Facultad de Ciencias y Tecnologías Computacionales

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

Módulo de carga de modelos CAD para simulaciones de microfluidos en el Proyecto: Estudio Computacional de Virus en Microfluidos (ECVM)

Autora: Melissa Cordero Alvarez

Tutores: Edisel Navas Conyedo

Co-tutor: Reyder Cruz de la Osa

La Habana, mayo de 2022

"Año 64 de la Revolución"

DECLARACIÓN DE AUTORÍA

El autor del trabajo de diploma con título **“**Módulo de carga de modelos CAD para simulaciones de microfluidos en el Proyecto: Estudio Computacional de Virus en Microfluidos (ECVM), concede a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la investigación, con carácter exclusivo. De forma similar se declara como único autor de su contenido. Para que así conste firma la presente a los <día> días del mes de <mes> del año 2022.

Melissa Cordero Alvarez

_____
Firma del Autor

Edisel Navas Conyedo

_____
Firma del Tutor

Reyder Cruz de la Osa

_____
Firma del Tutor

DATOS DE CONTACTO

Datos del Tutor:

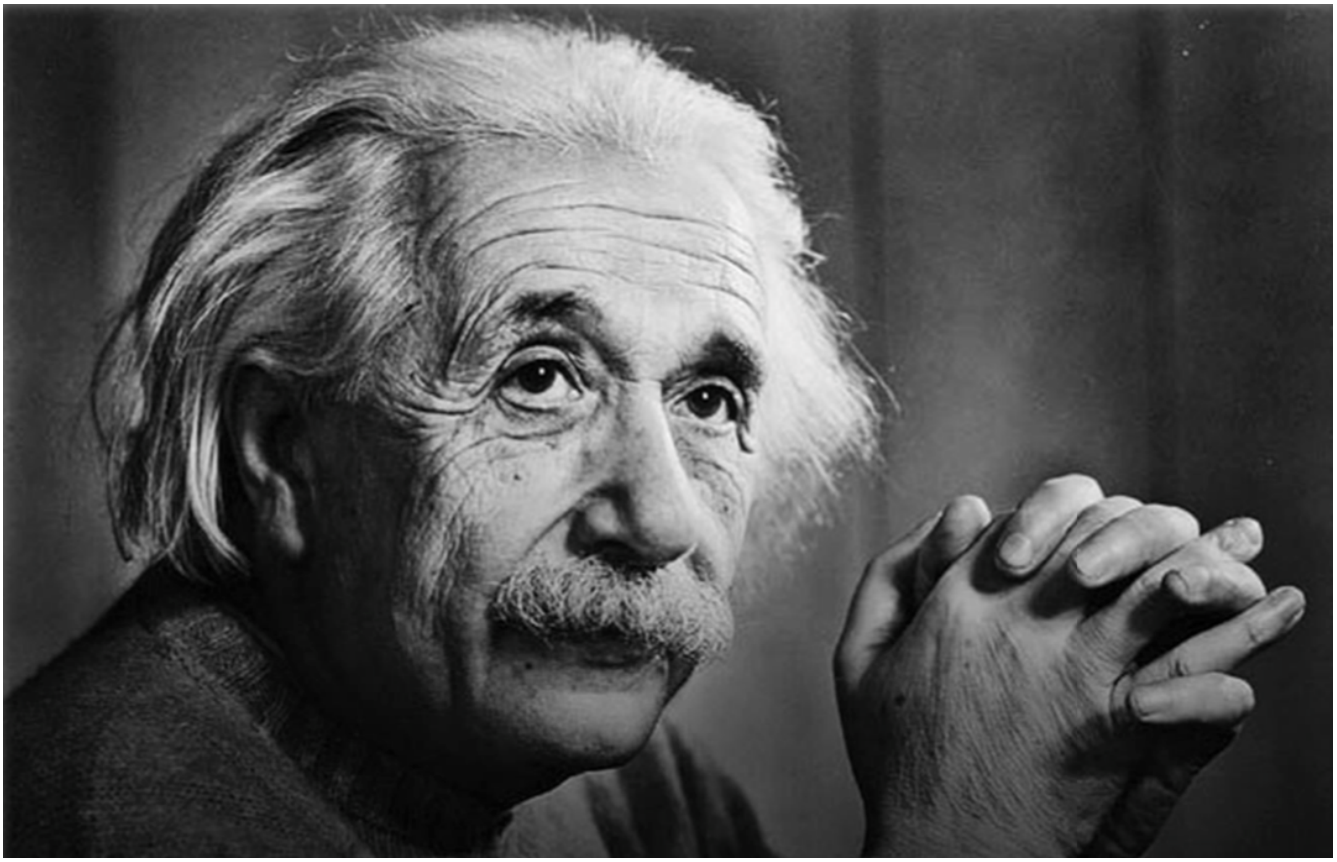
Profesor Edisel Navas Conyedo: Graduado de Licenciado en Física en 2003. Profesor Auxiliar e Investigador del Centro de Estudios de Matemática Computacional. Coordinador del grupo de Matemática y Física Computacional de la Línea de Computación Científica. Jefe de proyecto de Estudio Computacional de Virus en Microfluidos.

Correo electrónico: enavas@uci.cu

Datos de Co- tutor:

Profesor Reyder Cruz de la Osa: Graduado de Ciencias de la Computación en la Universidad de la Habana en 2007 e imparte clases de Inteligencia Artificial siendo profesor asistente. Presidente de la Cátedra Honorífica de Ajedrez "Remberto Fernández"

Correo electrónico: reyder@uci.cu



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

Albert Einstein

AGRADECIMIENTOS

El amor recibido, la dedicación y la paciencia con la que se preocupaban mis padres por mi avance y trayectoria de estos cinco años.

Gracias a todos mis amigos de la universidad que formaron parte de este largo y duro camino, por ayudarme en esas noches agotadoras de estudio.

A todas esas personas que formaron y forman parte de mi vida, y que influyeron en el resultado final.

DEDICATORIA

Dedico con todo mi corazón mi tesis principalmente a mi madre y a mi padre por motivarme a seguir estudiando y porque les cumplí el sueño de tener una hija Ingeniera, siendo su mayor orgullo en la vida. A mis abuelas porque su mayor anhelo es verme graduada, a mis hermanos por preocuparse por mí en todo momento y al resto de la familia.

RESUMEN

En nuestro país la biotecnología es una de las áreas donde tiene gran importancia la realización de simulaciones moleculares, debido a que se logra un mayor ahorro en el tiempo de investigación, recursos materiales y de producción de resultados efectivos trabajando en aras de mejorar cada día más la salud cubana. Gracias a la revolución tecnológica una de las herramientas más utilizadas para la obtención de resultados de las simulaciones moleculares es el simulador *Large-scaleAtomic Molecular MassivelyParallel Simulator (LAMMPS).* Actualmente la interacción con el simulador LAMMPS es tediosa debido a que no posee una interfaz gráfica de usuario, se trabaja con extensos códigos mediante una consola y la elaboración del archivo de entrada está expuesta a la ocurrencia de errores físicos y numéricos por lo que se retrasa el tiempo para la obtención de resultados necesarios en las investigaciones científicas.

La presente investigación expone el desarrollo de un módulo de carga de modelos CAD para simulaciones de microfluidos que tiene como objetivo fundamental: permitir la carga de diseños CAD de dispositivos de microfluidos en el software de simulación LAMMPS.

Para su desarrollo se realizó un estudio de los sistemas existentes con fines similares y se

determinaron tecnologías para su implementación. Se utilizó como metodología tradicional de desarrollo AUP - UCI, lenguaje de programación Python y fue montado en multiplataforma. El patrón arquitectónico implementado fue Tres Capas y se trabajó con el Visual Paradigm como herramienta CASE.

PALABRAS CLAVE

Simulador, LAMMPS, módulo, microfluidos, diseño

ABSTRACT

In our country, biotechnology is one of the areas where the realization of molecular simulations is very important, due to the fact that greater savings are achieved in research time, material resources and the production of effective results, working in order to improve every day. Cuban health. Thanks to the technological revolution, one of the most widely used tolls to obtain results from molecular simulations is the Large- scale Atomic Molecular Massively Parallel Simulator (LAMMPS). Currently, the interaction with the LAMMPS simulator is tedious because is does not have a graphical user interface, it works with extensive codes through a console and the preparation of the input file is exposed to the occurrence of physical and numerical errors, which is why it is delayed. The time to obtain the necessary results in scientific research.

This research exposes the development of a CAD model loading module for microfluidic simulations whose main objective is: to allow the loading of CAD designs of microfluidic devices in the LAMMPS simulation software.

For its development, a study of existing systems with similar purposes was carried out and technologies for its implementation were determined. The traditional AUP- UCI development methodology was used, the Python programming language and was mounted on a multiplatform. The implemented architectural pattern was Three Layers and the Visual Paradigm was used as a CASE tool.

KEYWORDS

Simulators, LAMMPS, module, microfluidics, design

TABLA DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

La microfluídica agrupa un gran número de ramas de la física, desde la dinámica de fluidos, hasta la electrónica, y está estrechamente ligada a las ciencias biológicas. La interdisciplinariedad asociada a ella es una característica propia de la época en que vivimos. En los últimos diez años, con el desarrollo de la biotecnología, la microelectrónica, la ciencia de materiales y otras aplicaciones de avanzada, el uso de dispositivos microfluídicos ha ido incrementándose. Si bien en Cuba algunos científicos han realizado trabajos relacionados con esta temática, hasta el momento no existe ninguna referencia en que se divulgue esta actividad *(Paredes Chicaiza y DT-Meléndez Tamayo 2012).*

La definición más aceptada de lo que se conoce como microfluídica aborda dispositivos y métodos para controlar y manipular fluidos con escalas de longitud menores a una décima de milímetros. Se suele aceptar que se habla de microfluidos cuando las cantidades que se manipulan son pequeñas, independientemente de que alguna parte del dispositivo sea relativamente más grande. Dicho de otro modo, la microfluídica es el conjunto de actividades en que se aprovechan las ventajas que proporcionan el uso y control del fluido a escalas inferiores a la milimétrica, donde las propiedades físicas pueden ser distintas de la escala convencional conocida por el hombre en su vida cotidiana, así como los fenómenos físicos *(Paredes Chicaiza y DT-Meléndez Tamayo 2012).*

