando el actor usa el sistema para llevar a cabo una tarea específica. Expresa una unidad coherente de funcionalidad, y se representa en el Diagrama de Casos de Uso mediante una elipse con el nombre del caso de uso en su interior. El nombre del caso de uso debe reflejar la tarea específica que el actor desea llevar a cabo usando el sistema (18). Aunque los casos de uso dirigen el proceso, no son elegidos de manera aislada, son desarrollados a la par con la arquitectura del sistema y maduran conforme avanza el ciclo de vida.

En relación con las características y necesidades del simulador FISIM los actores que se identificaron son los expuestos en el punto 2.4.1 y los casos de uso que se definieron son los descritos en el punto 2.4.3.

2.4.1. Descripción de los Actores del Sistema

Cuando hablamos de actor nos referimos al rol que algo o alguien juega cuando interactúa con el sistema. Un candidato a actor del sistema es cualquier individuo, grupo, organización o máquina que interactúa en los casos de uso. De acuerdo con esta idea un actor del sistema representa un tipo particular de usuario del sistema más que un usuario físico, ya que varios usuarios físicos pueden realizar el mismo papel en relación al negocio. Una vez que se han identificado los actores del sistema, se tiene identificado el entorno externo del sistema. A continuación en la tabla 1.1 se muestra la descripción de los actores identificados.

Actor	Descripción
Usuario	Persona con conocimientos de Física y Matemática que desea simular un fenómeno físico, puede ser un estudiante o un profesor.

Tabla 1: Descripción de los actores del sistema

2.4.2. Diagrama de Actores

A continuación se muestra el diagrama de actores correspondiente al software FISIM:

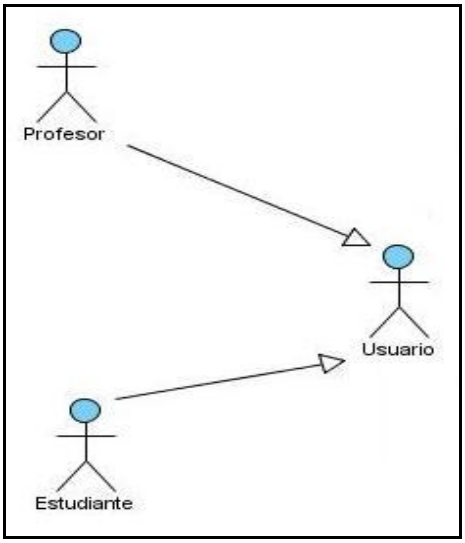


Figura 10 Diagrama de Actores.

2.4.3. Patrones de Casos de Uso utilizados

Los patrones de casos de uso son comportamientos que deben existir en el sistema, ayudan a describir qué es lo que el sistema debe hacer, es decir, describen el uso del sistema y cómo este interactúa con los usuarios. Estos patrones son utilizados generalmente como plantillas que describen como debería ser estructurados y organizados los casos de uso (19).

Patrón CRUD (Creating, Reading, Updating, Deleting)

Este patrón se basa en la fusión de casos de uso simples para formar una unidad conceptual.

Completo: Este patrón consta de un caso de uso, llamado Información CRUD o Gestionar información, modela todas las operaciones que pueden ser realizadas sobre una parte de la información de un tipo específico, tales como creación, lectura, actualización y eliminación. Suele ser utilizado cuando todos los flujos contribuyen al mismo valor del negocio, y estos a su vez son cortos y simples. A continuación se muestra un ejemplo de su uso del patrón en el diagrama:

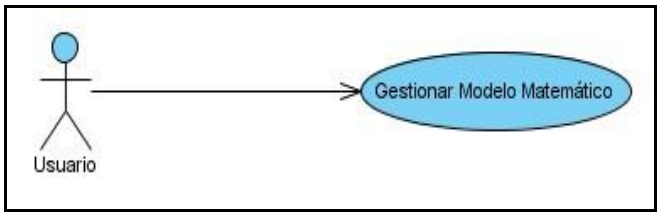


Figura 11: Patrón CRUD. Completo

Patrón Múltiples Actores

Roles común: Puede suceder que los dos actores jueguen el mismo rol sobre el CU. Este rol es representado por otro actor, heredado por los actores que comparten este rol. Es aplicable cuando, desde el punto de vista del caso de uso, solo exista una entidad externa interactuando con cada una de las instancias del caso de uso. A continuación se muestra un ejemplo de su uso del patrón en el diagrama:

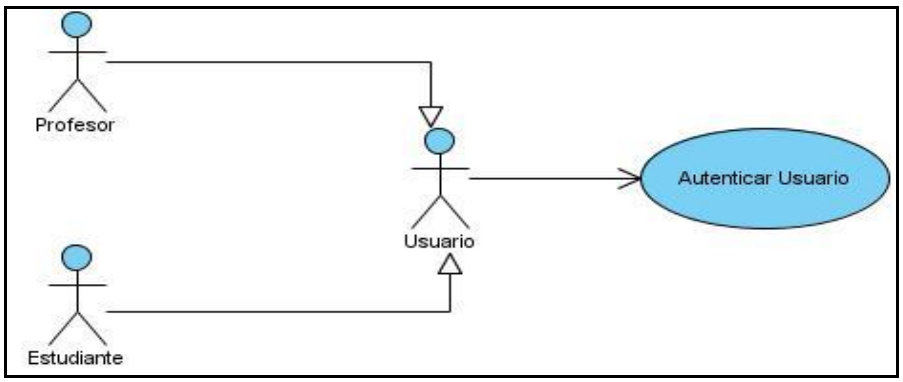


Figura 12: Patrón Múltiples Actores. Roles Comunes

2.4.4. Diagrama de Casos de Uso del Sistema

Un Diagrama de Casos de Uso muestra la relación entre los actores y los casos de uso del sistema. Representa la funcionalidad que ofrece el sistema en lo que se refiere a su interacción externa. En el diagrama de casos de uso se representa también el sistema como una caja rectangular con el nombre en su interior. Los casos de uso están en el interior de la caja del sistema, y los actores fuera, y cada actor está unido a los casos de uso en los que participa mediante una línea (18). A continuación se muestra el diagrama de casos de uso del sistema correspondiente al software FISIM:

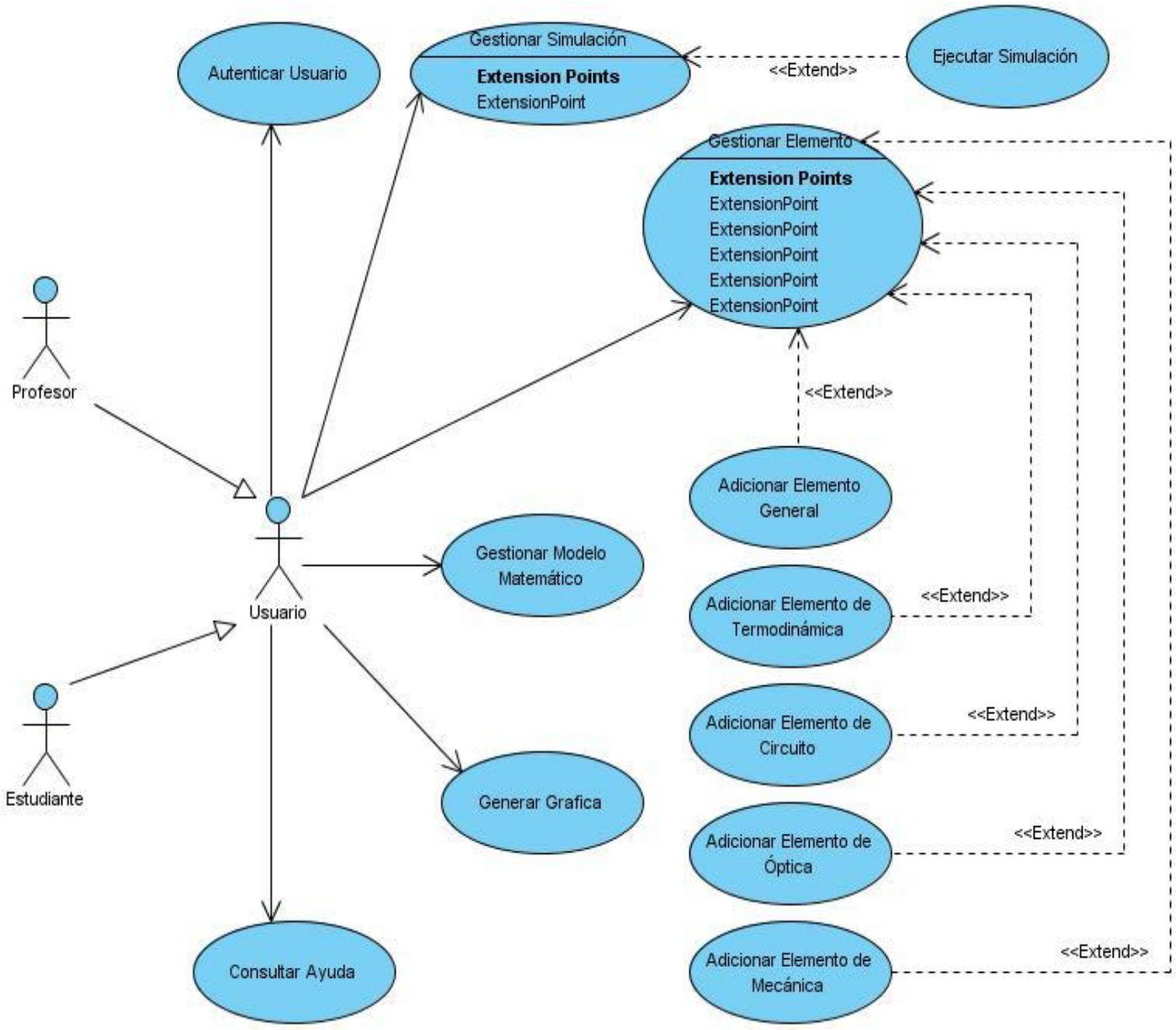


Figura 13. Diagrama de Caso de Uso del Sistema.

2.4.5. Descripción de los Casos de Uso del Sistema

Nombre	CU Autenticar Usuario.
Objetivo	Iniciar la sesión de un usuario en el sistema
Actores	Usuario (Inicia): Inicia la sesión.
Complejidad	Baja.
Nivel	Usuario.
Precondiciones	Debe haberse mostrado la interfaz de autenticación del sistema.
Poscondiciones	Se identificó y autenticó el usuario. Se mostró el escritorio de trabajo del usuario autenticado.
Descripción del Caso de Uso	
El caso de uso inicia cuando el usuario accede a la interfaz de autenticación del sistema. El sistema muestra opciones para la inclusión de usuario y contraseña, el usuario introduce los datos solicitados, el sistema consulta internamente que exista correspondencia entre los datos introducidos y los datos almacenados, luego el sistema permite el acceso mostrando el escritorio de trabajo del usuario autenticado. Terminando así el caso de uso.	
Requisitos Funcionales	
Iniciar sesión de trabajo por autenticación. Validar la integridad de los datos introducidos por el usuario. Mantener informado al usuario del resultado de las operaciones. Cerrar la interfaz de autenticación después de tres intentos. Registrar inicio de sesión: • Registrar fecha en la que se conectó. • Registrar hora en la que se conectó. • Registrar nombre de la Persona. • Registrar IP de la máquina que ejecuta la acción. Mostrar escritorio de trabajo del usuario autenticado.	

Tabla 2: Resumen de la descripción del CU Autenticar Usuario

Nombre	CU Gestionar Simulación.
Objetivo	Crear, abrir o modificar una simulación de un fenómeno o proceso físico.
Actores	Usuario (Inicia): Crea, abre o modifica una simulación.
Complejidad	Alta.
Nivel	Usuario.
Precondiciones	Debe haberse generado el escritorio de trabajo del usuario autenticado.
Poscondiciones	Se creó, abrió o modificó una simulación de un fenómeno o proceso físico.
Descripción del Caso de Uso	
El caso de uso se inicia cuando el usuario selecciona la opción de realizar una acción sobre una simulación. El usuario puede crear, abrir y modificar una simulación. En caso de que seleccione la opción de crear una simulación, el sistema dará la posibilidad de insertar los datos que se necesitan para confeccionar dicha simulación. Si el usuario elige la opción de abrir una simulación el sistema permitirá seleccionar una simulación previamente guardada. Si el usuario elige la opción de modificar una simulación, el sistema mostrará los datos que pueden ser editables dentro de la simulación, y una vez realizados los cambios permitirá guardar los mismos.	
Requisitos Funcionales	
Gestionar una simulación: • Crear una simulación determinada. • Abrir una simulación previamente elaborada. • Modificar una simulación que se haya creado o abierto. Guardar una simulación previamente elaborada. Guardar los cambios realizados a una simulación. Validar la integridad de los datos introducidos por el usuario. Mantener informado al usuario del resultado de las operaciones.	

Tabla 3: Resumen de la descripción del CU Gestionar Simulación

Nombre	CU Ejecutar Simulación.
Objetivo	Mostrar el comportamiento de los elementos de una

	simulación durante la misma.
Actores	Usuario (Inicia): Ejecuta la simulación.
Complejidad	Media.
Nivel	Usuario.
Precondiciones	Debe haberse generado el escritorio de trabajo del usuario autenticado. Debe haberse creado o abierto una simulación.
Poscondiciones	Se ejecutó, pausó o envió al inicio una simulación de un fenómeno o proceso físico
Descripción del Caso de Uso	
El caso de uso inicia cuando el actor selecciona la opción que le permite iniciar una simulación. El actor puede además pausar o poner en el inicio la simulación.	
Requisitos Funcionales	
Ejecutar una simulación para mostrar el comportamiento de sus elementos durante la misma. Pausar una simulación que se esté ejecutando. Poner en el inicio una simulación que se esté ejecutando. Medir durante la simulación, el valor de las variables definidas en el modelo matemático. Validar la integridad de los datos que forman parte de una simulación. Mantener informado al usuario del resultado de las operaciones.	

Tabla 4: Resumen de la descripción del CU Ejecutar Simulación

Nombre	CU Gestionar Modelo Matemático.
Objetivo	Crear, abrir o modificar un modelo matemático que describe el fenómeno o proceso a simular.
Actores	Usuario (Inicia): Crea, abre o modifica un modelo matemático.
Complejidad	Alta.
Nivel	Usuario.
Precondiciones	Debe haberse generado el escritorio de trabajo del usuario autenticado. Debe haberse seleccionado el tipo de simulación a

	realizar.
Poscondiciones	Se creó, abrió o modificó un modelo matemático que describe el fenómeno o proceso a simular.
Descripción del Caso de Uso	
El caso de uso se inicia cuando el usuario realiza una acción sobre la Hoja Lógica. El usuario puede crear, abrir y modificar un modelo matemático. En caso de que seleccione la opción de crear un modelo matemático, el sistema dará la posibilidad de insertar los datos que se necesitan para confeccionar dicho modelo, una vez creado permitirá interpretarlo y guardarlo. Si el actor elige la opción de abrir un modelo matemático el sistema permitirá seleccionar un modelo previamente guardado. Si el actor elige la opción de modificar un modelo matemático, el sistema mostrará los datos que pueden ser editables dentro del modelo, y una vez realizados los cambios permitirá guardar los mismos.	
Requisitos Funcionales	
Gestionar un modelo matemático: • Crear un modelo matemático que describa un proceso o fenómeno físico. • Abrir un modelo matemático previamente guardado. • Modificar un modelo matemático anteriormente creado o abierto. • Interpretar un modelo matemático. Guardar un modelo matemático creado. Guardar los cambios realizados a un modelo matemático. Mostrar los errores que contenga el modelo matemático definido. Validar la integridad de los datos introducidos por el usuario. Mantener informado al usuario del resultado de las operaciones.	

Tabla 5: Resumen de la descripción del CU Gestionar Modelo Matemático

Nombre	CU Gestionar Elemento.
Objetivo	Incluir, ver, modificar o eliminar un elemento.
Actores	Usuario (Inicia): Incluye, ve, modifica o elimina un elemento.
Complejidad	Alta.
Nivel	Usuario.

Precondiciones	Debe haberse generado el escritorio de trabajo del usuario autenticado. Para incluir un elemento debe haberse realizado la compilación del modelo matemático de manera correcta y debe haberse seleccionado el miso previamente. Para ver o modificar los datos de un elemento debe estar seleccionado anteriormente. Para eliminar un elemento debe estar seleccionado. Para crear un enlace entre puntos deben existir al menos dos puntos.
Poscondiciones	Se incluyó, vio, modificó o eliminó un elemento por el usuario.

Descripción del Caso de Uso
El caso de uso se inicia cuando el usuario selecciona la opción que le permite realizar una acción sobre un elemento. El actor puede incluir, ver, modificar o eliminar un elemento. En caso de que seleccione la opción de incluir un elemento, el sistema dará la posibilidad de insertar los datos que se necesitan para definir el comportamiento de cada elemento según las variables del modelo matemático definido en el intérprete. Si el actor elige la opción de ver un elemento el sistema mostrará las propiedades del elemento en cuestión. Si el actor elige la opción de modificar un elemento, el sistema mostrará los datos que pueden ser editables de un elemento, y una vez realizados los cambios permitirá guardar los cambios. Si el actor selecciona la opción de eliminar un elemento, el sistema eliminará el elemento seleccionado, y una vez realizados los cambios permitirá guardará las modificaciones.

Requisitos Funcionales
Gestionar un elemento. • Incluir un elemento al escenario. • Mostrar datos de un elemento incluido en el escenario. • Modificar datos de un elemento incluido en el escenario. • Eliminar un elemento existente en el escenario. Validar la integridad de los datos introducidos por el usuario. Mantener informado al usuario del resultado de las operaciones. Crear enlace entre dos puntos.

Mover un objeto previamente adicionado a cualquier posición dentro del escenario. Listar todos los elementos ubicados en el escenario. Seleccionar un objeto en el administrador de componentes. Mostrar en el escenario la posición (x; y) de un elemento seleccionado.

