Universidad de las Ciencias Informáticas

Facultad 2

Trabajo de Diploma para optar por el título de

Ingeniería en Ciencias Informáticas.

Título: ZEOSIM: Software para la simulación de los procesos de síntesis de materiales porosos tipo zeolitas por el método de Montecarlo Cinético

Autor: Yaqueline Mejías De la Torre.

Tutor: Lic. Yurisel Machado Batista.

La Habana, junio 2013

"Se pueden adquirir conocimientos y conciencia a lo largo de toda la vida, pero jamás en ninguna otra época de su existencia una persona volverá a tener la pureza y el desinterés con que, siendo joven, se enfrenta a la vida."

Declaración de Autoría

Declaro que soy el único autor de este trabajo y autorizo a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

Yaqueline Mejías De la Torre Yurisel Machado Batista

_____ _____

Datos de Contacto

Nombre del Tutor: Lic. Yurisel Machado Batista, Licenciado en Física.

Centro: Universidad de las Ciencias Informáticas.

E-mail: machadob@uci.cu

Nombre de Autor: Yaqueline Mejías De la Torre.

Centro: Universidad de las Ciencias Informáticas.

E-mail: ymdelatorre@estudiantes.uci.cu

Dedicatoria

A mis padres, que me han enseñado a ver la realidad con mis propios ojos, que sienten mis logros como si fueran suyos, que ven en mis sueños los suyos; a ustedes que admiro, respeto y amo. A mi mamá por todo su apoyo y comprensión, por estar siempre cuando la he necesitado y me ha ayudado a levantar en mis caídas y tropiezos; gracias por darme tu ejemplo y todo tu amor. Gracias a los dos por demostrarme que cambiarían todo por ofrecerme un instante de felicidad.

A Yanelis, por convertirte en una persona muy especial para mí, por guiarme, por apoyarme en los buenos y malos momentos. Gracias por tu cariño, y por estar siempre en mí.

A mi sobrina, que ha crecido lejos de mí, y que algún día podrá leer con madurez este trabajo; deseo igual nos haga testigos del suyo propio.

Agradecimientos

Quiero agradecer ante todo a nuestro Comandante Fidel Castro, porque gracias a sus ideas estamos aquí, y seguimos su ejemplo. Agradezco a la universidad el permitirme realizar mis estudios universitarios durante estos 5 años llenos de experiencias;a todas las personas que de una forma u otra han contribuido a mi formación profesional.

A mi tutor por todo su apoyo y paciencia durante la etapa de realización de este trabajo, y por ayudarme en todo momento,

A mis papitos, el más bello regalo que la vida me ha obsequiado, gracias por estar siempre pendientes de mí, por respetar mis decisiones aunque no les gusten. Gracias por ser un ejemplo a seguir, por estar orgullosos de mí, por darme la fuerza necesaria para enfrentar los momentos más difíciles. Gracias por sacrificar sus vidas y sus sueños porque yo cumpla los míos, por entregarme todo, y demostrarme que la felicidad de un hijo es, por siempre, la felicidad de una madre y un padre; los quiero infinitamente.

A mi hermano, por ser mi ejemplo de hombre trabajador, por su dedicación, por ser incondicional conmigo; gracias por ser mi hermano y mi amigo.

A mi poti, gracias por toda la espera, por darme las mayores alegrías desde que te conocí, por hacernos feliz y ayudarme a simplificar los obstáculos que muchas veces se nos imponen; gracias por creer en mí.

A mi familia que siempre me ha apoyado y respetado. A mis suegros y mi cuñada, por confiar en mí, y ver de cerca esta historia tan divina.

A Dianet, aunque no pueda estar presente mientras ahora leo, por ser mi mejor amiga desde hace 18 años y no defraudarme nunca, por no abandonar nuestra amistad; por saber

escucharme y apoyarme siempre; podré tener muchas amistades en mi vida, mas ninguna tan verdadera como la nuestra.

A mis hermanos de la casa, Sady, Pradito, Maylin, Daniela y Sulema, que han convivido conmigo todo este tiempo y unos a otros nos hemos ayudado, hemos aprendido a vivir independientes, a compartir, y construir momentos que no se olvidan nunca. A mis amigas Marvis, Anis, Irina, Ana y Yei por su amistad incondicional, por ser parte de mí, y darme la mano cuando les necesito, a todas las quiero mucho.

.

Resumen

El desarrollo tecnológico a nivel mundial ha posibilitado que se simulen diferentes procesos en sólidos poli cristalinos. La simulación computacional es una presentación informática que simula modelos abstractos de un determinado suceso real, y en estos procesos reduce el tiempo de espera para lograr un resultado considerable y aumentar la confiabilidad de los mismos.

La Zeolita es un mineral que pertenece al grupo de los aluminosilicatos, básicamente hidratados del sodio, del potasio, del calcio, en los cuales el agua se sostiene en las cavidades de los enrejados. Las sustancias que la componen así como sus propiedades hacen de éstas un material importante en la industria química por sus propiedades absorbentes catalíticas y de intercambio iónico.

El desarrollo de un algoritmo de simulación en una aplicación informática permitirá avanzar en el estudio de la Zeolita, en su aplicación en la industria farmacéutica en nuestro país por la facilidad de obtención de resultado en menor tiempo y ahorrando recursos. Por tanto, a través del presente trabajo, utilizando el lenguaje de Programación Java, con el apoyo de la metodología ágil XP, NetBeans como IDE de desarrollo, y el uso de herramientas informáticas para modelar el diseño, se desarrolla una aplicación informática que simula y permite diseñar gráficamente el comportamiento de los materiales porosos de tipo Zeolita empleando el algoritmo matemático de Montecarlo Cinético.

Palabras Claves:

Simulación computacional, síntesis, Zeolita, Montecarlo Cinético.

Índice

Índice de Tablas y Figuras

Introducción

Una simulación computacional es una presentación informática cuyo objetivo es introducir una simulación de un modelo abstracto de un sistema determinado. Dichas simulaciones se han convertido en una parte apreciable y útil de los modelos matemáticos de muchas técnicas naturales de ciencias, tales como la física, la química y la biología; así como de sistemas humanos de economía, psicología y ciencias sociales. Estas simulaciones abarcan desde programas informáticos cuya ejecución dura unos minutos, hasta conjuntos de ordenadores conectados en red cuya ejecución dura horas, e incluso pueden extenderse varios días.

En el área de la física, la simulación computacional se introdujo como una herramienta para tratar sistemas de muchos cuerpos a comienzo de la década de los ’50, con el trabajo pionero de Metrópolis[1]. Actualmente, gracias al vertiginoso desarrollo de la tecnología de las computadoras, la simulación computacional se ha convertido en una herramienta de cálculo esencial, tanto para los investigadores experimentalistas como para teóricos. Las simulaciones requieren un menor coste económico, permiten alcanzar condiciones complicadas en el laboratorio (por ejemplo trabajar a muy altas presiones y las bajas temperaturas), ejercen un perfecto control de todos los parámetros que intervienen en el proceso y además proporcionan información de todos y cada uno de los compuestos que forman parte del sistema. Mediante un buen modelo computacional no sólo se pueden reproducir experimentos de laboratorio, sino que además, gracias a que se pueden variar libremente los parámetros usados, permite probar modelos teóricos existentes en rangos de parámetros imposibles de alcanzar experimentalmente por ahora, resolviendo así conflictos entre explicación teórica y observación. Un papel fundamental también lo juega hoy día la visualización de los resultados obtenidos, no sólo se obtienen datos numéricos que pueden ser contrastados con los experimentos, sino también se puede obtener una imagen gráfica del proceso en cuestión.

Los materiales Zeolíticos o Zeolitas son aluminosilicatos que presentan una estructura porosa a nivel molecular bien organizada[2]. La síntesis de estos materiales es una temática muy atractiva y abordada en investigaciones científicas recientes[3, 4], debido a sus muchas aplicaciones[5]. Los procesos que ocurren en la formación de estos materiales son complejos y difíciles de monitorear con métodos experimentales. Es aquí donde los métodos de simulación computacional entran a jugar un papel importante, convirtiéndose en una herramienta que permite conocer los detalles de estos procesos y determinar las propiedades físico-químicas de estos materiales.

En Cuba durante los últimos años ha cobrado fuerza la creación de sistemas informáticos para dar solución a disímiles problemas de diversos sectores de la sociedad. En el Laboratorio de Ingeniería de Zeolitas del Instituto de Ciencias y Tecnologías de Materiales de la Universidad de la Habana trabajan en conjunto dos grupos de investigadores que se dedican, uno a la síntesis y caracterización por métodos experimentales de materiales porosos tipo Zeolitas, y el otro grupo al diseño y modelación de estos materiales. Este último grupo hace uso de técnicas de simulación computacional para simular entes y procesos que tienen lugar en la obtención de Zeolitas, y cuyos resultados son de gran utilidad porque permiten reducir gastos de tiempo y recursos en el desarrollo experimental. Varios son los software que les permiten realizar estas simulaciones[6], pero la mayoría son muy generales (casi siempre en una simulación se tiene que usar más de uno) y además el usuario necesita tener amplios conocimientos teóricos de las técnicas de simulación computacional. Por otro lado, el grupo de investigadores que se dedica a la rama experimental carece de conocimientos teóricos sobre estas técnicas, por lo que se les hace complejo utilizar los software para realizar sus propios experimentos. La capacitación a estos investigadores sobre el tema no sería factible, debido al tiempo que se necesita para llevar a cabo este proceso. Además, lo más adecuado sería que estos investigadores simularan sus propios experimentos con el uso de un solo software sin necesidad de convertirse en especialistas teóricos.

Luego de realizarse el análisis anterior de las dificultades existentes surge la siguiente **situación problemática**: para que los investigadores puedan llevar a cabo sus estudios teóricos en el proceso de simulación de materiales zeolitas necesitan emplear más de un software que realicen simulación computacional, mientras que los que se dedican a la rama experimental se les dificulta la utilización de estos software para simular sus experimentos, debido a que carecen de conocimientos teóricos sobre técnicas de simulación computacional.

Se presenta el siguiente **problema a resolver**: ¿cómo lograr que los investigadores que se dedican al estudio de la simulación de los materiales tipo zeolitas puedan simular el proceso de síntesis con el uso de un solo software y sin necesidad de tener amplios conocimientos sobre técnicas de simulación computacional?

En comunicación con lo formulado anteriormente, el **objeto de estudio** estará centrado en la simulación computacional del proceso de síntesis de materiales porosos tipo zeolitas para su representación.

El **campo de acción** lo constituyen los métodos de simulación computacional de entes y procesos, aplicados a la obtención de materiales porosos tipo zeolitas.

Para lograr este propósito se plantea como **objetivo general**: desarrollar una herramienta que permita la simulación computacional de los procesos de síntesis de materiales porosos tipo zeolitas a través del método de Montecarlo Cinético, que pueda ser usada por los investigadores que no son especialistas en la simulación computacional, permitiéndoles, de forma dinámica y visual, la selección de todos los elementos necesarios para establecer la simulación e introducir los datos que necesiten sin implementar funcionalidades en el software ni consultar otras herramientas.

Guiados por lo expuesto anteriormente y para apoyar la investigación se plantea la siguiente **idea a defender**: Si los investigadores que se dedican a la rama experimental contaran con una herramienta que les permitiera realizar experimentos sin que necesiten tener amplios conocimientos sobre técnicas de simulación, implementar códigos en el software para su funcionamiento, y que pudieran realizar sus operaciones sin necesidad de consultar otro software; entonces no presentarían dificultades para simular procesos de síntesis de materiales porosos tipo zeolitas.

Para resolver la situación problemática, se llevarán a cabo las siguientes **tareas de investigación**:

- Análisis de conceptos relacionados al marco teórico y las técnicas de simulación existentes.
- Caracterización de los sistemas informáticos de simulación de procesos de síntesis aplicados a la zeolita en el plano nacional e internacional.
- Selección de la metodología y herramientas de desarrollo que proporcionen la creación y garanticen la calidad de la herramienta a obtener.
- Diseño de la aplicación.
- Implementación de la solución propuesta.
- Ejecución de pruebas para comprobar el correcto funcionamiento del software.
- Validación del software para corroborar su utilidad en el campo de la simulación de materiales porosos tipo zeolita.

En el desarrollo de este trabajo se emplearán los métodos investigativos que a continuación se presentan:

Métodos Teóricos:

- Analítico–sintético: este método es empleado para el análisis de la situación problemática y determinar posibles variantes de solución.
- Modelación: se crean modelos para diseñar las actividades que se implementan durante el desarrollo de la aplicación.

Métodos Empíricos:

- Consulta de las fuentes de información: estudiar y documentarse sobre las técnicas de simulación computacional, los materiales zeolitas, y las herramientas adecuadas para realizar este trabajo.
- Pruebas: se realizarán pruebas para detectar posibles errores y evaluar la calidad de los requerimientos pedidos por el cliente.

La presente tesis se estructura de cuatro capítulos, los cuales brindan una información detallada del desarrollo de la aplicación para lograr, en su conjunto, su entendimiento y aprendizaje:

Capítulo 1. Fundamentación Teórica: aborda la fundamentación teórica vinculada al objeto de estudio y las herramientas y metodología a utilizar.

Capítulo 2. Características del Sistema: se siguen las pautas de la metodología escogida: identificación de los requisitos, descripción de la arquitectura base y modelación de los diagramas requeridos para la descripción de las funcionalidades.

Capítulo 3. Diseño del Sistema: define el diseño de la solución apoyándose en la metodología establecida, describiendo los patrones de diseño aplicados al software.

Capítulo 4. Implementación y Prueba del Sistema: detalla la implementación de cada requisito funcional y describe los casos de pruebas elaborados.

Capítulo 1: Fundamentación Teórica

1.1 Introducción

En el presente capítulo se brinda una panorámica general acerca del estudio de las Zeolitas y sus aplicaciones. Además, se caracterizan diferentes software del método de simulación computacional empleado para simular el proceso de síntesis de zeolitas con el software a desarrollar. Se describen además el lenguaje de programación a utilizar durante el desarrollo del capítulo, se realiza la descripción de la metodología de análisis y diseño escogida, así como las herramientas a emplear en la configuración y confección de un sistema que resuelva la situación problemática planteada.

1.2 Zeolitas: características, aplicaciones y proceso de síntesis

Las zeolitas forman un grupo de más de 80 variedades minerales que los mineralogistas conocen desde hace unos 200 años. La primera zeolita fue descubierta en 1756 por A.F. Cronstedt, quien la llamó "piedra que hierve" en base a la propiedad que tienen de perder el agua al ser calentadas. Hacia 1920 se conocía que estos minerales tenían propiedades de absorción selectiva de sustancias, por lo que se denominaron "tamices moleculares". Al culminar la década del 40 ya existía una gran demanda comercial, sin embargo la escasez de materias primas hizo que comenzara el desarrollo de zeolitas sintéticas[1].Hoy en día, coexisten en el mercado las zeolitas sintéticas y las naturales. Las zeolitas naturales son recursos geológicos relativamente abundantes en áreas volcánicas e intrusivas [7].

Teniendo en cuenta las características y propiedades de las Zeolitas, muchas son las aplicaciones que encuentra este material. El estudio de las Zeolitas constituye una de las ramas de mayor interés en la comunidad científica, motivado fundamentalmente por la introducción de estos materiales en la industria[8]. Las excelentes propiedades adsortivas[2] de la zeolita vistas en aplicaciones en el secado, purificación y separación; de intercambio iónico[3] y catálisis[4], son un

[1]Las Zeolitas sintéticas consiguen una mayor pureza que las Zeolitas naturales, lo que las hace aptas para múltiples usos entre los que destaca la detergencia.
[2]Se aplica a la sustancia que adsorbe. Atraer y retener en la superficie de un cuerpo, moléculas o iones de otro cuerpo.
[3]Cationes hidratados dentro de los poros de la zeolita están unidos débilmente y preparados para intercambiarse con otros cationes cuando se encuentran en un medio acuoso.

punto de partida importante para su uso en aplicaciones farmacéuticas, logrando aumentar significativamente la capacidad de adsorción de fármacos como ejemplo de aplicaciones en nuestro país[9].

Los materiales zeolitas, desde el punto de vista del control ambiental, presentan diversas aplicaciones por su alto potencial en la descontaminación. Por ejemplo, se han utilizado para retener metales en aguas ácidas, adsorber e inmovilizar metales pesados de aguas y suelos contaminados y separar compuestos orgánicos volátiles. En la industria se utilizan como rellenos, fertilizantes, suplementos dietéticos en nutrición animal, cementos, adsorbentes y catalizadores.

- Aplicaciones en el ámbito internacional

En México, la empresa Zeolitas e Insumos Nacionales S.A utiliza la zeolita en plantas de tratamiento, dichas zeolitas están tratadas con un intercambio iónico de acuerdo al tipo de agua y al grado de contaminantes que contenga el agua residual[10].

También se creó unDepartamento de Zeolitas en 1982 con el proyecto "Síntesis y aplicaciones de Zeolitas para la Industria Petroquímica", en su laboratorio, actualmente, se manejan los procesos hidrotérmicos en la obtención de zeolitas del tipo A, X, Y, Sodalita, y Filipsita[11].

En Colombia se hace uso de las zeolitas naturales en el tratamiento anaerobio de aguas residuales industriales para la remoción integrada de materias orgánicas, compuestas de azufre y nutrientes[12].