Una prueba de la importancia que ha cobrado la microfluídica es que esta temática se incluye cada vez más en las sesiones técnicas de casi todas las organizaciones científicas de todo el mundo como por ejemplo: AIChe, American Institute of Chemical Engineers, e IEEE, Institute of Electrical and Electronic Engineers, entre otros. Por consiguiente, el número de publicaciones, congresos y reuniones ha crecido de manera casi exponencial en los últimos años. Además de las publicaciones tradicionalmente relacionadas con los fluidos, otras como Lab-on-a-Chip, Sensors and Actuators, y Analytical Chemistry han publicado numerosos trabajos sobre microfluidos, lo que confirma el carácter multidisciplinario de esta actividad. Su efervescencia se demuestra en el creciente número de publicaciones nuevas relacionadas con esta temática *(Rojas Lazo 2014).*

Los microfluidos representan un reto para los científicos de hoy y los que se preparan para el futuro. Los problemas que se avecinan requieren un enfoque multidisciplinario e integrador, como ha quedado demostrado por los avances ya logrados. En un dispositivo microfluídico es típico encontrar tecnología empleada en la microelectrónica, diseños copiados de la naturaleza, con sondas y sensores desarrollados para reacciones químicas complejas, sistemas ópticos avanzados para visualizar, y la omnipresente computadora para controlar y procesar datos. Es de esperar que los avances logrados en los microfluidos abran la puerta a las aplicaciones de nanofluídica (materiales porosos, autoensamblaje, na-

nomodificación de superficies) que comienzan a llamar la atención. Los microfluidos sirven ya como interfaz entre los dispositivos con escalas nanométricas y los equipos convencionales que los humanos podemos manipular *(Caribe 2010).*

En el diseño y fabricación de los dispositivos de microfluidos se ha dado mayor importancia a la simplicidad por encima de la complejidad, y otro tanto a la funcionalidad sobre la miniaturización. Estos factores han estado condicionados por los elevados costos de las tecnologías de fabricación, las limitaciones de empleo de diferentes materiales, así como la dificultad de bombeo a tales escalas. Estas limitaciones impulsaron la búsqueda de nuevos materiales y conceptos de utilización de dispositivos lo más simples posibles y que, a la vez, no fueran costosos, ya que en muchos casos deben ser desechados luego de ser empleados. El principal método de fabricación consiste en el empleo de moldes reutilizables de modo que la replicación de los dispositivos no sea un problema. La interconexión de los dispositivos se logra utilizando monturas especiales que pueden ser utilizadas una y otra vez, y que tienen los sensores y sistemas de conexión debidamente instalados *(Rojas Lazo 2014).*

Ahora bien, los dispositivos microfluídicos se pueden modelar a través de los llamados modelos CAD; pues en la actualidad, las nuevas tecnologías de la informática suponen una herramienta básica para los profesionales. En el entorno concreto de un ingeniero muchas de las actividades que puedan desarrollar en su profesión requieren el uso habitual de un programa que permita diseñar modelos en 3D de piezas.

El diseño asistido por ordenador en inglés: “Computer Aided Design” (CAD), es un conjunto de herramientas informáticas que permiten a un profesional diseñar modelos 3D de piezas, montaje de conjuntos para, finalmente obtener planos. Los programas incluyen en muchos casos, la posibilidad de realizar un análisis por elementos finitos (FEM Finit Element Analysis) e incluso información adicional para su integración en PLM (Product Life Management).

Es importante tener en cuenta que, hoy en día, el PLM (Product Life Management) supone una nueva filosofía que hace frente a todo el ciclo de vida del producto, integrando el primer diseño con las sucesivas etapas, haciendo frente a modificaciones de forma que toda la información que afecta a distintas áreas de la empresa está permanentemente actualizada.

El objetivo directo final del uso de una herramienta de CAD puede ser la fabricación de prototipos que conducen a la fabricación de piezas, la fabricación de piezas directamente en las máquinas de control numérico (CNC) o proporcionar información para el montaje de conjuntos. Para una correcta utilización de estas herramientas es imprescindible tener conocimientos básicos tanto de dibujo y diseño como

tener un grado básico de conocimientos de informática *(Paredes Chicaiza y DT-Meléndez Tamayo 2012).*

Más recientemente surgió el término de ingeniería asistida por ordenador o “Computer Aided Engineering (CAE) que se basa en el uso del ordenador para realizar los diferentes cálculos y dimensionamientos de los diferentes proyectos. Todo esto lleva a un estrechamiento en la relación entre la ingeniería y el diseño asistido por ordenador con la expectativa de poder llegar a la construcción integrada por ordenador *(Caño, Cruz y Solano 2007).*

También aparece el término de fabricación asistida por ordenador o “Computer-Aided Manufacturing (CAM), que implica el uso del hardware y del software para obtener un proceso de fabricación parcial o totalmente automatizado, procesos que pueden ser para producir elementos estructurales u otros elementos.
En la actualidad ya se dispone de programas de CAD que tienen la capacidad de crear instrucciones para el mecanizado automático de piezas o bien que sean capaces de llevar a cabo una integración con programas tipo CAM para poder relacionarse entre ellos para que estos últimos generen dichas instrucciones *(Caño, Cruz y Solano 2007).*

Son obvias las ventajas de los programas CAD respecto al dibujo manual, ya casi obsoleto. Rapidez de ejecución, incremento en la precisión de los diseños y disponer de una biblioteca de diseños y de conocimiento. Este software tiene múltiples aplicaciones en la ingeniería como son la elaboración de diagramas de diversos tipos, gráficos estadísticos, análisis con elementos finitos y representación normalizada de piezas para su diseño, entre otros *(Rojas Lazo 2014).*

Los dispositivos microfluídicos se pueden modelar a través de los modelos CAD y ese modelo se utiliza para realizar simulaciones de prueba o calibración a través de simuladores como: High Performance Customizable Molecular Simulation (OpenMM), Chemestry at Harvard Macromolecular Mechanics (CHARMM), Assisted Model Building with Energy Refinement (AMBER) y Large- scale Atomic/Molecular Massively Parallel Simulator (LAMMPS); el cual se utiliza para simular microfluidos, pues los modelos CAD se representan por canales por los cuales fluyen los microfluidos y esos canales tienen tamaño de micrómetro.

LAMMPS, es un programa para la simulación de dinámica molecular con el que se pueden modelar conjuntos de partículas en estado sólido, líquido o gaseoso, bajo diferentes condiciones de contorno y campos de fuerza, ya sean sistemas poliméricos, atómicos, biológicos, metálicos o granulados. LAMMPS puede aprovechar el poder del procesamiento en paralelo utilizando técnicas de descomposición espacial que dividen el dominio de simulación en pequeños subdominios 3D y está

optimizado para algunos modelos de procesadores Intel, en los que puede ofrecer rendimientos acelerados de CPU.

Este software cuenta con varias ventajas, pero evidentemente las más impresionantes son la capacidad de trabajar con simulaciones de manera simultánea, lo que se conoce en términos informáticos como trabajo en paralelo (HPC - High Performance Computer: computación de alto rendimiento), y por otra parte es que este sistema es libre de pago.

Desgraciadamente, entre sus desventajas más notables es la dificultad de trabajo dado que todo el proceso se realiza de manera manual por consola a través de la edición de archivos de descripción del sistema y un archivo o script de ejecución de órdenes, lo cual hace bastante tedioso el proceso cuando se enfoca en una investigación complicada o extensa. A pesar de su versatilidad, LAMMPS no posee una interfaz gráfica de usuario, por lo que la interacción es compleja. La elaboración del archivo de entrada de LAMMPS requiere de una descripción detallada de los datos de inicio, integra diversos parámetros que definen el sistema y las condiciones ambientales de la simulación, por lo que este proceso puede generar una serie de posibles errores físicos o numéricos, por ejemplo, elegir un paso de tiempo demasiado grande o especificar un coeficiente de campo de fuerza no válido, a los cuales el simulador no puede dar solución. De igual forma la gestión de los ficheros de entrada y salida conlleva la interacción con la plataforma de simulación y la recuperación de los resultados, proceso que demanda de recursos computacionales y tiempo, al estar estos ficheros expuestos a posibles pérdidas y corrupción de la información dado su almacenamiento local, por lo que su gestión es ineficiente y no garantiza la integridad de los datos.

Utilizar un software de diseño CAD para la descripción del sistema permite el ahorro en esfuerzo y configuración de las estructuras espaciales y de interacción entre los componentes, donde además el diseño resulta reutilizable. Se diseña un dispositivo con un objetivo para que sea un modelo hábil y la forma de validar de que cumpla su objetivo es primeramente hacer una simulación donde se compruebe si funciona con el propósito final. Si funciona, se procede a fabricar el dispositivo microfluídico, que al tener un alto costo no se puede construir en vano, y en caso de que no sea un modelo hábil se cambia el diseño hasta que este dispositivo cumpla con su finalidad *(Figura 1).* Por eso resulta necesario incorporarle a LAMMPS la funcionalidad de cargar los modelos CAD a través de un plugin o fase intermedia que transforma la información presente en los archivos CAD a un formato de celda (.cell) de LAMMPS.

Figura 1: Modelado del proceso de validación de un dispositivo hábil.

Fuente: Elaboración propia.

Este módulo que se va a desarrollar es parte del proyecto de investigación: Estudio Computacional de Virus en Microfluidos (ECVM), donde las temáticas que desarrolla corresponden con la línea de Computación Científica en las materias de Física y Matemática Computacional, Computación de Altas Prestaciones y Bioinformática. Este se basa en el desarrollo de simulaciones de fenómenos físicos complejos con la presencia de estructuras virales que contribuye a desarrollar otras áreas aplicadas en la universidad vinculadas con la docencia y la investigación. Tiene como base la hipótesis de que los complejos funcionales de las proteínas de la Coagulación y el Complemento que unen al Virus del Dengue pueden alcanzar dimensiones de importancia en comparación con las dimensiones del virus y modificar significativamente su geometría.

Por todo lo expresado anteriormente se han definido los siguientes parámetros de investigación. Se evidencia el siguiente **Problema de investigación científica:** ¿Cómo facilitar la carga de modelos CAD de dispositivos hábiles de microfluidos en el software LAMMPS garantizando un uso óptimo de los recursos computacionales?

Se determina como **objeto de estudio:** Representación de estructuras 3D en el software CAD y LAMMMPS.

Campo de acción: Carga de archivos de representación 3D de modelos hábiles de microfluidos mediante CAD en el proyecto SIMBOX/ECVM.

Objetivo general: Desarrollar un módulo de carga de modelos CAD de dispositivos microfluídicos para el software LAMMPS en el proyecto SIMBOX/ECVM.