Tabla 6: Resumen de la descripción del CU Gestionar Elemento

Nombre	CU Adicionar Elemento General
Objetivo	Incluir al escenario cualquier elemento general sin importar el tipo de simulación.
Actores	Usuario (Inicia): Adiciona elemento general al escenario
Complejidad	Alta.
Nivel	Usuario.
Precondiciones	Debe haberse generado el escritorio de trabajo del usuario autenticado. Para incluir un elemento general debe haberse realizado la compilación del modelo matemático de manera correcta y debe haberse seleccionado el miso previamente.
Poscondiciones	Se adicionó un elemento general al escenario.
Descripción del Caso de Uso	
El caso de uso se inicia cuando el usuario selecciona la opción que le permita adicionar un elemento general, dichos elementos pueden ser: Imágenes, Cursores, Etiquetas, Puntos, Gráficas, Medidores y Vectores. El usuario puede mostrar el valor de una variable definida en una etiqueta, cursor o un medidor, así como, observar el comportamiento de la misma de forma puntual o vectorial.	
Requisitos Funcionales	
Adicionar al escenario sin importar el tipo de simulación cualquiera de los siguientes elementos: Cursores, Etiquetas, Puntos, Vectores, Imágenes, Gráficas y Medidores. Mostrar de manera puntual o vectorial el comportamiento de las variables definidas. Definir el comportamiento de cada elemento según las variables del modelo matemático definido en el intérprete. • Definir la posición en el eje x y/o en el eje y de un elemento general. • Definir las dimensiones de un elemento general.	

• Definir la visibilidad de un elemento general. Mantener informado al usuario del resultado de las operaciones.

Tabla 7: Resumen de la descripción del CU Adicionar Elemento General

Nombre	CU Adicionar Elemento de Termodinámica.
Objetivo	Incluir al escenario cualquier elemento de termodinámica si la simulación a realizar es de termodinámica.
Actores	Usuario (Inicia): Adiciona elemento de termodinámica al escenario.
Complejidad	Alta.
Nivel	Usuario.
Precondiciones	Debe haberse generado el escritorio de trabajo del usuario autenticado. Para incluir un elemento de termodinámica debe haberse realizado la compilación del modelo matemático de manera correcta, debe haberse seleccionado realizar una simulación de termodinámica y el miso debe haber sido seleccionado previamente.
Poscondiciones	Se adicionó un elemento de termodinámica al escenario.
Descripción del Caso de Uso	
El caso de uso se inicia cuando el usuario selecciona la opción que le permita adicionar un elemento de termodinámica, dichos elementos pueden ser: un recipiente, un foco, un tubo, un compresor o una máquina.	
Requisitos Funcionales	
Adicionar al escenario si la simulación a desarrollar es de Termodinámica cualquiera de los siguientes elementos: Recipiente, Foco, Tubo, Compresor, y Máquina. Validar si los elementos a adicionar corresponden a la tipología de simulación en cuestión. Definir el comportamiento de cada elemento según las variables del modelo matemático definido en el intérprete. • Definir la posición en el eje x y/o en el eje y de un elemento general. • Definir las dimensiones de un elemento general.	

• Definir la visibilidad de un elemento general. Mantener informado al usuario del resultado de las operaciones.

Tabla 8: Resumen de la descripción del CU Adicionar Elemento de Termodinámica

Nombre	CU Adicionar Elemento de Óptica.
Objetivo	Incluir al escenario cualquier elemento de óptica si la simulación a realizar es de óptica.
Actores	Usuario (Inicia): Adiciona elemento de óptica al escenario.
Complejidad	Alta.
Nivel	Usuario.
Precondiciones	Debe haberse generado el escritorio de trabajo del usuario autenticado. Para incluir un elemento de óptica debe haberse realizado la compilación del modelo matemático de manera correcta, debe haberse seleccionado realizar una simulación de óptica y el miso debe haber sido seleccionado previamente.
Poscondiciones	Se adicionó un elemento de óptica al escenario.
Descripción del Caso de Uso	
El caso de uso se inicia cuando el usuario selecciona la opción que le permita adicionar un elemento de óptica, dichos elementos pueden ser: un espejo, una lámina de vidrio, un lente, un medio, un rayo luminoso, un objeto, una pizarra o una lupa.	
Requisitos Funcionales	
Adicionar al escenario si la simulación a desarrollar es de Óptica cualquiera de los siguientes elementos: Espejo, Lámina de Vidrio, Lente, Medio, Rayo Luminoso, Objeto, Pizarra y Lupa. Validar si los elementos a adicionar corresponden a la tipología de simulación en cuestión. Definir el comportamiento de cada elemento según las variables del modelo matemático definido en el intérprete. • Definir la posición en el eje x y/o en el eje y de un elemento general. • Definir las dimensiones de un elemento general. • Definir la visibilidad de un elemento general.	

Mantener informado al usuario del resultado de las operaciones.

Tabla 9: Resumen de la descripción del CU Adicionar Elemento de Óptica

Nombre	CU Adicionar Elemento de Circuito.
Objetivo	Incluir al escenario cualquier elemento de circuito si la simulación a realizar es de circuito.
Actores	Usuario (Inicia): Adiciona elemento de circuito al escenario.
Complejidad	Alta.
Nivel	Usuario.
Precondiciones	Debe haberse generado el escritorio de trabajo del usuario autenticado. Para incluir un elemento de circuito debe haberse realizado la compilación del modelo matemático de manera correcta, debe haberse seleccionado realizar una simulación de circuito y el miso debe haber sido seleccionado previamente.
Poscondiciones	Se adicionó un elemento de circuito al escenario.
Descripción del Caso de Uso	
El caso de uso se inicia cuando el usuario selecciona la opción que le permita adicionar un elemento de circuito, dichos elementos pueden ser: un cable conductor, un interruptor, una pila, una batería, una bombilla, un amperímetro, un voltímetro, un condensador, una resistencia, una fuente de corriente alterna o un motor.	
Requisitos Funcionales	
Adicionar al escenario si la simulación a desarrollar es de Circuito cualquiera de los siguientes elementos: Cable Conductor, Interruptor, Pila, Batería, Bombilla, Amperímetro, Voltímetro, Condensador, Resistencia, Fuente de Corriente Alterna o Motor. Validar si los elementos a adicionar corresponden a la tipología de simulación en cuestión. Definir el comportamiento de cada elemento según las variables del modelo matemático definido en el intérprete. • Definir la posición en el eje x y/o en el eje y de un elemento general.	

• Definir las dimensiones de un elemento general. • Definir la visibilidad de un elemento general. Mantener informado al usuario del resultado de las operaciones.

Tabla 10: Resumen de la descripción del CU Adicionar Elemento de Circuito

Nombre	CU Adicionar Elemento de Mecánica.
Objetivo	Incluir al escenario cualquier elemento de mecánica si la simulación a realizar es de mecánica.
Actores	Usuario (Inicia): Adiciona elemento de mecánica al escenario.
Complejidad	Alta.
Nivel	Usuario.
Precondiciones	Debe haberse generado el escritorio de trabajo del usuario autenticado. Para incluir un elemento de mecánica debe haberse realizado la compilación del modelo matemático de manera correcta, debe haberse seleccionado realizar una simulación de mecánica y el miso debe haber sido seleccionado previamente.
Poscondiciones	Se adicionó un elemento de mecánica al escenario.
Descripción del Caso de Uso	
El caso de uso se inicia cuando el usuario selecciona la opción que le permita adicionar un elemento de mecánica, dichos elementos pueden ser: un péndulo, una polea, una masa, un bloque de madera, una pendiente, un carrito, un cañón, un resorte o una superficie.	
Requisitos Funcionales	
Adicionar al escenario si la simulación a desarrollar es de Mecánica cualquiera de los siguientes elementos: Péndulo, Polea, Masa, Bloque de Madera, Pendiente, Carrito, Cañón, Resorte o Superficie. Validar si los elementos a adicionar corresponden a la tipología de simulación en cuestión. Definir el comportamiento de cada elemento según las variables del modelo matemático definido en el intérprete.	

• Definir la posición en el eje x y/o en el eje y de un elemento general. • Definir las dimensiones de un elemento general. • Definir la visibilidad de un elemento general. Mantener informado al usuario del resultado de las operaciones.

Tabla 11: Resumen de la descripción del CU Adicionar Elemento de Mecánica

Nombre	CU Generar Gráfica.
Objetivo	Graficar el comportamiento de una variable durante la simulación.
Actores	Usuario (Inicia): Grafica comportamiento de variable.
Complejidad	Media.
Nivel	Usuario.
Precondiciones	Debe haberse generado el escritorio de trabajo del usuario autenticado. Para crear una gráfica debe haberse realizado la compilación del modelo matemático de manera correcta y debe haberse seleccionado previamente la opción de graficar.
Poscondiciones	Se graficó el comportamiento de una variable durante la simulación.
Descripción del Caso de Uso	
El caso de uso se inicia cuando el Usuario selecciona la opción de crear una grafica. El mismo puede introducir el nombre de la gráfica y seleccionar las variables a las que desea graficarle el comportamiento en los ejes de coordenadas X y Y.	
Requisitos Funcionales	
Graficar el comportamiento de una variable durante la simulación. Validar la integridad de los datos. Mantener informado al usuario del resultado de las operaciones.	

Tabla 12: Resumen de la descripción del CU Generar Gráfica

Nombre	CU Consultar Ayuda.
Objetivo	Consultar la ayuda del simulador FISMAT.
Actores	Usuario (Inicia): Consultar la Ayuda del simulador FISMAT.
Complejidad	Baja.
Nivel	Usuario.
Precondiciones	Debe haberse generado el escritorio de trabajo del usuario autenticado.
Poscondiciones	Se mostró la ayuda del Simulador al Usuario.
Descripción del Caso de Uso	
El caso de uso se inicia cuando el usuario selecciona la opción que le permite abrir la ayuda del simulador FISMAT. El actor puede seleccionar el contenido sobre el que desea obtener información. La ayuda contiene la sintaxis y las principales reglas que debe cumplir el modelo matemático definido, así como el procedimiento a llevar realizar para confeccionar una simulación. En si es una guía para mejorar el trabajo del usuario con el simulador.	
Requisitos Funcionales	
Mostrar la ayuda del simulador. Listar los contenidos a consultar. Seleccionar un contenido existente en la ayuda del simulador.	

Tabla 13: Resumen de la descripción del CU Consultar Ayuda

2.5. Conclusiones

Durante el desarrollo de este capítulo se han definido los requerimientos que debe tener el sistema para su desarrollo, los cuales se han representado mediante un diagrama de casos de uso. También se describieron las acciones que debe realizar cada actor que interactúa con el sistema a través de una descripción resumida de los casos de uso identificados. Además se explicaron los patrones de casos de uso que se evidencian en el diagrama general de casos de uso y se abordaron los principales conceptos relacionados con el dominio del problema, los que fueron representados en un modelo de dominio, así como las relaciones existentes entre ellos. Todo ello para servir de base en la confección del análisis y el diseño de FISIM.

3. CAPÍTULO 3. ANÁLISIS Y DISEÑO DEL SISTEMA

Durante el análisis, examinamos los requerimientos que se describieron en la captura de requisitos, refinándolos y estructurándolos. Todo ello es con el objetivo de llegar a una comprensión más precisa de los requisitos y una descripción de los mismos que sea fácil de mantener y que nos ayude a estructurar el sistema entero, incluyendo su arquitectura. El propósito fundamental del análisis es darle solución a los requerimientos utilizando el lenguaje de los desarrolladores para describir los resultados (11).

Por otra parte en el diseño se modela el sistema y se encuentra su forma para que soporte todos los requisitos. Los propósitos del diseño son (11):

- Adquirir una comprensión en profundidad de los aspectos y restricciones con los lenguajes de programación, componentes reutilizables, sistemas operativos, tecnologías de distribución y concurrencia, entre otras.
- Crear una entrada apropiada y un punto de partida para actividades de implementación, capturando los requisitos o subsistemas individuales, interfaces y clases.
- Ser capaces de descomponer los trabajos de implementación en partes más manejables que puedan ser llevadas a cabo por diferentes equipos de desarrollo, teniendo en cuenta la posible concurrencia.
- Capturar las interfaces entre los subsistemas en el ciclo de vida del software.
- Ser capaces de visualizar y reflexionar sobre el diseño utilizando una notación común.
- Crear una abstracción de la implementación del sistema.

En este capítulo se expone el análisis y el diseño propuesto para la solución, modelándose los artefactos necesarios que contribuyen a la implementación del sistema, los cuales son el Modelo de Análisis y el Modelo de Diseño, los que se encargan de describir en términos de clases cómo deben funcionar los casos de uso del sistema. También en este capítulo se logra representar la colaboración entre las clases y se dan a conocer los patrones de diseño que se utilizan en la elaboración de las mismas.

3.1. Modelo de Análisis

El Modelo de Análisis es la primera representación técnica de un sistema. Se representa mediante un sistema de análisis que denota el paquete de más alto nivel del sistema. La utilización de otros paquetes de análisis es una forma de organizar el modelo de análisis en partes más manejables que representan abstracciones de subsistemas y posiblemente capas completas del diseño del sistema. Dentro del modelo

de análisis los casos de uso se describen mediante clases de análisis y sus objetos. Este modelo debe lograr tres objetivos primarios (11):

- Describir lo que requiere el cliente.
- Establecer una base para la creación de un diseño de software.
- Definir un conjunto de requisitos que se pueda validar una vez que se construye el software.

Este modelo es usado para representar la estructura global del sistema. El mismo describe la realización de casos de uso y sirve como una abstracción del Modelo de Diseño. El Modelo de Análisis puede contener las clases y paquetes de análisis, las realizaciones de los casos de uso, las relaciones y los diagramas.

3.1.1. Diagrama de Clases del Análisis

Una clase de análisis y sus objetos normalmente participan en varias realizaciones de casos de uso, y algunas de las responsabilidades, atributos y asociaciones de una clase concreta suelen ser solo relevantes para una única realización de casos de uso. Por tanto es importante durante el análisis coordinar todos los requisitos sobre una clase y sus objetos que pueden tener diferentes casos de uso. Para hacerlo es que se utiliza el Diagrama de Clases del Análisis. Este es un artefacto en el que se representan los conceptos en un dominio del problema. Representa las cosas del mundo real, no de la implementación automatizada de estas. Incorporan, además, las definiciones y relaciones entre las clases. Estas clases se clasifican en (11):

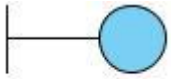
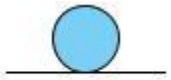
Nombre	Características	Representación
Interfaz	Se utiliza para modelar la interacción entre el sistema y sus actores. Esta interacción a menudo implica recibir información y peticiones de los usuarios y los sistemas externos.	 ClassInterfaz
Entidad	Se utiliza para modelar información que posee una vida larga y que es a menudo persistente. Modelan la información y el comportamiento asociado de algún fenómeno o proceso.	 ClassEntidad
Control	Representan coordinación, secuencia, transacciones y control de otros objetos y se usan para encapsular el control de un caso de uso en concreto. También se utilizan para representar derivaciones y cálculos complejos.	 ClassControl

Tabla 14: Descripción de las Clases del Análisis

A continuación se muestra los diagramas de clases de análisis de los casos de uso Autenticar Usuario y Gestionar Simulación del software FISIM. Para consultar los diagramas de clases del análisis correspondientes al resto de los casos de uso remitirse al Anexo 1.