- Aplicaciones en el ámbito nacional

Las investigaciones desarrolladas han demostrado que las zeolitas naturales poseen propiedades biológicas y una alta estabilidad química y física en su interacción con el medio biológico; también se ha comprobado una alta especificidad en algunos materiales derivados de la Clinoptilolita[5], lo que implica que sean consideradas como materiales promisorios para ser utilizados en la práctica médica[13, 14].

[4]Las zeolitas son útiles como catalizadores para reacciones importantes con moléculas. Las más importantes son craqueo, isomerización y síntesis de hidrocarbonos, promueven reacciones catalíticas como ácido-base y reacciones de metal inducido.
[5]Zeolita natural.

Los trabajos teóricos-computacionales realizados en Cuba abarcan fundamentalmente estudios de propiedades estructurales de zeolitas. En trabajos recientes se estudia la hidratación de zeolitas como un primer paso para abordar el intercambio iónico. Otros estudios presentados demuestran las posibilidades de la simulación computacional en zeolitas, cuyo resultado recoja además estudios encaminados a determinar la conformación estructural de moléculas con interés farmacéutico. Sin embargo, no se han obtenido antecedentes de software que desarrollen investigaciones realizadas sobre los métodos de Montecarlo Cinético en el uso del material zeolita.

La introducción paulatina de zeolitas naturales cubanas en la agricultura ha mejorado la producción de tomates, con mezclas de urea, estos han duplicado los rendimientos permitiendo el aumento del número de frutos por planta y su peso. La Empresa de Cítricos de Jagüey Grande, en Matanzas, emplea la zeolita como sustrato para la producción intensiva, lo que reduce el uso de abonos[15].

- Síntesis de Zeolitas

Para cada aplicación de la zeolita, estas necesitan características y propiedades específicas, es por ello que los científicos se dedican a su creación o reproducción a través del mecanismo de síntesis. En la síntesis intervienen variables que determinan las propiedades físico-químicas finales del material, la relación entre estas variables y/o las condiciones iniciales del sistema a sintetizar con el resultado final es lo que se conoce como rutas de síntesis.

El desarrollo de un experimento para la obtención de materiales noveles requiere de grandes inversiones, razón por la que es indispensable realizar estudios preliminares para asegurar su conveniencia, de acuerdo a su eficiencia y ejecución económica para proyectos de cualquier tamaño. Una herramienta para ejecutar estudios de este tipo lo constituye la simulación computacional, y se ha convertido en un instrumento muy importante que juega un papel fundamental en la obtención de características y propiedades de materiales y la definición de experimentos necesarios con excelentes aplicaciones en la industria.

1.3 Simulación Computacional.

Thomas T. Goldsmith Jr. y EstleRay Mann expone lo siguiente: *"Simulación es una técnica numérica para conducir experimentos en una computadora digital. Estos experimentos*

comprenden ciertos tipos de relaciones matemáticas y lógicas, las cuales son necesarias para describir el comportamiento y la estructura de sistemas complejos del mundo real a través de largos períodos".

Otra definición formulada por R.E. Shannon[16]es: *"La simulación es el proceso de diseñar un modelo de un sistema real y llevar a término experiencias con él, con la finalidad de comprender el comportamiento del sistema o evaluar nuevas estrategias -dentro de los límites impuestos por un cierto criterio o un conjunto de ellos - para el funcionamiento del sistema".*

La presente investigación se guía por el concepto establecido por Thomas T. Goldsmith Jr. y Estle Ray Mann, pues expresa de forma sintetizada la simulación a diseñar y presenta mayor afinidad con el trabajo.

Entre los métodos de simulación clásicos, los más usados por la comunidad científica son la Dinámica Molecular (DM) y el método de Montecarlo (MC)[17-21]. La DM es un método determinista, que permite estudiar el comportamiento temporal de un sistema dinámico. Esto hace que sea más preciso que el MC para la simulación de un sistema físico, sin embargo el intervalo de tiempo que puede simular esta técnica es relativamente corto (del orden de los nanosegundos) y la cantidad de partículas es del orden de los 100 mil átomos. En la Figura 1 se muestran algunas características de estos métodos de simulación, demostrando la capacidad del sistema para cada método y el tiempo en el que es capaz de realizar el experimento. En dicha figura se añadió el método cuántico de ab initio para una mejor comparación, dicho método calcula en picosegundos un orden de 5 a 100 átomos.

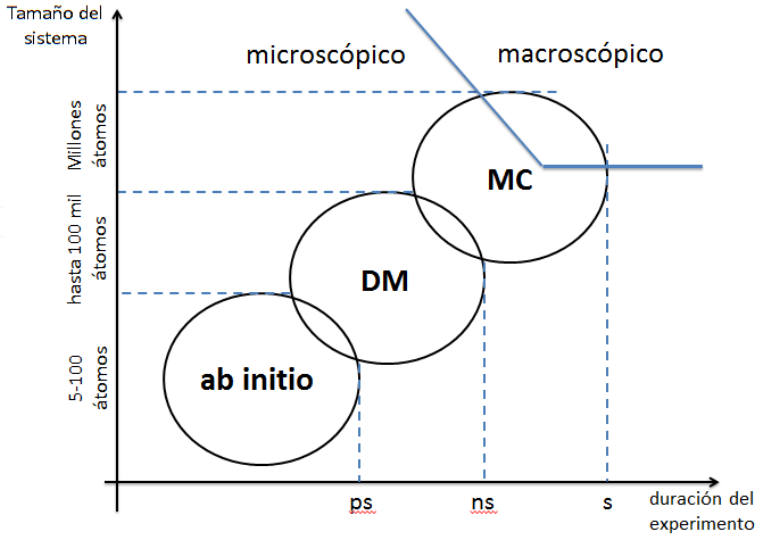

Figura 1. Características de los principales métodos de simulación computacional (ps: picosegundos; ns: nanosegundos; s: segundos)

La síntesis de materiales zeolíticos implica además de un número grande de átomos, una duración temporal que puede ser de minutos, horas o días. Por tal razón el MC se convierte en la herramienta adecuada para simular estos procesos de síntesis.

- Método de Montecarlo

El MC es un método de simulación probabilístico o estocástico, pues usa números aleatorios para generar las posibles configuraciones del sistema atómico o molecular[22]. En una simulación de MC, cada configuración (conjunto de posiciones espaciales de los átomos o partículas que forman el sistema) depende solamente de su predecesor y no de cualquier otra configuración previamente visitada.

El método de Montecarlo genera configuraciones al azar y emplea un conjunto especial de criterios para decidir si la nueva configuración debe ser aceptada o no. El criterio está basado en la suposición de que las configuraciones están distribuidas de acuerdo a la energía del sistema. El algoritmo básico de este método se conoce como algoritmo de metrópolis[23],el cual se describe de la siguiente forma:

1- Se selecciona una partícula n_i cualquiera y se le produce un desplazamiento aleatorio Δr(r'=r+Δr donde r' es la nueva posición de la partícula).
2- Se calcula la nueva energía del sistema U'(r) (se halla $\Delta U=U'(r)-U(r)$)

3- Si ΔU<0 se acepta la movida.

4- Si ΔU>0 se genera un número aleatorio *α* entre [0,1].

- Si $\alpha < exp(-\Delta U / K_B T)$ se acepta la movida y se regresa al paso 1.
- Si $\alpha < exp(-\Delta U / K_B T)$ se rechaza la movida y se regresa al paso 1.

(K_B es la constante de Boltzman[6] y T es la temperatura)

5- Luego de suficientes movidas se termina la simulación. STOP.

➢ Montecarlo Cinético

Una de las metas de este trabajo, es simular la evolución temporal de la formación de materiales porosos tipo Zeolitas, pero el algoritmo de Metrópolis no permite estudiar este comportamiento. Sin embargo han sido desarrolladas técnicas de MC tales como el Montecarlo Cinético (KMC) con el cual es posible determinar estocásticamente la evolución de un sistema dependiente del tiempo.

El algoritmo para implementar el KMC en la aplicación a realizares el siguiente:

1- Inicio (t=0)

2- Determinar todas las transiciones de estados posibles y calcular la función cumulativa.

Las transiciones de estados posibles que se identificaron teniendo en cuenta que el sistema es molecular son:

- Trasladar una molécula.
- Rotar una molécula.
- Enlazar dos moléculas.

Cada una de estas transiciones tiene asignada una probabilidad de ocurrencia, la cual está dada por:

$p_i = \rho\ exp(-\Delta U / K_B T)$

Donde ρ se conoce como frecuencia de ocurrencia y se determina por la teoría de Transición de Estados[24, 25]:

$\rho = (K_B T/\hbar) * exp(-\Delta F/K_B T)$

Donde h es la constante de Planck[7] y ΔF es la diferencia entre la máxima y la mínima energía libre de la superficie de energía potencial. De esta forma la función cumulativa se calcula como:

[6] La constante de Boltzmann (k o k_B) es la constante física que relaciona temperatura absoluta y energía. Se llama así por el físico austriaco Ludwig Boltzmann.

$R = \Sigma\, p_i$

Donde *i* corre sobre todos los eventos o configuraciones posibles a ocurrir.

3- Generar un número aleatorio *u* (0<u<1) y encontrar el evento *i* según el criterio:

$\Sigma\, p_{i-1} < u\mathrm{R} \leq \Sigma\, p_i$

4- Producir el evento i.

5- Generar un número aleatorio v (0<v<1) y adelantar el tiempo: t=t+Δt donde Δt = - ($\ell og\ v$/R)

6- Regresar al paso 2. Luego de suficientes movidas, detener la simulación. STOP.

- Potenciales de interacción y ensembles de la simulación.

Para el cálculo de la energía (U(r)) del sistema es necesario tener en cuenta todas las interacciones que pueden estar presentes entre los átomos y moléculas. Esta energía se calcula como la suma de todas las interacciones presentes. Teniendo en cuenta que muchas de esas interacciones son muy pequeñas, en una buena aproximación pueden ser despreciadas. Por tal motivo se da la posibilidad de escoger los potenciales que se consideren necesarios, a partir de una lista donde se recojan la mayor cantidad posible de estos.

La relación entre los resultados de la simulación y el experimento lo establece la Física Estadística [26], por lo que es necesario tener en cuenta los parámetros termodinámicos que permanecen constantes durante una corrida de KMC; esto lo define el ensemble termodinámico. Entre los más usados se encuentran el micro canónico (NVE), el canónico (NVT) y el isotérmico-isobárico (NPT); donde N es la cantidad de partículas, V es el volumen, E es la energía, P es la presión y T es la temperatura, que son las magnitudes que permanecen constantes según el ensemble.
Existen software que utilizan estas técnicas de simulación, tales como:

- Crystal: es un programa de química cuántica, diseñado principalmente para los cálculos en los cristales (3 dimensiones), las losas (2 dimensiones) y polímeros (1 dimensión) utilizando la simetría de traslación, pero también puede ser utilizado para moléculas individuales[27].

[7] La constante de Planck, simbolizada con la letra h, es una constante física que representa al cuanto elemental de acción. Es la relación entre la cantidad de energía y de frecuencia asociadas a un cuanto o a una partícula. Desempeña un papel central en la teoría de la mecánica cuántica y recibe su nombre de su descubridor, Max Planck.

- Open PaperOpt: Es una plataforma de la simulación de código abierto utilizando Montecarlo para la simulación del nivel de la partícula de dispersión luminosa en estructuras de papel generadas.[28].
- DL- Poly: es un software de simulación de dinámica molecular clásica, es un paquete de subrutinas, programas y ficheros de datos, diseñado para facilitar las simulaciones de dinámica molecular de macromoléculas, polímeros, sistemas iónicos y sus soluciones[29].

Desarrollar un software con calidad depende de actividades y etapas, donde el impacto de elegir la mejor metodología para un equipo, en un determinado proyecto, es trascendental para el éxito del producto[30], donde incrementen su potencial, aumenten la calidad del producto con los recursos y tiempos destinados a esa producción.

1.4 Metodología de desarrollo

Dentro de las clasificaciones de metodologías de desarrollo que se pueden emplear para la modelación de la herramienta, se encuentran las metodologías tradicionales o pesadas, y las metodologías ágiles o ligeras. Las metodologías tradicionales generan mucha documentación, están destinadas para un equipo de desarrollo grande y para proyectos a largo plazo. Además, no están definidas para cambios frecuentes y tratan de especificar todos los requisitos del software antes de comenzar su desarrollo, lo cual conlleva a la decisión de utilizar las metodologías ágiles en la presente investigación.

Las metodologías ágiles se basan en el desarrollo de software en plazos cortos, y ponen de relevancia que la capacidad de respuesta a un cambio es más importante que el seguimiento estricto de un plan de desarrollo del trabajo. Entre dichas metodologías se encuentra Programación Extrema[8] (XP, por sus siglas en inglés) que se centra en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores y propiciando un buen clima de trabajo. XP se basa en retroalimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y audacia para enfrentar los cambios. XP es principalmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico. Corrige los errores antes de

[8] Extreme Programming

añadir una nueva funcionalidad, realizando entregas frecuentes del software para analizar su funcionamiento[31].

Dentro de las condiciones de ambiente para el desarrollo de este trabajo se tiene poco personal de desarrollo, pues solo se dispone de un integrante, se cuenta con poco tiempo para el desarrollo de la aplicación, los requerimientos cambian a lo largo de todo el ciclo de vida de la aplicación. La existencia de estos factores principales determinó la selección de la metodología ágil XP que, como especifican sus características, constituye una de las metodologías destacada dentro de los procesos ágiles para el desarrollo de un software.

1.5 Lenguaje de Modelado.

Un lenguaje de modelado es un conjunto estandarizado de símbolos para modelar un diseño de software, y permite a los creadores y diseñadores de sistemas generar diseños que capturen las ideasen una forma convencional y fácil de comprender para el resto de las personas[32]. Para el desarrollo de la investigación se selecciona el Lenguaje Unificado de Modelado (UML)[9].

- Lenguaje Unificado de Modelado

UML es la sucesión de una serie de métodos de análisis y diseño orientados a objetos, tiene una notación gráfica muy expresiva que permite representar en mayor o menor medida todas las fases de un proyecto informático. UML se ha convertido en un estándar para representar y modelar la información con la que se trabaja en las fases de análisis y, especialmente, de diseño. UML sirve para el modelado completo de sistemas complejos, tanto en el diseño de los software como para la arquitectura hardware donde se ejecuten.[33]

En el presente trabajo se hace necesaria la utilización de UML para el modelado de los diagramas del proceso de negocio y representación de los componentes de diseño que se realizan al software, y muestran una detallada visión de lo que presenta la aplicación en cada una de sus funcionalidades.

- Notación de Modelado

[9]Unified Modeling Language

Notación para el Modelado de Procesos de Negocio (BPMN)[10] es una notación para diseñar de forma estandarizada el modelado de procesos de negocio del sistema. Su principal objetivo es proporcionar una notación que sea fácilmente legible y entendible por parte del equipo que desarrolla el negocio, y su modelado se realiza mediante diagramas simples con un conjunto pequeño de elementos gráficos[34], lo que permite que los usuarios del negocio y desarrolladores de la aplicación les sea fácil comprender el flujo y el proceso.

1.6 Herramientas CASE

Una herramienta CASE (Ingeniería de Software Asistida por Computadora) es un programa informático destinada a detallar la productividad en el desarrollo de software reduciendo el costo de las mismas en términos de tiempo y gastos.

Para realizar los modelos visuales de la aplicación se escoge la herramienta CASE Visual Paradigm, herramienta que brinda un número considerable de estereotipos para la elaboración de los diagramas y soporta el ciclo de vida completo del desarrollo de software. Permite diseñar todos los tipos de diagramas de clases, código inverso, componer código desde diagramas y generar documentación[35].Favorece su uso en el presente desarrollo porque es eficiente para el proceso de negocio de la aplicación, es disponible para múltiples versiones y plataformas, y el equipo de desarrollo posee experiencia con el trabajo en esta herramienta.

1.7 Lenguaje de Programación

Los lenguajes de programación están diseñados para representar un conjunto de acciones sucesivas que un equipo debe realizar, es un modo práctico para que las personas puedan dar instrucciones a dicho equipo. Contiene un grupo de símbolos y pautas sintácticas y semánticas que puntualizan su estructura y el significado de sus expresiones.

Entre estos lenguajes se encuentra Java, que es basado en C++[11], es más entendible para el equipo de desarrollo[36], basado también en clases y orientado a objetos, implementando varias características como la herencia, instancias de objetos, reutilización de datos; y es un lenguaje que puede ser ejecutado en múltiples plataformas, sin necesidad de cambiar el código. Evita la

[10]Business Process Modeling
[11]C++ es un lenguaje orientado a objetos. Creado para añadirle cualidades y características de las que necesitaba C, conservando una considerable potencia para programación a bajo nivel.

corrupción de datos, y tiene un fácil manejo de excepciones; traduce el código fuente a código intermedio, y es multitarea.

Se ha escogido el lenguaje de programación Java para la implementación de la herramienta informática propuesta, debido a las potenciales características que brinda, permitiendo este lenguaje la implementación de un código legible para la aplicación sin complejidades, y con el uso de recursos y librerías que facilitan la obtención de funcionalidades óptimas en el trabajo.