Objetivos específicos:

1 Desarrollar una herramienta de traducción de los modelos CAD a modelos hábiles para LAMMPS.

2 Validar mediante simulaciones la calidad de la transformación de los modelos CAD en modelos hábiles para LAMMPS.

Aporte Práctico: El aporte práctico de la presente investigación consiste en la implementación de un módulo informático, tanto en 2D como 3D, permitiéndole proveer a LAMMPS de los archivos de entrada que representan correctamente las estructuras del diseño del sistema de microfluidos. Esto permitirá facilitar en gran escala el trabajo y análisis de estos datos de las simulaciones, lo cual permitirá una mejor comprensión de varios fenómenos que se analizan en el campo de la microfluídica.

Tareas de Investigación

1 Analizar la estructura de los archivos CAD en la representación espacial de los componentes de dispositivos microfluídicos.
2 Diseñar la arquitectura de trasformación de los archivos CAD para su incorporación en simulaciones con LAMMPS.
3 Implementar un módulo de carga de modelos CAD.
4 Evaluar la calidad de la trasformación de modelos CAD en modelos hábiles para LAMMPS.

Estructura general del documento

El documento se encuentra estructurado en introducción, tres capítulos, conclusiones, recomendaciones y bibliografía. Los temas abordados en cada uno de los capítulos son descritos a continuación:

1 Capítulo 1: “Fundamentos de la investigación”: se presentan un conjunto de consideraciones referidas al trabajo y análisis de datos de las simulaciones en el simulador LAMMPS, así como su relación con los modelos CAD, que son los que se desean integrar como herramienta de modelación, además de su propia integración con el sistema de desarrollar.
2 Capítulo 2: “Propuesta de solución”: se introducen ideas enfocadas a la solución de problema de investigación. Específicamente se propone un subsistema de modelación de datos de las simulaciones en LAMMPS para la optimización del trabajo y análisis de datos de simulaciones.

3 Capítulo 3: “Pruebas y análisis de resultados”: se describen el conjunto de pruebas y posibles cambios a los que se somete el sistema, luego de realizar las debidas pruebas experimentales.

Capítulo I: FUNDAMENTOS DE LA INVESTIGACIÓN

En el presente capítulo, dedicado a comprender el objeto de estudio y el campo de acción de la investigación, se abordan los conceptos, propiedades y características principales para poder leer un archivo CAD, obtener el modelo de superficie y hacer su representación en LAMMPS. Se exponen si existen o no soluciones similares existentes y las tecnologías empleadas en el desarrollo del presente módulo.

1.1 Conceptualización de los principales términos.

El diseño asistido por ordenador (CAD) consiste en el uso de programas de ordenador para crear, modificar, analizar y documentar representaciones gráficas 2D o 3D de objetos físicos como una alternativa a los borradores manuales y a los prototipos de producto.

Existe una gran variedad de sectores en los que se utiliza CAD de manera muy regular, sobre todo en ingenierías, civil o aeronáutica, pero también lo usan arquitectos o la industria de la automatización.

Esta tecnología permite la agilización del trabajo, automatizando los procesos manuales del proceso de diseño de producto, reduciendo errores, ganando velocidad y aumentando la calidad. Con esto las empresas consiguen así una mayor eficacia y productividad, puesto que permite visualizar de manera previa el producto final y además jugar de forma interactiva con diferentes diseños, sin necesidad de crear un número elevado de prototipos.

Existen 2 tipos de software CAD, el 2D y el 3D. El primero de ellos, CAD 2D trabaja con dibujos técnicos bidimensionales simples que suelen ser la base para otros proyectos mayores, en cambio el 3D permite crear dibujos tridimensionales con mayor precisión y detalle y que además muestran el espacio de trabajo y la profundidad, por lo que ofrece una visión más real de los objetos [CITATION Exp19 \l 23562].

Dentro de la comunidad científica, específicamente las ramas asociadas a la química, biología y bioinformática, se hace uso de un sistema informático llamado **LAMMPS** para el desarrollo y análisis de simulaciones a nivel atómico/molecular. LAMMPS es un código de dinámica molecular clásica que permite modelar conjuntos de partículas en los diferentes estados (líquido, sólido y gaseoso). Además, ofrece la modelación de sistemas atómicos, poliméricos, biológicos, sistemas metálicos, granulados, y de grano grueso utilizando una variada serie de campos de fuerza y condiciones de contorno. El programa integra las ecuaciones de movimiento de Newton para las interacciones entre los átomos, moléculas y/ partículas macroscópicas que se encuentran en el espacio de simulación definido. La modelación de los sistemas de partículas puede ser tanto en 2D como en 3D, dependiendo de la finalidad de la investigación.

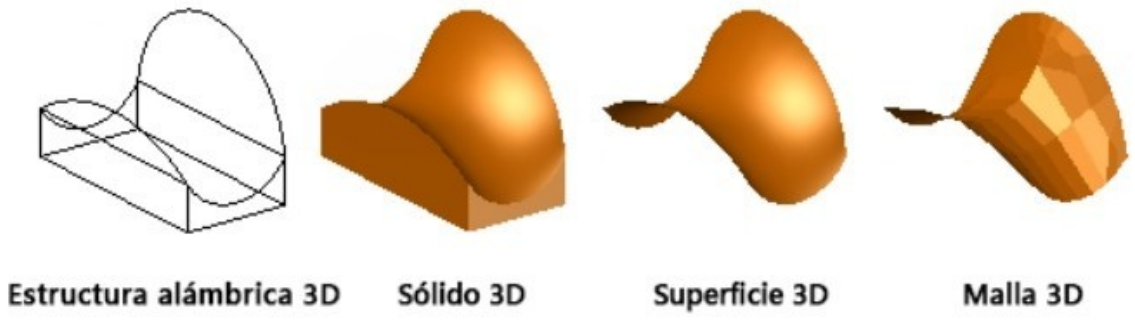


Figura 2: Tipos de modelos 3D [CITATION AUT22 \l 23562]

Estructura de un Espacio de Simulación

Los Espacios de Simulación para el software LAMMPS son almacenados en ficheros no binarios de extensiones tipo: data y cell. No existe diferencia estructural entre una extensión y otra, sólo cambia su acrónimo. La organización interna del fichero, en el cual se almacenan todas las partículas del sistema, está bien definida en la página oficial de LAMMPS.

La primera regla a tener en cuenta es que las líneas en blanco son importantes, después de la sección de encabezado todos los bloques de instrucciones son separados por líneas en blanco. Las identaciones y espacios entre palabras y números en una línea no son relevantes, exceptuando las palabras claves.

El primer elemento del fichero es el encabezado. Este debe estas siempre presente y estar ubicado en la primera línea del archivo. Los demás elementos, aunque pueden encontrarse en orden arbitrario, se les ha asignado un orden lógico con el objetivo de lograr uniformidad en los ficheros generados.

El segundo bloque de instrucciones es el encargado de inicializar las entradas del fichero. En él pueden encontrarse varios atributos que indican la cantidad de átomos (atoms), enlaces (bonds), ángulos (angles), interacciones diédricas (dihedrals) e interacciones impropias (impropers).

En el tercer bloque debe especificarse la cantidad de tipos de cada entrada que se ha definido anteriormente en el bloque de inicialización. Ejemplo de ello es decir que existen 5 tipos de átomos (5 atom types) en el fichero que se está construyendo. En caso de que un atributo no haya sido inicializado en las entradas, tampoco debe de inicializarse en este bloque.

El cuarto bloque define el tamaño del Espacio de Simulación en formatos de ejes de coordenadas (x, y, z) para sistemas en tres dimensiones. En caso de que el sistema sea no-periódico, estos datos definen

la mínima y máxima extensión de los átomos. Para sistemas en dos dimensiones, el valor de z se define en cero.

El quinto bloque se especializa en la definición de todos los coeficientes antes inicializados en las capas posteriores. Se representa de manera iterativa, empezando por la primera propiedad y se define cada uno de los N componentes. Hasta este punto, es obligatorio que el encabezado, los átomos y las masas estén en el fichero.

En el último bloque se localiza toda la información espacial de los átomos, sus relaciones y propiedades.

Esta división del fichero contiene los N átomos del sistema listados secuencialmente. El primer número de su estructura es el *atom-tag*, el cual se emplea para identificar el átomo a través de la simulación. El *molecule-tag* es el segundo identificador que está adherido al átomo y representa la molécula a la que pertenece el átomo. El tercer elemento representa el tipo de átomo y el cuarto elemento es definido como *q* y representa la carga del átomo. Esto es seguido de las coordenadas iniciales (x, y, z) del átomo. Por último, y de manera opcional, se encuentran los valores (nx, ny, nz) que solo serán leídos por LAMMPS si el comando *true flag* es especificado en el script de entrada a la simulación, de otro modo se inicializan como cero.

Las entradas para la velocidad, enlaces, ángulos, interacciones diédricas e interacciones impropias deben aparecer luego de la entrada de los átomos [CITATION Leo \l 23562]

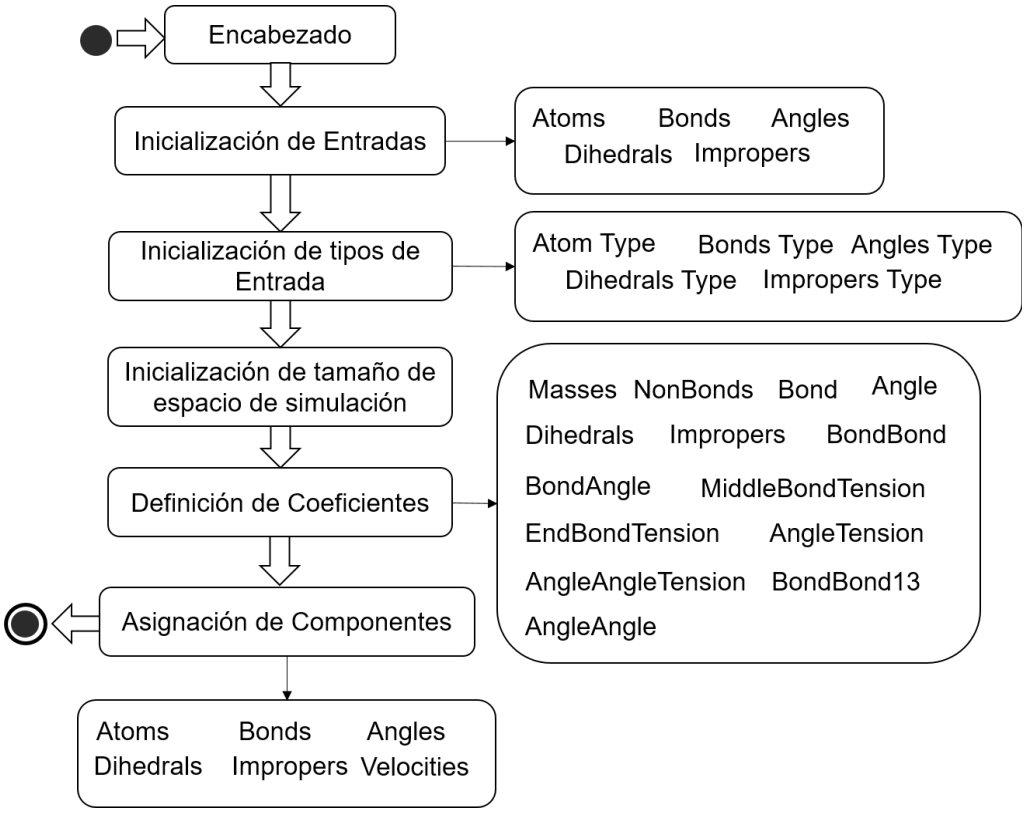


Figura 3: Bloques representativos de un fichero de Espacio de Simulación [CITATION Leo20 \l 23562]

Método SPH

SPH es un método continuo, que no requiere una cuadrícula predefinida para evaluar las ecuaciones de campo diferenciales parciales asociadas de la mecánica continua. En cambio, SPH discretiza el campo de distribución de masas en masas puntuales que se mueven con el material, de acuerdo con las ecuaciones de movimiento de Newton. Las posiciones de las masas puntuales sirven como nodos de integración para las ecuaciones de campo de la mecánica continua. Los campos variables requeridos se construyen sobre la marcha utilizando núcleos de interpolación, que se centran en las masas puntuales. Debido a su naturaleza de partículas, SPH es directamente compatible con la arquitectura de código existente y las estructuras de datos presentes en LAMMPS.