Diagrama de Clases del Análisis del CU Autenticar Usuario

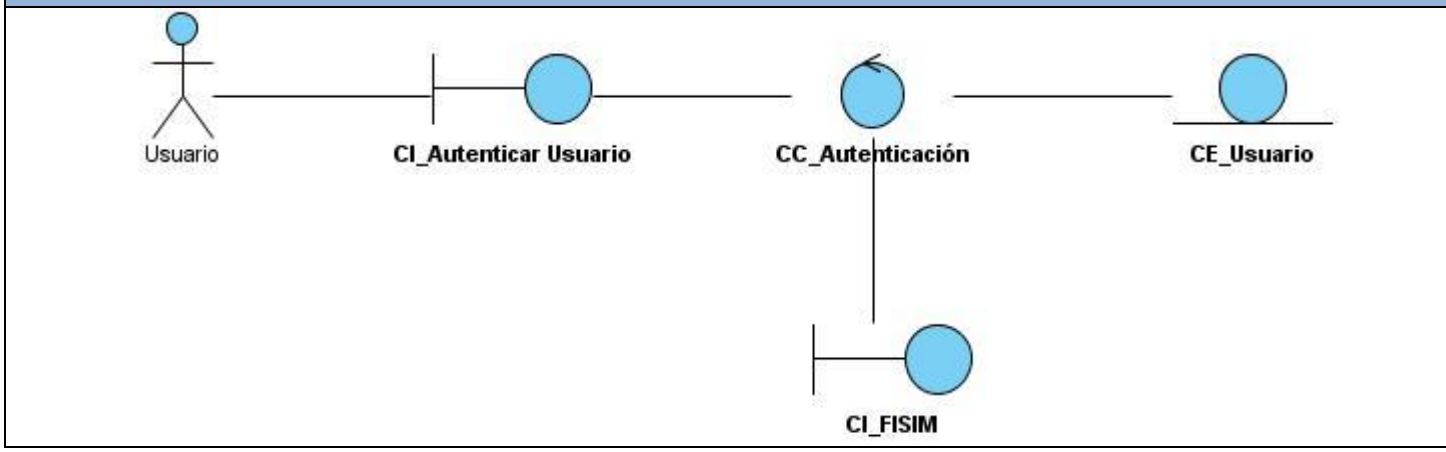


Figura 14: Diagrama de Clases del Análisis del CU Autenticar Usuario

Diagrama de Clases del Análisis del CU Gestionar Simulación

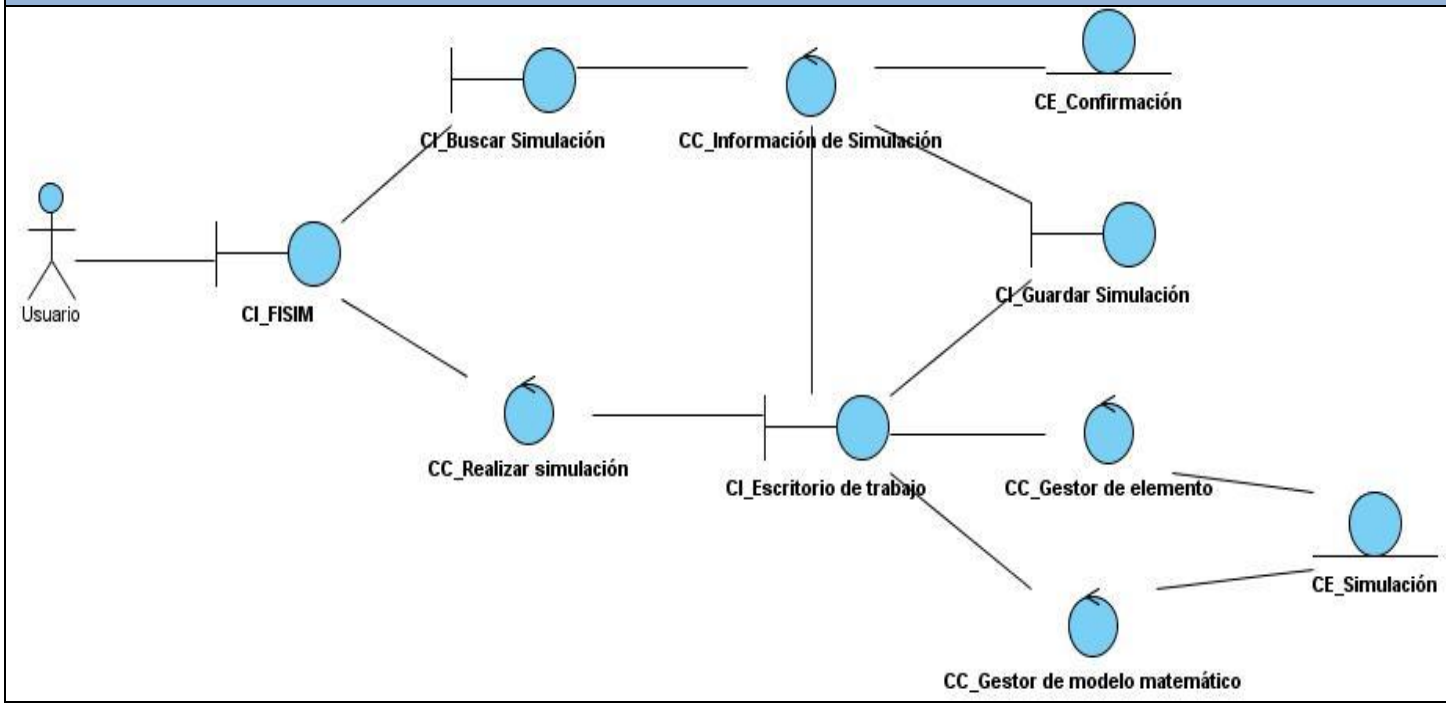


Figura 15: Diagrama de Clases del Análisis del CU Gestionar Simulación

3.1.2. Diagrama de Interacción del Análisis

La secuencia de acciones en un caso de uso comienza cuando un actor invoca el caso de uso mediante el envío de algún tipo de mensaje del sistema. Los diagramas de interacción se utilizan para modelar los aspectos dinámicos de un sistema, lo que conlleva a modelar instancias concretas o prototípicas de clases interfaces, componentes y nodos, junto con los mensajes enviados entre ellos, todo en el contexto de un escenario que ilustra un comportamiento. En el contexto de las clases describen la forma en que grupos de objetos colaboran para proveer un comportamiento. Se dividen en dos categorías los diagramas de colaboración y los diagramas de secuencia (11).

Un diagrama de colaboración es un diagrama de interacción que destaca la organización estructural de los objetos que envían y reciben mensajes, muestran las relaciones entre los objetos y los mensajes que intercambian. Un diagrama de secuencia es un diagrama de interacción que destaca la ordenación temporal de los mensajes, muestran las interacciones expresadas en función de secuencias temporales.

A continuación se muestran los diagramas de colaboración y secuencia de los casos de uso Autenticar Usuario y Gestionar Simulación del software FISIM. Para consultar los diagramas de colaboración correspondientes al resto de los casos de uso remitirse al Anexo 2 y para consultar los diagramas de secuencia correspondientes al resto de los casos de uso remitirse al Anexo 3.

Diagrama de colaboración del CU Autenticar Usuario

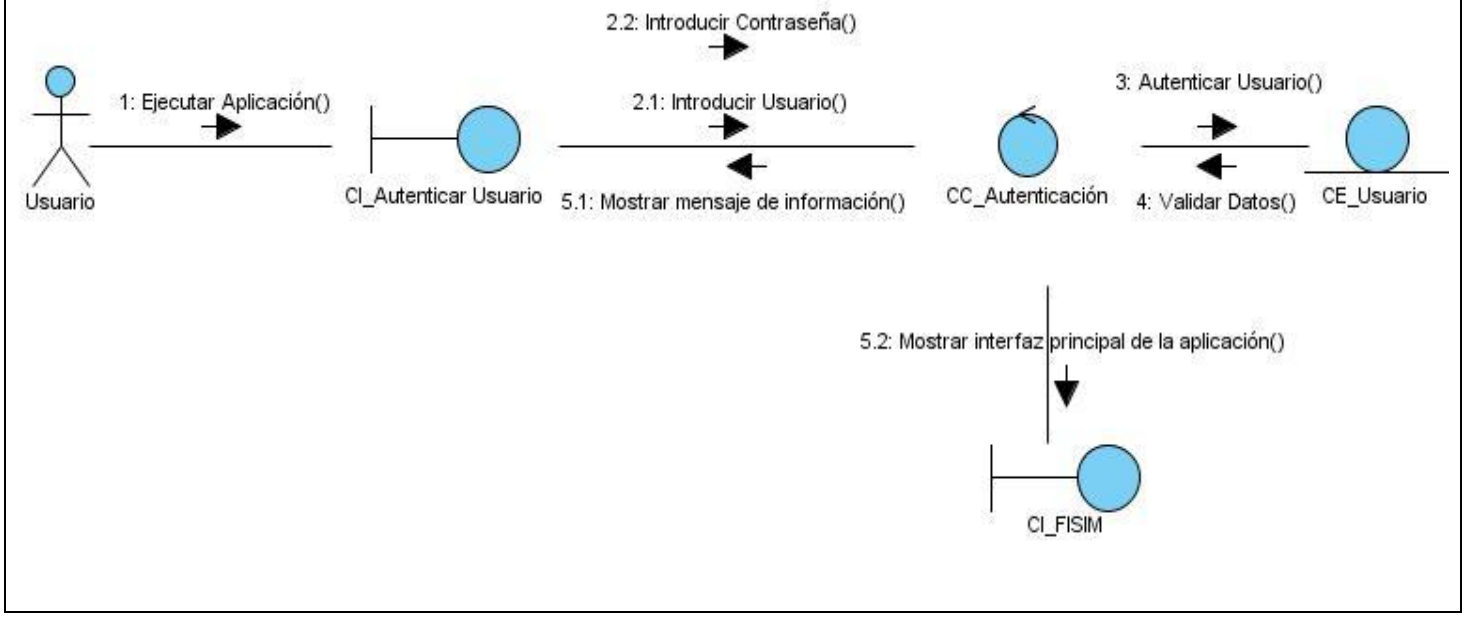


Figura 16: Diagrama de colaboración del CU Autenticar Usuario

Diagrama de colaboración de la sección Buscar Simulación del CU Gestionar Simulación

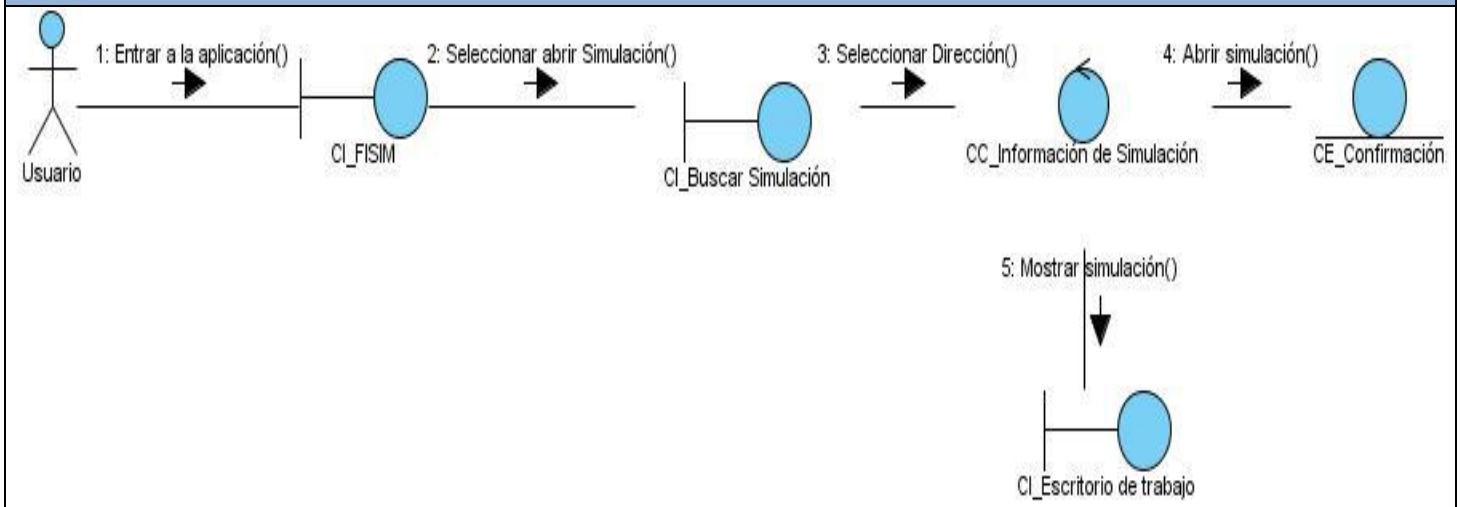


Figura 17: Diagrama de colaboración de la sección Buscar Simulación del CU Gestionar Simulación

Diagrama de colaboración de la sección Guardar Simulación del CU Gestionar Simulación

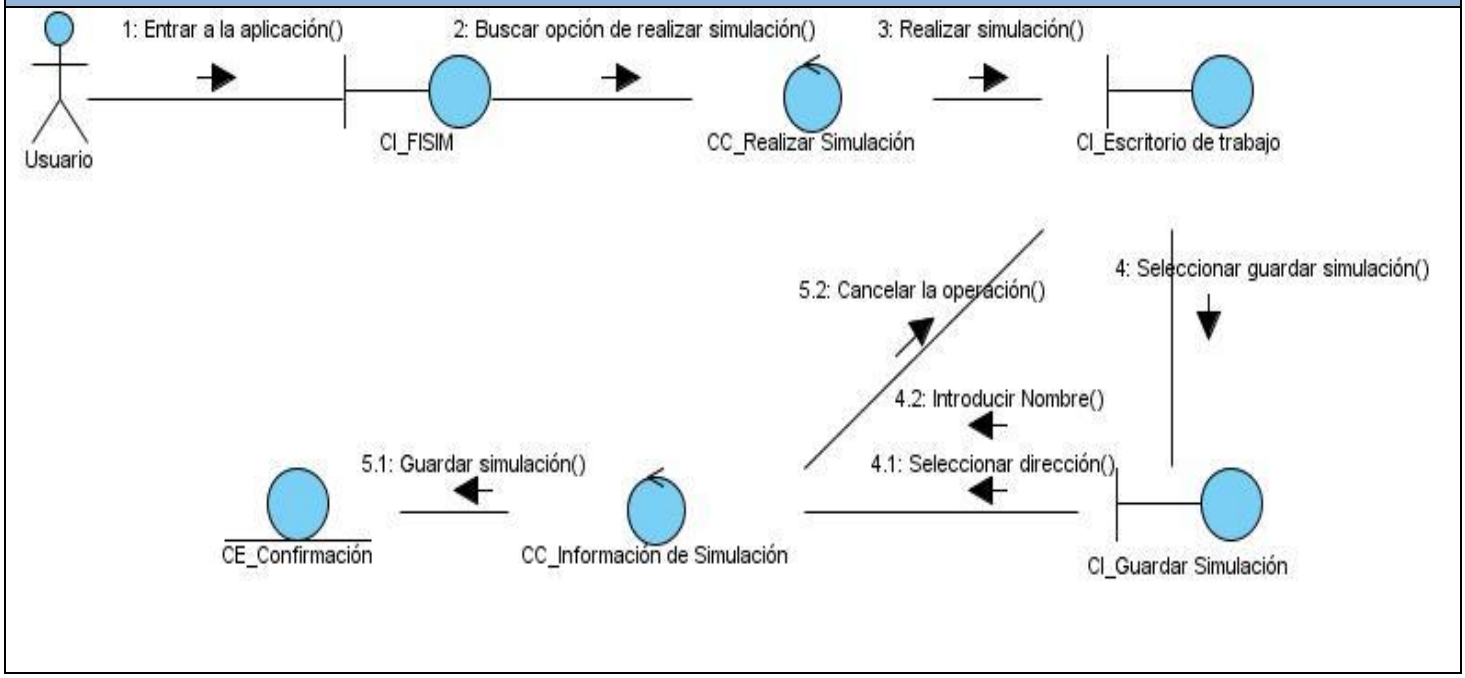


Figura 18: Diagrama de colaboración de la sección Guardar Simulación del CU Gestionar Simulación

Diagrama de colaboración de la sección Elemento del CU Gestionar Simulación

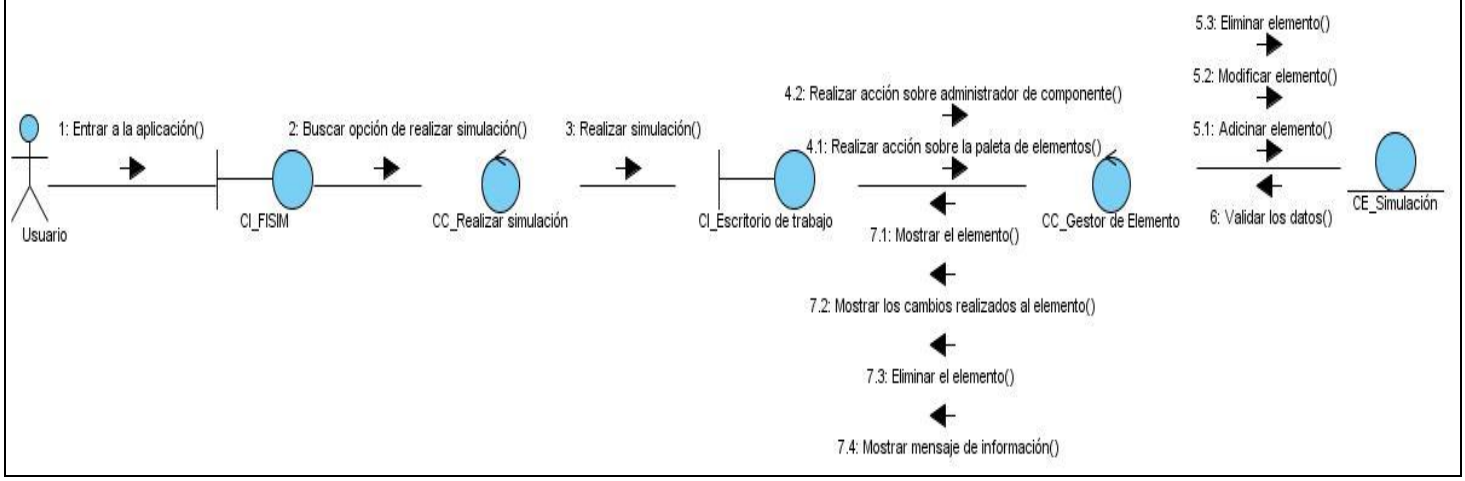


Figura 19: Diagrama de colaboración de la sección Elemento del CU Gestionar Simulación

Diagrama de colaboración de la sección Modelo Matemático del CU Gestionar Simulación

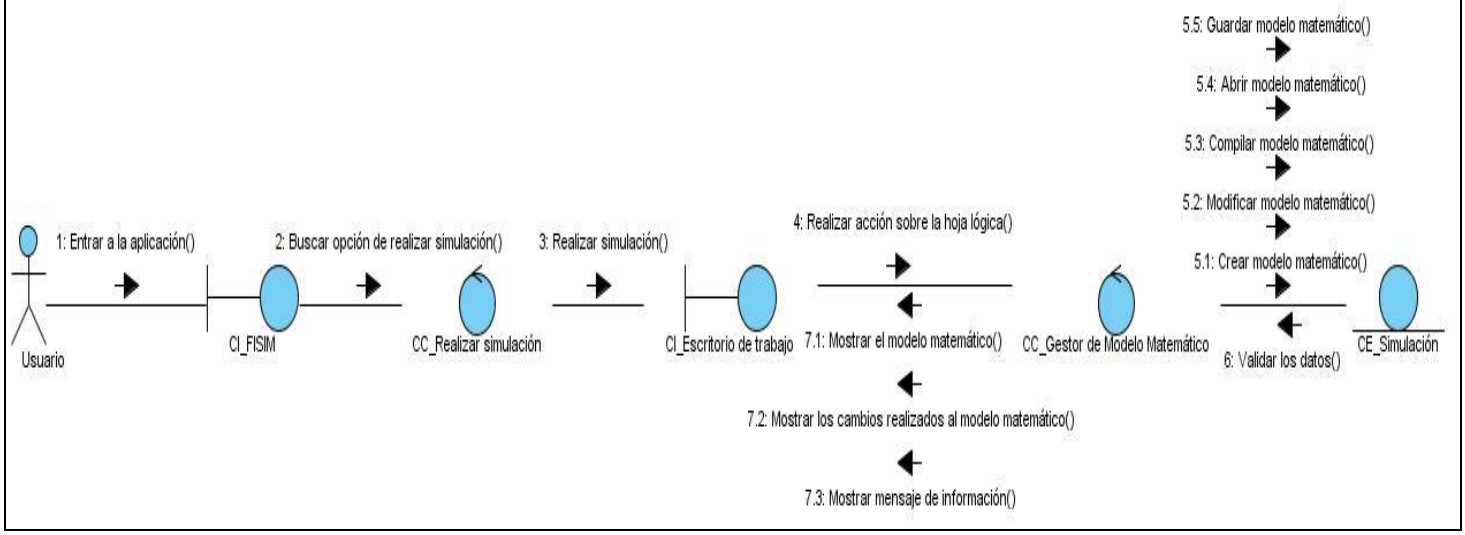


Figura 20: Diagrama de colaboración de la sección Modelo Matemático del CU Gestionar Simulación