- Bibliotecas utilizadas.

Librería iText, una biblioteca Open Source[12] para crear y manipular archivos PDF y HTML en Java. Su principal objetivo es la creación de archivos de determinada extensión que pueden ser exportados en múltiples formatos, o múltiples instancias del mismo formato. En el presente trabajo se utiliza esta biblioteca para la construcción de un archivo *.pdf que le va a permitir al usuario final conservar los resultados y propiedades físicas del sistema simulado.

Jmol: un visor Java de código abierto para estructuras químicas en tres dimensiones, con prestaciones para compuestos químicos, cristales y biomoléculas. En la aplicación es utilizada para visualizar los resultados obtenidos luego de la simulación, mediante un archivo *.xyz que el usuario, en la ventana de resultados finales, tiene como opción extraer.

1.8 Entorno de Desarrollo Integrado

Entorno de Desarrollo Integrado (IDE)[13], es un programa informático constituido por un conjunto de herramientas de programación. Los lenguajes de programación necesitan de un IDE de desarrollo como soporte para su implementación, existen IDE que presentan soporte para el lenguaje de programación Java:

- NetBeans

NetBeans es un entorno de desarrollo integrado disponible para Windows, Mac, y Linux. Este IDE está integrado por código abierto y una plataforma de aplicación. Se compone de herramientas que presentan una gran estabilidad, es un producto libre y sin restricciones de uso.

[12]Código abierto (o *fuente abierta*) es el término con el que se conoce al software distribuido y desarrollado libremente.
[13]Integrated Development Environment

Para el trabajo se escoge como editor de texto el IDE NetBeans, pues tiene acceso a todos los ficheros del proyecto, contiene una ventana de depuración y errores, información de los parámetros que usa una función, subrayado de errores automático, es autocompletado; contiene una ventana con un completo listado de todas las variables, funciones, objetos, métodos, y demás del fichero que se esté editando, y tiene acceso rápido a varios proyectos a la vez; siendo estas características una parte fundamental que permite el desarrollo de la aplicación sin mayores dificultades a la hora de su implementación y con acceso fácil a sus componentes visuales para el desarrollo de la interfaz visual sin necesidad de emplear otra herramienta informática.

1.9 Plataforma de Desarrollo

Una plataforma de desarrollo es un entorno físico o lógico sobre el que se puede ejecutar un programa[37].Es una solución completa para desarrollar software y sistemas basados en software. Permite al equipo de desarrollo que utilice una plataforma de desarrollo de software construir, integrar, extender, modernizar e implantar software y sistemas basados en software[38].

- Plataforma Java.

Para el desarrollo de la aplicación se selecciona la plataforma Java 2 Estándar Edition (J2SE, por sus siglas en inglés), que es la propuesta de la empresa Oracle[14] para el desarrollo y la implementación de aplicaciones de escritorio; esta plataforma se apoya por completo en el lenguaje Java.

El JRE (Java Runtime Environment, o Entorno en Tiempo de Ejecución de Java) es el software necesario para ejecutar cualquier aplicación desarrollada para la plataforma Java. El usuario final utiliza el JRE como parte de paquetes software. Ofrece también el JDK (Java Development Kit) donde reside el JRE, e incluye herramientas como el compilador de Java, y puede también obtenerse como un paquete independiente, considerándose como el entorno necesario para ejecutar una aplicación Java. Para el desarrollo de la aplicación se utilizará la versión JDK 1.6.

[14]Empresa multinacional de tecnología informática estadounidense que se especializa en el desarrollo y comercialización de sistemas informáticos de hardware y productos de software para empresas

1.10 Frameworks empleados la solución

Framework es un concepto genérico, haciendo referencia al ambiente de trabajo, y ejecución[39]. Constituyen soluciones que contemplan herramientas de apoyo al ambiente de trabajo o desarrollo y motores de ejecución.

- Framework de Presentación

Swing Application Framework es una especificación de Java para marcos de aplicaciones simples con una interfaz gráfica de usuario, define una infraestructura común para la mayoría de las aplicaciones de escritorio, por lo que las aplicaciones Swing son más fáciles de crear.

1.11 Conclusiones Parciales

En el presente capítulo se ha realizado un estudio de las tendencias actuales, así como una investigación puntualizada de lo que ha logrado el mundo en esta rama. La selección del lenguaje de programación, las herramientas, la metodología a utilizar en el desarrollo de la aplicación propuesta, hacen del trabajo una definición de lo que se quiere lograr realizar, teniéndose así varias conclusiones, tales como:

- Realizar el diseño y modelado de la aplicación con la metodología XP, utilizando UML como lenguaje de modelado, análisis y diseño, y BPMN como notación para el diseño del negocio del sistema.
- Realizar la implementación de la herramienta informática con el lenguaje de programación Java, porque permite definir de forma legible los códigos concretos que construyen la aplicación final, y presenta un manejo sencillo de sus funciones gráficas para elaborar el modelo visual.
- Utilizar el IDE de desarrollo NetBeans, porque contiene soporte para dicho lenguaje y es ventajoso por su completamiento de código y compatibilidad, además que puede ser utilizado en cualquier plataforma.

Una vez conocidas las herramientas y los conceptos claves para el desarrollo y comprensión de la herramienta que se desea realizar, se comenzará a desarrollar la propuesta de solución.

Capítulo 2: Características del Sistema

2.1 Introducción

La descripción de las características esenciales que debe tener la solución que se propone es el punto de partida del presente capítulo, las cuales se encuentran respaldadas por la modelación de los procesos del negocio. Se describe y fundamenta la primera etapa que describe la metodología de software escogida, guiando paso a paso el proceso de desarrollo de la aplicación; y los requerimientos funcionales y no funcionales se llevan a procedimientos diseñados para su posterior implementación.

2.2 Descripción del proceso de negocio

El proceso de simular la síntesis de zeolitas mediante el método de Montecarlo Cinético comienza cuando el usuario crea un nuevo experimento, para ello debe elegir las moléculas que formarán parte de la simulación a efectuar. Se cuenta con la información necesaria de las moléculas, tales como el nombre, su fórmula química, un diagrama que agrupa los átomos que componen cada molécula y el listado de átomos que contiene cada una, teniendo de estos el nombre, carga, masa y sus parámetros potenciales. El usuario puede modificar los datos de los átomos que estime conveniente así como agregar la cantidad de un tipo de molécula específico que desee, eliminarla luego de ser agregada a la caja de simulación; así como crear una nueva molécula para adicionarla al sistema y utilizarla posteriormente.

El usuario determina la dimensión de la caja de simulación (coordenadas X, Y, Z) dada en Angström (símbolo Å, 10^{-10}m). Dicha caja agrupa todas las moléculas que van a realizar transiciones entre sí durante el proceso de simulación (trasladar o rotar); también se debe determinar el valor de separación entre moléculas para establecer la distancia mínima entre ellas. Para complementar los parámetros necesarios que establecen la simulación, se necesita elegir los potenciales, ensembles, y valores de temperatura con los que trabajará el sistema.

El usuario tiene la opción de equilibrar el proceso de simulación, que le permite organizar las moléculas que quedan agrupadas en la caja, calculando la energía total para tratar de obtener una energía menor que la inicial, llevando el proceso a una configuración próxima a su estado de equilibrio; dicha operación es opcional para el usuario pues la no realización de la misma no impide el ejecutar el proceso de simulación. Seguidamente, el usuario puede realizar la

simulación, estableciendo el tiempo de duración de la misma, y se puede comprobar mediante los resultados que se obtienen, siendo esto el principal objetivo de todo el experimento, pues el listado de las moléculas con sus coordenadas finales de ubicación le indica al usuario el comportamiento de cada una durante el proceso de simulación. El usuario puede guardar las configuraciones que vaya realizando, así como los resultados que obtiene finalmente. Ver Figura 2.

2.3 Modelación del negocio

- Diagramas de procesos de negocio

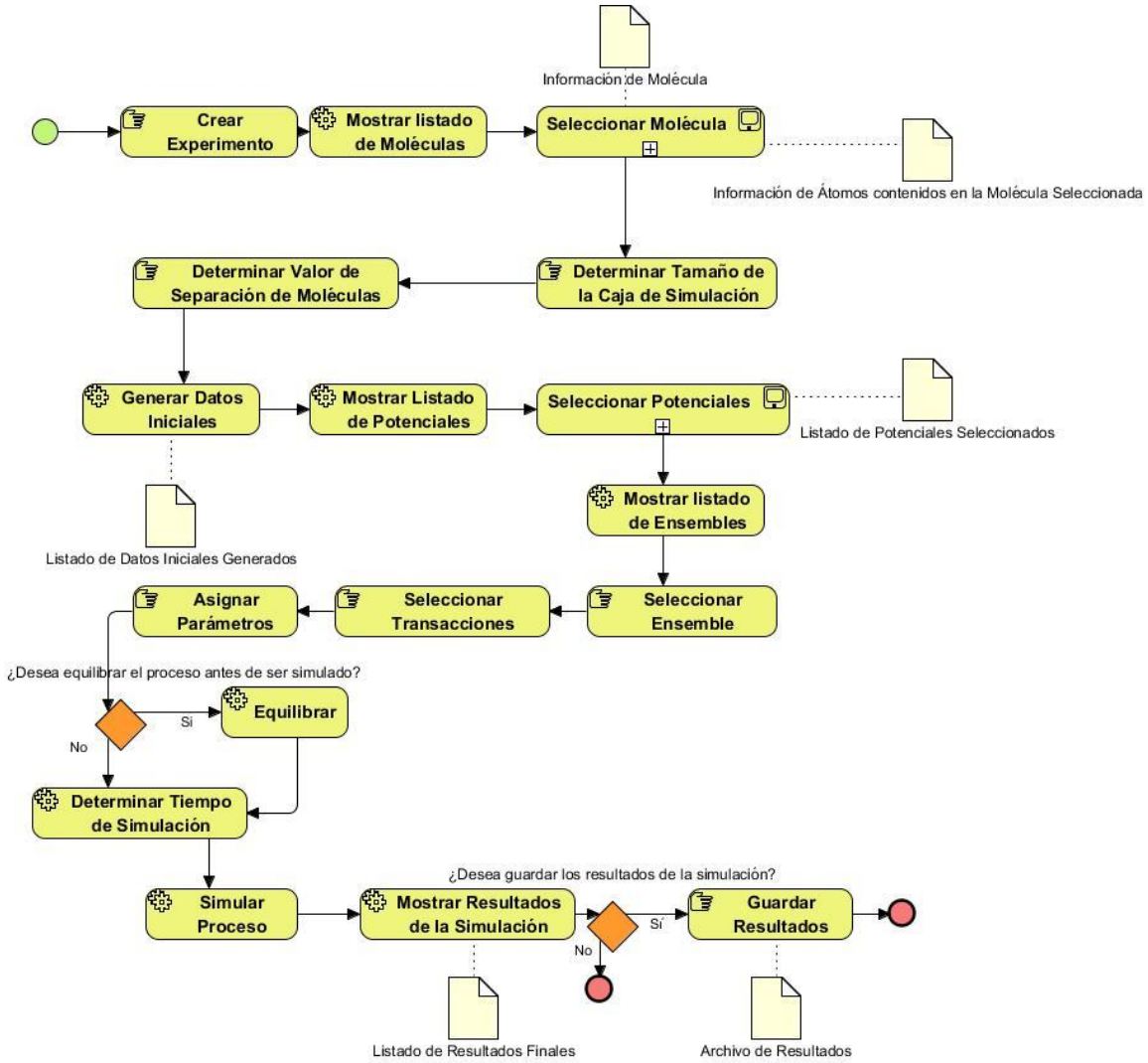

Figura 2. Crear Nuevo Experimento.

- Especificación de Procesos:

Actividad 1: Crear Experimento.

Responsable: Equipo de Desarrollo.

Participantes: Cliente.

Misión: acordar las tareas que se realizarán inicialmente, fijar los momentos de encuentro entre las partes, y comprender lo que desea el cliente.

Entradas:
- Cuestionario de la entrevista.

Salidas:
- Formulario con todos los elementos que desea tener el cliente en la aplicación.

Actividad 2: Mostrar listado de moléculas.

Responsable: Equipo de Desarrollo.

Participantes: Cliente.

Misión: mostrar información inicial de las moléculas que, a petición del cliente, debe contener el sistema para facilitar su uso.

Entradas:
- Datos de moléculas.

Salidas:
- Listado de las moléculas.

Actividad 3: Seleccionar Moléculas.

Responsable: Equipo de Desarrollo.

Participantes: Cliente.

Misión: permitir la selección de las moléculas que se encuentran listadas, adicionarlas para su utilización en la aplicación, modificar los datos de sus átomos, y eliminarlas en caso de no utilizarla.

Entradas:
- Listado de Moléculas.

Salidas:
- Moléculas seleccionadas.

Actividad 4: Determinar tamaño de la caja de simulación.

Responsable: Equipo de Desarrollo.

Participantes: Cliente.

Misión: permitir la asignación del tamaño de la caja de simulación que se involucra en el proceso de simulación de las moléculas en la aplicación.

Entradas:
- Tamaño de la caja de simulación

Salidas:
- Tamaño de la caja de simulación otorgada por el usuario.

Actividad 5: Determinar valor de separación de moléculas.

Responsable: Equipo de Desarrollo.

Participantes: Cliente.

Misión: separar, mediante un valor numérico entrante, las moléculas que se seleccionan para realizar la simulación.

Entradas:
- Valor de separación mayor o igual a 3.0 Angstrom.

Salidas:
- Moléculas separadas a dicho valor entrante.

Actividad 6: Generar datos iniciales.

Responsable: Equipo de Desarrollo.

Participantes: Cliente.

Misión: confeccionar el sistema molecular inicial que el usuario desea simular, y la posibilidad de guardar los datos que se modifiquen así como las posiciones iniciales de las moléculas antes de establecer la separación entre ellas.

Entradas:
- Listado de moléculas seleccionadas, tamaño de la caja de simulación y separación entre moléculas.

Salidas:
- Datos generados para su utilización en la simulación.

Actividad 7: Mostrar listado de potenciales.

Responsable: Equipo de Desarrollo.

Participantes: Cliente.

Misión: mostrar el listado de los potenciales que contiene la aplicación para el cálculo de las energías que presenta el sistema molecular al realizar la simulación.

Entradas:
- Información de los potenciales.

Salidas:
- Listado de los potenciales.

Actividad 8: Seleccionar Potencial.

Responsable: Equipo de Desarrollo.

Participantes: Cliente.

Misión: permitir la selección de los potenciales que el usuario necesite para involucrarlos en el proceso de simulación.

Entradas:
- Listado de potenciales.

Salidas:
- Potenciales seleccionados, selección que puede ser eliminada si el usuario desea.

Actividad 9: Mostrar listado de ensembles.

Responsable: Equipo de Desarrollo.

Participantes: Cliente.

Misión: mostrar el listado de los ensembles que contiene la aplicación para establecer parámetros que presenta el sistema molecular al realizar la simulación.

Entradas:
- Información de los ensembles.

Salidas:
- Listado de ensembles.

Actividad 10: Seleccionar Ensemble.

Responsable: Equipo de Desarrollo.

Participantes: Cliente.

Misión: permitir la selección del ensemble que el usuario necesita en el proceso de simulación.

Entradas:
- Listado de ensembles.

Salidas:
- Ensembles seleccionado.

Actividad 11: Seleccionar Transacciones.

Responsable: Equipo de Desarrollo.

Participantes: Cliente.

Misión: permitir la selección de los eventos que el usuario desea realicen las moléculas al comenzar el proceso de simulación. Los eventos pueden ser: Trasladar molécula, Rotar Molécula, Enlazar Molécula.

Entradas:
- Los eventos que la aplicación es capaz de realizar.

Salidas:
- Al realizar el proceso de simulación se calcula la probabilidad de que ocurran dichos eventos, y la molécula posible realiza el evento correspondiente.

Actividad 12: Asignar Temperatura.

Responsable: Equipo de Desarrollo.

Participantes: Cliente.

Misión: definir el valor de temperatura que va a tener el sistema.

Entradas:
- Valor de temperatura.

Salidas:
- El sistema tendrá el valor de temperatura asignado por el usuario, cuyo valor influirá en los cálculos que se involucran en el proceso de simulación.

Actividad 13: Equilibrar.

Responsable: Equipo de Desarrollo.

Participantes: Cliente.

Misión: establecer un equilibro en el sistema, obteniendo una posible menor energía para favorecer al proceso de simulación.

Entradas:
- Selección de uno o más potenciales, y un valor de temperatura.

Salidas:
- Se calcula la energía del sistema, se le dan nuevas posiciones a las moléculas en el sistema, y se calcula la nueva energía que obtiene.

Actividad 14: Determinar tiempo de simulación.

Responsable: Equipo de Desarrollo.

Participantes: Cliente.

Misión: permitir la determinación de un tiempo de duración para el proceso de simulación.

Entradas:
- El valor del tiempo de duración en minutos. Ej.: 1, Ej.: 2.

Salidas:
- El proceso de simulación se realizará hasta el tiempo de duración que se haya establecido.

Actividad 15: Simular Proceso.

Responsable: Equipo de Desarrollo.

Participantes: Cliente.

Misión: realizar el proceso de simulación de las moléculas en el sistema.