SPH es un método para resolver problemas en la mecánica continua lagrangiana, donde las ecuaciones diferenciales parciales gobernantes describen la evolución del movimiento conjunto de la densidad ρ, las coordenadas r, la velocidad v y la energía por unidad de masa e en términos de gradientes de velocidad, presión tensor2 P, y el vector de flujo de Q = κ∇T, con conductividad térmica κ gradiente de temperatura ∇T.

$$\frac{\mathrm{d}\rho}{\mathrm{d}t} = -\rho\nabla\cdot\mathbf{v}$$
$$\frac{\mathrm{d}\mathbf{v}}{\mathrm{d}t} = -\frac{1}{\rho}\nabla\cdot P$$
$$\frac{\mathrm{d}e}{\mathrm{d}t} = -\frac{1}{\rho}P\colon\nabla\mathbf{v} - \frac{1}{\rho}\nabla\cdot Q$$

SPH interpola el conjunto de variables de campo {ρ, v, e, P, Q} por medio de la interpolación kernel.
En la distribución de LAMMPS, el SPH se distribuye como un módulo adicional, lo que significa que no se compila de forma predeterminada con el resto de LAMMPS.
Algunas de las ventajas del método SPH son su adaptabilidad a métodos complejos, su capacidad para simular superficies libres, el posible tratamiento de las salpicaduras o gotas, la simplicidad de las ecuaciones a resolver o la conservación de las magnitudes dinámicas que definen el fluido [CITATION Lie11 \l 23562].

Application Programming Interface **(API)** (traducido Interfaz de Programación de Aplicaciones) es un conjunto de definiciones y protocolos que se utiliza para desarrollar e integrar el software de las aplicaciones. En concepto, podemos entender la API como un código que indica a las aplicaciones cómo pueden mantener una comunicación entre sí. Estas reglas permiten que los distintos programas mantengan interacciones. Hace referencia a los procesos, las funciones y los métodos que brinda una determinada biblioteca de programación a modo de capa de abstracción para que sea empleada por otro programa informático [CITATION Gom21 \l 23562].

Las API permiten que sus productos y servicios se comuniquen con otros, sin necesidad de saber cómo están implementados. Esto simplifica el desarrollo de las aplicaciones y permite ahorrar tiempo y dinero. Además, le otorgan flexibilidad; simplifican el diseño, la administración y el uso de las aplicaciones, y proporcionan oportunidades de innovación, lo cual es ideal al momento de diseñar herramientas y productos nuevos o para gestionar las actuales [CITATION Gom21 \l 23562].

LAMMPS se puede compilar como una biblioteca ya sea estática o compartida, de modo que pueda llamarse mediante otro código, usarse de forma conjunta con otros códigos o controlarse a través de una secuencia de comandos de Python. Incluso el ejecutable independiente de LAMMPS es esencialmente un contenedor delgado sobre la biblioteca de LAMMPS, que crea una instancia de este, donde procesa la entrada y luego existe.

La mayoría de las API se basa en funciones de contenedor de lenguaje C en los archivos *src/library.h y src/library.cpp*, pero también es posible usar C++ directamente. El procedimiento básico es siempre

el mismo: crea una o más instancias de LAMMPS, pasa comandos como cadenas o desde archivos a esa instancia de LAMMPS para ejecutar cálculos y/o funciones de llamada que leen, manipulan y actualizan datos de las instancias de clases activas dentro de LAMMPS para realizar análisis o realizar operaciones que no son posibles con los comandos de script de entrada existentes.

API de Python de LAMMPS

El módulo de Python de LAMMPS permite llamar a la API de la biblioteca C de LAMMPS desde Python mediante la carga dinámica de funciones en la biblioteca compartida de LAMMPS a través del módulo ctypes de Python. Debido a la carga dinámica, se requiere que LAMMPS se compile en modo "compartido". La interfaz de Python está orientada a objetos, pero por lo demás trata de ser muy similar a la API de la biblioteca C. Hay disponibles tres clases de Python diferentes para ejecutar LAMMPS y se complementan entre sí.

```
def __initLAMMPS__(self, inputFilename=None, logFilename=None, verbose = False):
        """Initialize a LAMMPS object using the provided input file.

        Parameters
        ----------
        inputFilename : string (None)
            Path to input file.

        logFilename : string (None)
            Path to log file.

        verbose : bool
            If true, write output to standard out.
        """

        try:
            from lammps import lammps
        except:
            print "ERROR: The LAMMPS python interface could not be found. Check your
PYTHONPATH and compiled LAMMPS directory to make sure it is compiled and
importable."


        # check to see we are handing an absolute pathname.
        if not os.path.isabs(inputFilename):
            inputFilename = os.path.abspath(inputFilename)

        # we're going to assume here that the user is taking care of the fact that
LAMMPS needs to know where to find things but we will issue a worning in case they
aren't.
        if os.path.dirname(inputFilename) != os.getcwd():
            print "WARNING: Getting lammps input file from directory other than
current working directory. Be sure pathnames are correct in input files."

        # initialize the lammps object
        if verbose == True:
            self.lmp=lammps()
            #self.lmp = lammps("", ["-c", "on", "-sf", "cuda"])
        else:
            args = ["-sc", "none", "-echo", "none", "-log", "none"]
            self.lmp = lammps("", args)

        # after the lammps object is created, initialize the lammps simulation.
        # specify general log file parameters
        #self.command("echo none")

        # write out log file if needed
        if logFilename is not None:
            if not os.path.isabs(logFilename):
                logFilename = os.path.abspath(logFilename)
```

Figura 4: Ejemplo del código del API de LAMMPS.

1.2 Búsqueda y obtención de contenidos.

El Proyecto GESES-PETRI (con nombre de producto GETRIX), consiste en el desarrollo de un sistema web cuya única y exclusiva función es facilitar el trabajo de simulaciones con LAMMPS para todos los científicos de su comunidad. Para esto se ha diseñado un sistema que consta de 3 componentes fundamentales y un almacén de datos.

Dichos componentes serán abordados de manera breve a continuación solo para ofrecer una mayor comprensión a los lectores:

Subsistema para la Creación y Control de Simulaciones (SIMBOX)- Este microsistema tendrá como función principal la comunicación con el simulador LAMMPS para administrar los parámetros y controlar las simulaciones. Solamente tendrá comunicación con el sistema GESES, el simulador LAMMPS y con el almacén de datos.

Subsistema para la Creación de Espacios de Simulación (PETRI)- Este microsistema estará solamente relacionado con GESES y tendrá como función principal la creación de espacios de simulación. Tendrá acceso al almacén de datos del Sistema GESES-PETRI.

Sistema Gestor de Simulaciones y Espacios de Simulación (GESES)- sistema que permitirá al científico gestionar varias instancias de simulaciones LAMMPS desde una misma ventana matriz, ajustar sus parámetros y observar la consola de progreso de la simulación. Además, podrá crear espacios de simulación con moléculas previamente seleccionadas o de su preferencia, ajustar el tamaño del espacio y definir las cantidades de moléculas individuales a insertar. Este sistema será accedido vía web, con autenticación única para cada científico que garantiza seguridad en sus datos de estudio y su alojamiento será en un servidor o HPC que permita ejecutar las instancias de simulaciones LAMMPS. Este constituye el sistema central de todo el proyecto y está conectado tanto a PETRI como a SIMBOX.

Archivos CAD, acrónimo de archivos de diseño asistido por ordenador, son una especie de imagen almacenada en un formato bidimensional o tridimensional. Diseños CAD en 2D se conocen comúnmente como dibujos, mientras que los diseños CAD en 3D se llaman modelos. Los archivos CAD se utilizan normalmente para almacenar modelos de herramientas de ingeniería, planos de arquitectura y diseños relacionados. Los archivos CAD son muy buenos para el almacenamiento de la información precisa para diseños de alta calidad, así como ilustraciones técnicas [CITATION Sol221 \l 23562].

Un archivo CAD es una salida de un software CAD que contiene información clave sobre el objeto diseñado: su representación de geometría y topología, jerarquía del modelo 3D, metadatos y atributos visuales según el formato del archivo. La característica definitoria de un archivo CAD es su formato. Hay formatos de archivo CAD neutrales, nativos y "kernel"

- Los formatos de archivos neutrales, como STEP e IGES, se elaboraron como estándares de la industria legibles por la mayoría de las plataformas de modelado CAD. Las especificaciones para esos formatos están disponibles públicamente y son mantenidas por empresas o consorcios. Por ejemplo, IGES es desarrollado por la Oficina Nacional de Normas, mientras que STEP es un producto de CAX IF.
- Formatos nativos,p. SOLIDWORKS, CATIA Y DWG son producidos por los principales proveedores de CAD. Para importarlos en herramientas CAD de terceros, se requiere conversión.
- Los formatos de kernel se derivan de los populares kernels de modelado geométrico, siendo ACIS y Parasolid los más comunes. Un kernel de modelado CAD es un componente de un software de modelado, que define cómo describe matemáticamente una forma. Los formatos de kernel funcionan mejor con el software construido sobre el kernel correspondiente. [CITATION Sol221 \l 23562].

Tabla 1: Formatos de archivos CAD [CITATION Mar21 \l 23562].