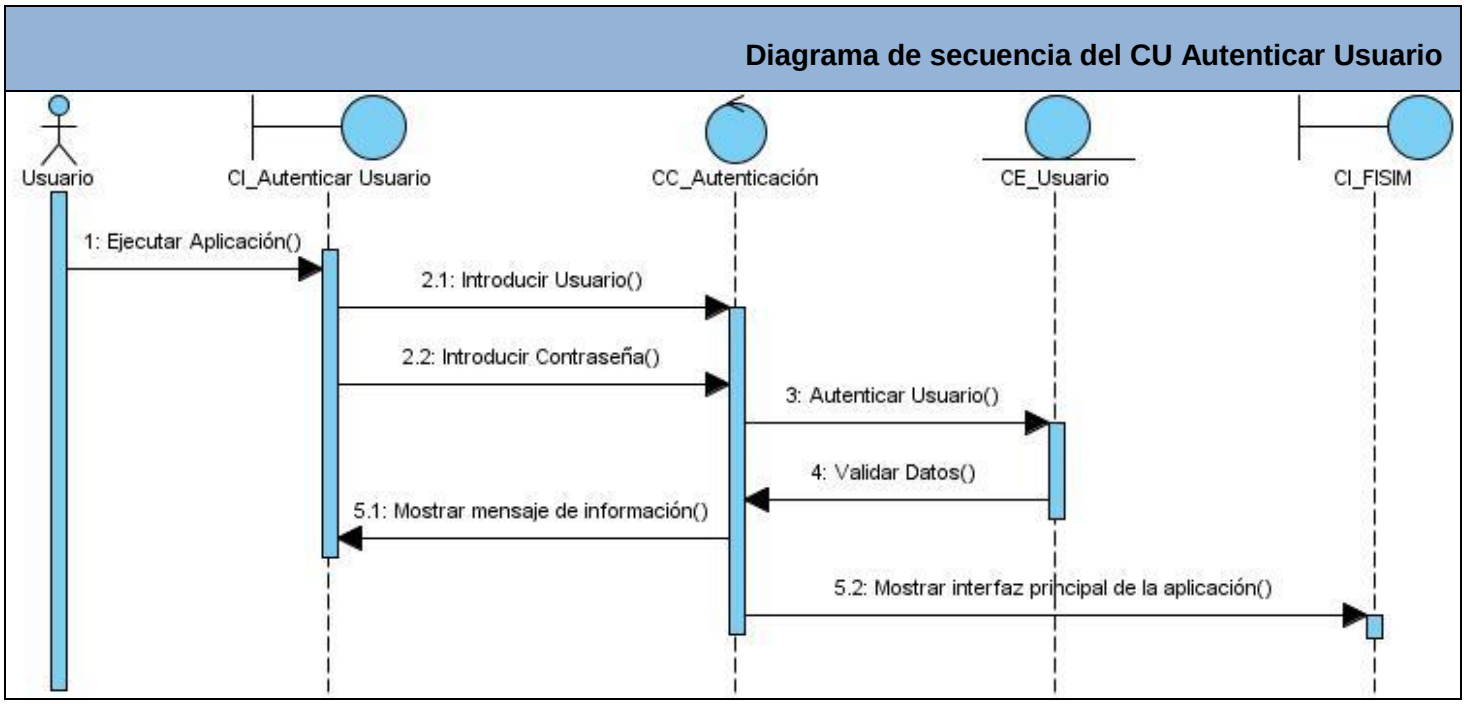


Figura 21: Diagrama de secuencia del CU Autenticar Usuario

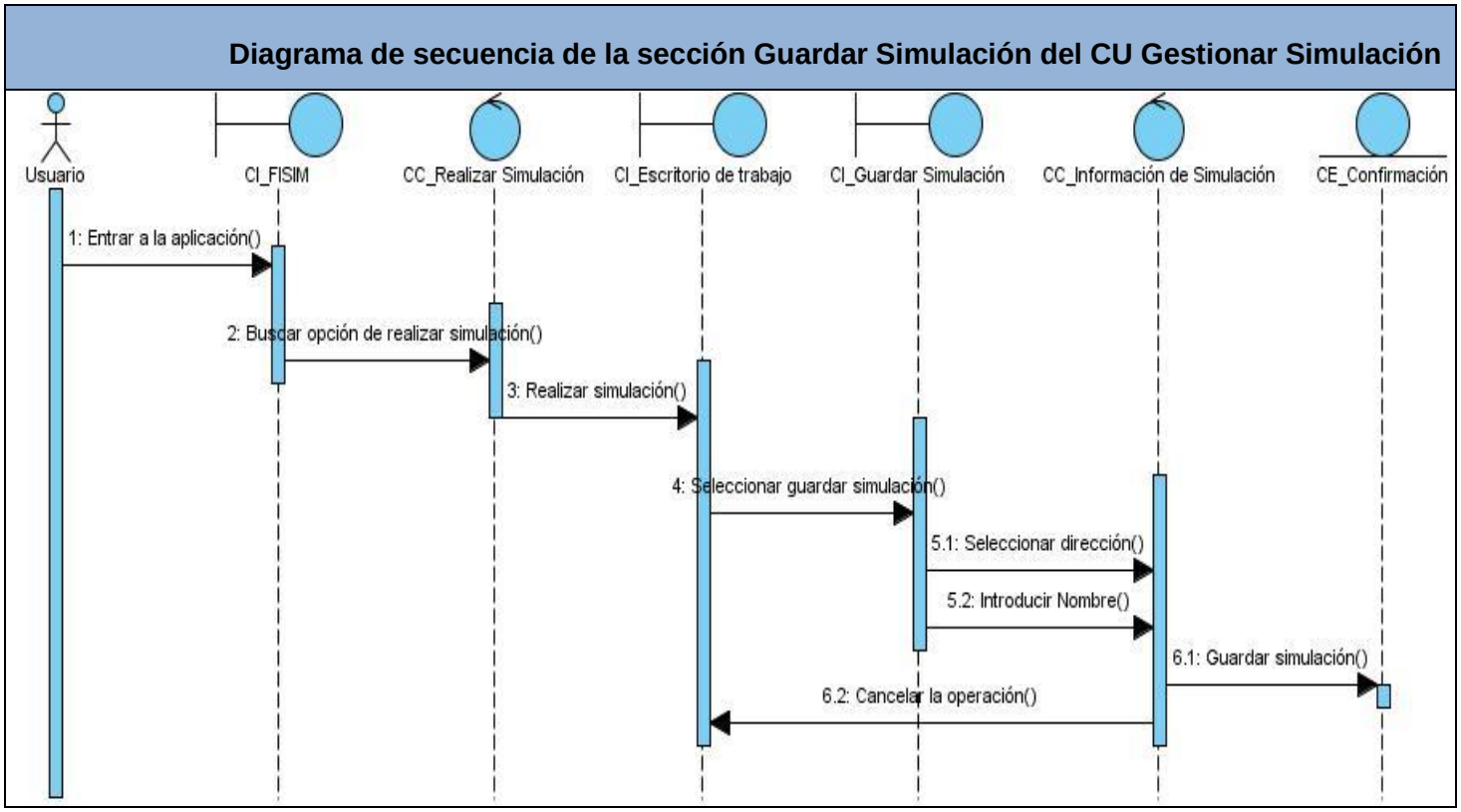


Figura 22: Diagrama de secuencia de la sección Guardar Simulación del CU Gestionar Simulación

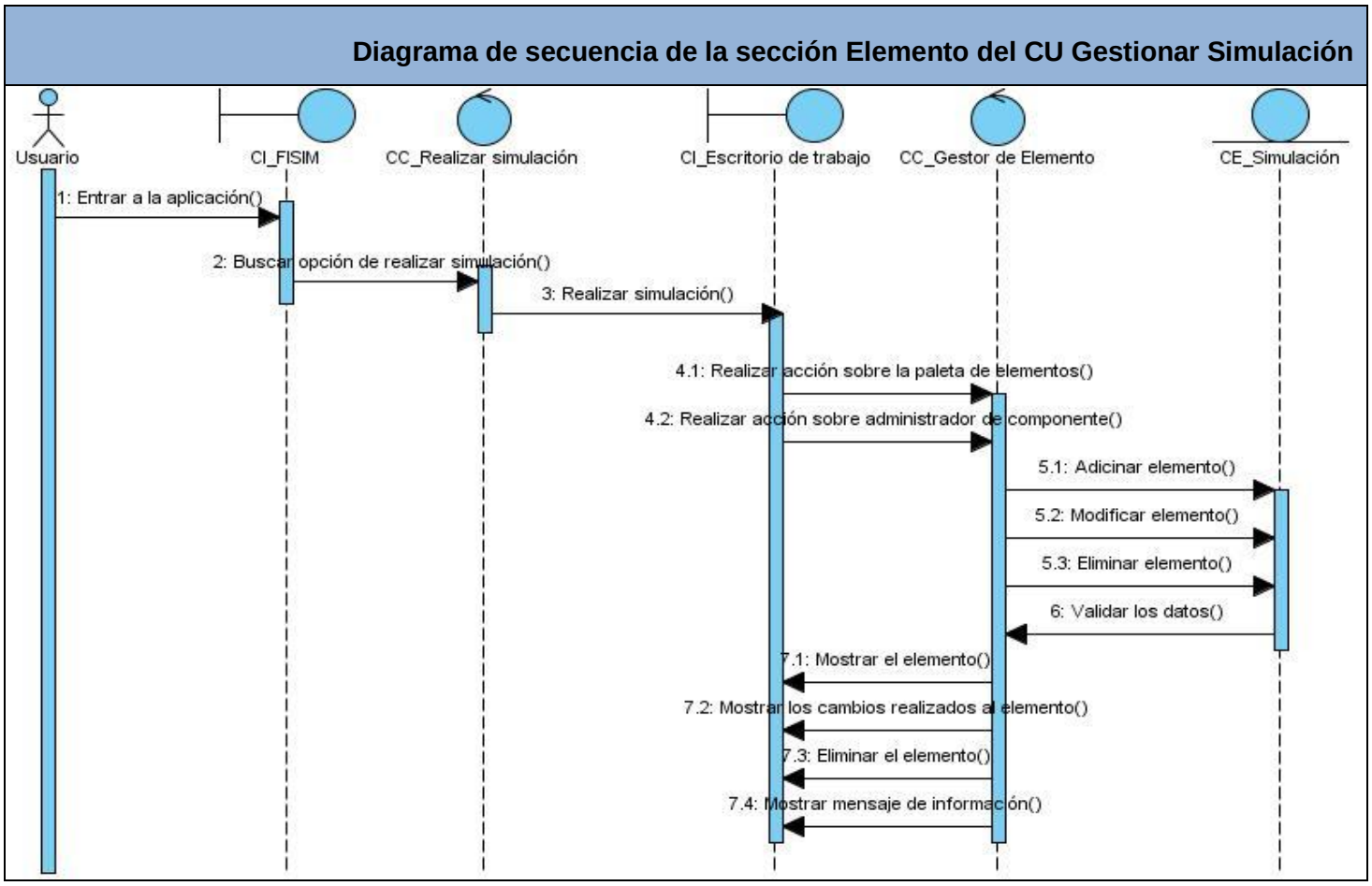


Figura 23: Diagrama de secuencia de la sección Elemento del CU Gestionar Simulación

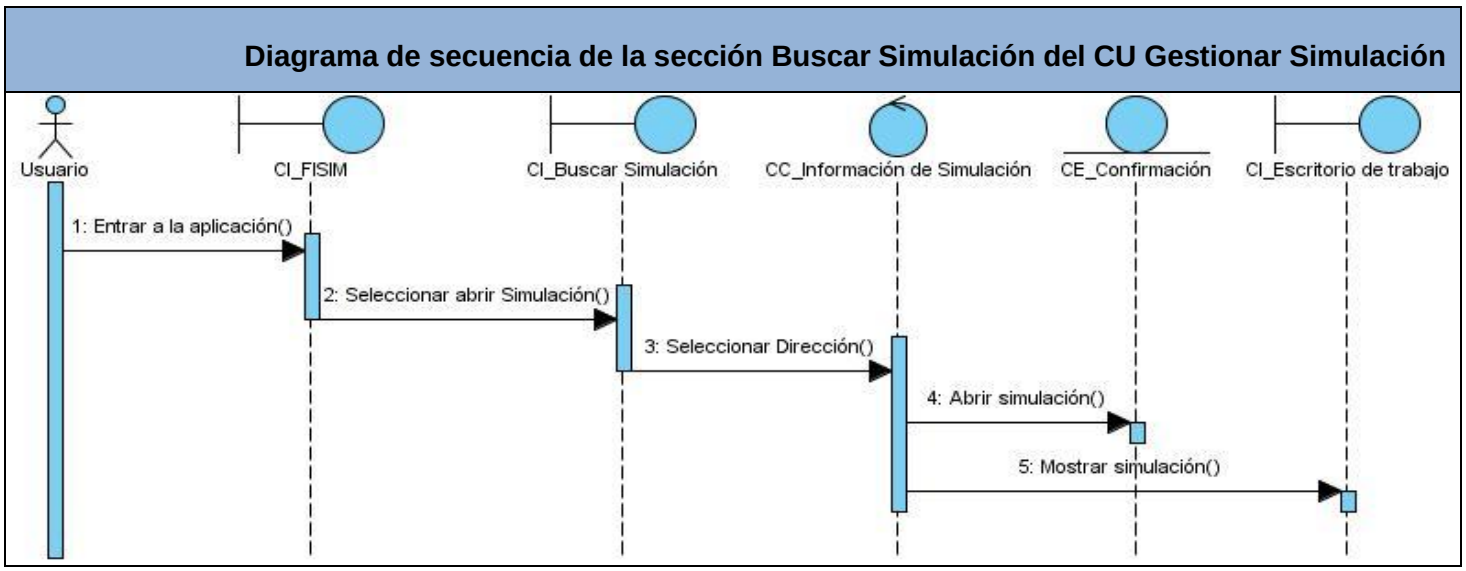


Figura 24: Diagrama de secuencia de la sección Buscar Simulación del CU Gestionar Simulación

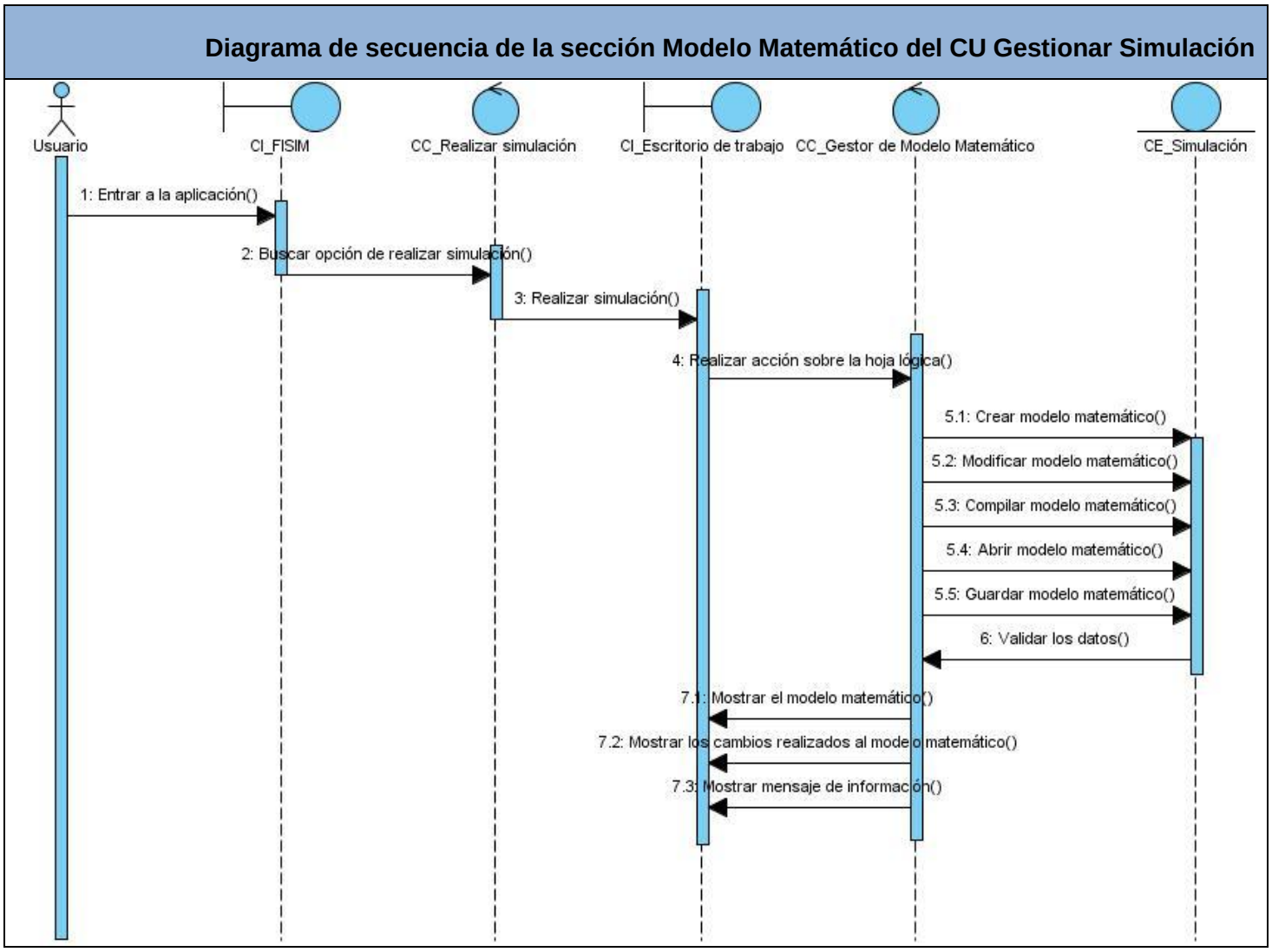


Figura 25: Diagrama de secuencia de la sección Modelo Matemático del CU Gestionar Simulación

3.1.3. Diagrama de Paquetes del Análisis

Los paquetes del análisis proporcionan un medio para organizar los artefactos del modelo de análisis en piezas manejables. Un paquete de análisis puede constar de clases del análisis, realizaciones de casos de uso y de otros paquetes del análisis. Estos paquetes tienen las siguientes características (11):

- Pueden representar una separación de intereses de análisis, o sea en un sistema grande algunos paquetes del análisis pueden analizarse de manera separada, posiblemente de manera concurrente por diferentes desarrolladores con distintos conocimientos del dominio.
- Deberían crearse basándose en los requisitos funcionales y en el dominio del problema y deberían

ser reconocibles por las personas con conocimiento del dominio. No deberían basarse en los requisitos no funcionales o en el dominio de la solución.