Entradas:
- Selección de uno o más potenciales, un valor de temperatura, seleccionar el ensemble, el tiempo de duración y las transacciones posibles a realizar.

Salidas:
- Durante el tiempo que se determinó, se calculan las probabilidades de cada molécula para realizar alguna transacción, y en caso de cumplir con el criterio establecido por la técnica aplicada realizan dichas transacciones.

Actividad 16: Mostrar Resultados de la simulación.

Responsable: Equipo de Desarrollo.

Participantes: Cliente.

Misión: mostrar los resultados que se obtienen luego del proceso de simulación, así como las propiedades físicas del sistema.

Entradas:
- Realizar la simulación del sistema.

Salidas:
- Resultados de la simulación.

Actividad 17: Guardar Resultados.

Responsable: Equipo de Desarrollo.

Participantes: Cliente.

Misión: permitir exportar los resultados para la persistencia de los datos finales.

Entradas:
- Selección del tipo de archivo en que desea guardar los resultados.

Salidas:
- Archivo con los datos almacenados.

2.4 Propuesta del Sistema.

Se implementará una aplicación Desktop para la simulación de los procesos de síntesis de materiales porosos tipo zeolitas, mediante el método de Montecarlo Cinético. Contará con interfaces independientes de la principal, compuestas por varios componentes visuales que permitirán: gestionar moléculas, determinar el tamaño de la caja de simulación y la separación entre moléculas, mostrar las posiciones de los átomos, seleccionar los potenciales, ensembles y eventos posibles a realizarse, establecer el tiempo de ejecución de la simulación y mostrar los resultados del proceso de simulación.

La aplicación permitirá un sencillo manejo de sus funcionalidades, el usuario sólo necesitará tener un conocimiento básico para la manipulación de la aplicación desde el punto de vista informático, podrá seleccionar, de manera visual, los datos que desee utilizar, modificarlos, y emplearlos en su experimento, conocer las propiedades físico-químicas de la solución obtenida, así como visualizarla; pues el software estará enfocado en ofrecer lo necesario para abarcar todas las funcionalidades del proceso de simulación, sin que el usuario necesite implementar ninguna de ellas ni consultar otro software.

2.5 Especificación de los requisitos de software.

Los requisitos son un componente principal en el proceso de creación de una herramienta informática, y están presentes en diversos componentes de la aplicación para satisfacer las especificaciones pendientes a través del usuario. Es una condición o capacidad que tiene que ser alcanzada por un sistema, o componente de un sistema, para satisfacer un contrato, estándar, u otro documento impuesto formalmente.[41]

Dichos requisitos presentan dos categorías, pueden ser requisitos funcionales y requisitos no funcionales. Los requisitos funcionales expresan una acción que debe ser capaz de realizar el sistema, es la declaración de servicios que el sistema debe proporcionar, cómo debería reaccionar el sistema a determinadas entradas y cómo debería comportarse en cada situación. Los requisitos no funcionales son restricciones de los servicios o funciones ofrecidas por el sistema que expresan una propiedad o cualidad que el mismo debe presentar, teniendo en cuenta aspectos como el rendimiento, usabilidad, seguridad e interfaz de dicha aplicación.

La captura de estos requerimientos permite desarrollar un modelo del sistema que se va a construir, y para extraerlos se utilizó la técnica de entrevistas, realizándose frecuentes encuentros con el cliente.

2.5.1 Requisitos Funcionales

RF1. Crear Experimento
RF2. Abrir Experimento
RF3. Gestionar Molécula
- **RF3.1** Crear Molécula
- **RF3.2**AdicionarMolécula
- **RF3.3**Modificar Molécula
- **RF3.4**Eliminar Molécula

RF4. Modificar Átomo
RF5. Otorgar Dimensiones de Caja de Simulación
RF6. Separar Moléculas
RF7. Guardar Datos
RF8. Gestionar Potencial
- **RF8.1**Adicionar Potencial
- **RF8.2**Eliminar Potencial

RF9. SeleccionarEnsemble
RF10. Seleccionar Transacciones
RF11. Asignar Temperatura al Sistema
RF12. Equilibrar Proceso.
RF13. Simular Proceso
RF14. Guardar Posiciones Actuales.
RF15. Mostrar Resultados
RF16. Guardar Resultados

2.5.2 Requisitos no funcionales

- Requisitos del Software

Se necesita tener instalado en el ordenador cliente JDK (máquina virtual de Java) para el funcionamiento de la aplicación.

- Requisitos del Hardware

Memoria RAM: Mínimo 1 GB.

- Requisitos de Interfaz de usuario

La interfaz contará con los componentes visuales para las operaciones que el usuario necesita, sin sobrecarga de imágenes.

2.6 Fase de Exploración

Exploración es la primera fase definida por la metodología XP, define el alcance general que tiene el proyecto, y su actividad comienza con la creación de las historias de usuario, que describen las características y funcionalidades que se requieren para la construcción del software.

En esta fase, los clientes plantean a grandes rasgos las historias de usuario que son de interés para la primera entrega del producto. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto. Se prueba la tecnología y se exploran las posibilidades de la arquitectura del sistema construyendo un prototipo[31].

2.6.1 Historias de Usuario

Uno de los artefactos que describe XP son las historias de usuario, que constituyen la forma en que se especifican los requisitos del sistema. Estas se escriben desde la perspectiva del cliente aunque los desarrolladores pueden brindar ayuda en su identificación. Corresponden a la técnica utilizada para especificar los requisitos del software. Se trata de formatos en los cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales. El tratamiento de las historias del usuario es muy dinámico y flexible. Cada historia de usuario es lo suficientemente comprensible y delimitada para que el programador pueda implementarla en unas semanas[42].

Las historias de usuario se dividen en dos clasificaciones, definidas como:

Escala Nominal de Prioridad en el Negocio:

- ✓ Alta: son las historias de usuario que constituyen funcionalidades de gran importancia en el desarrollo del software.
- ✓ Media: lo constituyen aquellas funcionalidades a tener en cuenta sin que haya una afectación directa sobre la aplicación.

- ✓ Baja: aquellas historias de usuario que constituyen funcionalidades que sirven de ayuda al control de la estructura del software.

Escala Nominal de Riesgo en Desarrollo:

- ✓ Alta: aquellas historias de usuario que, para su implementación, se considera la posible existencia de errores que lleven a la interrupción del desarrollo del código.
- ✓ Media: cuando se retrasa la entrega de la aplicación debido a los errores que puedan aparecer en la implementación de la historia de usuario.
- ✓ Baja: cuando aparecen errores que serán tratados con coherente facilidad sin que traigan dificultades para el desarrollo de la aplicación.

A continuación se presentan las historias de usuarios de mayor prioridad para el desarrollo de la aplicación:

Historia de Usuario	
Número: 1	**Usuario:** Usuario.
Nombre historia: Crear Experimento	
Prioridad en negocio: Alta.	**Riesgo en desarrollo:** Alto.
Puntos estimados:1	**Iteración asignada:** 1
Programador responsable: Yaqueline Mejías De la Torre	
Descripción: Se selecciona la opción Crear Experimento en la ventana principal para comenzar la utilización del sistema.	
Observaciones: Al crear un nuevo experimento se muestra la primera ventana que le permitirá al usuario comenzar a definir las condiciones de su sistema para establecer la simulación.	
Prototipo de Interfaz	

Tabla 1. Crear Experimento

Historia de Usuario

Número: 3	**Usuario:** Usuario.
Nombre historia: Gestionar Molécula	
Prioridad en negocio: Alta.	**Riesgo en desarrollo:** Alto.
Puntos estimados:1	**Iteración asignada:** 1
Programador responsable: Yaqueline Mejías De la Torre	
Descripción: **Crear Molécula:** Permite al usuario adicionar al software una molécula que no esté almacenada en la aplicación. Esta funcionalidad se encuentra en el menú de Acciones en la ventana Generar Sistema. **Adicionar Molécula:** Se adiciona una molécula a la caja de simulación para generar el sistema. **Modificar Molécula:** Permite al usuario modificar la cantidad del tipo de molécula seleccionada, cambiando el valor de cantidad que tiene por defecto, y quedarán adicionadas a la caja de simulación n cantidad de ese tipo de molécula. **Eliminar Molécula:** Elimina la molécula que el usuario seleccione de la tabla que contiene el listado de todas las moléculas adicionadas por él.	
Observaciones: La molécula queda adicionada y sus datos se muestran en una tabla donde el usuario puede modificar la cantidad del tipo de molécula seleccionada que desee introducir en la caja, y puede eliminarla seleccionando la que desea extraer del listado adicionado.	

Tabla 2. Gestionar Molécula

	Historia de Usuario
Número: 8	**Usuario:** Usuario.
Nombre historia: Gestionar Potencial	
Prioridad en negocio: Alta.	**Riesgo en desarrollo:** Alto.
Puntos estimados:1	**Iteración asignada:** 2
Programador responsable: Yaqueline Mejías De la Torre	
Descripción: **Adicionar Potencial:**	

Se adicionan los potenciales que se involucran en el proceso de simulación. **Eliminar Potencial:** Permite al usuario seleccionar un potencial de la lista de potenciales adicionados y eliminarlo.
Observaciones: Para cada potencial existen varios tipos que, según lo que desea el cliente, podrá seleccionar para su solución experimental.

Tabla 3. Gestionar Potencial

Historia de Usuario	
Número: 9	**Usuario:** Usuario.
Nombre historia: Seleccionar Ensemble	
Prioridad en negocio: Alta.	**Riesgo en desarrollo:** Alto.
Puntos estimados:1	**Iteración asignada:** 2
Programador responsable: Yaqueline Mejías De la Torre	
Descripción: Se selecciona el ensemble que se involucra en el proceso de simulación.	
Observaciones:	

Tabla 4. Seleccionar Ensemble

Historia de Usuario	
Número: 11	**Usuario:** Usuario.
Nombre historia: Asignar Temperatura al Sistema.	
Prioridad en negocio: Alta.	**Riesgo en desarrollo:** Alto.
Puntos estimados:1	**Iteración asignada:** 2
Programador responsable: Yaqueline Mejías De la Torre	
Descripción: Se asigna el valor de presión y temperatura con el que se va a simular el proceso.	
Observaciones:	

Tabla 5. Asignar Temperatura al Sistema.

Historia de Usuario	
Número: 13	**Usuario:** Sistema.
Nombre historia: Simular Proceso.	
Prioridad en negocio: Alta.	**Riesgo en desarrollo:** Alto.
Puntos estimados:3	**Iteración asignada:** 3
Programador responsable: Yaqueline Mejías De la Torre	
Descripción: Se simula el proceso con todos los datos seleccionados, presionando el botón que se encuentra en esa ventana.	
Observaciones: Toma todos los datos y valores asignados por el usuario, y simula el proceso para la obtención de los resultados que se desea.	

Tabla 6. Simular Proceso

Historia de Usuario	
Número: 15	**Usuario:** Sistema.
Nombre historia: Mostrar Resultados.	
Prioridad en negocio: Alta.	**Riesgo en desarrollo:** Medio.
Puntos estimados:2	**Iteración asignada:** 4
Programador responsable: Yaqueline Mejías De la Torre	
Descripción: Cuando el sistema culmina el proceso de simulación le da la opción al usuario de mostrar los resultados que se obtienen en la caja de simulación, o el usuario puede ir al menú principal de la ventana y elegir la opción de Mostrar Resultados, solo luego de haber realizado la simulación.	
Observaciones:	

Tabla 7. Mostrar Resultados

2.6.2 Planificación de la Entrega

En este punto se establece la prioridad de cada historia de usuario, y en correspondencia, los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente; dicha planificación se puede realizar basándose en el tiempo o el alcance[31].

Las estimaciones, relacionadas a la implementación de las historias, las establece el programador utilizando como medida el punto. Cada punto equivale a una semana de implementación, teniendo como máximo 3 puntos. Por otro lado, el equipo de trabajo mantiene un registro de desarrollo, lo cual establece en puntos de iteración, basando esto en la suma de puntos que corresponde a las historias de usuario que fueron culminadas en la última iteración.

Cada HU se traduce en tareas específicas de programación. Del mismo modo, para cada HU se establecen las pruebas de aceptación. Estas pruebas se realizan al final del ciclo en el que se desarrollan, pero también al final de cada uno de los ciclos siguientes, para verificar que sucesivas iteraciones no han afectado a las anteriores. Para un alto desempeño y resultados más satisfactorios del equipo de desarrollo, se definieron las iteraciones según el nivel de importancia y aparición de cada funcionalidad en el sistema.

- Estimación de esfuerzo por historias de usuarios

Para el desarrollo de la aplicación se tienen en cuenta situaciones referentes a HU, tales como la aparición de nuevas historias, aplazamiento en cuanto a la iteración en la cual se implementa, el no cumplimiento en la iteración planificada, y transformación durante la iteración, ejecutándose una estimación del esfuerzo para cada una de las HU identificadas. La *Tabla 8* muestra los resultados de lo planteado:

No. HU	Historia de Usuario	Puntos de estimación
1.	Crear Experimento	1
2.	Abrir Experimento	2
3.	Crear Molécula	1

4.	Adicionar Molécula	1
5.	Modificar Molécula	1
6.	Eliminar Molécula	1
7.	Modificar Átomo	1
8.	Otorgar Dimensiones de la Caja de Simulación	1
9.	Separar Moléculas	1
10.	Guardar Datos	2
11.	Adicionar Potencial	1
12.	Eliminar Potencial	1
13.	Adicionar Ensemble	1
14.	Seleccionar Transacciones	1
15.	Asignar Temperatura al Sistema.	1
16.	Equilibrar Proceso	2
17.	Guardar Posiciones Actuales	3
18.	Simular Proceso.	1
19.	Mostrar Resultados	2
20.	Guardar Resultados	2

Tabla 8. Resultados de Puntos de Estimación.

2.6.3 Plan de Iteraciones.

Al ser identificadas las historias de usuario y estimado el esfuerzo dedicado a la realización de cada una de ellas, se procede a la realización de la planificación para la etapa de implementación del software.

En la primera iteración se puede establecer una arquitectura del sistema que se pueda emplear durante el resto del proyecto, el cliente decide qué historias se implementarán en cada iteración para maximizar el valor de negocio. Al culminar la última iteración el sistema estará listo para entrar en producción. Los elementos que deben tomarse en cuenta durante la elaboración del

Plan de la Iteración son la rapidez del proyecto, pruebas de aceptación no superadas en alguna iteración, y tareas no finalizadas en la anterior iteración.

Cada iteración comprende un período de tiempo de desarrollo del proyecto entre una y cuatro semanas. Las siguientes iteraciones se han definido para lograr un mayor desempeño y resultados satisfactorios del equipo de desarrollo:

Iteración 1: En esta iteración se implementarán las historias de usuario correspondientes a la captura de datos para generar el sistema inicial, es decir, desde la HU 1 hasta la10, pues tienen una prioridad alta sobre el sistema, dándole posibilidad al usuario de probarlas antes de continuar su desarrollo.

Iteración 2: Esta iteración es para la implementación de las historias 11 hasta la 18, que se encargan de equilibrar y simular el sistema, guardar la información que se tiene hasta el momento si el usuario desea, y reúne todos los aspectos que se hacen necesarios para realizar una simulación satisfactoria. Una vez concluida esta iteración se realizar las pruebas suficientes para mostrarle al cliente el avance hasta el momento, con el objetivo de retroalimentar al grupo de trabajo.

Iteración 3: En la tercera iteración se implementan las HU que terminarán con los objetivos propuestos para el sistema, es decir, las historias 19 y 20. Al finalizar la misma se contará con una versión oficial del producto final.

- Plan de duración de las iteraciones

Mediante el uso de la metodología de desarrollo XP, y para obtener una mejor distribución del trabajo, resulta necesario un plan de duración de iteraciones como parte del ciclo de vida del proyecto. Tiene como objetivo mostrar la duración de cada iteración y el orden en que serán implementadas las HU. En el presente trabajo se realizó un solo plan debido a que existe un único y pequeño equipo de desarrollo. La *Tabla 9* expone la distribución de las HU según el orden en que serán abordadas en cada iteración, teniendo en cuenta su prioridad y el tiempo de duración de las mismas.

Iteración	Orden de las HU a implementar	Duración (Semanas)
Iteración 1	1. Crear Experimento.	5
	2. Abrir Experimento.	

	3. Crear Molécula.	
	4. Adicionar Molécula.	
	5. Modificar Molécula.	
	6. Eliminar Molécula.	
	7. Modificar Átomo.	
	8. Otorgar Dimensiones de la Caja de Simulación.	
	9. Separar Moléculas.	
	10. Guardar Datos.	
Iteración 2	1. Adicionar Potencial.	4
	2. Eliminar Potencial.	
	3. Adicionar Ensemble.	
	4. Seleccionar Transacciones.	
	5. Asignar Temperatura al Sistema.	
	6. Equilibrar Proceso.	
	7. Guardar Posiciones Actuales.	
	8. Simular Proceso.	
Iteración 3	1. Mostrar Resultados.	5
	2. Guardar Resultados.	

Tabla 9. Orden de Historias de Usuario a Implementar

- Plan de entregas

Las historias de usuario determinan el plan de entrega, que permiten crear los planes de iteración, y con cada historia previamente evaluada, el cliente las agrupa en orden de importancia. Para trazar el plan de entregas del software para la simulación de los procesos de síntesis de materiales porosos tipo zeolitas por el método de Montecarlo Cinético, se basa en dos parámetros: tiempo de desarrollo del sistema y el nivel de importancia para el usuario final.