Neutral CAD formats	Kernel CAD formats	Native CAD formats
DXF	ACIS	CATIA
IFC	Open	DWG
IGES	CASCADE	PTC Creo
JT	Parasolid	Siemens NX
STEP	Rhino	Solid Edge
3D PDF		Solidworks
3DS		3D XML
3MF		
Collada		
FBX		
OBJ		
PLY		
PRC		
STL		
U3D		
USD		
VRML		
X3D		
glTF		

FreeCAD es una aplicación libre de diseño asistido por computadora en tres dimensiones, ingeniería asistida por computadora, para la asistencia en ingeniería mecánica y el diseño de elementos mecánicos. Está basado en Open Cascade y programado en los lenguajes C++ y Python. Además del propio formato de archivo de FreeCAD, pueden manejarse los siguientes formatos de archivo: DXF, SVG (Scalable Vector Graphics), STEP, IGES, STL (STereoLithography), OBJ (Wavefront), DAE (Collada), SCAD (OpenSCAD), IV (Inventor) y IFC. El formato de archivo STEP 3D es un formato de archivo muy útil para la trasferencia de datos entre herramientas CAD y de herramientas CAD a otras herramientas 3D con fines de visualización. Su capacidad para representar superficies matemáticas perfectas significa que satisface las necesidades de los ingenieros que se desean que se fabriquen sus diseños.

Representación de geometría

Las dos formas más comunes de representar datos 3D en archivos CAD son mallas y B-Rep, también conocidas como representaciones poligonales y de límites. Los cuerpos B-Rep se definen a través de un conjunto de entidades geométricas precisas y entidades topológicas. El modelo malla, por otro lado, es un gemelo de R-Rep con todos los límites precisos reemplazados por un conjunto de facetas aproximadas.

Las entidades geométricas son un marco para la visualización futura. Almacena la información sobre dimensiones y propiedades geométricas de las piezas: volumen, área superficial y centro de masa.
Los formatos de archivo CAD difieren en su alcance de representación geométrica: IGES admite polígonos B-Rep, STL y VRML, mientras que JT puede contener ambos [CITATION Mar21 \l 23562].

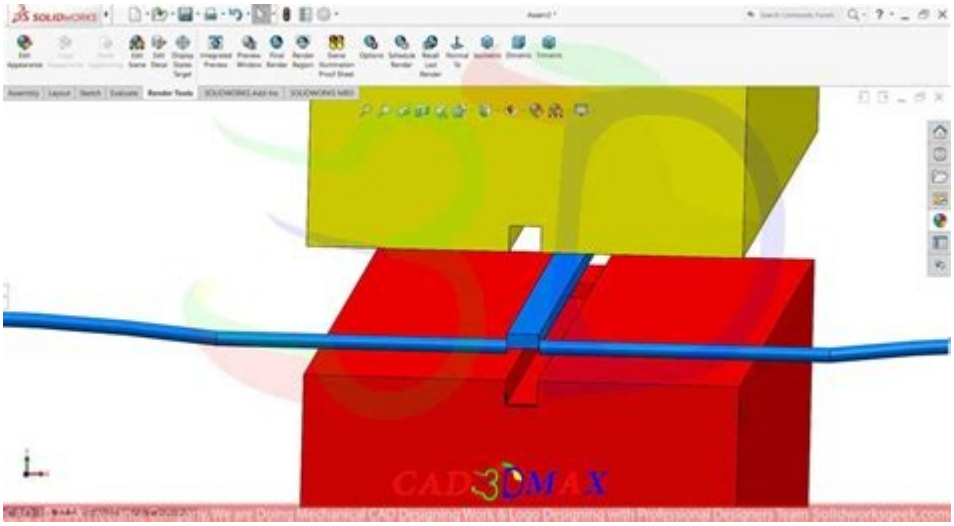


Figura 5: Ejemplo de una representación CAD.

Topología

La topología es solo un factor para las representaciones B-Rep. Las formas topológicas incluyen cuerpos (sólidos, láminas, estructuras alámbricas, bellotas) y elementos (carcasas, caras y aristas)
Mientras que la geometría define la forma de un cuerpo, la topología define el recorte de la geometría subyacente y almacena información de conectividad. Una buena topología permite notificaciones y animaciones predecibles y ocupa menos espacio en la memoria.

Dependiendo del formato CAD, puede haber diferentes tipos de entidades topológicas y diferentes requisitos para su representación. La estructura de datos topológica común de un modelo CAD surge del hecho de que cada borde está conectado a varias caras y es un requisito previo para la formación de sólidos. Pero siempre hay excepciones a la regla. Por ejemplo, en IGES no existe una entidad

como Edge. En su lugar, hay una lista de bordes que no proporciona conectividad de caras durante la conversión. Otro ejemplo de diversidad en la topología CAD es que mientras que en STEP el límite exterior es obligatorio y tiene su propia entidad (FACE_OUTER_BOUND), otro formato puede no tener tal requisito, por ejemplo, Parasolid [CITATION Mar21 \l 23562].

Operaciones booleanas sobre estructuras espaciales en CAD

Las operaciones booleanas hacen referencia al proceso de creación de un elemento a partir de la combinación de dos o más formas geométricas. Cuatro opciones disponibles que se refieren a este tipo de operaciones y que se aplican desde una capa superior a una capa inferior son:

- Unión: opción que nos permite sumar dos figuras o formas geométricas
- Sustracción: elimina el trazo de una figura que va encima de la que está en la posición de abajo
- Intersección: ofrece la posibilidad de conservar el área en donde las formas originales se sobreponen
- Diferencia: genera un efecto contrario a la intersección *(KeepCoding 2022).*

PyMOL:

Es un visor de molecular de código abierto, apropiado para producir imágenes 3D de alta calidad de moléculas pequeñas y de macromoléculas biológicas, como las proteínas.
PyMOL es una de las pocas herramientas de visualización de fuente abierta disponibles para la

biología estructural. La parte Py de su nombre alude al hecho de que extiende a, y es extensible mediante el lenguaje de programación Python, debido a lo cual puede ser extendido para realizar análisis complejos de estructuras moleculares utilizando bibliotecas disponibles para Python como NumPy o pylab. [CITATION wik19 \l 23562].

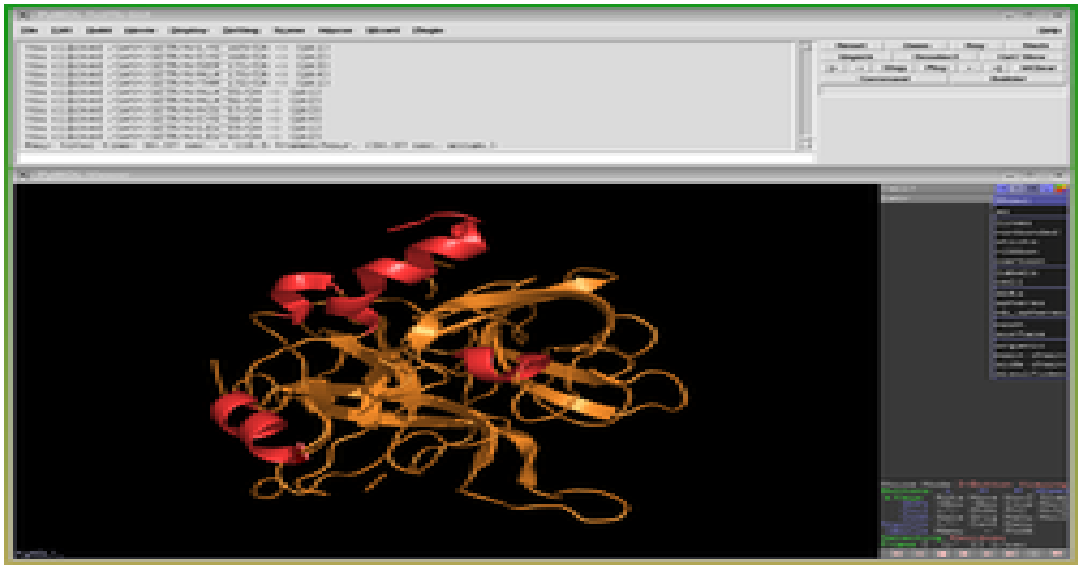


Figura 6: Captura de pantalla de Pymol. [CITATION wik19 \l 23562]

Los dispositivos microfluídicos son unos sistemas que están revolucionando los campos del análisis químico y biológico. El motivo, según explican los expertos es que permiten el estudio de determinados fluidos a una escala muy pequeña. Estos microsistemas, que integran las operaciones de ensayo y la preparación de las muestras en un chip, desarrollan las funciones de laboratorio convencionales pero de un modo portable y con un menor tiempo de respuesta. Las aplicaciones de estos sistemas son numerosas, pasando por la medicina, con capacidades para analizar fluidos del cuerpo humano como la sangre o la orina, hasta capacidades en Ingeniería Ambiental, con el propósito de estudiar la concentración de algún contaminante [CITATION Med18 \l 23562].

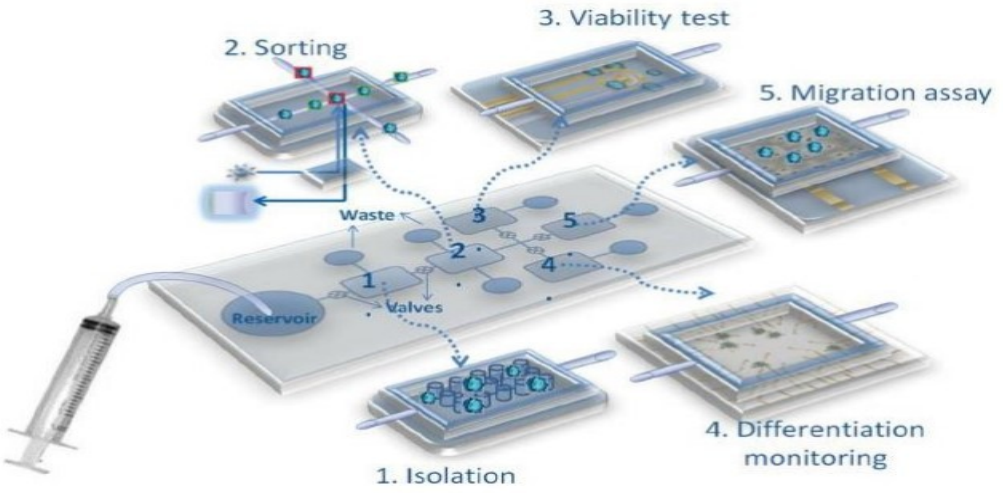


Figura 7: Dispositivo microfluídico modular [CITATION Med18 \l 23562].