- Se convertirán probablemente en subsistemas en las dos capas de aplicación superiores del modelo de diseño o se distribuirán entre ellos. En algunos casos podrían reflejar una capa completa de primer nivel en el modelo de diseño.

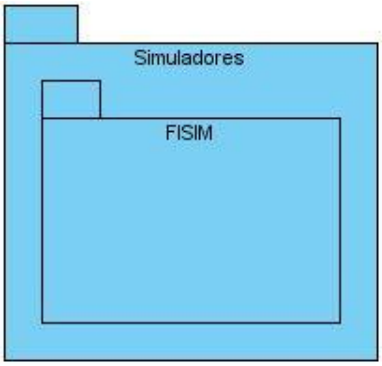


Figura 26: Diagrama de paquetes del análisis

3.2. Modelo de Diseño

El modelo de diseño es un modelo de objetos que describe la realización física de los casos de uso centrándose en cómo los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de implementación, tienen impacto en el sistema. Además sirve de abstracción de la implementación del sistema y es utilizado como una entrada fundamental de las actividades de implementación (11).

En el modelo de diseño los casos de uso son realizados por las clases del diseño y sus objetos. Esto se representa por colaboraciones en el modelo y denota realización de casos de uso del diseño. Adviértase que realización de casos de uso del diseño es diferente a realización de casos de uso del análisis. Lo primero describe como se realiza un caso de uso en términos de interacción entre objetos de diseño, mientras que lo segundo describe como se realiza un caso de uso en términos de interacción entre objetos del análisis (11).

3.2.1. Diagrama de Clases del Diseño

Una clase del diseño y sus atributos participan en varias realizaciones de casos de uso. En un diagrama de clases del diseño se exponen las clases que intervienen en las realizaciones de los casos de uso del sistema. En este tipo de diagrama se representa un nivel de detalle más alto que en los diagramas de

clases del análisis, relacionándose con el lenguaje de programación del cual se hará uso en la implementación del sistema (20).

El diagrama de clases de diseño describe gráficamente las especificaciones de las clases de software y de las interfaces en una aplicación. Normalmente contiene la siguiente información (21):

- Clases, asociaciones y atributos.
- Interfaces con sus operaciones y constantes.
- Métodos.
- Información sobre los tipos de los atributos, navegabilidad y dependencias.

A continuación se muestra los diagramas de clases de diseño de los casos de uso Autenticar Usuario y Gestionar Simulación del software FISIM. Para consultar los diagramas de clases del diseño correspondientes al resto de los casos de uso remitirse al Anexo 4.

Diagrama de clases del diseño del CU Autenticar Usuario

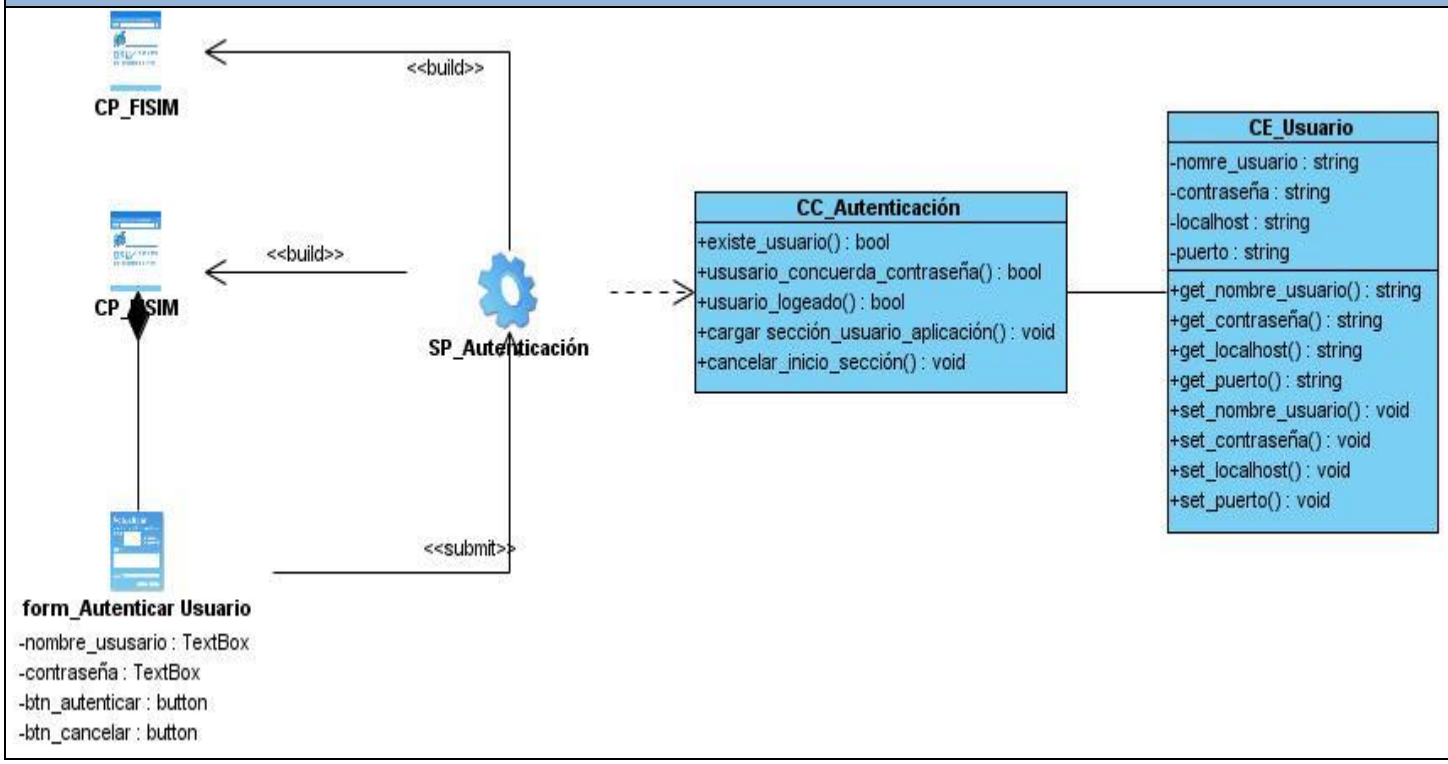


Figura 27: Diagrama de clases del diseño del CU Autenticar Usuario

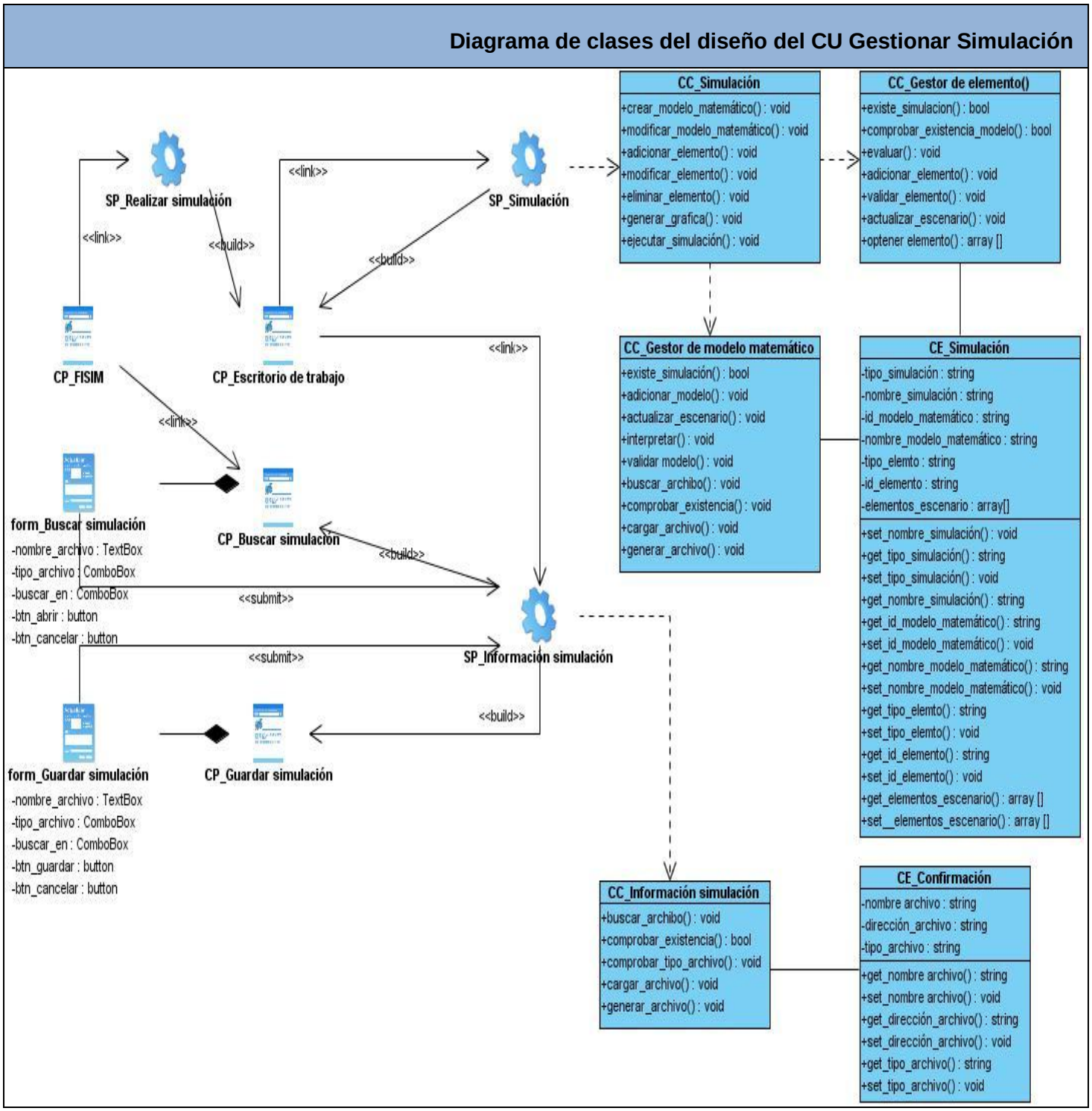


Figura 28: Diagrama de clases del diseño del CU Gestionar Simulación

3.2.2. Diagrama de Interacción del Diseño

A continuación se muestra los diagramas de secuencia del diseño de los casos de uso Autenticar Usuario y Gestionar Simulación del software FISIM. Para consultar los diagramas de secuencia correspondientes al resto de los casos de uso remitirse al Anexo 5.

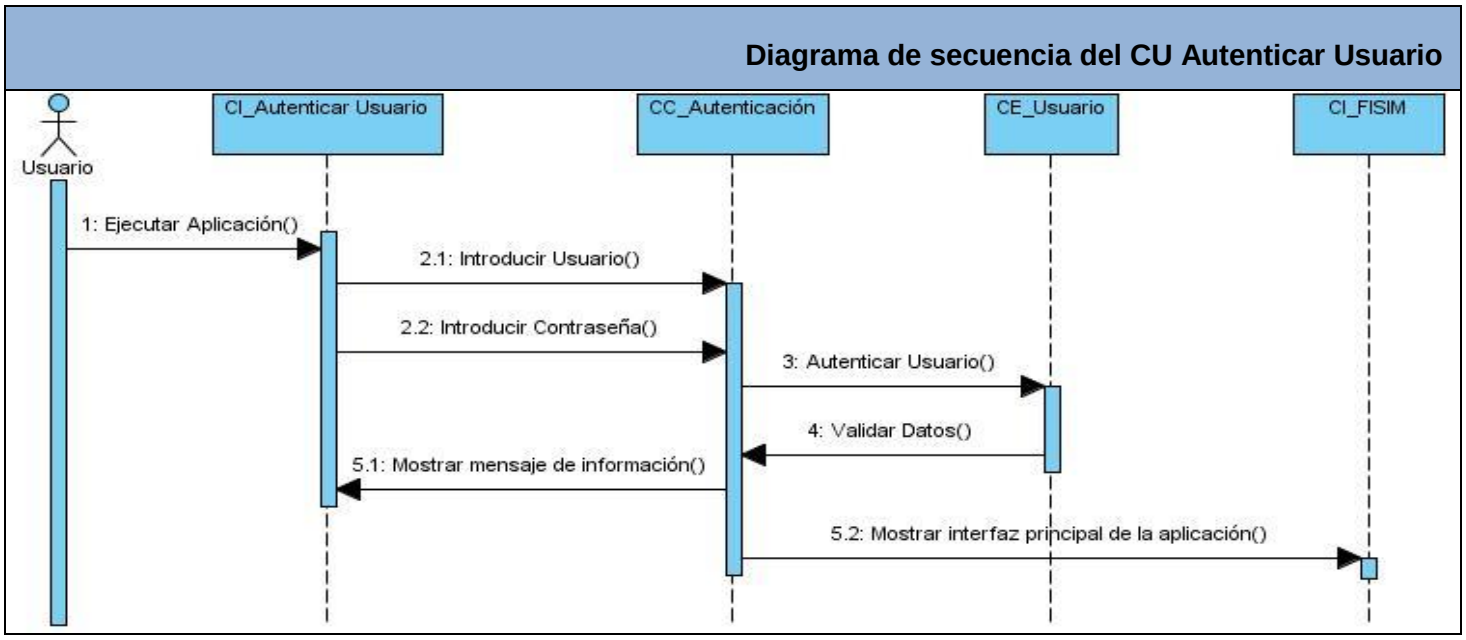


Figura 29: Diagrama de secuencia del CU Autenticar Usuario

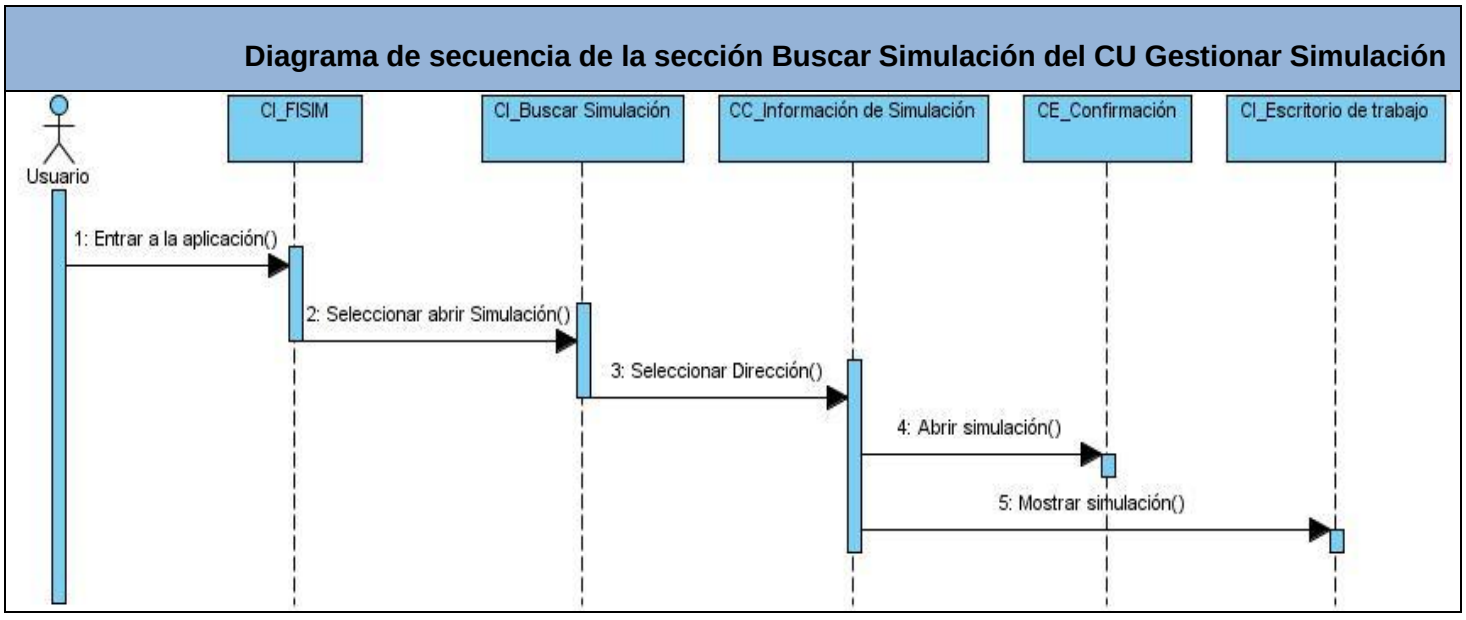


Figura 30: Diagrama de secuencia de la sección Buscar Simulación del CU Gestionar Simulación