El plan de entregas debe ser lo más exacto posible, pues constituye un documento oficial por el cual los clientes le exigen a los desarrolladores la entrega de las distintas iteraciones del producto. A continuación se muestra el plan estimado de entregas para la fase de implementación, realizándose al final de cada iteración versiones entregables del sistema.

Sistema	Fin iteración 1 (Febr. 2013)	Fin iteración 2 (Abril. 2013)	Fin iteración 3 (Mayo. 2013)
Software para la simulación de los procesos de síntesis de materiales porosos tipo zeolitas por el método de Montecarlo Cinético	Captura De Información.	Proceso de equilibrar y simular el sistema.	Muestra de Resultados finales.

Tabla 10. Plan Estimado de Entrega.

2.7 Conclusiones Parciales

El análisis realizado durante el presente capítulo indica un mejor entendimiento del trabajo y se crean las bases para la comprensión del sistema a implementar. La descripción de las Historias de Usuario es un paso fundamental en el ciclo de vida de desarrollo de un proyecto, artefacto generado por XP para la descripción de los requisitos funcionales. Seexpuso la planificación del equipo de desarrollo, compuesta por iteraciones que de manera incremental se implementarán en el sistema propuesto. Con el resultado del desarrollo del capítulo, se da paso a la próxima fase de desarrollo del sistema.

Capítulo 3. Diseño del Sistema

3.1 Introducción.

Establecer un correcto diseño de la aplicación es la forma de realizar una arquitectura consistente y una implementación correcta y eficiente. En el presente capítulo se detalla el diseño arquitectónico de la solución apoyándose de la metodología establecida, implantando así semejanza entre el trayecto del desarrollo del software y el objetivo trazado del mismo.

3.2 Diseño.

El diseño de un software es un proceso que contiene un conjunto de pasos que le permiten al diseñador describir los aspectos que ha de tener el sistema a construir, implementa los requisitos contenidos en el modelo de caracterización del sistema y agrupa todos los requerimientos implícitos que el usuario desea. Además, facilita una completa idea de lo que consiste el software a realizar, enfocando el proceso de negocio descrito, y su comportamiento desde el punto de vista de la implementación. Ayuda a identificar los errores y obstáculos comunes que ocurren al crear sistemas[43].

La etapa del Diseño del Sistema encierra las siguientes etapas:

- ✓ El diseño de los datos, que transforma el proceso de negocio, creado durante el análisis, en estructuras de datos vitales para implementar el software.
- ✓ El diseño arquitectónico, el cual define la relación entre los elementos estructurales de la aplicación.
- ✓ El diseño de interfaz de usuario, que describe la comunicación que estable el software con los usuarios y desarrolladores que lo emplean.

En el diseño están contenidos patrones de diseño, que constituyen la búsqueda de soluciones a determinado problema en el diseño o programación del software, proporcionando mayor eficiencia en la aplicación, y la relación entre clases; son soluciones que permiten reutilizar código, lo que significa que son aplicables a distintos problemas de diseño en varias circunstancias. Un patrón de diseño identifica las clases e instancias participantes, sus roles y colaboraciones, y la distribución de responsabilidades. Cada patrón de diseño se centra en un problema de diseño orientado a objetos en particular. En el mismo se describe cuándo se aplica, si se puede aplicar a la vista de otras limitaciones de diseño, y las consecuencias y compensaciones de su uso.

3.2.1 Patrones GRASP

GRASP (Responsabilidad General de asignación de Patrones de Software)[43] constituyen un conjunto de patrones genéricos del software para la asignación de responsabilidades, son una representación de buenas prácticas aconsejable para el diseño de la aplicación.

Experto: Experto en información específica que la responsabilidad de la creación de un objeto o la implementación de un método, ha de incurrir sobre la clase que conoce toda la información necesaria para crearlo. De este modo se logra un diseño con mayor cohesión[15], y la información se mantiene encapsulada, disminuyendo el acoplamiento. Cada objeto utiliza su información propia para ejecutar sus tareas, se trata el comportamiento entre las clases que contienen la información requerida siendo así más factibles de comprender y conservar. Su uso se evidencia la clase Simulación_Zeosim, responsable de realizar el proceso de simulación, pues cuenta con la información necesaria para ejecutar esta acción.

Alta Cohesión: Este patrón permite que la información que almacena una clase sea coherente y esté, en la medida de lo posible, relacionada con dicha clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen gran trabajo. Esto se evidencia en las clases Moléculas y Átomos, donde sus responsabilidades están relacionadas pero cada una realiza un número relativamente pequeño de responsabilidades.

Bajo Acoplamiento: Este patrón indica tener las clases lo menos liadas entre sí que se pueda, de forma talque al producirse una modificación en alguna de ellas, se tenga la mínima ramificación posible en el resto de clases, fortaleciendo la reutilización, y disminuyendo la dependencia entre las clases. Una especificación que contiene este patrón es el acoplamiento de control donde un módulo le envía a otro un elemento de control que determina la lógica de ejecución del mismo.

Controlador: El patrón controlador es el intermedio entre una interfaz determinada y el algoritmo que implementa dicha interfaz, es la que recibe los datos que el usuario determina y los envía a las diferentes clases según la llamada del método. Este patrón se relaciona estrechamente con la arquitectura establecida en este software, pues sugiere que la lógica del negocio debe estar separada de la capa de presentación, lo cual facilita aumentar la reutilización de códigos y, de la

[15]Cohesión, medida de cuán relacionadas y enfocadas están las responsabilidades de una clase.

misma forma, tener un mayor control; mientras mayor número de controladores se logren, aumentará la cohesión y disminuirá el acoplamiento.

3.2.2 Patrones GOF.

Los patrones GOF definen la estructura y comportamiento del sistema. Los empleados en el software fueron:

Singleton: Garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia. Su uso se evidencia en la clase Servicios_Implementación de la capa de Negocio, la cual implementa a la interfaz Servicios. En dicha clase se hace un llamado a una instancia de la clase principal Simulación_Zeosim, permitiendo que solo se cree un objeto de esta clase y que pueda ser utilizado en varios métodos.

```
public class Simulacion_Zeosim {

        private static Simulacion_Zeosim instance = null;

    public static Simulacion_Zeosim getInstance() {
        if (instance == null) {
            instance = new Simulacion_Zeosim();
        }
        return instance;
    }
```

Figura 3. Instancia de la clase Simulación_Zeosim

```
public class Servicios_Implement implements Servicios
{

    private Simulacion_Zeosim clase_principal;

    public Servicios_Implement() {

        this.clase_principal=Simulacion_Zeosim.getInstance();
    }
```

Figura 4. Llamado a la instancia de la clase Simulación_Zeosim

Iterator: Se utiliza para acceder al contenido de una colección de datos. Se evidencia tanto en las clases modelo como en métodos contenidos en las interfaces de usuario.

```
public double CalcularNuevaEnergia(Molecula actual) {
    double constanteK = 9 * Math.pow(10, 9);
    double nuevaEnergia = 0;
    int pos = -1;
    LinkedList<Atomo> atomos = new LinkedList<Atomo>();
    for (int i = 0; i < mol_agregadas.size(); i++) {
        if (mol_agregadas.get(i).getId() != actual.getId()) {
            for (int j = 0; j < mol_agregadas.get(i).getList_atomos().size(); j++) {
                atomos.add(mol_agregadas.get(i).getList_atomos().get(j));
            }
        }
    }
    for (int i = 0; i < actual.getList_atomos().size(); i++) {
        for (int j = 0; j < atomos.size(); j++) {
            double sigma = CalcularSigmaAtomo(actual.getList_atomos().get(i), atomos.get(j));
            double exilo = CalcularExiloAtomo(actual.getList_atomos().get(i), atomos.get(j));
            double distance = CalcularDistanciaAtomo(actual.getList_atomos().get(i), atomos.get(j));
            double carga = CalcularCargaAtomo(actual.getList_atomos().get(i), atomos.get(j));

            if (pot_agregados.size() == 1) {
                //....Si es el potencial de Lennard Jones....//
                if (pot_agregados.get(0).getId() == 1) {
                    nuevaEnergia += 4 * exilo * (Math.pow((sigma / distance), 12) - Math.pow((sigma
                } else {   //....Si es el potencial de Coulomb....//
                    nuevaEnergia += constanteK * (carga / distance);
                }
            } else {   //....Si son los dos potenciales....//
                nuevaEnergia += 4 * exilo * (Math.pow((sigma / distance), 12) - Math.pow((sigma / di
            }
```

Figura 5. Uso del patrón para recorrer el listado de moléculas adicionadas al sistema y calcular la energía actual del sistema.

3.2.3 Patrones de Arquitectura.

Un patrón arquitectónico describe a una familia de sistemas informáticos en términos de su organización estructural, describe además componentes y las relaciones entre ellos con las restricciones de la aplicación, la composición asociada y el diseño para su construcción. El patrón empleado en el desarrollo de la presente aplicación es la programación por capas (conocida Arquitectura N Capas), cuyo objetivo primordial es la separación de la lógica del negocio de la lógica de diseño; se ve reflejado en el desarrollo de esta aplicación al separar la capa de implementación de la capa de presentación con la que interactúa el usuario.[44]

Este estilo arquitectónico aplicado al software presenta una ventaja principal y es que su desarrollo se lleva a cabo en tres niveles y, si durante la implementación de sus módulos ocurren cambios, solo se ataca el nivel requerido sin tener que revisar el resto del código asociado.

En este trabajo los componentes del software se definen en tres capas lógicas, las cuales están organizadas de la forma:

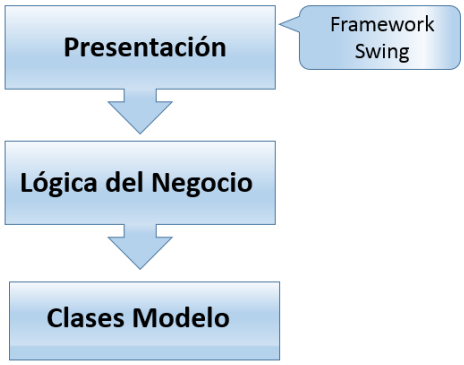

Figura 6. Estructura del diseño de la arquitectura.

1era capa: la capa de presentación, esta capa se encarga de proveer una interfaz entre el sistema y el usuario. Primordialmente, se responsabiliza de que se le comunique información al usuario por parte del sistema y viceversa. El *framework* que opera en esta capa es el *Swing*.

2da capa: lógica de negocio, es la capa que contiene los procesos a realizar con la información recibida a través de la capa de presentación, las peticiones que el usuario realiza, y responsabilizándose de que se le envíen las respuestas adecuadas.

3era capa: clases modelo, en esta capa se encuentran las clases persistentes que encapsulan los objetos del negocio y la implementación de las funcionalidades.

3.3 Diagrama de Paquetes.

Cuando se realiza un software que presenta gran cantidad de clases, dependencias o relaciones, es preciso dividir el proyecto en unidades más pequeñas, como en paquetes, con el objetivo de que las personas puedan trabajar con una cantidad de información limitada a la vez y que el equipo de trabajo no interfiera con el trabajo de los otros. El siguiente diagrama de paquetes del diseño tiene el objetivo de estructurar una visión general, donde las clases del sistema se encuentran agrupadas en cada paquete según el objetivo de su creación, y la estructura organizativa de los paquetes responde a la arquitectura definida para la implementación.

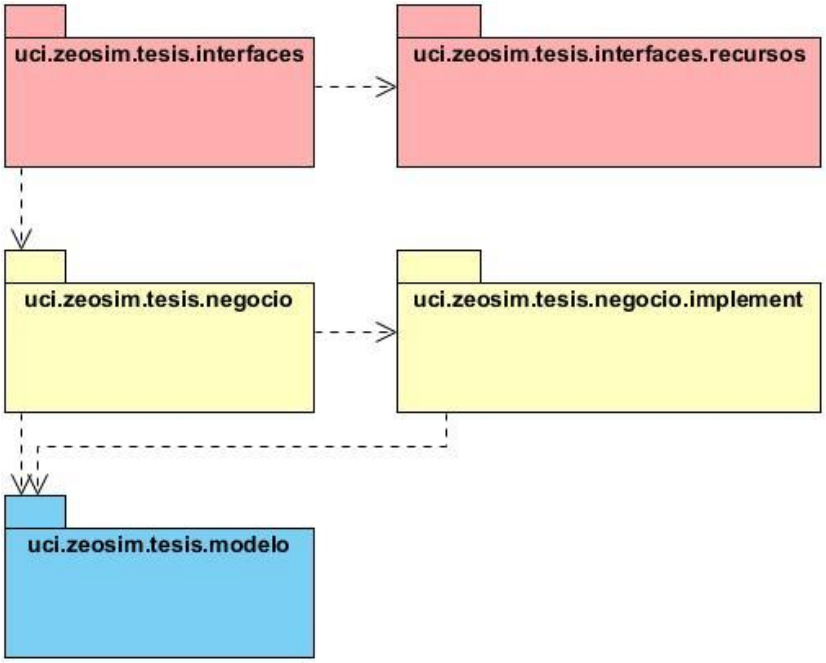

Figura 7. Estructura de los paquetes del diseño.

Paquete uci.zeosim.tesis.interfaces: se encuentran todas las interfaces de usuario.

Paquete uci.zeosim.tesis.interfaces.recursos: se encuentra el mapa de configuración, encargado de gestionar las imágenes que contiene la aplicación.

Paquete uci.zeosim.tesis.negocio: se encuentra la interfaz Servicio, encargada de la comunicación entre las capas Presentación y Lógica del negocio.

Paquete uci.zeosim.tesis.negocio.implement: se encuentra la implementación de la interfaz Servicio.

Paquete uci.zeosim.tesis.modelo: se encuentran todas las clases correspondientes al negocio del sistema.

3.4 Diagrama de clases del diseño.

El diagrama de clases del diseño describe gráficamente las especificaciones de las clases del software y las interfaces de la aplicación, se desarrolla enfocado a la arquitectura del sistema, el siguiente diagrama de clases del diseño corresponde al sistema desarrollado:

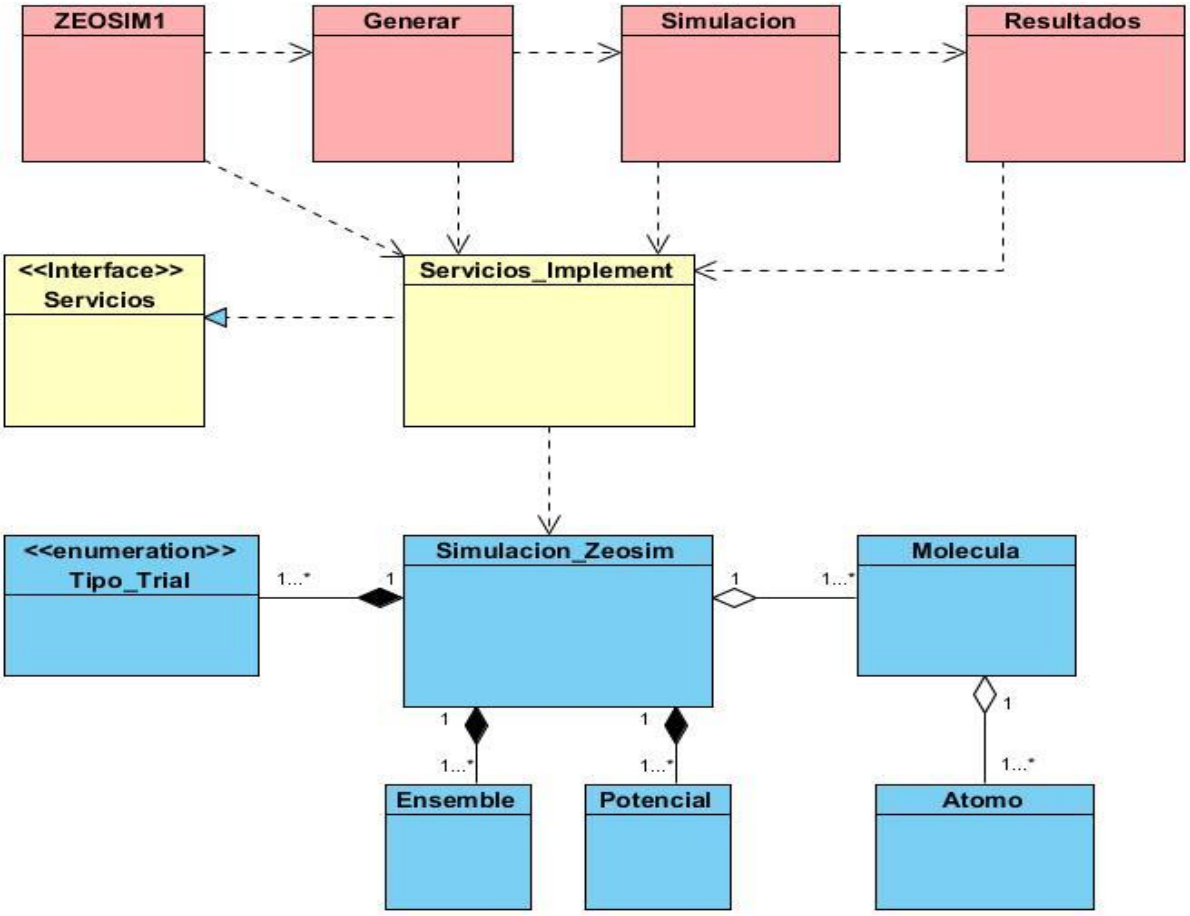

Figura 8. Diagrama de clases del diseño de la aplicación.