1.3 Análisis de soluciones informáticas similares existentes

No se encontró ninguna investigación que haga alusión a la incorporación de modelos CAD dentro del simulador LAMMPS, aunque existen herramientas para generar los archivos CAD y también existen herramientas que hacen simulaciones de microfluidos.

1.4 Metodología para dar solución al problema de investigación

Desarrollar un buen software depende de un sinnúmero 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. El papel preponderante de las metodologías es sin duda esencial en un proyecto y en el paso inicial, que debe encajar en el equipo, guiar y organizar actividades que conlleven a las metas trazadas en el grupo *(Claro 2010).*

El éxito del producto depende en gran parte de la metodología escogida por el equipo, ya sea tradicional o ágil, donde los equipos maximicen su potencial, aumenten la calidad del producto con los recursos y tiempos establecidos.

Según [CITATION Bry18 \l 23562], las metodologías de desarrollo tradicionales o clásicas son también llamados modelos de proceso prescriptivo, y fueron planteadas originalmente para poner orden en el caos del desarrollo de software que existía cuando se empezó a generar masivamente. La historia revela que estos modelos tradicionales que fueron presentados en la década de los 60, dieron cierta estructura útil al trabajo de la Ingeniería de software y constituyen un mapa razonablemente eficaz para los equipos de desarrollo.

Hoy en día, el mundo empresarial opera en un entorno global que cambia rápidamente; por ende, se debe responder a las nuevas necesidades y oportunidades del mercado, teniendo en cuenta que el software es partícipe de casi todas las operaciones empresariales, se debe desarrollar soluciones informáticas de manera ágil para poder dar una respuesta de calidad a todo lo necesario[CITATION Bry18 \l 23562].

Las metodologías ágiles presentan como principal particularidad la flexibilidad, los proyectos en desarrollo son subdivididos en proyectos más pequeños, incluye una comunicación constante con el usuario, son altamente colaborativos y es mucho más adaptable a los cambios. De hecho, el cambio de requerimientos por parte del cliente es una característica especial, así como también las entregas, revisión y retroalimentación constante *(Cadavid 2013).*

La metodología de desarrollo de software Variación de AUP para la UCI es una variante realizada por la Universidad de las Ciencias Informáticas a la metodología ágil AUP (Proceso Ágil Unificado) y está definida por la universidad como la metodología rectora de la actividad productiva. La metodología Variación de AUP para la UCI está formada por tres fases, (Inicio, Ejecución y Cierre) para el ciclo de vida de los proyectos de la universidad [CITATION Ecu191 \l 23562].

- Inicio

Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo y decidir si se ejecuta o no el proyecto.

- Ejecución

En esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, se obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto. Durante esta fase el software es transferido al ambiente de los usuarios finales o

entregado al cliente junto con la documentación. Además, en esta transición se capacita a los usuarios finales sobre la utilización de la aplicación.

- Cierre

En el cierre se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto [CITATION Ecu191 \l 23562].

1.5 Herramientas y tecnologías para dar solución al problema de investigación

Actualmente el desarrollo de las tecnologías es cada vez más elevado. Esto trae consigo que exista gran diversidad de herramientas de software para el trabajo de desarrollo de una aplicación informática. Por tal motivo, se hace imprescindible seleccionar cuidadosamente las herramientas a utilizar para desarrollar un buen producto informático. A continuación se describen las principales herramientas y tecnologías que se utilizarán para el desarrollo de la investigación y la implementación de la solución.

1.5.1 Herramienta CASE

Las herramientas de ingeniería de software asistida por computadora, conocidas como **CASE** por sus siglas en inglés (Computer Aided Software Engineering), son diversas aplicaciones informáticas destinadas a apoyar de forma automatizada las actividades del proceso de desarrollo de software. Permiten representar el problema a través de diagramas y modelos teniendo en cuenta las características del negocio, lo cual contribuye al entendimiento de los involucrados en el desarrollo del sistema. De igual modo facilitan la automatización del ciclo de vida del desarrollo de sistemas de información, completamente o en alguna de sus fases, reduciendo así el costo de los mismos en términos de tiempo y de dinero, aumentando la productividad en el desarrollo de software.

Entre este tipo de herramientas figura el **Visual Paradigm**, programa diseñado para una amplia gama de usuarios interesados en la construcción de sistemas de software de forma fiable a través de la utilización de un enfoque orientado a objetos. Propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación. Ha sido concebida para soportar el ciclo de vida completo del proceso de desarrollo del software a través de la representación de todo tipo de diagramas. También permite la reutilización del software y facilita la portabilidad y estandarización de la documentación. [CITATION Ecu13 \l 23562]

Entre sus principales ventajas se destaca que ofrece un entorno de creación de diagramas **UML**,

generación de código y de base de datos, transformación de diagramas de entidad-relación en tablas de base de datos, es multiplataforma y además permite representar todo tipo de aplicaciones. También tiene capacidad de ingeniería directa e inversa, sus diseños son centrados en casos de usos y enfocados al negocio. Por otra parte, la Universidad de las Ciencias Informáticas cuenta con la licencia para el uso de este software por lo cual está dentro de las tecnologías definidas por la dirección de producción. Además, el equipo de desarrollo ha trabajado con esta herramienta con anterioridad. Todo esto tributa a que sea precisamente Visual Paradigm la herramienta CASE idónea para el desarrollo de la aplicación en cuestión. En el transcurso de la presente investigación, se hace uso de la herramienta "Visual Paradigm para UML", en su versión 8.0.

Editor de Código Fuente Ligero Visual Studio Code

Visual Studio Code es un editor de código fuente ligero, incluye soporte para depuración, control de Git integrado, resaltado de sintaxis, finalización de código inteligente, fragmentos de código y refactorización de código. Se encuentra disponible para sistemas operativos como Windows, macOS y Linux, posee soporte incorporado para JavaScript y Node.js y un variado ecosistema de extensiones para otros lenguajes como C++, C #, Jva, Python, PHP, Go y tiempos de ejecución .NET y Unity. Una de las características más importantes de este software puede ser actualizado con frecuencia y rapidez de actualización de la extensión para que al menos se vea un número de actualizaciones que se suelen ver con las nuevas revisiones en cada nueva versión. Visual Studio Code no sólo es una herramienta de cifrado, se pueden instalar muchas extensiones para compilar, depurar, embellecer el código y más por la instalación de extensiones *(taiwebs).*

Lenguaje de modelado UML 2.0

Para el desarrollo de la aplicación se utilizará **UML 2.0** como el lenguaje de modelado con el que se realizarán los artefactos necesarios en el proceso de desarrollo del software. UML es un lenguaje gráfico para visualizar, especificar y documentar cada una de las partes que comprende el desarrollo de software. Entrega una forma de modelar sucesos conceptuales como lo son procesos de negocio y funciones de sistema, además de sucesos concretos como lo son escribir clases en un lenguaje determinado, esquema de base de datos y componentes de software reusables

Entre sus principales características se encuentran:

- Permite modelar sistemas utilizando técnicas de programación orientadas a objeto.
- Permite documentar todos los artefactos de un proceso de desarrollo dígase requisitos, arquitectura, pruebas entre otros.

Se logra especificar todas las decisiones de análisis, diseño e implementación mediante la

construcción de modelos precisos y no ambiguos. *[CITATION His14 \l 21514]*

Lenguaje de programación a utilizar

Como lenguaje de programación a utilizar para darle solución al problema de investigación se seleccionó Python en su versión 3.9.9, debido a la necesidad de las librerías científicas que tiene en su biblioteca y permite grandes cálculos numéricos *(Virtanen et al. 2020).*

Python es un lenguaje de escritura rápida de código abierto, robusto y escalable. Permite plasmar ideas complejas con solo pocas líneas de código. Puede ser utilizado en todos los sistemas operativos y donde se utiliza en la creación de cualquier tipo de aplicación ya sea minería de datos hasta aplicaciones web complejas Su curva de aprendizaje es sencilla. Otra de sus ventajas se debe a que es un lenguaje de alto nivel orientado a objetos donde se agregan tanto datos como funcionalidades favoreciendo la reusabilidad, el mantenimiento y la fiabilidad del código. Es multiparadigma, ofreciendo una programación estructurada, imperativa y funcional. No es necesario compilarlo debido a que es un lenguaje interpretado con un desarrollo mucho más rápido. Puede integrarse a otros programas escritos en lenguajes como C y C++. Se debe tener en cuenta demás que contiene un conjunto de librerías disponibles que dan la posibilidad de incorporar y utilizar funcionalidades extras.

Librería SciPy

SciPy es una biblioteca de rutinas numéricas para el lenguaje de programación Python que proporciona bloques de construcción fundamentales para modelar y resolver problemas científicos. SciPy incluye algoritmos de optimización, integración, interpolación, problemas de valores propios, ecuaciones algebraicas, ecuaciones diferenciales y muchas otras clases de problemas; también proporciona estructuras de datos especializadas, como matrices dispersas y árboles k-dimensionales. SciPy está construido sobre NumPy, que proporciona estructuras de datos de matrices y rutinas numéricas rápidas relacionadas, y SciPy es en sí mismo la base sobre la que se construyen bibliotecas científicas de nivel superior, incluyendo scikit-learn y scikit-image. Científicos, ingenieros y otras personas de todo el mundo confían en SciPy. Por ejemplo, los scripts publicados utilizados en el análisis de las ondas gravitacionales importan varios subpaquetes de SciPy, y el proyecto de imágenes de agujeros negros M87 cita a SciPy *(Virtanen et al. 2020).*

PythonOCC: es un proyecto que tiene como objetivo proporcionar toda la gama de Tecnología OpenCASCADE (OCCT), funciona a través del módulo Python OCC. Por otro lado, proporciona acceso a todas las clases y funciones de OCCT, por lo que es complejo pero también muy potente.

El Módulo de las Piezas tiene los métodos Parte._aPythonOCC_() y Parte._dePythonOCC_() para intercambiar TopoDS_Shape (Piezas TopoForma) entidades hacia y desde PythonOCC. Estos métodos nos permiten utilizar toda la potencia de OCCT en Python y luego poner las formas resultantes de nuevo en los objetos de FreeCAD [CITATION Fre21 \l 23562].