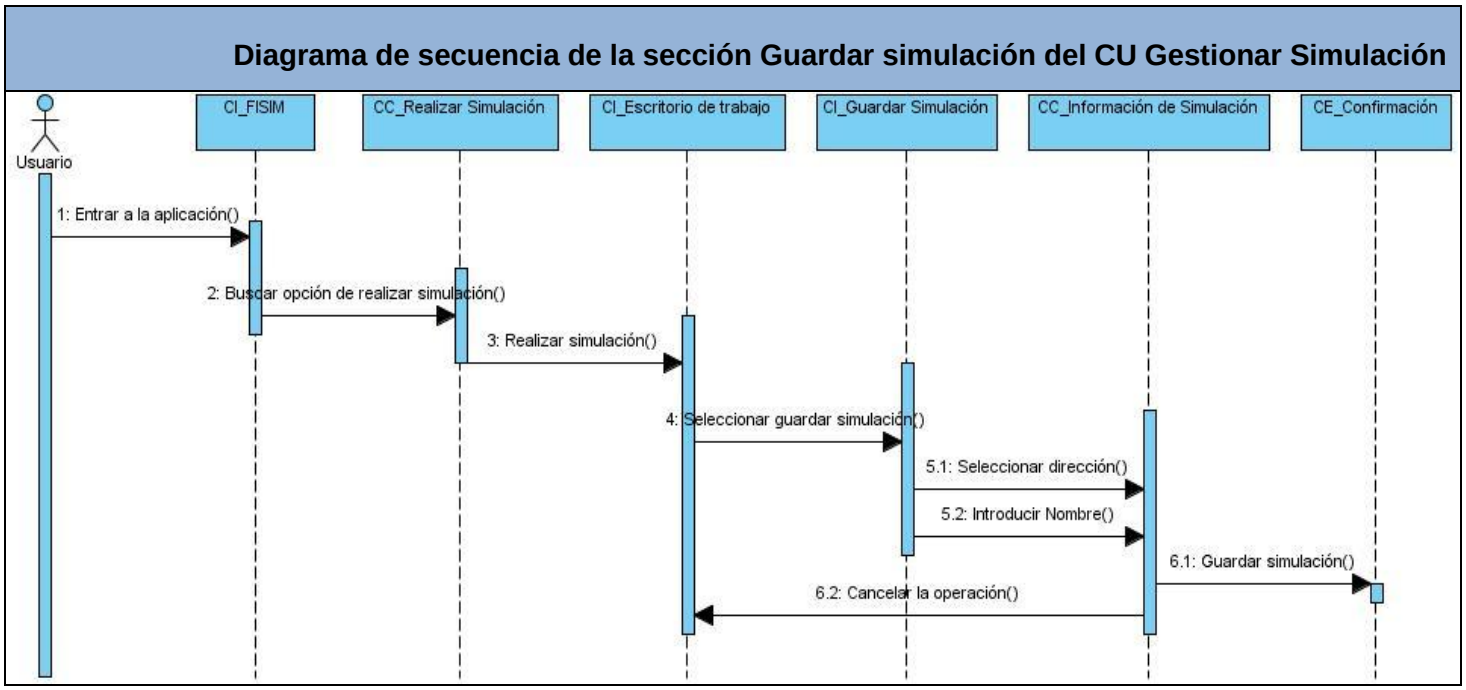


Figura 31: Diagrama de secuencia de la sección Guardar Simulación del CU Gestionar Simulación

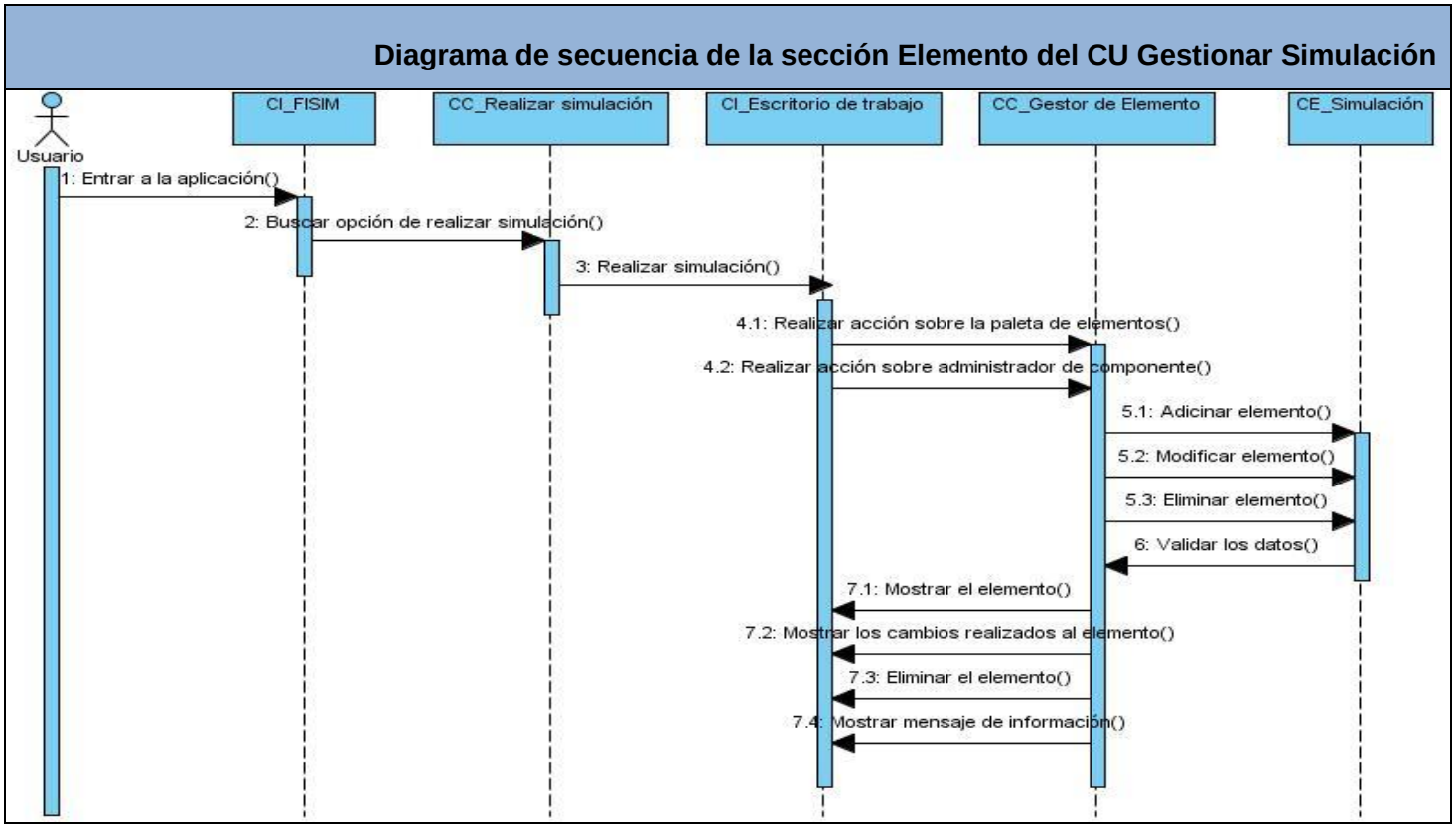


Figura 32: Diagrama de secuencia de la sección Elemento del CU Gestionar Simulación

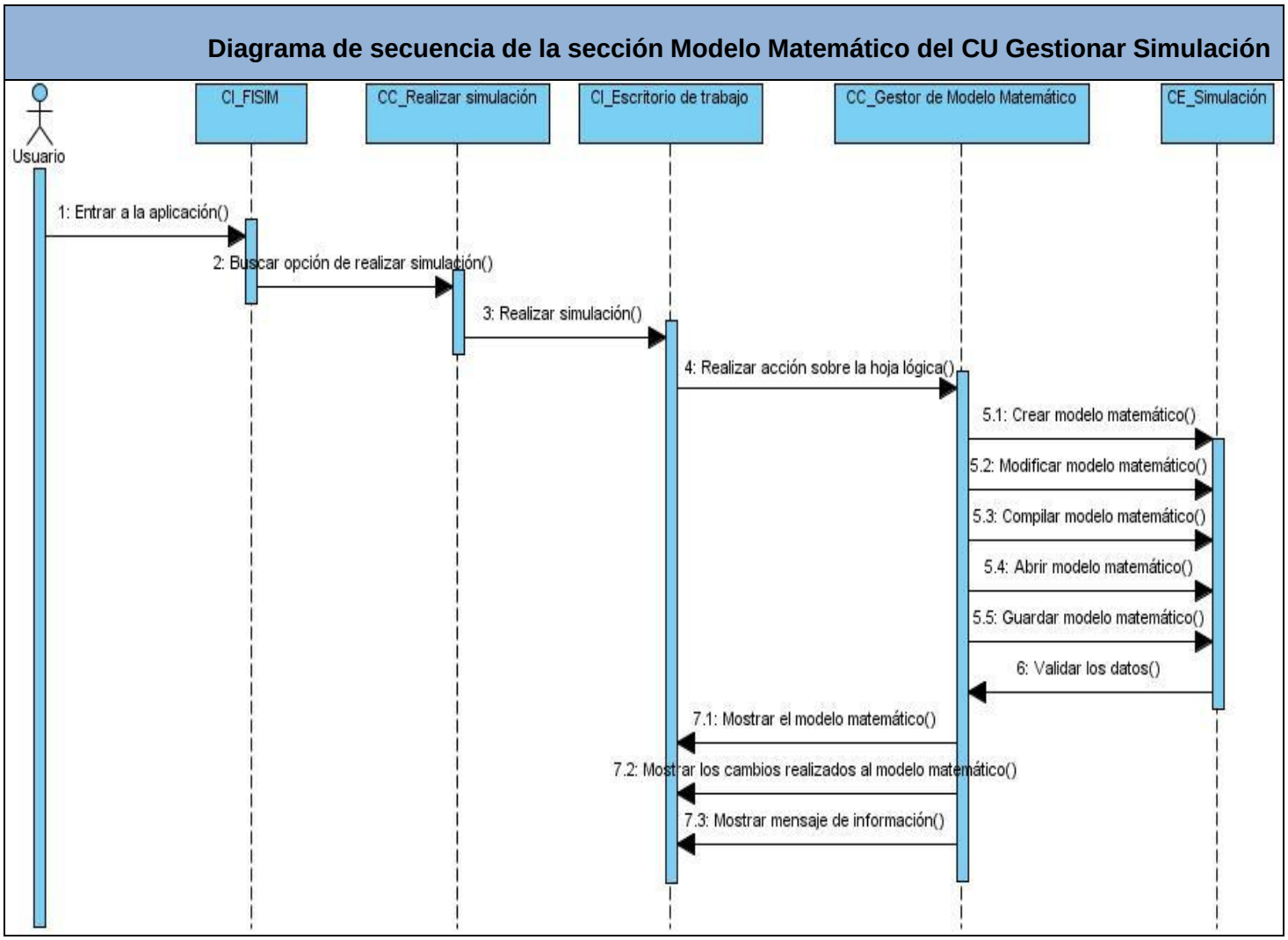


Figura 33: Diagrama de secuencia de la sección Modelo Matemático del CU Gestionar Simulación

3.2.3. Diagrama de paquetes del diseño

A continuación se muestra el diagrama de paquetes del diseño, el mismo está compuesto por tres paquetes fundamentales:

- El paquete FISIM que contiene los principales elementos de la vista y las clases que permiten al subsistema controlar sus componentes visuales y conexiones a la plataforma ZERA.
- El paquete Modelo que contiene todas las clases entidades de una simulación así como la clase controlador Escritorio de Trabajo y la clase principal Simulación.

- El paquete Alfacompiler que contiene todas las clases necesarias para interpretar el modelo matemático definido para una simulación.

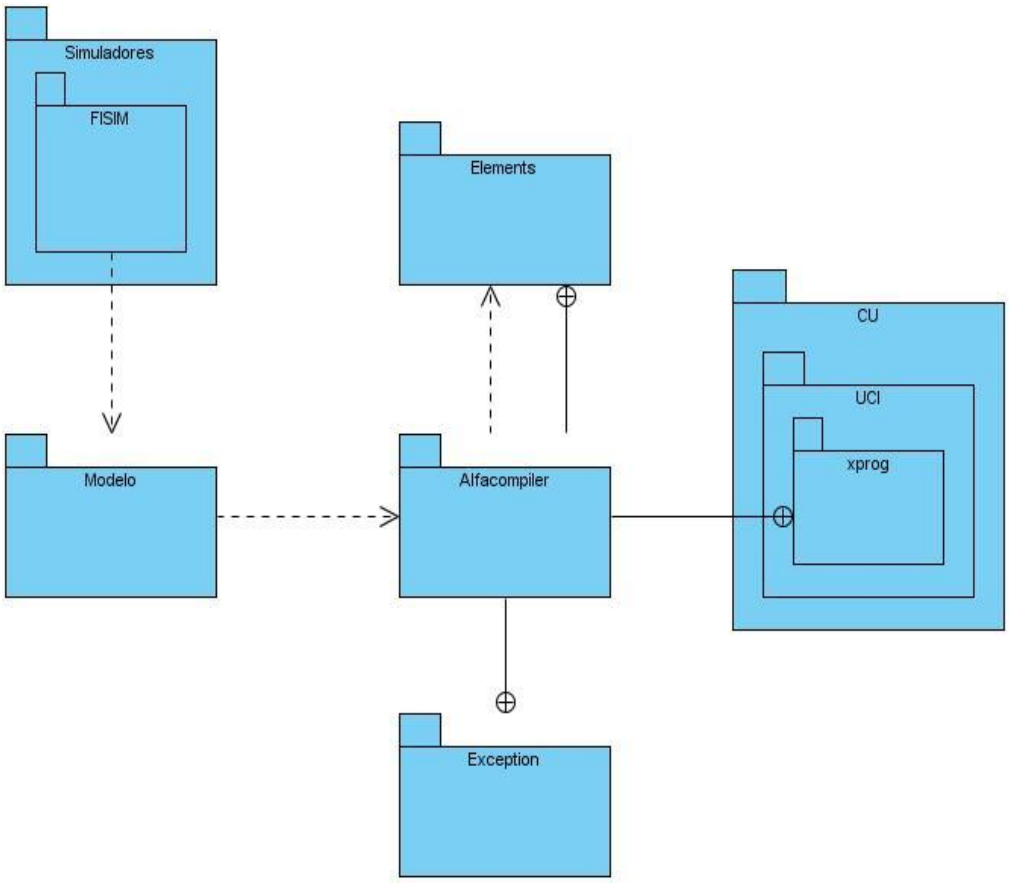


Figura 34: Diagrama de paquetes del diseño

3.3. Patrones de Diseño

Los patrones son soluciones de sentido común que deberían formar parte del conocimiento de un diseñador experto. Mejoran la comunicación entre diseñadores, pues establecen un marco de referencia. Facilitan el aprendizaje al programador inexperto, pudiendo establecer parejas problema-solución. En la programación orientada a objetos resulta complicado descomponer el sistema en objetos, estos patrones permitirán identificar a los objetos apropiados de una manera mucho más sencilla. Además, los patrones de diseño, también ayudarán a especificar las interfaces, identificando los elementos claves en las interfaces y las relaciones existentes entre distintas interfaces. De igual modo facilitarán la especificación de las implementaciones. También ayudan a reutilizar código, facilitando la decisión entre herencia o composición.

Relacionan estructuras en tiempo de compilación y en tiempo de ejecución y permiten hacer un diseño preparado para el cambio (20).

La arquitectura propuesta para el subsistema FISIM está basada en implementar la solución orientada a objetos mediante una variante del patrón arquitectónico Modelo Vista Controlador el cual permite separar en tres niveles las funcionalidades de una aplicación con el objetivo de incrementar la usabilidad de las mismas. Es decir, separa el modelado del dominio, la presentación y las acciones basadas en datos ingresados por el usuario. Los niveles en los que se divide son:

- **Modelo:** Esta es la representación específica de la información con la cual el sistema opera. Se limita a lo relativo de la vista y su controlador facilitando las presentaciones visuales complejas. El sistema también puede operar con más datos no relativos a la presentación, haciendo uso integrado de otras lógicas de negocio y de datos afines con el sistema modelado.

- **Vista:** Esta presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario.

- **Controlador:** Responde a eventos, usualmente acciones del usuario e invoca peticiones al modelo y probablemente a la vista.

Para el desarrollo de los diagramas de clases del diseño se utilizaron los patrones GRASP (General Responsibility Asignment Software Patterns), los que proporcionaron que los estos sean más reusables, flexibles y tengan una mayor robustez. Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. Constituyen un apoyo para la enseñanza que ayuda a entender el diseño de objeto esencial y aplica el razonamiento para el diseño de una forma sistemática, racional y explicable. Dentro de esta clasificación de patrones se encuentran entre otros los siguientes (19):

- **Experto**: Asignar una responsabilidad al experto en información: La responsabilidad de realizar una labor es de la clase que tiene o puede tener los datos involucrados. Una clase, contiene toda la información necesaria para realizar la labor que tiene encomendada. Este patrón se usa más que cualquier otro al asignar responsabilidades; es un principio básico que suele utilizarse en el diseño orientado a objetos; expresa simplemente la "intuición" de que los objetos hacen cosas relacionadas con la información que poseen.

- **Creador**: Asignar a la clase B la responsabilidad de crear una instancia de clase A en alguno de los siguientes casos: B agrega los objetos de A; B contiene a los objetos de A; B registra las instancias de los objetos de A; B utiliza específicamente los objetos de A; B tiene los datos de inicialización que serán transmitidos a A cuando este objeto sea creado. El propósito fundamental de este patrón es encontrar un creador que debemos conectar con el objeto producido en cualquier evento. Al escogerlo como creador, se da soporte al bajo acoplamiento.

- **Controlador**: Asignar la responsabilidad de controlar el flujo de eventos del sistema, a clases específicas. Esto facilita la centralización de actividades. La mayor parte de los sistemas reciben eventos de entrada externa, por lo cual, si se recurre a un diseño orientado a objetos, hay que elegir los controladores que manejen esos eventos de entrada.

- **Alta Cohesión:** Asignar una responsabilidad de modo que la cohesión siga siendo alta. Una clase tiene responsabilidades moderadas en un área funcional y colabora con las otras para llevar a cabo las tareas. Una clase con mucha cohesión es útil porque es bastante fácil darle mantenimiento, entenderla y reutilizarla. Su alto grado de funcionalidad, combinada con una reducida cantidad de operaciones, también simplifica el mantenimiento y los mejoramientos. La ventaja que significa una gran funcionalidad también soporta un aumento de la capacidad de reutilización.