Zeosim1: Interfaz principal que permite crear o abrir un experimento.

Generar: Interfaz que permite introducir los datos iniciales del sistema.

Simulación: Interfaz donde se realiza la simulación del proceso.

Resultados: Interfaz que muestra los resultados de la simulación.

Servicios: Conecta las interfaces con las clases necesarias para la obtención de datos.

Servicios_Implement: Implementa la interfaz servicio.

Simulación_Zeosim: Clase principal que ejecuta las funcionalidades del sistema.

3.5 Tarjeta CRC.

Para diseñar un sistema adecuado, eficiente se debe cumplir con tres principios: Cargo o Clase, Responsabilidad y Colaboración (CRC), llamada tarjeta CRC que permite deshacerse del método de trabajo que se basa en procedimientos, y trabajar con una metodología basada en objetos; es la forma más simple de representar las responsabilidades del sistema, además de fácil de visualizar y entender por el usuario y el equipo de desarrollo. El nombre de la clase se coloca como título en la tarjeta, las responsabilidades se ubican a la izquierda, y las clases que se implican en cada responsabilidad van a la derecha, en la misma línea que su requerimiento correspondiente. A continuación la tarjeta CRC que conforma la aplicación:

Simulación_Zeosim	
Responsabilidades	**Colaboración**
CalcularSigmaAtomo	
CalcularExiloAtomo	
CalcularDistanciaAtomo	
CalcularCargaAtomo	
CalcularMasaAtomo	
CalcularNuevaEnergia	Molécula
EnergiaActualDelSistema	
EquilibrarElSistema	Potencial, Átomo
CopiaLista	Molécula, Átomo
RotarMolecula	
CalcularR	
CantidadParticula	
FrecuenciaDeOcurrencia	
SimularProcesoDelSistema	Átomo, TipoTrial
VolumenCaja	
DensidadDelSistema	
PresionDelSistema	
FuerzasDeParticulas	
ComprobarCoordenadas	
TrasladarMolecula	

SepararMoleculas	Molécula, Átomo
AgregarSeparacion	

Tabla 11. Tarjeta CRC

3.6 Conclusiones Parciales.

En este capítulo se precisó y describió el diseño de la aplicación, lo cual permitió obtener una visión avanzada del sistema basado en la comprensión de los requisitos que se establecieron en el capítulo anterior. Se definieron y caracterizaron los patrones de diseño reflejados en la implementación del software, mostrando además la descripción de la tarjeta principal de Clase-Responsabilidad-Colaboración.

Capítulo 4. Implementación y Prueba del Sistema

4.1 Introducción.

En el presente capítulo se describen los elementos necesarios para la implementación, a través del resultado obtenido del diseño. Se realiza el diseño de las pruebas pertinentes, quedando así conformado el modelo de implementación del sistema.

4.2 Implementación.

La implementación es la fase en la cual los desarrolladores escriben el código fuente de las aplicaciones informáticas, y no culmina hasta que todas las funcionalidades hayan sido implementadas y se haya probado su funcionamiento.

- Diagrama de componentes.

Los diagramas de componentes describen los elementos físicos del sistema y sus relaciones. El siguiente diagrama de componentes refleja la integración de cada uno de los elementos que intervienen en la implementación, conteniendo Zeosim.jar todos los paquetes que conforman el sistema.

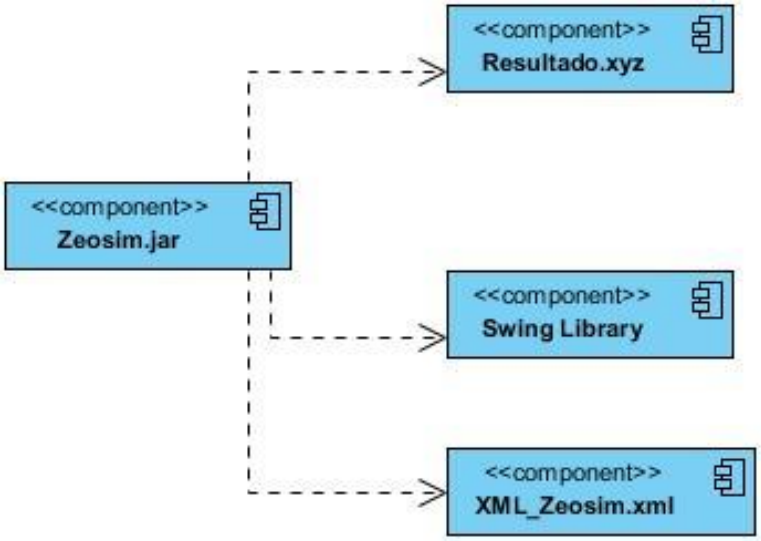

Figura 9. Diagrama de componentes del sistema.

En la fase de planificación fueron descritas las historias de usuarios que se realizarían en cada iteración del proceso de producción. En esta fase se realiza el desarrollo de las mismas, teniendo presente realizar una revisión del plan de iteraciones, haciéndose modificaciones de ser

necesario. Como parte de este plan, se descomponen las historias de usuario en tareas de desarrollo. Estas tareas, son para el uso estricto de los programadores, pueden ser escritas en lenguaje técnico, y no necesariamente descifrables por el cliente, y se denominan: Tareas de Ingeniería. A continuación se muestran las tareas realizadas, separadas mediante las iteraciones en que fueron realizadas cada una:

- Tareas de la Iteración 1:

Tarea de Ingeniería	
Número Tarea: 1	**Número Historia de Usuario:** 1
Nombre Tarea: Crear Experimento.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:**1
Fecha Inicio:	**Fecha Fin:**
Programador Responsable: Yaqueline Mejías De la Torre.	
Descripción: Definir y crear los componentes visuales necesarios para mostrar la información a utilizar y ejecutar el procedimiento de crear un nuevo experimento.	

Tabla 12. Tarea de Ingeniería 1. Crear Experimento.

Tarea de Ingeniería	
Número Tarea: 4	**Número Historia de Usuario:** 4
Nombre Tarea: Adicionar Molécula	
Tipo de Tarea : Desarrollo	**Puntos Estimados:**1
Fecha Inicio:	**Fecha Fin:**
Programador Responsable: Yaqueline Mejías De la Torre.	
Descripción: • Definir una estructura de entidades como soporte para la información solicitada. • Definir y crear los componentes visuales necesarios para almacenar la información relacionada con la molécula y ejecutar el procedimiento de agregar una molécula al proceso de simulación.	

Tabla 13. Tarea de Ingeniería 4. Adicionar Molécula.

Tarea de Ingeniería	
Número Tarea: 7	**Número Historia de Usuario:** 7
Nombre Tarea: Modificar Átomo	

Tipo de Tarea : Desarrollo	**Puntos Estimados:**1
Fecha Inicio:	**Fecha Fin:**
Programador Responsable: Yaqueline Mejías De la Torre.	
Descripción: Realizar una funcionalidad que permita modificar los datos de los átomos que contiene cada molécula según la elección realizada por el usuario. Tener en cuenta para ello la selección de la molécula, cuyos datos de átomos desea modificar.	

Tabla 14. Tarea de Ingeniería 7. Modificar Átomo.

Tarea de Ingeniería	
Número Tarea: 8	**Número Historia de Usuario:** 8
Nombre Tarea: Otorgar Dimensiones de la Caja de Simulación	
Tipo de Tarea : Desarrollo	**Puntos Estimados:**1
Fecha Inicio:	**Fecha Fin:**
Programador Responsable: Yaqueline Mejías De la Torre.	
Descripción: Permitir al usuario introducir los valores de dimensión para la caja de simulación del sistema, que serán guardados en variables creadas como atributos en la clase principal de la aplicación.	

Tabla 15. Tarea de Ingeniería 8. Otorgar Dimensiones de la Caja de Simulación.

Tarea de Ingeniería	
Número Tarea: 9	**Número Historia de Usuario:** 9
Nombre Tarea: Separar Moléculas	
Tipo de Tarea : Desarrollo	**Puntos Estimados:**1
Fecha Inicio:	**Fecha Fin:**
Programador Responsable: Yaqueline Mejías De la Torre.	
Descripción: Realizar una funcionalidad que permita, dado el valor que introduce el usuario de separación, separar las moléculas que se adicionaron al proceso de simulación, el valor debe ser mayor o igual a 3.0 Angstrom.	

Tabla 16. Tarea de Ingeniería 9. Separar Moléculas.

- Tareas de la Iteración 2:

Tarea de Ingeniería

Número Tarea: 11	**Número Historia de Usuario:** 11
Nombre Tarea: Adicionar Potencial	
Tipo de Tarea : Desarrollo	**Puntos Estimados:**1
Fecha Inicio:	**Fecha Fin:**
Programador Responsable: Yaqueline Mejías De la Torre.	
Descripción: Crear una funcionalidad que permita adicionar uno o más potenciales para su utilización en el proceso de simulación. Su objetivo es la aplicación del cálculo que establece cada potencial para hallar una energía en el sistema molecular.	

Tabla 17. Tarea de Ingeniería 11. Adicionar Potencial.

Tarea de Ingeniería	
Número Tarea: 13	**Número Historia de Usuario:** 13
Nombre Tarea: Adicionar Ensemble	
Tipo de Tarea : Desarrollo	**Puntos Estimados:**1
Fecha Inicio:	**Fecha Fin:**
Programador Responsable: Yaqueline Mejías De la Torre.	
Descripción: Permitir al usuario seleccionar el ensemble para realizar la simulación con el parámetro establecido según el ensemble seleccionado.	

Tabla 18. Tarea de Ingeniería 13. Adicionar Ensemble.

Tarea de Ingeniería	
Número Tarea: 14	**Número Historia de Usuario:** 14
Nombre Tarea: Seleccionar Transacciones	
Tipo de Tarea : Desarrollo	**Puntos Estimados:**1
Fecha Inicio:	**Fecha Fin:**
Programador Responsable: Yaqueline Mejías De la Torre.	
Descripción: Permitir al usuario seleccionar los eventos que desea realice el sistema durante el proceso de simulación de las moléculas (Trasladar, Rotar, Enlazar). En esta funcionalidad influye el cálculo de las probabilidades que tiene cada molécula de realizar dichos eventos.	

Tabla 19. Tarea de Ingeniería 14. Seleccionar Transacciones.

Tarea de Ingeniería

Número Tarea: 15	**Número Historia de Usuario:** 15
Nombre Tarea: Asignar Temperatura al Sistema	
Tipo de Tarea : Desarrollo	**Puntos Estimados:**1
Fecha Inicio:	**Fecha Fin:**
Programador Responsable: Yaqueline Mejías De la Torre.	
Descripción: Permitir al usuario otorgar un valor de temperatura para el sistema, este valor determina la temperatura que tendrá el sistema durante el proceso de simulación.	

Tabla 20. Tarea de Ingeniería 15. Asignar Temperatura al Sistema.

Tarea de Ingeniería	
Número Tarea: 18	**Número Historia de Usuario:** 18
Nombre Tarea: Simular Proceso	
Tipo de Tarea : Desarrollo	**Puntos Estimados:**3
Fecha Inicio:	**Fecha Fin:**
Programador Responsable: Yaqueline Mejías De la Torre.	
Descripción: Implementar una funcionalidad que realice la simulación del sistema, teniendo en cuenta la probabilidad de cada molécula a realizar alguno de los eventos seleccionados por el usuario	

Tabla 21. Tarea de Ingeniería 18. Simular Proceso.

- Tareas de la Iteración 3:

Tarea de Ingeniería	
Número Tarea: 19	**Número Historia de Usuario:** 19
Nombre Tarea: Mostrar Resultados	
Tipo de Tarea : Desarrollo	**Puntos Estimados:**2
Fecha Inicio:	**Fecha Fin:**
Programador Responsable: Yaqueline Mejías De la Torre.	
Descripción: Desarrollar la funcionalidad que muestre los resultados obtenidos luego de realizar la simulación. Los resultados muestran las posiciones finales que obtiene cada molécula al ser simulado el sistema. Se incluyen los cálculos de las propiedades físicas del sistema, para cada propiedad se debe realizar una funcionalidad que calcule dichos parámetros, tales como la	

Densidad del sistema, la Energía final, el Volumen de la caja de simulación, la Presión del sistema, así como la duración real del proceso de simulación.

Tabla 22. Tarea de Ingeniería 19. Mostrar Resultados.

- Estándares de Codificación.

Al comenzar un software, es necesario establecer un estándar de codificación para asegurarse de que todos los programadores del proyecto trabajen de forma conjugada, de esta forma se logra que todo el código de la aplicación sea legible y equivalente.

Un estándar de codificación es un conjunto de reglas de notación y nomenclatura, específicas de cada lenguaje de programación, que se usan y se siguen durante la fase de implementación de una aplicación, y reducen el riesgo de que los desarrolladores introduzcan errores que no son detectados por los compiladores, minimizando así el tiempo y coste de las actividades de depuración y pruebas necesarias para la detección y corrección de los mismos.[45]

Mediante un estándar de codificación se logra que el código se encuentre en correcto estado y que cualquier desarrollador del equipo pueda modificar cualquier parte del código. En la siguiente tabla se especifican las convenciones de nombres empleadas:

Tipos de Identificadores	Reglas para Nombres	Ejemplos
Clases	Los nombres de las clases comenzarán siempre con letra mayúscula, incluyendo nombre de clases compuesto, siendo éstas últimas separadas por un guión bajo (“_”).	class Potencial class Simulación_Zeosim
Variables	Las variables no tienen restricciones de escritura, ya que pueden ser variables, atributos o instancias de clases, y comenzar en mayúscula o minúscula, dependiendo de la decisión del programador.	double energiaSistema; double num_aleatorio;
Listas	Las listas son de tipo LinkedList<>() y se escribirán siempre con minúscula y cada palabra que contenga el nombre separada por un guión bajo.	LinkedList<Molecula>lista_moleculas LinkedList<TipoTrial>transaccion

Métodos	Los métodos comenzarán siempre con mayúsculas, incluyendo si son nombres compuestos.	CalcularDistanciaAtomo() SumaDeEnergias()

Tabla 23. Tablas de Estándares de Codificación.

4.3 Prueba.

Las pruebas son uno de los pilares de XP siendo la última fase que define esta metodología, constituyen una etapa dentro del desarrollo del software que permiten comprobar y revelar la calidad de un producto final. Son utilizadas para identificar fallos en la implementación o usabilidad del programa.

Las pruebas del sistema, XP las separa en dos grupos diferentes: pruebas unitarias, que se encargan de verificar el código y son diseñadas por los programadores; y las pruebas de aceptación destinadas a evaluar si al finalizar una iteración se logró la funcionalidad requerida, y son diseñadas por el cliente final. Las pruebas del sistema tienen como objetivo verificar que el sistema cumple los requisitos establecidos por el usuario por lo que también pueden enlazar dentro de la categoría de pruebas de aceptación.

- **Pruebas Unitarias.**

El objetivo de las pruebas unitarias es aislar cada parte del software y mostrar que cada funcionalidad implementada está correcta, es una forma de probar el correcto funcionamiento de un módulo determinado de código. A continuación se ejemplifican estas pruebas aplicadas a varias funcionalidades del software durante su implementación.

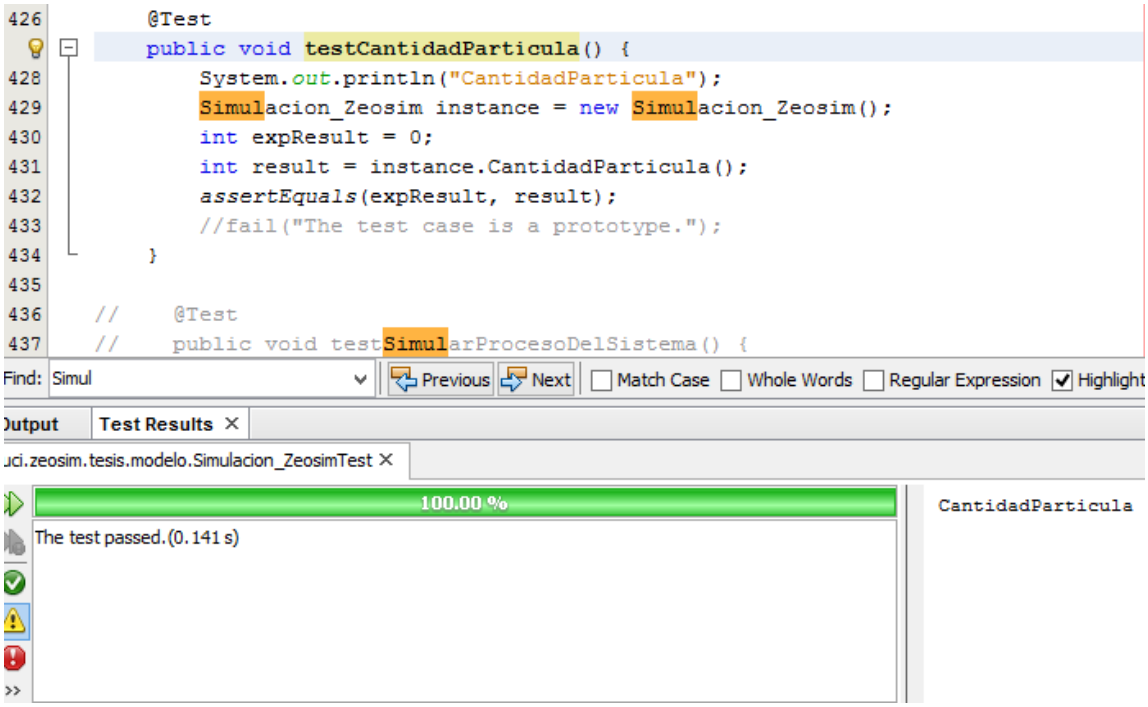

Figura 10. Prueba Unitaria Cantidad de partículas.