Framework web a utilizar

Este módulo que se integra al proyecto SIMBOX/ECVM está implementado en Django por lo que el requisito fundamental para darle solución al problema de investigación es seguir utilizando como tecnología el framework Django en su versión 4.8. Es uno de los frameworks más populares, completos y toma especial importancia en los temas de seguridad ayudando a los desarrolladores a detectar fallos que se encuentren en el código que comprometan la seguridad de la aplicación. Es una tecnología gratuita de código abierto que está en constante mejora de sus funcionalidades *(Holovaty y Kaplan-Moss 2009).*

Elección del framework Django

Se seleccionó Django como tecnología a llevar a cabo debido a sus grandes ventajas sobre las demás tecnologías de desarrollo web y por ser la utilizada en el proyecto SIMBOX/ECVM. Django además de haber sido utilizado en más de 367 000 proyectos de desarrollo web, también cuenta con la opción que incorpora su propio ORM y dispone de sus propios modelos de datos. Cuenta con un sistema de autenticación de usuarios superior al de Flask y tiene gran rendimiento y flexibilidad permitiendo escalar proyectos de forma sencilla. Trabaja bajo el patrón Modelo-Vista-Controlador garantizando el desarrollo ágil y reutilizable. Django es el *framework* de desarrollo en Python con mayor cantidad de recursos accesibles, incluye opciones de protección para las aplicaciones por ejemplo contra ataques de SQL injection o ataques XSS. Proporciona una estructura de código autogenerador ayudando a un desarrollo más ágil y cuenta con su propio panel de administración para base de datos *(Holovaty y Kaplan-Moss 2009).*

JSON

Una de las funcionalidades que posee Django es la capacidad de devolver una respuesta JavaScript Object Notation (JSON) que es un formato de un conjunto de consultas y datos de un objeto almacenado en una base de datos tanto en Java como en Python. Común y ligero con una fácil legibilidad para los desarrolladores. La generación de un JSON será necesaria en el proceso de peticiones de generación de Espacios de Simulación *(Holovaty y Kaplan-Moss 2009).*

Sistema de Gestión de Base de Datos

Un sistema de gestión de bases de datos consiste en una colección de datos interrelacionados y un conjunto de programas para acceder a los mismos, donde se permite el almacenamiento, manipulación y consulta de datos pertenecientes a una base de datos organizada en uno o varios ficheros. Durante el transcurso de la investigación, se emplea PostgreSQL Admin, el cual es un sistema de gestión de bases de datos objeto-relacional. Se encuentra distribuido bajo licencia BSD y tiene su código fuente disponible libremente. Es uno de los sistemas de gestión de bases de datos de código abierto más potente del mercado. Utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Esto ofrece como ventaja que un fallo en uno de los procesos no afecte al resto, garantizando así que el sistema continúe funcionando *(Momjian 2000).*

El uso de esta Base de Datos está asociado al almacenamiento de los modelos convertidos de CAD a LAMMPS de forma reutilizable. Entre sus principales características se encuentra la creación de copias de seguridad mientras se usa la base de datos, lo que permite salvar las operaciones realizadas cuando ocurra un suceso inesperado. Además, utiliza para el control de concurrencia un modelo multiversión, creando una “copia” de la base de datos para cada usuario. Los cambios que se realizan solo se reflejan en la base de datos del sistema cuando hayan finalizado todas las operaciones, actualizando posteriormente todas las “copias” a todos los usuarios conectados. De esta forma se protegen las transacciones y permite que múltiples usuarios trabajen simultáneamente sobre una misma base de datos sin que ocurran interbloqueos en la ejecución de las consultas *(Momjian 2000).*

Conclusiones del capítulo

A partir del análisis teórico de la investigación fue posible comprender el objetivo planteado. El estudio de los conceptos relacionados permitió la comprensión de términos que inusualmente se conocen; facilitando vocablos concretos para familiarizar al autor con la temática. Como no se hace alusión a la incorporación de modelos CAD dentro del simulador LAMMPS es de total importancia continuar con el desarrollo de una solución. Como parte del análisis de las tecnologías de desarrollo se escogieron las necesarias y las más fáciles de utilizar en teoría para poder llevar a cabo dicho módulo. Finalmente, se utilizó la metodología AUP UCI por su adecuación a los procesos productivos de la universidad.

CAPITULO II: Análisis, diseño e implementación del módulo de carga de modelos CAD para simulaciones de microfluidos.

Introducción

Para realizar la carga de las estructuras de un dispositivo microfluídico, representado como un archivo CAD, en las simulaciones que utilizan el software LAMMPS, es necesario establecer un nexo entre la información contenido en el estándar CAD, los modelos de superficie de objetos sólidos y su representación física en un software de simulación de Dinámica Molecular de partículas. Se realiza un estudio sobre la carga e interpretación de los modelos de superficie contenidos en el archivo CAD, diseñando un algoritmo de transformación para la representación mediante partículas coloidales superficiales de la estructura superficial de las cavidades del dispositivo microfluídico.

2.1 Modelo de dominio

Un Modelo de dominio es la entrada para muchos de los artefactos que se construyen en un proceso software; mostrando así las clases conceptuales significativas en un dominio del problema. Además, es el artefacto clave del análisis orientado a objetos [CITATION Peñ18 \l 23562].

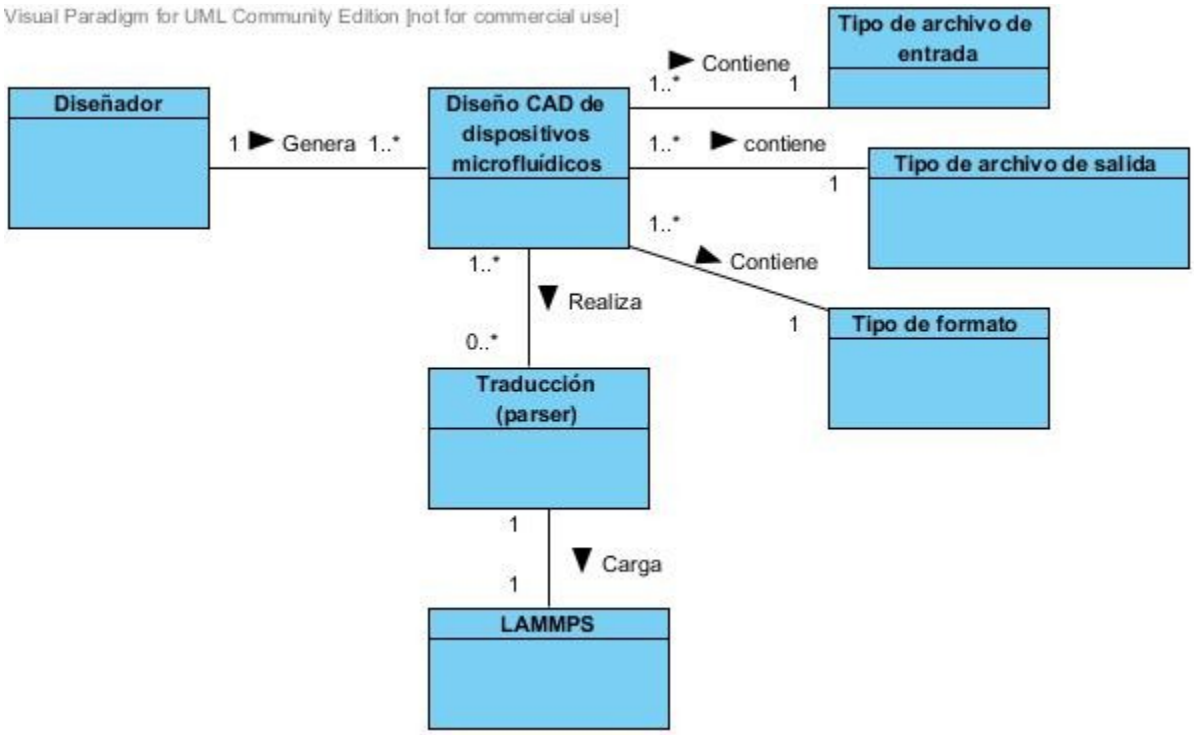


Figura 8: Diagrama de dominio del módulo de carga de modelos CAD para dispositivos microfluídicos en LAMMPS.

Fuente: Elaboración propia

2.1.1 Descripción de los conceptos

Tabla 2: Descripción de los conceptos.

Concepto	Descripción
Desarrollador	Actor real principal del proceso de carga de modelos CAD
Diseño CAD de dispositivos microfluídico	Contiene todo el diseño que permite recrear infinidad de procesos a una escala microscópica.
Traducción (parser)	Programa informático que analiza una cadena de símbolos según las reglas de una gramática formal
Tipo de archivo de entrada	El tipo de archivo de entrada es STEP
Tipo de archivo de salida	El tipo de archivo de salida es .cell
Tipo de formato	El formato con el que se va a desarrollar es FreeCAD
LAMMPS	Sistema que simula procesos moleculares.

2.2 Propuesta de Solución

Se propone crear un módulo de carga de modelos CAD para simulaciones de microfluidos en el proyecto ECVM, teniendo como cliente a usuarios generales de LAMMPS.

En el módulo lo primero que se va a hacer es cargar el archivo stl. De aquí se extraen todos los objetos que representan las distintas estructuras geométricas que hay dentro, sobre estos objetos se realiza un conjunto de operaciones booleanas para extraer la superficie exterior (sería la parte exterior del objeto) e interior (serían las cavidades internas de los canales de los microfluidos). A través de estas operaciones se obtienen los vértices exteriores y las caras interiores; siendo esto lo que se lee y se grafica. Al traducirlo se pueden situar en estas posiciones las partículas coloidales que se representan en archivos de LAMMPS. La forma en que se guarda esta información en LAMMPS es en un archivo .cell que tiene un conjunto de etiquetas.

2.3 Concepción del sistema

La estructura, el diseño y las funcionalidades que debe ejecutar el módulo para alcanzar su objetivo, son vitales para entender el curso de la investigación. Es asociado a esta idea que se hace preciso la gestión documental de la concepción del módulo, mostrando los principales detalles del producto de software que se propone como solución. Esto permitirá un mayor entendimiento del proyecto, no solo para su desarrollo, sino también para el cliente y la creación de futuras versiones.

Para el proceso de carga y parser del modelo CAD que no es más que un programa informático que analiza una cadena de símbolos según las reglas de una gramática formal; se utilizó la librería pythonOCC, la cual proporciona un contenedor de Python para la tecnología OpenCASCADE (OCCT). Hay un doble beneficio al usar OCCT junto con Python:

- sin necesidad de conocimientos de C++, entorno complejo de compilador de C++ o flujos de trabajo de compilación, desarrollo multiplataforma nativo
- acceda al amplio ecosistema de ciencia de datos de Python (NumPy, scipy, meshers, bibliotecas de IA, Jupyter y Voila)

Esto hace que la complejidad de OCCT sea fácilmente accesible para los usuarios finales empresariales.