- **Bajo Acoplamiento**: Asignar una responsabilidad para mantener bajo acoplamiento. Debe haber pocas dependencias entre las clases. Si todas las clases dependen de todas ¿cuánto software podemos extraer de un modo independiente y reutilizable en otro proyecto? .Para determinar el nivel de acoplamiento de clases, son muy buenos los diagramas de colaboración de UML. Uno de los principales síntomas de un mal diseño y alto acoplamiento es una herencia muy profunda. Siempre hay que considerar las ventajas de la delegación respecto de la herencia**.** Además soporta el diseño de clases más independientes, que reducen el impacto de los cambios, y también más reutilizables, que acrecienta la oportunidad de una mayor productividad.

El patrón experto se evidencia en los diagramas sobre todo en las clases controladoras de los distintos análisis puesto que estas poseen la mayor información necesaria para resolver las funcionalidades requeridas. También se evidencia el bajo acoplamiento una vez que las relaciones entre las clases que participan en el desarrollo de los distintos casos de usos son mínimas asegurando así que exista entre ellas una alta cohesión. Además se utiliza el patrón creador con el objetivo de asignar la responsabilidad de crear instancias de otras clases a la clase adecuada para este fin.

También se utiliza el patrón de creación Singleton. De forma general los patrones de creación abstraen la forma en la que se crean los objetos, permitiendo tratar las clases a crear de forma genérica dejando para más tarde la decisión de qué clases crear o cómo crearlas. Según donde se tome dicha decisión se pueden clasificar a los patrones de creación en *patrones de creación de clase* y *patrones de creación de objeto*. En específico Singleton está diseñado para restringir la creación de objetos pertenecientes a una clase o el valor de un tipo a un único objeto. Su intención consiste en garantizar que una clase sólo tenga una instancia y proporcionar un punto de acceso global a ella. Este hace que la clase sea responsable de su única instancia, quitando así este problema a los clientes (21).

3.4. Conclusiones

El análisis y diseño brinda a un software la primera visión de lo que pudiera ser la solución en el desarrollo del mismo. Durante el desarrollo de este capítulo se definieron las clases del análisis y el diseño mostrando sus relaciones a través del modelado de los diagramas de clases del análisis y del diseño, enfocados al cumplimiento de los requerimientos funcionales de FISIM. También se mostró la colaboración entre esas clases mediante diferentes diagramas de interacción. Para lograr un mejor diseño se valoró la aplicación de los patrones de diseño teniendo en cuenta las características del sistema para ayudar a la optimización del mismo.

CONCLUSIONES GENERALES

Luego de finalizada la investigación y realizadas todas las tareas propuestas, se puede afirmar que se cumplieron los objetivos planteados ya que:

- Se realizó un estudio de simuladores existentes que tenían características similares a las del simulador FISIM, facilitando la creación de funcionalidades para el mismo.
- Se realizó una comparación de diferentes metodologías de desarrollo de software existentes escogiéndose RUP para el desarrollo del presente trabajo.
- Luego de haber realizado un análisis de algunas de las herramientas CASE existentes, se definió utilizar para el presente trabajo la herramienta Visual Paradigm, la cual se vincula muy bien con la metodología seleccionada.
- Se generó un Modelo de Dominio logrando combinar los principales conceptos existentes en el simulador FISIM.
- Se especificaron todas las funcionalidades que debía cumplir el sistema mediante los requisitos funcionales y no funcionales del mismo.
- Se generó un diagrama de casos de uso y se utilizaron patrones para lograr un mejor entendimiento de los mismos.
- Se realizó la especificación de los casos de uso del sistema, dejando plasmada en las mismas las descripciones de cada uno de los casos de uso identificados en el simulador.
- Se realizó el análisis y el diseño del software generándose todos los artefactos necesarios para completar la información necesaria sobre todas las actividades realizadas y lograr una mejor comprensión del negocio.
- La documentación generada sirve de base para la futura implementación del simulador FISIM.

RECOMENDACIONES

Luego de haberse realizado el presente trabajo de diploma, a raíz de la investigación que se realizó para la confección del mismo se recomienda:

- Incluir nuevas tipologías de simulaciones y elementos necesarios para la realización de las mismas, con el objetivo de aumentar los objetos de aprendizaje del producto.
- Agregar nuevas funcionalidades a medida que avance el desarrollo del sistema de acuerdo a las expectativas de los usuarios finales.
- Agregar una sección en la aplicación FISIM que permita a los estudiantes ver las tareas que les asignó el profesor.
- Lograr una mayor integración del subsistema FISIM y la plataforma ZERA con el fin de mejorar la trazabilidad de los usuarios y la asignación de tareas.

REFERENCIAS BIBLIOGRÁFICAS

1. **Cuéllar Vazquez, Edith, Rodríguez Gómez, Gustavo y Muñoz Arteaga, Jaime.** Inaoe. *Inaoe.* [En línea] [Citado el: 02 de 11 de 2010.] http://ccc.inaoep.mx/~grodrig/Descargas/PatSSD.pdf. 1.

2. **S.M, Alessi y S.R, Trollip.** *Computer Sased Instruction, Method and Development.* New Jersey : Ed. Prentice Hall Inc., 1985. 2.

3. Universidad Autónoma de Aguascalientes. *Universidad Autónoma de Aguascalientes.* [En línea] Universidad Autónoma de Aguascalientes. [Citado el: 02 de 11 de 2010.] http://sned.dgd.uaa.mx/educadis/index.php?option=com_content&task=view&id=14&Itemid=28.

4. **Sánchez, Jaime.** Monografias.com. *Monografias.com.* [En línea] 1999. [Citado el: 06 de 11 de 2010.] http://www.monografias.com/trabajos31/software-educativo-cuba/software-educativo-cuba.shtml#softeducat.

5. **Bonet, Jordi.** Softonic. *Softonic.* [En línea] [Citado el: 06 de 11 de 2010.] http://phet.softonic.com/.

6. la informacion.com. *la informacion.com.* [En línea] [Citado el: 06 de 11 de 2010.] http://www.microsiervos.com/archivo/ordenadores/phun-simulador.html.

7. **Sanchez, M.** *Metodologías de Desarrollo de Software.* 2004.

8. **Molpeceres, A.** Procesos de desarrollo: RUP, XP y FDD. *Procesos de desarrollo: RUP, XP y FDD.* [En línea] 2002. [Citado el: 07 de 11 de 2010.] http://www.javahispano.org/articles.article.action?id=76.

9. **Rodríguez Trujillo, Isyed.** *Ingeniería de Requerimientos.* Ciudad de la Habana : s.n., 2008.

10. **Calero Solís, Manuel.** willi.net. *willi.net.* [En línea] 2003. [Citado el: 07 de 11 de 2010.] http://www.willydev.net/descargas/prev/ExplicaXp.pdf.

11. **Jacobson, Ivar, Booch, Grady y Rumbaugh, Jame.** *El Proceso Unificado de Desarrollo de Software.* Ciudad de la Habana : Félix Varela, 2004.

12. **Rumbaugh, Jame, Jacobson, Ivar y Booch, Grady.** *El Lenguaje Unificado de Modelado. Manual de Referencia.* s.l. : Addison Wesley, 2000.

13. **Périssé, Marcelo Claudio.** *Proyecto Informático. Una Metodología Simplificada.* 2001.

14. **García Montero, María de los Angeles.** *Ingeniería de Requerimientos aplicada al proceso de Inicio de Investigación y Registro de Notificaciones en el CICPC.* Ciudad de la Habana : s.n., 2008.

15. IBM. [En línea] [Citado el: 08 de 11 de 2010.] http://www.ibm.com/software/awdtools/developer/rose/&ei=OaGBTY_1E8OEtgfilP3WBA&sa=X&oi=translate&ct=result&resnum=1&ved=0CCoQ7gEwAA&prev=/search%3Fq%3Drational%2Brose%26hl%3Des%26biw%3D1024%26bih%3D544%2.

16. **Delgado Dapena, Martha D.** [En línea] [Citado el: 06 de 01 de 2011.] http://www.inf.udec.cl/~revista/ediciones/edicion8/Rbc.pdf.

17. **Pressman, Roger S.** *Ingeniería del Software "Un enfoque prático".* Madrid : McGraw-Hill, 2002.

18. Clikear.com. *Clikear.com.* [En línea] [Citado el: 06 de 01 de 2011.] http://www.clikear.com/manuales/uml/diagramascasouso.aspx.

19. **Larman, C.** *UML y Patrones. Introducción al análisis y diseño orientado a objeto.* Mexico : s.n., 1999.

20. **Carvajal Pérez, Erlys. 2009.** *Análisis y diseño del subsistema de Análisis de Resultados de un Simulador de Procesos Químicos.* Ciudad de La Habana : s.n., 2009.

21. **Visconti, M., Astudillo, H.** *Fundamentos de Ingeniería de Software,* 2003.

BIBLIOGRAFÍA

1. **Jacobson, Ivar, Christerson, Magnus, Jonsson, Patrik y Overgaard Gunnar.** *Object-Oriented Software Engineering: A Use-Case-Driven Approach*. Ed. Wesley, 1992.

2. **Rumbaugh, James, Blaha, M, Premerlani, W, Eddy, F, Lorensen, W.** *Object-Oriented Modeling and Desing*. 1991.

3. **Cuéllar Vazquez, Edith, Rodríguez Gómez, Gustavo y Muñoz Arteaga, Jaime.** *Aplicación de Patrones de Software en el Dominio de los Simuladores de Procesos Dinámicos.* [En línea]. http://ccc.inaoep.mx/~grodrig/Descargas/PatSSD.pdf. 1.

4. **S.M, Alessi y S.R, Trollip.** *Computer Sased Instruction, Method and Development.* New Jersey : Ed. Prentice Hall Inc., 1985. 2.

5. **Sanchez, M.** *Metodologías de Desarrollo de Software.* 2004.

6. **Molpeceres, A.** Procesos de desarrollo: RUP, XP y FDD. *Procesos de desarrollo: RUP, XP y FDD.* [En línea] 2002. http://www.javahispano.org/articles.article.action?id=76.

7. **Jacobson, Ivar, Booch, Grady y Rumbaugh, Jame.** *El Proceso Unificado de Desarrollo de Software.* Ciudad de la Habana : Félix Varela, 2004.

8. **Rumbaugh, Jame, Jacobson, Ivar y Booch, Grady.** *El Lenguaje Unificado de Modelado. Manual de Referencia.* s.l. : Addison Wesley, 2000.

9. **Périssé, Marcelo Claudio.** *Proyecto Informático. Una Metodología Simplificada.* 2001.

10. **García Montero, María de los Angeles.** *Ingeniería de Requerimientos aplicada al proceso de Inicio de Investigación y Registro de Notificaciones en el CICPC.* Ciudad de la Habana : s.n., 2008.

11. **IBM**. [En línea] http://www.ibm.com.

12. **Pressman, Roger S.** *Ingeniería del Software "Un enfoque prático".* Madrid : McGraw-Hill, 2002.

13. **Larman, C.** *UML y Patrones. Introducción al análisis y diseño orientado a objeto.* Mexico : s.n., 1999.

14. **Carvajal Pérez, Erlys. 2009.** *Análisis y diseño del subsistema de Análisis de Resultados de un Simulador de Procesos Químicos.* Ciudad de La Habana : s.n., 2009.

15. **Visconti, M., Astudillo, H.** *Fundamentos de Ingeniería de Software,* 2003.

16. **Jacobson, Ivar, Griss, Martin y Jonsson, Patrik.** *Software Reuse: Architecture, Process and Organization for Business Success*. Ed. Wesley, 1997.

17. **Jacobson, Ivar, Bylund, Stefan, Jonsson, Patrik, Ehnebom, Staffan.** *Usign contractsand use case to build plugable architectures, Journal of Object-Oriented Programming.* 1995.

18. **Génova, Gonzalo. y Fuentes, José M.** *Evaluación comparativa de herramietnas CASE para UML desde el punto de vista notacional.* Revista de la Asociación de Técnicos de Informática, Nº 181. 2006. ISNN: 0211-2124.

19. **Software, Departamento Central de Ingeniería de. 2009.** *Introducción a la Ingeniería de Software.* Universidad de las Ciencias Informáticas, 2009.

20. **Software, Departamento Central de Ingeniería de. 2009.** *Entorno Virtual de Aprendizaje.* Noviembre de 2009.

21. **Software, Departamento central de Ingeniería de. 2009.** *Fase de Inicio. Disciplina de Requisitos.* Universidad de las Ciencias Informáticas, 2009.

22. **Périssé, Marecelo Claudio. 2001.** *Proyecto Informático. Una Metodología Simplificada.* 2001.

23. **Corporation, Rational Software. 2002.** *Product: Rational Software Corporation.* 2002. *Rational Unified Process. Best Practices for Software Development Teams.* 1998.

24. **Poley, Rob y Stevens, Pedita.** *Using UML: Software Engineering whit Objects and Components.*