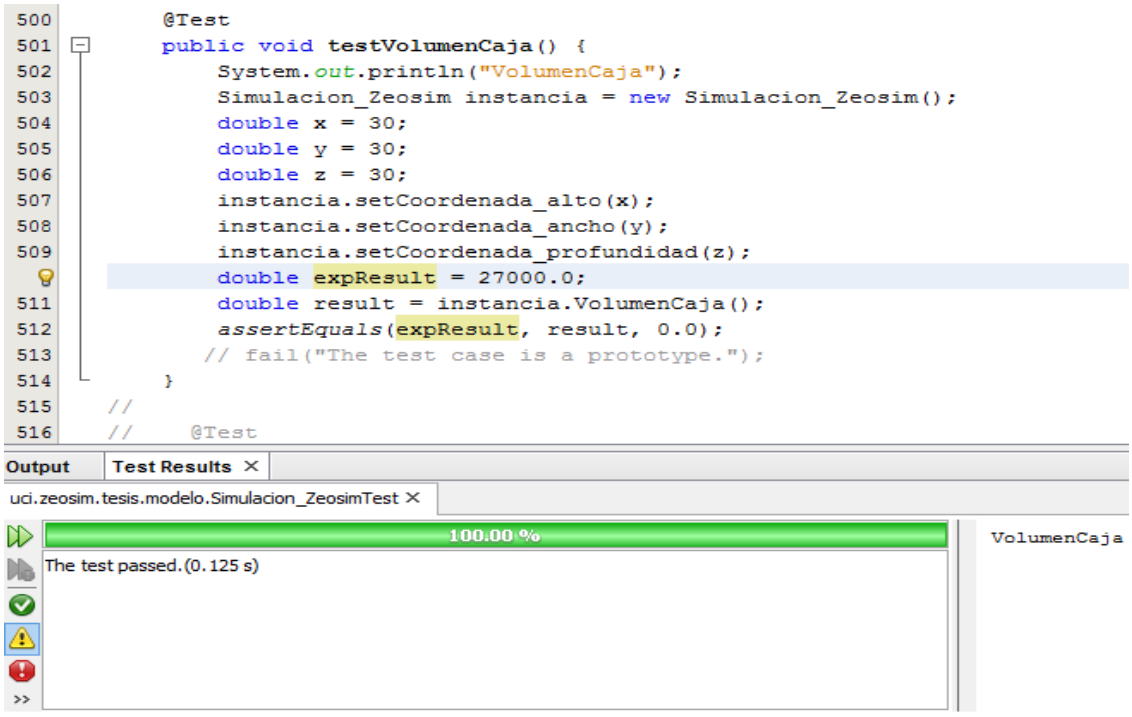

Figura 11. Prueba Unitaria Volumen de la caja.

```
365         @Test
366         public void testCalcularNuevaEnergia() {
367             System.out.println("CalcularNuevaEnergia");
368             Molecula agua = new Molecula();
369             Atomo H = new Atomo();
370             H.setCarga(1.2);
371             H.setMasa(12.3);
372             H.setEnergia(1.76);
373             H.setSigma(0.21);
374             H.setCoord_X(2.123);
375             H.setCoord_Y(1.123);
376             H.setCoord_Z(3.123);
377             LinkedList<Atomo> lista = new LinkedList<Atomo>();
```

Output | Test Results ×

uci.zeosim.tesis.modelo.Simulacion_ZeosimTest × | Rerun failed tests ×

100.00 %

The test passed. (0.119 s)

Figura 12. Prueba Unitaria Calcular nueva energía.

```
591         @Test
592         public void testTrasladarMolecula() {
593             System.out.println("TrasladarMolecula");
594              Molecula agua = new Molecula();
595             Atomo H = new Atomo();
596             H.setCarga(1.2);
597             H.setMasa(12.3);
598             H.setEnergia(1.76);
599             H.setSigma(0.21);
600             H.setCoord_X(2.123);
601             H.setCoord_Y(1.123);
```

Output | Test Results ×

uci.zeosim.tesis.modelo.Simulacion_ZeosimTest × | Rerun failed tests ×

100.00 %

The test passed. (0.124 s)

Figura 13. Prueba Unitaria Trasladar molécula.

❖ **Pruebas de Aceptación.**

Las pruebas de aceptación son creadas a partir de la realización de las historias de usuario. Durante cada iteración, la historia de usuario escogida en la planificación de iteraciones se convertirá en una o varias pruebas de aceptación; y el usuario especifica los aspectos a probar cuando una historia de usuario ha sido correctamente implementada.

Las pruebas de aceptación son más importantes que las pruebas unitarias, pues significan la satisfacción del cliente con el producto desarrollado, el final de una iteración y comienzo de la próxima; es por ello que el cliente es la persona indicada para diseñar dichas pruebas. El objetivo de estas pruebas es comprobar los requerimientos, siendo éstos la principal fuente de información para realizar las pruebas de aceptación, y la publicación de los resultados debe ser lo más rápido posible, para que los desarrolladores puedan realizar con rapidez los cambios que sean necesarios.

❖ **Diseño de Casos de Prueba.**

A continuación se muestran los diseños de casos de prueba realizados en el sistema:

Caso de Prueba de Aceptación	
Código caso de prueba:3	**Nombre HU:** Gestionar Molécula
Nombre de la persona que realiza la prueba: Yaqueline Mejías De la Torre.	
Descripción de la prueba: Se debe comprobar que se adicione una molécula, así como modificar la cantidad de la misma a utilizar, o eliminarla. Si se adiciona nuevamente la misma molécula, el sistema muestra un mensaje con el tipo de información que se requiere para corregir el error. En el caso de estar correcto estos datos, se guardan en el sistema para su uso en el transcurso de la aplicación.	
Condiciones de ejecución: ➢ La aplicación no puede tener errores durante su compilación.	
Entrada / Pasos de ejecución: **Escenario: Crear Molécula.** Se selecciona la opción Crear Molécula del Menú principal de acciones, y el usuario puede	

introducir los datos de la nueva molécula que desea adicionar al sistema. **Escenario: Adicionar Molécula.** Se selecciona la molécula, mostrándose un listado con los nombres de las mismas, y posteriormente se preciosa el botón Adicionar, quedando agregada al proceso en cuestión. **Escenario: Modificar Molécula.** La opción de modificar la molécula es para modificar la cantidad de moléculas que el usuario desea tener en el sistema, para ello se selecciona la molécula mostrada en el listado y se modifica la cantidad del tipo de molécula seleccionada. **Escenario: Eliminar Molécula.** Se selecciona la molécula que se desea eliminar y se procede a presionar el botón Eliminar para borrarla del listado de moléculas adicionadas.
Resultado esperado: **Escenario: Crear Molécula.** En el listado de moléculas del sistema le aparecerá la nueva molécula creada, permitiéndole al usuario su selección y posterior uso en el proceso de simulación. **Escenario: Adicionar Molécula.** En una tabla presentada en la interfaz se muestra la molécula añadida. **Escenario: Modificar Molécula.** En la tabla presentada en la interfaz de modifica la cantidad del tipo de molécula seleccionada, y quedan agregadas en el sistema dicha cantidad de la molécula especificada. **Escenario: Eliminar Molécula.** Se muestra un mensaje de confirmación.
Evaluación de la Prueba: Satisfactoria

Tabla 24. Caso de Prueba de Aceptación: Gestionar Molécula.

Caso de Prueba de Aceptación	
Código caso de prueba:8	**Nombre HU:** Gestionar Potencial
Nombre de la persona que realiza la prueba: Yaqueline Mejías De la Torre.	
Descripción de la prueba: Se debe comprobar que se adicione un potencial, así como eliminarlo luego de su selección si el usuario lo necesita. Si se adiciona nuevamente el mismo potencial, el sistema muestra un mensaje con la información al usuario para corregir el error. En el caso de estar correcto estos datos, se guardan en el sistema para su uso en el transcurso de la aplicación.	
Condiciones de ejecución:	

➢ La aplicación no puede tener errores durante su compilación.
Entrada / Pasos de ejecución: **Escenario: Adicionar Potencial.** Se selecciona el potencial, mostrándose un listado con los nombres de las mismas, y posteriormente se preciosa el botón Adicionar, quedando agregada al proceso en cuestión. **Escenario: Eliminar Potencial.** Se selecciona el potencial que se desea eliminar y se procede a presionar el botón Eliminar para borrarlo del listado de potenciales agregados.
Resultado esperado: **Escenario: Adicionar Potencial.** En una tabla presentada en la interfaz se muestra el potencial seleccionado. **Escenario: Eliminar Potencial.** El potencial deja de estar presente en la tabla de adicionados.
Evaluación de la Prueba: Satisfactoria

Tabla 25. Caso de Prueba de Aceptación: Gestionar Potencial.

Caso de Prueba de Aceptación	
Código caso de prueba:13	**Nombre HU:** Simular Proceso.
Nombre de la persona que realiza la prueba: Yaqueline Mejías De la Torre.	
Descripción de la prueba: Se realiza el proceso de simulación, para ello se debe verificar que el usuario haya agregado al menos un potencial, que haya seleccionado al menos una transacción, y también debe introducir un valor de temperatura.	
Condiciones de ejecución: ➢ La aplicación no puede tener errores durante su compilación.	
Entrada / Pasos de ejecución: Se oprime el botón Simular proceso, el usuario introduce el tiempo que desea dure la simulación, y posteriormente se mostrará un mensaje al concluir dicho proceso.	
Resultado esperado: Se muestra la opción de visualizar los resultados obtenidos tras realizar la simulación.	

Evaluación de la Prueba: Satisfactoria

Tabla 26. Caso de Prueba de Aceptación: Simular Proceso.

4.4 Resultados de las Pruebas.

Como resultado de las pruebas realizadas, en la primera etapa de pruebas se le aplicaron 3 iteraciones y se detectaron 8 no conformidades en la primera procediendo todas a ser modificadas, 4 en la segunda procediendo cada una a su corrección y 1 en la última no procedida, la siguiente figura muestran dichos resultados.

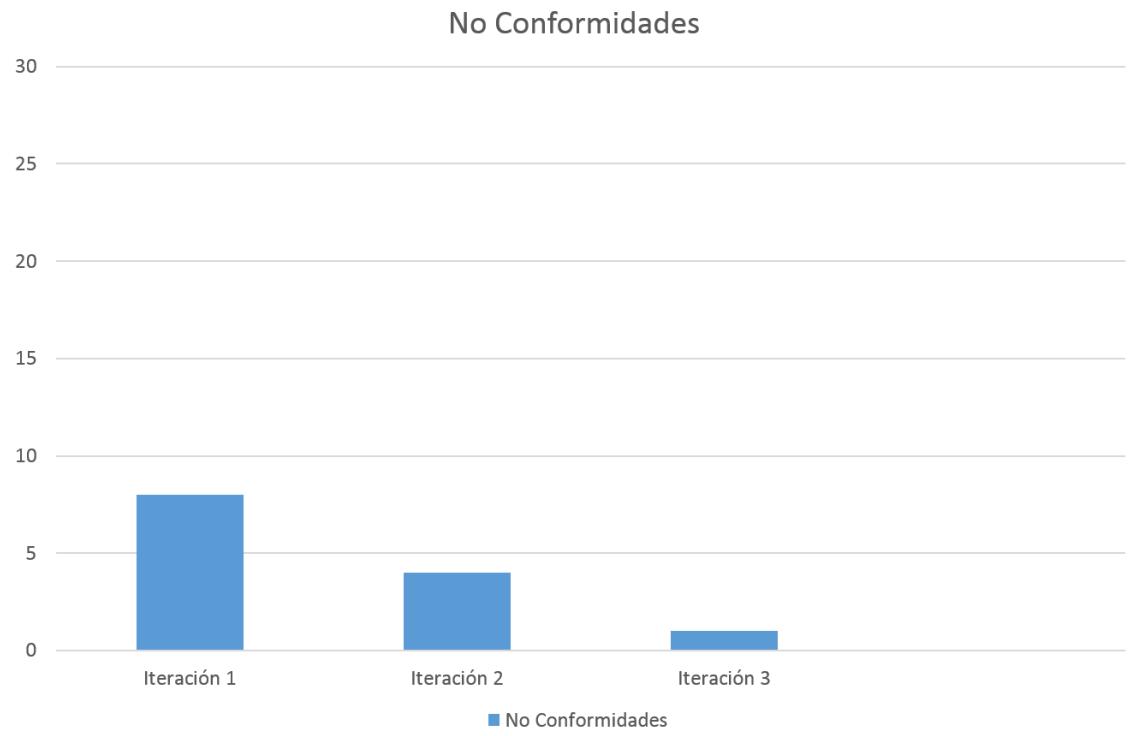

Figura 14. No conformidades Iteración 1.

En la segunda etapa se probaron todas las funcionalidades del sistema, los resultados obtenidos fueron: 14 no conformidades para la primera iteración, 7 en la segunda y 2 en la última iteración.

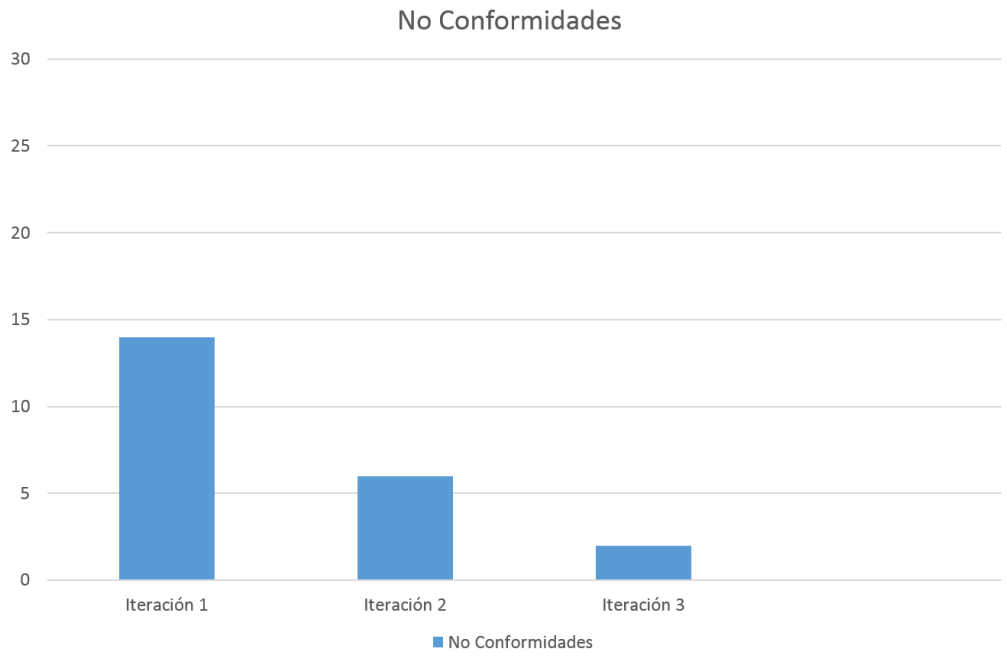

Figura 15. No conformidades Iteración 2.

En la tercera etapa quedaron aprobadas las funcionalidades del sistema, los resultados obtenidos fueron: 9 no conformidades para la primera iteración, 5 en la segunda donde se procedió a modificar todas las no conformidades, y resultó 0 no conformidades en la última iteración.

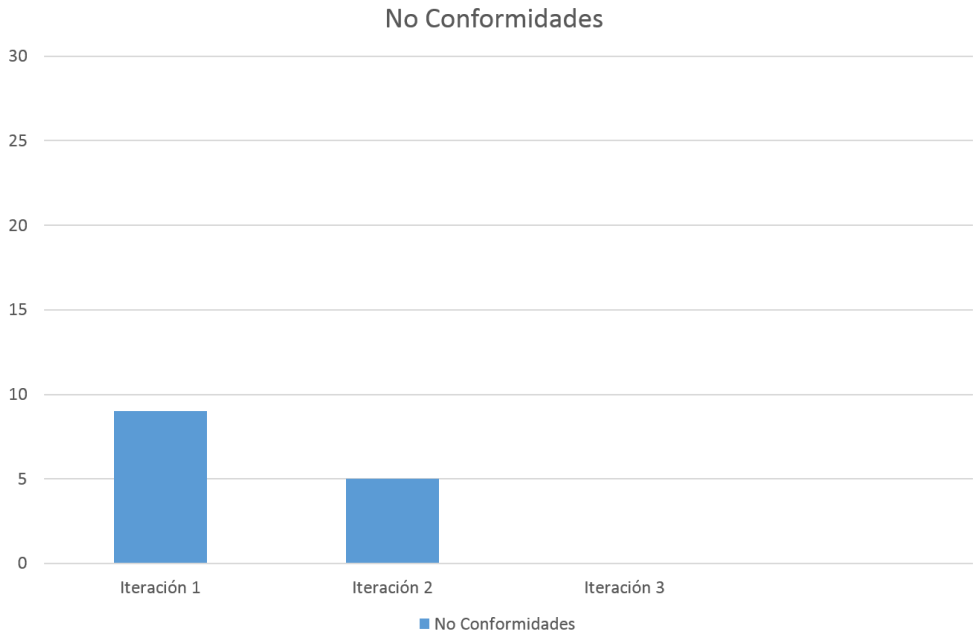

Figura 16. No conformidades Iteración 3.

Durante el proceso de ejecución de pruebas al sistema, fue realizada una valoración sobre el requerimiento de hardware que necesita el software. Para esta prueba, se realizaron diferentes corridas con un sistema de 980, 4900 y 1 millón de partículas respectivamente, con un tiempo de duración de 4 minutos dado por el usuario. Midiendo el uso de memoria RAM se dividieron los procesos del sistema en tres etapas: Generar el sistema inicial, Equilibrar proceso y Simular proceso. La siguiente gráfica muestra los resultados obtenidos, con la cual se valida la necesidad de una memoria RAM de 1Gb para un mejor desempeño del software.

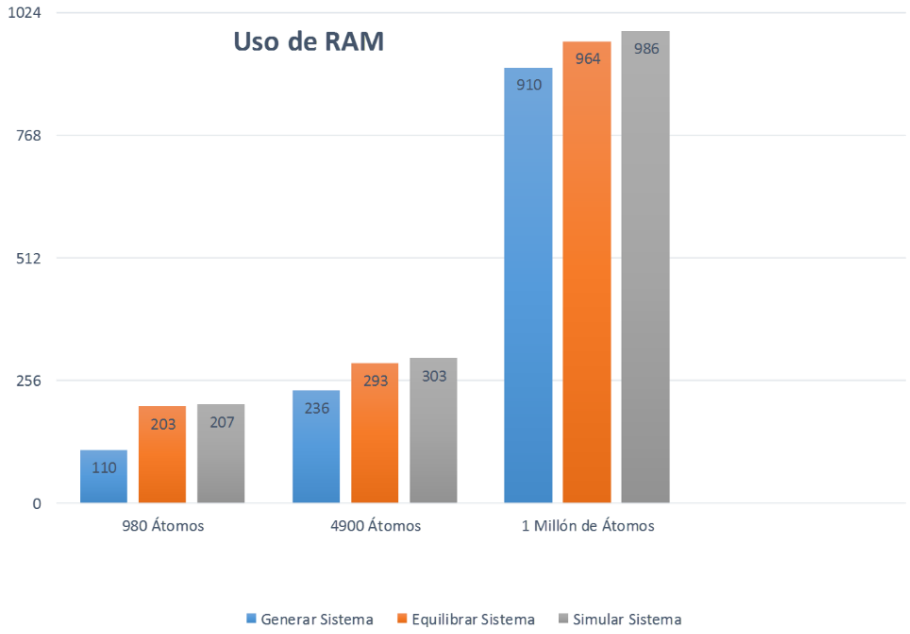

Figura 17. Uso de la memoria RAM durante el uso del software.

4.5 Conclusiones Parciales.

En este capítulo quedó conformado el modelo de implementación, basada en el diseño descrito, estableciendo casos de pruebas que le permite al usuario validar el funcionamiento de las funcionalidades. Se obtuvieron los artefactos propuestos por la metodología XP para la etapa de producción, demostrando el desarrollo satisfactorio del software en el período de tiempo establecido. Se definieron los estándares de codificación puestos en práctica en la implementación Además, se pudo verificar que el software no presentaba errores al ser satisfactorias las pruebas aceptación del cliente y las pruebas desarrolladas por el equipo de trabajo, arrojando excelentes resultados por cada una de las pruebas.

Conclusiones Generales

Con el desarrollo de este trabajo se profundizó en el conocimiento de los materiales porosos tipo zeolitas vinculadas a diferentes ramas de la industria. Se aplicó la metodología XP para guiar el proceso de desarrollo de software, mediante el cual se obtuvo la implementación de un programa de interfaz para desarrollar simulaciones de procesos de síntesis de estos materiales a través de la técnica de Montecarlo Cinético. El diseño y la implementación se rigieron por herramientas, tales como NetBeans 7.3 y el lenguaje Java para la programación. Se puede concluir que se ha cumplido satisfactoriamente el objetivo general trazado para este trabajo, enfatizando en los siguientes puntos:

- Se implementó una interfaz que permite simular procesos de síntesis de materiales porosos tipo Zeolitas, utilizando la técnica computacional de Montecarlo Cinético, y que puede ser utilizada por especialistas teóricos o no, sin necesidad de tener amplios conocimientos sobre técnicas de simulación computacional.
- Se logró, con el diseño de la aplicación, ahorro en tiempo y recursos para las personas que utilicen el software.
- Correcto manejo de las interfaces de la aplicación, de modo que se le puedan integrar nuevas aplicaciones.
- El desarrollo de esta herramienta significa, además, un aporte práctico trascendental, debido a que constituye una novedad tecnológica que marca un paso de avance en el desarrollo de software multiplataforma.
- Se evaluó la solución desarrollada mediante el uso de pruebas de aceptación dirigidas por el usuario y las pruebas unitarias realizadas por el equipo de desarrollo, arrojando resultados satisfactorios.

Recomendaciones

A partir de los resultados obtenidos en la investigación realizada durante la elaboración de este trabajo, y con la finalidad de asegurar la posterior ampliación, modificación y mejora dela aplicación propuesta, se exponen algunas recomendaciones:

- Implementar otras técnicas de simulación computacional que puedan integrarse a la aplicación.
- Introducir la utilización de otros potenciales y ensembles de simulación para aumentar los parámetros involucrados en el proceso de simulación del sistema.
- Extender la simulación computacional de procesos de síntesis a otros tipos de materiales diferentes a la zeolita.

Bibliografía

1. Olguín Gutiérrez M. T. “Zeolitas características y propiedades”. Instituto Nacional de Investigaciones Nucleares, Depto. de Química, A. P. 18-1027, Col. Escandón, Delegación Miguel Hidalgo, C. P. 11801, México, D. F., México.
2. Bosch, Pedro. Schifter, Isaac, LA ZEOLITA. Una piedra que hierve. Capítulo 3. USOS DE LAS ZEOLITAS. Disponible en: http://bibliotecadigital.ilce.edu.mx/sites/ciencia/volumen1/ciencia2/55/htm/sec_5.html
3. Canós, José H., Letelier, Patricio y Penadés, Mª Carmen. Metodologías Ágiles en el Desarrollo de Software. DSIC -Universidad Politécnica de Valencia.
4. Marchesi, Michele., Succi, Giancarlo., Williams, Laurie. Extreme Programming Perspectives. Addison Wesley, ISBN: 0-201-77005-9.
5. Simulación molecular de la adsorción de hidrocarburos en zeolitas con cationes de intercambio. Disponible en: http://dialnet.unirioja.es/servlet/articulo?codigo=1996644
6. El Lenguaje de Modelado Unificado UML: Disponible en: http://www.docirs.cl/uml.htm
7. Proceso Unificado de Rational. Disponible en:
8. PROGRAMACION EXTREMA XP. Disponible en: http://www.ingenieriadesoftware.mex.tl/52753_XP---Extreme-Programing.html
9. Visual Paradigm for UML (ME). Disponible en: http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%28M%C3%8D%29_14720_p/
10. J Rumbaugh, I Jacobson, G Booch, Libro El lenguaje unificado de modelado. 1999 - en.scientificcommons.org
11. Open PaperOpt: A Monte Carlo Simulation Tool for Simulating Light Scattering in Paper and Print Sundsvall, 2010. Digital Printing Centre / FSCN – Fibre Science and Communication Network, Mid Sweden University.
12. Schildt, Herbert. The Complete Reference, Java 2. Fifth Edition.
13. Wesley, Addison. - Java Tutorial 3th Edition - A Short Course on the Basics. 2000.
14. Pressman, Roger.Ingenieria.del.Software.-6th.Ed.McGraw-Hill.
15. Mendoza, Sanchez, Maria A. Metodologías De Desarrollo De Software. 2002 Grupo Informatizate.

16. Rodríguez Corbea, Maite, Ordóñez Pérez, Meylin. LA METODOLOGÍA XP APLICABLE AL DESARROLLO DEL SOFTWARE EDUCATIVO EN CUBA. Trabajo de Diploma para optar por el título de Ingeniería en Ciencias Informáticas. 2007.
17. De la Madrid Tejeda, Alejandro. Zeolitas e Insumos Nacionales S.A, 1989. Veracruz, México. Disponible en: http://www.zeolitas.info/index.htm.
18. W. Smith, T.R. Forester and I.T. Todorov. THE DL POLY 2 USER MANUAL. STFC Daresbury Laboratory, Daresbury, Warrington WA4 4ADCheshire, UK

Referencias

1. N. Metropolis, A.W.R., M. N. Rosenbluth, A. H. Teller, and E. Teller, *Equation of State Calculations by Fast Computing Machines.* J.Chem. Phys. 21: p. 1087.

2. Su, Z.-Y.Y.a.B.-L., *Insights into hierarchically meso–macroporous structured materials.* Journal of Materials Chemistry.

3. Frias, D.R. *Molecular Dynamics Simulation of Self-Assembly of n-Decyltrimethylammonium Bromide Micelles*. March 5, 2008 [cited.

4. Ateeque Malani, S.M.A.a.P.A.M., *Monte Carlo Simulations of Silica Polymerization and Network Formation.* Physical Chemistry.

5. Mark W. Ackley, S.U.R., Himanshu Saxena, *Application of natural zeolites in the purification and separation of gases*. p. 25-42.

6. Falk, M.L., *Atomlab: Una herramienta para la enseñanza de ciencias materiales y simulacion en escala atómica.* Journal of Materials Education, 2004. 26: p. 235-242.

7. *Historia de las Zeolitas, Departamento de Ciencias Geológicas, UBA.* [cited; Available from: http://zeolitas.gl.fcen.uba.ar/historia.htm.

8. Angel Rabdel Ruíz Salvador, A.G.G., Anabel Lam Barandela *Propiedades estructurales de materiales zeolíticos: estudios vía simulación computacional*.

9. Rivera, D.A., *Desarrollo, Diseño y Caracterización de Materiales Zeolíticos y Porosos de Interés Farmacéutico y Medioambiental.*

10. *De la Madrid Tejeda, Alejandro. 1989 Zeolitas e Insumos Nacionales S.A* [cited; Available from: http://www.zeolitas.info/index.htm.

11. *Departamento de Investigación en Zeolitas ICUAP*: Complejo de Ciencias, Ciudad Universitaria Puebla, México.

12. Dra. Lorna Guerrero, I.D.C. *PROYECTO USM 27.08.13*. [cited; Available from: http://www.dgip.utfsm.cl/publico/Proy2008/pqu08.html#pag1.

13. Pavelic K, S.B., Colic M. En: editors. 2001. pp., *Zeolites and mesoporous materials at the dawn of the 21st Century*, in *Studies in surface science and catalysis*, D.R.F. Galarneau A, Fagula F, Vedrine J, Editor. 2001, Elsevier Science: Amsterdam p. 7.

14. Rivera A, R.-F.G., Altshuler E., *Time evolution of a natural Clinoptilolite in aqueous medium: Conductivity and pH experiments. .*Micropor Mesopor Mat, 2000. 40: p. 173.

15. *Uso de zeolitas cubanas mejoran rendimientos agrícolas*. Juventud Rebelde.

16. Shannon, R.J., James D., *Systemssimulation: the art and science». IEEE TransactionsonSystems.*, ed. M.a. Cybernetics6 (10). 1976.

17. Heerman, D.W. (1986) *Computer Simulation Methods in Theoretical Physics*. Springer-Verlag Volume,

18. M. P. Allen and D. Tildesley, *Computer Simulations of Liquids,* 1987.

19. Hansen, J.P., *An introduction to Molecular Dynamics with an applicaction to glass transition,on Computer Simulation in Materials Science*, M.M.a.V. Pontikis, Editor. 1991, Kluwer Academic.

20. Haile, J.H., *Molecular Dynamics Simulation*. 1992, J. Wiley: New York.

21. D. E. Rappaport, *The Art of Molecular Dynamics Simulation*, C.U. Press, Editor. 1996.

22. Coddington, P.D., *Analysis of Random Number Generators Using Monte Carlo Simulation*. October 14, 1993, Northeast Parallel Architectures Center, Syracuse University.

23. Tildesley, M.P.A.a.D.J., *Computer Simulation of Liquids*. 1987, New York: Oxford University Press.

24. S. Glasstone, K.J.L., and H. Eyring, *The Theory of Rate Processes*. McGraw-Hill. 1941, New York.

25. Vineyard., G.H., *Frequency factors and isotope effects in solid state rate processes.* .J.Phys. Chem. Solids, 1957. 3: p. 121.

26. Reif, F., *Fundamentos de física estadística y térmica.*1967: Madrid (Espaha). Mcgraw-Hill.

27. *CRYSTAL: software for guided crystal structure analysis.* Journal of Applied Crystallography, December, 2003. Volume 36, Part 6.

28. Coppel, L.G., *Open Source Monte Carlo Simulation Plataform For Particle Level Simulation Of Ligth Scattering From Generated Paper Structures*. June 2009.

29. Youngs, D.T.G.A., *A Molecular Dynamics Study of Glucose Solvation in the Ionic Liquid.* Journal of Polymer Science. Volume 7(Issue 11): p. 2279–2281.

30. Roberth G. Figueroa, C.J.S., Armando A. Cabrera *METODOLOGÍAS TRADICIONALES VS. METODOLOGÍAS ÁGILES*. Volume,

31. Penadés, P.L.y.M.C., *Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP)*: Facultad de Informática. Universidad Politécnica de Valencia.

32. Orallo, E.H., *El Lenguaje Unificado de Modelado (UML)*. 2002.

33. James Rumbaugh, I.J., Grady Booch, *The Unified Modeling Language Reference Manual*. 1999: Addison Wesley Longman, Inc.

34. Remco M. Dijkmana, M.D., Chun Ouyangc, *Semantics and analysis of business process models in BPMN.*Information and Software Technology., November 2008. Volume 50(Issue 12): p. 1281–1294.

35. Caballero, A.V.F.Ó.G.I., *Una Herramienta CASE para ADOO: Visual Paradigm Análisis y Diseño Orientado a Objetos*, in *Prácticas Ingeniería del Software*.

36. JAVA, S.O. *Sitio Oficial JAVA*. 2009 [cited 23 de 3 de 2012]; Available from: http://java.com/es.

37. Aumaille, B.s.l.E., *JAVA 2.* 2000.

38. *GSInnova. 2007. GSInnova* 2007.

39. *Frameworks*. [cited; Available from: http://www.lsi.us.es/~javierj/investigacion_ficheros/Framework.pdf.

40. Johnson, R.H., Jürgen; Arendsen, Alef; Risberg, Thomas; Sampaleanu, Colin, Wrox, *Professional Java Development with the Spring Framework*. First Edition ed. July 8, 2005.

41. *IEEE: Standard Glossary of Software Engineering Terminology*.

42. Duarte, A.O., *The Methodologies of Agile Development like an Opportunity for the Engineering of Educative Software.*, in *Avances en Sistemas e Informatica*. Junio, 2008, Grupo de Investigacion CICOM: Universidad de Pamplona, Colombia.

43. Craig., L., *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* Primera edición ed. 1999. Pág. 187.

44. Pressman, R.S., *Ingeniería del Software.* Sexta Edición ed.

45. Charte, F., *Visual C#.Net.* 2002, Anaya Multimedia, SA.

Glosario de Términos

XP: Extreme Programming.

UML: Lenguaje de Modelado Unificado.

HU: Historias de Usuario.

Montecarlo Cinético: Es un método estocástico que permite resolver problemas físicos y matemáticos mediante la simulación de variables aleatorias dependientes del tiempo.

GRASP: En diseño orientado a objetos, GRASP son patrones generales de software para asignación de responsabilidades, sus siglas significan "General Responsibility Assignment Software Patterns".

Software: Conjunto de programas, instrucciones y reglas informáticas para ejecutar tareas en un ordenador.

Simulación: Es una presentación informática para modelar situaciones de la vida real o el funcionamiento de un sistema cuando evoluciona en el tiempo, por medio de un programa de computadora.

Hardware: Conjunto de los componentes que integran la parte material de una computadora.

Código Abierto: Es una tendencia internacional del desarrollo de software que sigue la distribución del código junto a las aplicaciones, se rigen por licencias tales como GNU/GPL.

Framework: En el desarrollo de software, constituye una estructura de soporte para organizar y desarrollar una aplicación. Puede incluir soporte de programas, bibliotecas, entre otros sistemas para ayudar a desarrollar y unir los diferentes componentes de un proyecto.