ANEXOS

Anexo 1: Diagramas de clases del análisis

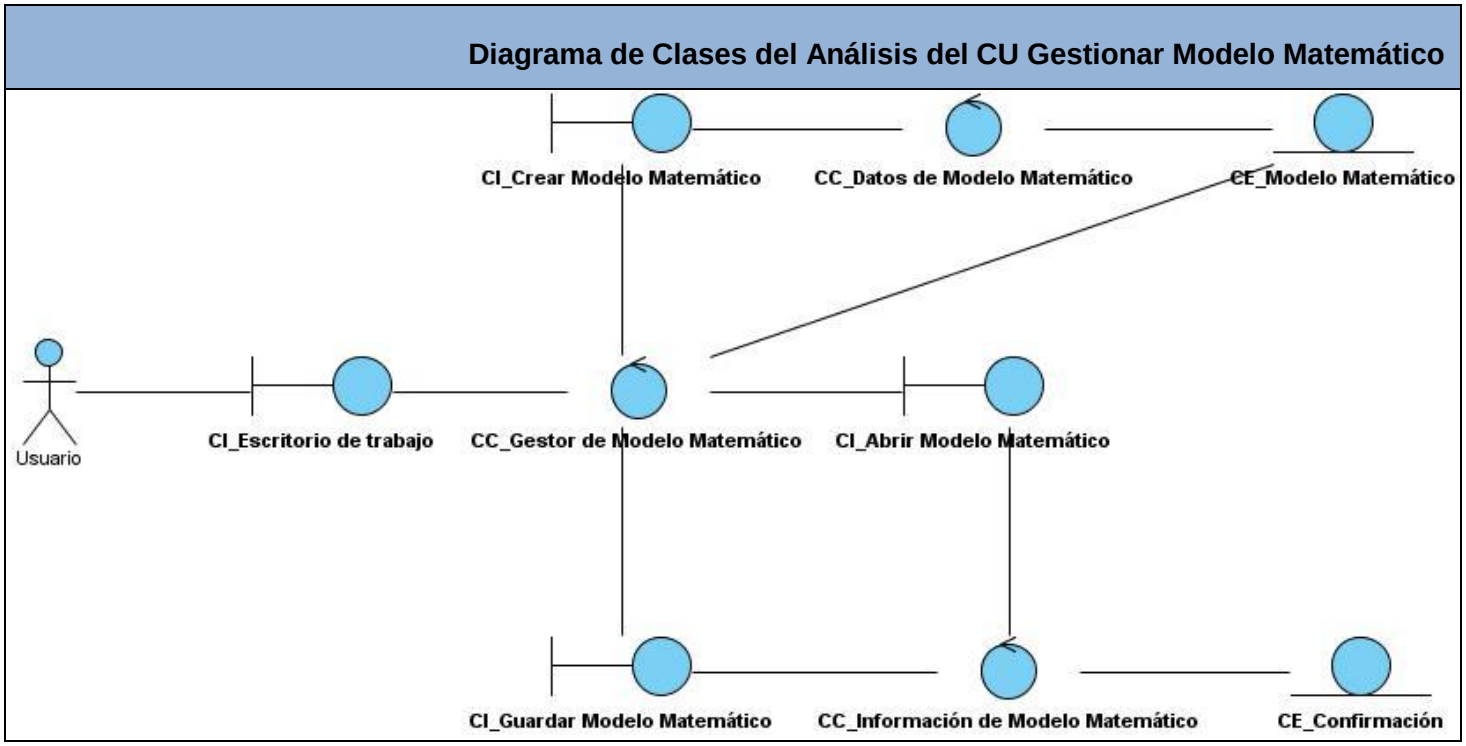


Figura 35: Diagrama de Clases del Análisis del CU Gestionar Modelo Matemático

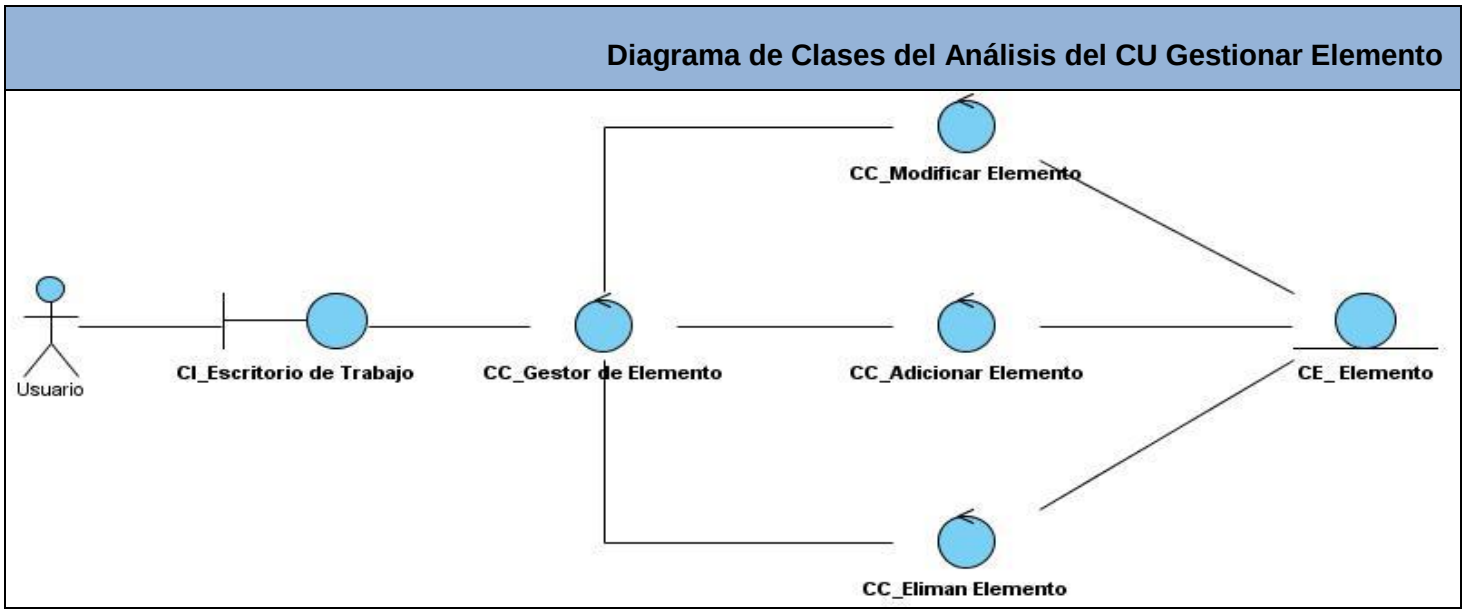


Figura 36: Diagrama de Clases del Análisis del CU Gestionar Elemento

Diagrama de Clases del Análisis del CU Adicionar Elemento de Termodinámica

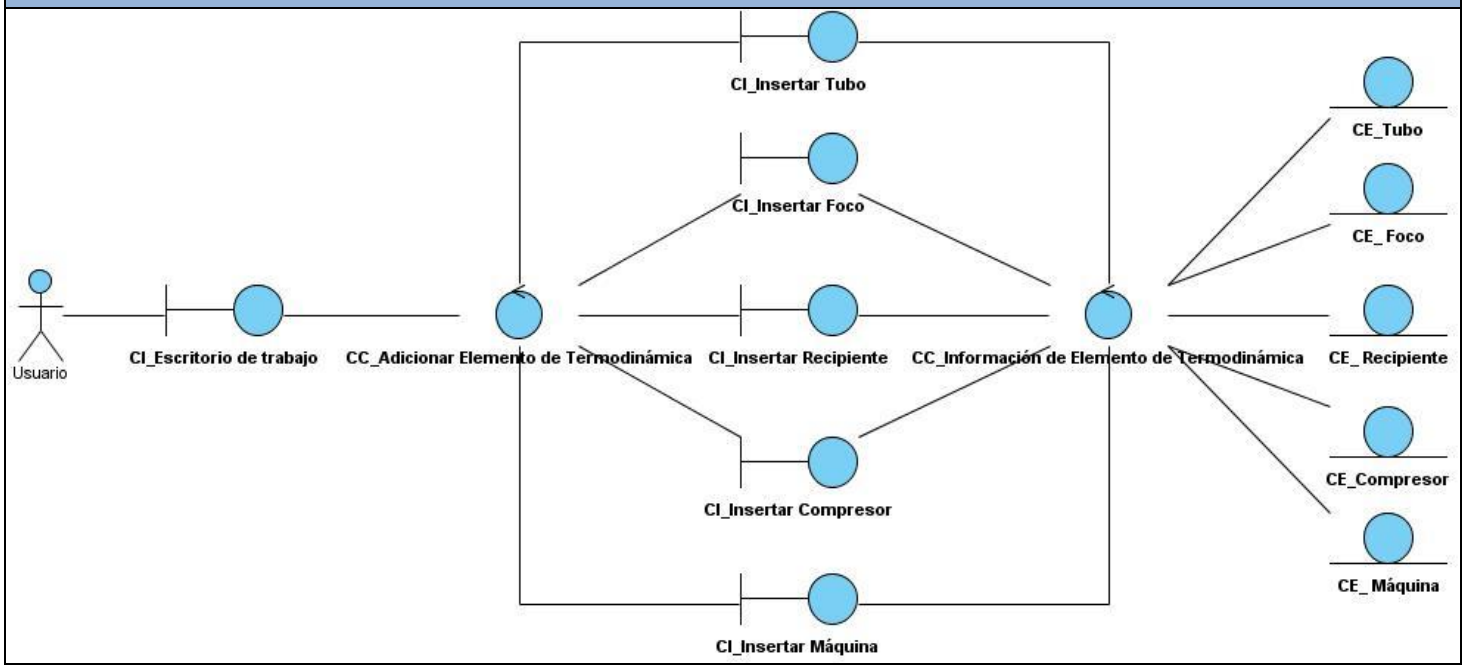


Figura 37: Diagrama de Clases del Análisis del CU Adicionar Elemento de Termodinámica

Diagrama de Clases del Análisis del CU Adicionar Elemento de Mecánica

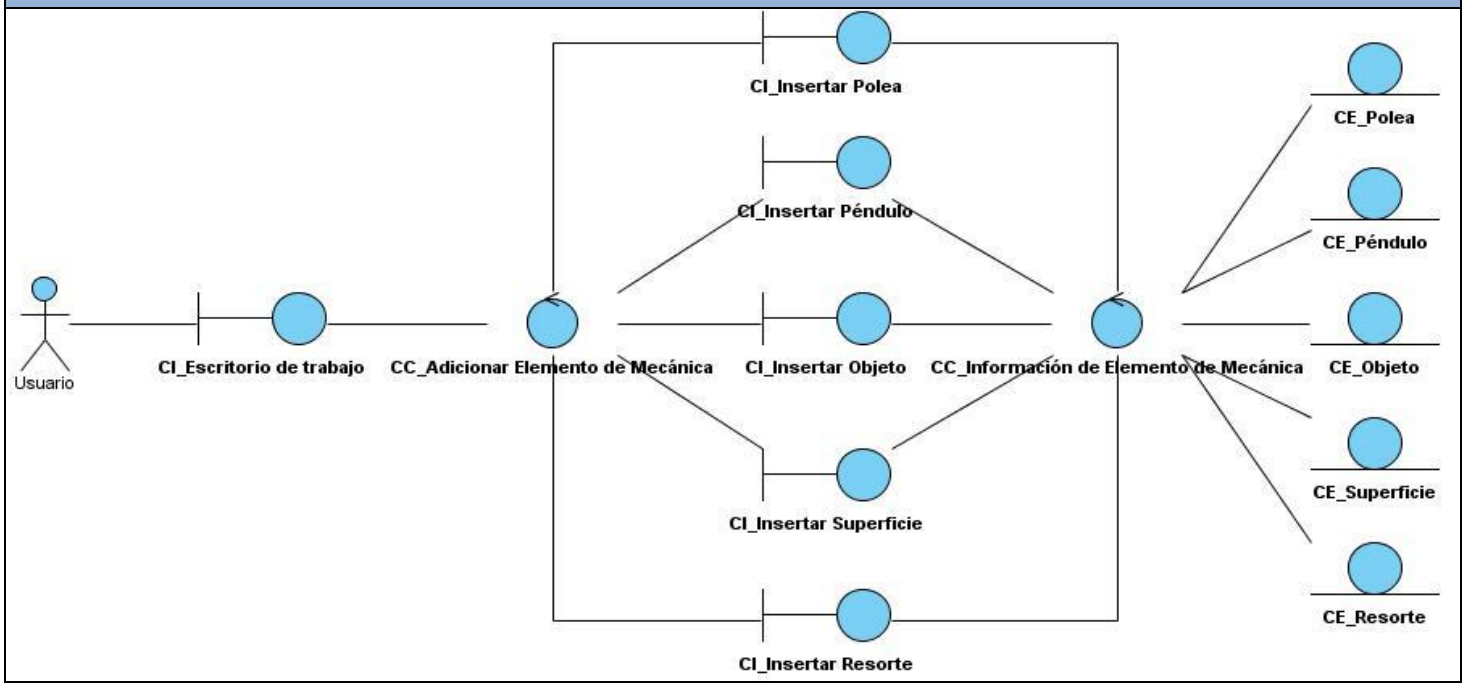


Figura 38: Diagrama de Clases del Análisis del CU Adicionar Elemento de Mecánica

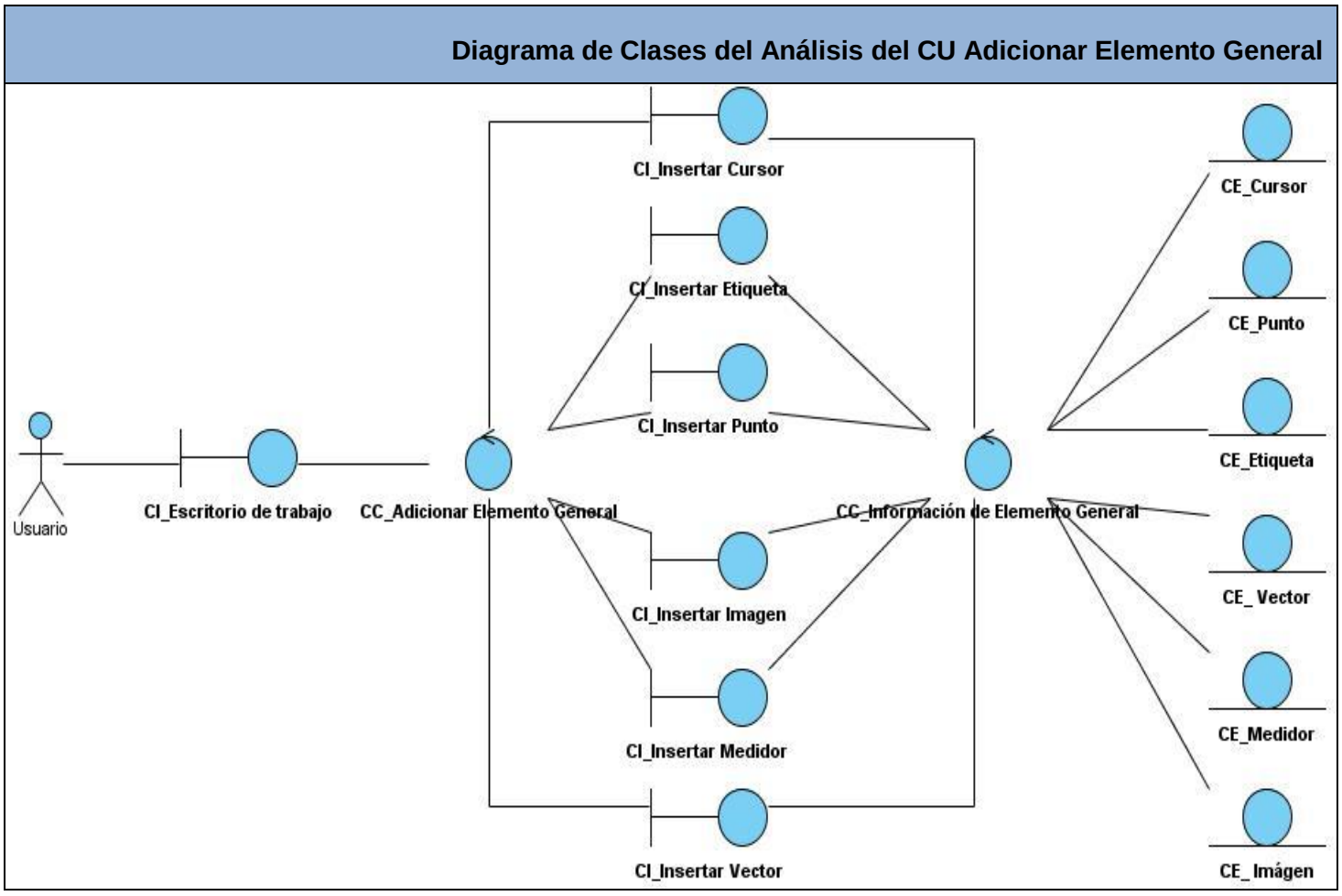


Figura 39: Diagrama de Clases del Análisis del CU Adicionar Elemento General

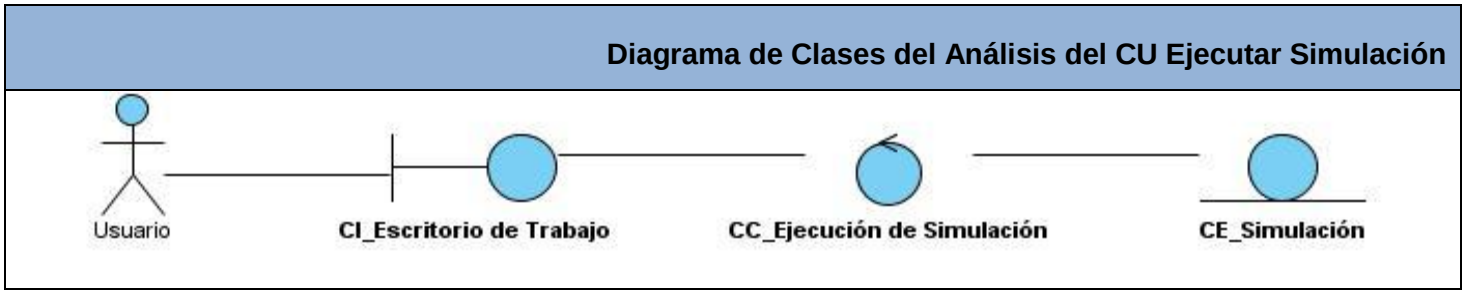


Figura 40: Diagrama de Clases del Análisis del CU Ejecutar Simulación

Diagrama de Clases del Análisis del CU Adicionar Elemento de Óptica

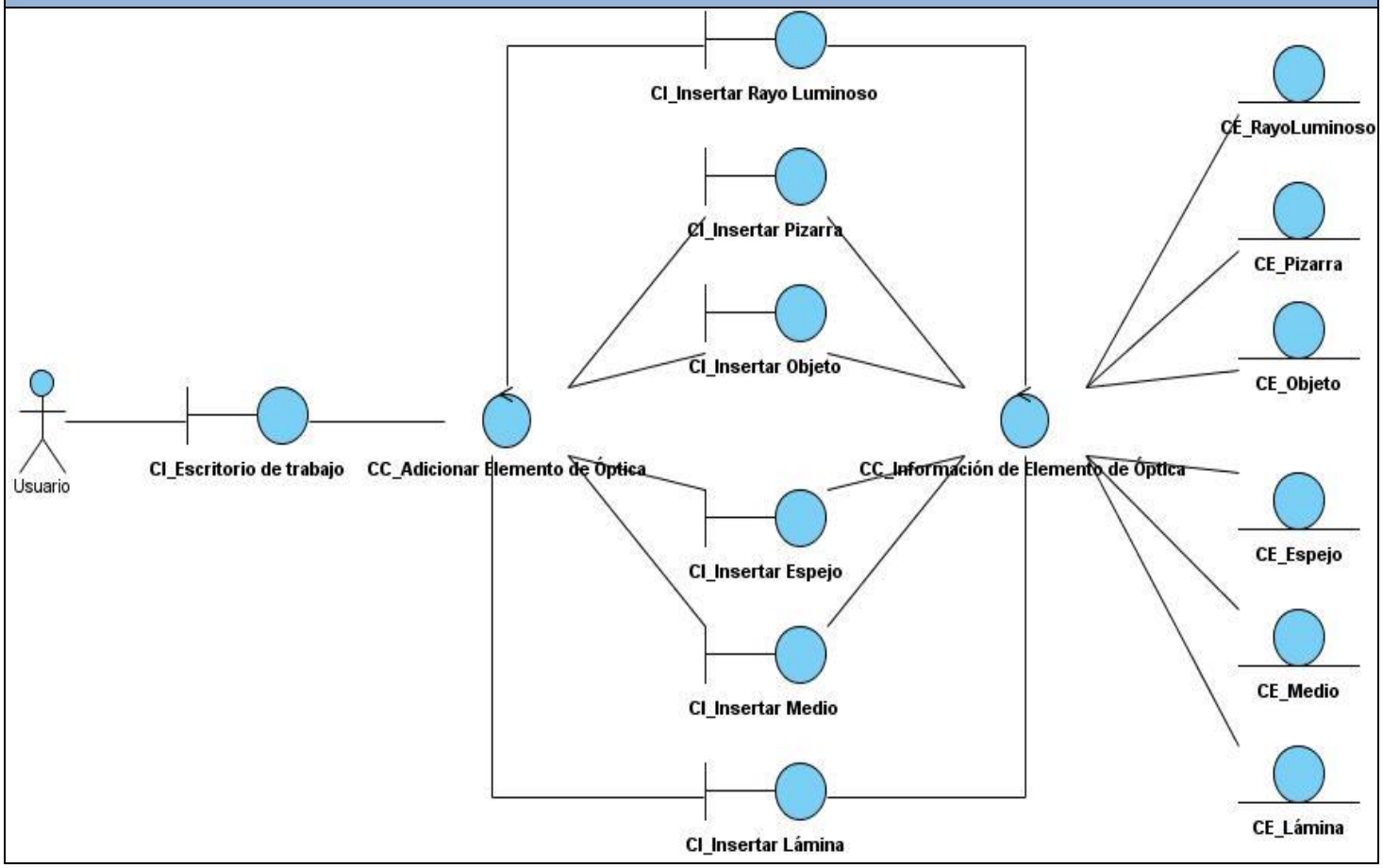


Figura 41: Diagrama de Clases del Análisis del CU Adicionar Elemento de Óptica

Diagrama de Clases del Análisis del CU Generar Gráfica

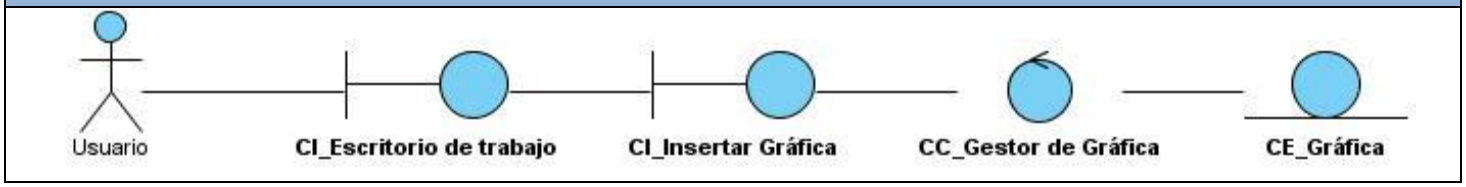


Figura 42: Diagrama de Clases del Análisis del CU Generar Gráfica