Open Cascade Technology (OCCT) es una colección de bibliotecas C++ que juntas constituyen un núcleo de diseño asistido por ordenador (CAD) profesional para modelar objetos 2D y 3D, y construir herramientas especializadas para la fabricación, simulación o visualización. OpenCASCADE es el corazón de las capacidades geométricas de FreeCAD. Las clases geométricas de OCCT están en su mayoría implementadas y disponibles en FreeCAD. También proporciona funciones internas para leer

y escribir diferentes formatos de archivos como STEP e IGES, y para realizar proyecciones 2D [CITATION Fre22 \l 23562].

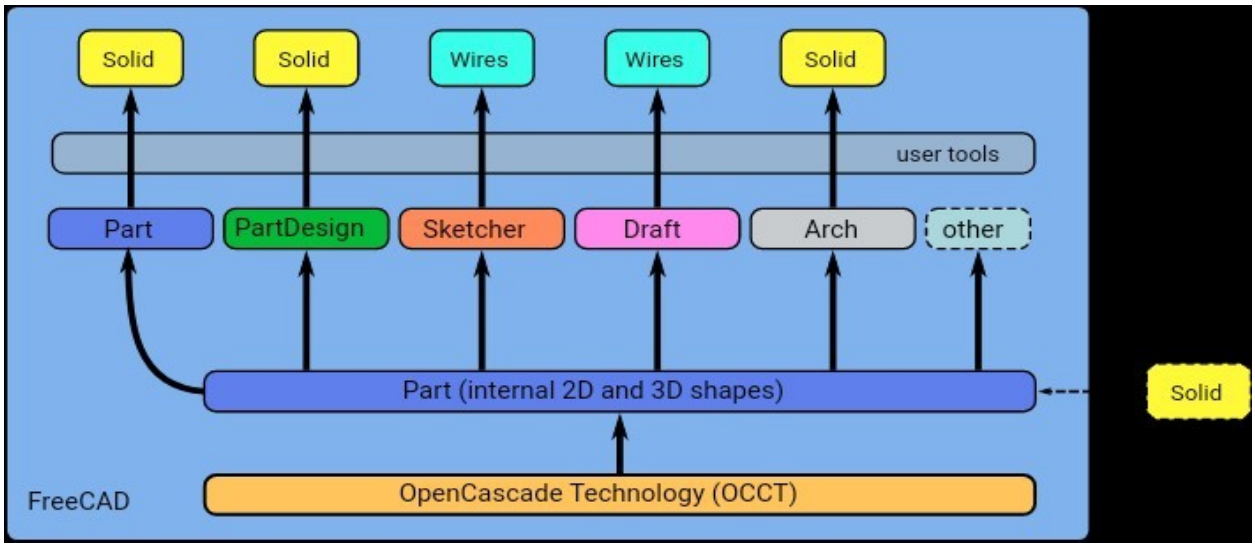


Figura 9: OpenCascade proporciona las clases geométricas básicas y las funciones de dibujo al módulo Pieza, que luego son utilizadas por todos los ambientes de trabajo en FreeCAD. [CITATION Fre22 \l 23562].

2.3.1 Visión y alcance del sistema

El módulo de carga tendrá como objetivo fundamental:

- Permitir la carga de diseños CAD de dispositivos de microfluidos en el software de simulación LAMMPS.

Para el cumplimiento de estos objetivos se implementan funcionalidades óptimas desde el punto de vista del código fuente, que permitan una fácil interacción y una respuesta rápida a las peticiones de los usuarios.

2.3.2 Planificación del proyecto por roles

La planificación del proyecto es la ordenación sistemática de las tareas para lograr un objetivo, donde se expone lo que se necesita hacer y cómo debe llevarse a cabo. Además, es importante definir quién realizará cada tarea y darle un nivel de importancia a cada una, pues de su cumplimiento en tiempo depende el éxito del producto final. Para el desarrollo de la solución y acorde a las necesidades del sistema a implementar, siguiendo las líneas de la metodología de software AUP, se definen los siguientes roles:

- **Jefe de proyecto**: es el responsable de tomar las decisiones finales, acerca de estándares y convenciones a seguir durante el desarrollo de un producto determinado. Participa en la selección de objetivos y requerimientos.

Tareas fundamentales:

1 Realiza la planificación del proyecto, a lo largo de todo el ciclo de vida, incluida la planificación en detalle de cada sprint.
2 Acelera que se consigan los objetivos propuestos en la reunión de planificación del Sprint.
3 Controle el progreso del desarrollo del producto.

- **Desarrollador:** Por medio del mismo es posible que las especificaciones del sistema se conviertan en código fuente ejecutable. El éxito del software depende en gran medida del conocimiento del lenguaje de programación seleccionado para el desarrollo, una buena definición de requisitos y un correcto entendimiento entre el analista y programador.

- **Stakeholder (cliente):** En la metodología AUP-UCI el cliente forma parte del equipo de desarrollo. En ese caso este rol es responsable de, junto al analista, establecer los requisitos del sistema. Acompaña a cada rol dentro del equipo de desarrollo en todas las fases del software, apoyando el trabajo que se realiza y dando su criterio acerca de los resultados que se obtienen en cada etapa. Participa en el diseño y ejecución de las pruebas de software.

Figura 10: Representación de diferentes roles en un proyecto

A continuación se muestran los roles definidos por personas para el desarrollo del módulo de carga de modelos CAD para simulaciones de microfluidos.

Tabla 3: Roles y responsabilidades del módulo

Roles	**Responsabilidades**	**Nombre**
Jefe de proyecto	Representa al equipo de desarrollo, guía y organiza el proceso de desarrollo del software, conoce el proceso de negocio establecido y conoce las tecnologías a utilizar.	Edisel Navas Conyedo
Desarrollador	Escribe, testea y construye software	Participantes del proyecto ECVM
Stakeholder (cliente)	Establece el proceso de negocio, define los requisitos funcionales del sistema, apoya y participa en cada fase del desarrollo del software, participa en el proceso de pruebas del	Usuarios generales de LAMMPS

	software	

2.4 Especificación de requisitos

La especificación de requisitos de software es una descripción completa del comportamiento del módulo que se va a desarrollar. Está dirigida tanto al cliente como al equipo de desarrollo.

Requisitos candidatos

1. Leer archivos de FreeCAD.
2. Generar los modelos de superficies.
3. Generar un mallado de partículas que representen las paredes interiores de los canales en el dispositivo.
4. Incluir un potencial de interacción entre las partículas de superficie y los componentes del microfluidos.

2.4.1 Requisitos Funcionales

A decir de Ian Somerville, los requisitos funcionales (RF) “son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que este debe reaccionar ante entradas particulares y de cómo se debe comportar en situaciones particulares” (Sommerville, 2005). A continuación, se listan los requisitos funcionales de la solución del módulo:

Tabla 4: Requisitos funcionales del módulo.

Prioridad	Ítem*	Descripción	Estimación	Estimado por
Baja	RF1	Leer archivos de FreeCAD.	2	Desarrollador
Baja	RF2	Leer los modelos de superficie	2	Desarrollador
Baja	RF3	Identificar las superficies a representar	2	Desarrollador
Baja	RF4	Identificar los vértices exteriores de la superficie	4	Desarrollador
Media	RF5	Obtener las caras de la superficie	5	Desarrollador
Baja	RF6	Mostrar los modelos de superficie cuando se va a reutilizar un modelo anterior	2	Desarrollador

Baja	RF7	Listar los modelos de superficie cuando se va a reutilizar un modelo anterior	2	Desarrollador
Media	RF8	Definir los modelos de superficie cuando se va a reutilizar un modelo anterior	3	Desarrollador
Baja	RF9	Leer la información de los vértices	2	Desarrollador
Baja	RF10	Leer la información de las caras de la superficie	2	Desarrollador
Media	RF11	Crear el mallado de partículas	4	Desarrollador
Media	RF12	Guardar el archivo .cell de LAMMPS incluyendo un potencial de interacción tipo SPH	4	Desarrollador
Alta	RF13	Mostrar el mayado de partículas en PYMOL	3	Desarrollador
Media	RF14	Adicionar potencial de interacción	2	Desarrollador
Baja	RF15	Eliminar potencial de interacción	2	Desarrollador
Baja	RF16	Actualizar potencial de interacción	2	Desarrollador
Media	RF17	Modificar potencial de interacción	3	Desarrollador

2.4.2 Requisitos no funcionales

Somerville afirma que los requisitos no funcionales (RNF) “como su nombre lo sugiere, son aquellos requerimientos que no se refieren a las funciones específicas que proporciona el sistema, sino a las propiedades emergentes de este como la fiabilidad, el tiempo de respuesta y la capacidad de almacenamiento” (Sommerville, 2005). En resumen, son un conjunto de reglas que hacen del sistema, un producto agradable, usable, rápido o confiable. Estos requisitos se encargan de especificar las propiedades que debe tener el producto, así como las restricciones del entorno o de la implementación, rendimiento, dependencias de la plataforma, facilidad de mantenimiento, extensibilidad, fiabilidad, entre otras. Seguidamente se definen los RNF pertenecientes al módulo a desarrollar:

Tabla 5: Requisitos no funcionales del módulo.

Requisitos no funcionales	
Fiabilidad	- Las funcionalidades principales del módulo estarán orientadas a íconos para un mayor reconocimiento por parte del usuario.
Usabilidad	- El módulo podrá ser usado por personas con conocimientos básicos en el manejo de computadoras. Las funcionalidades principales del sistema estarán orientadas a iconos para un mayor reconocimiento por parte del cliente. - La propuesta de solución debe brindar un acceso fácil y rápido.
Restricciones de diseño	- Lenguaje de desarrollo: Python. - El sistema operativo a usar en el entorno de desarrollo puede ser Linux o Windows. - Los artefactos del análisis se realizarán con Visual Paradigm. - La interfaz de administración será implementada con el framework Django 2.0.
Rendimiento	- El módulo debe responder en un tiempo menor de los 10 segundos.
Portabilidad	- El módulo debe ser multiplataforma
Eficiencia	- El tiempo de respuesta estará dado por la cantidad de información a procesar, entre mayor cantidad de información mayor será el tiempo de procesamiento. De igual modo, la velocidad de procesamiento de la información, la actualización y la recuperación dependerán de la cantidad de información que tenga que procesar el módulo.

2.5 Historias de Usuario

Las historias de usuario son descripciones cortas y simples de una característica contada desde la perspectiva de la persona que desea la nueva capacidad. Estas son parte de un enfoque ágil que

ayuda a cambiar el enfoque de escribir sobre los requisitos a hablar sobre ellos [CITATION Use18 \l 23562].

En el cuarto escenario de AUP, en su variante UCI, los requisitos se administran utilizando historias de usuario, por lo que para la solución descrita se describen algunas Historias de Usuario.

Tabla 6: Historia de Usuario 1

Número: HU_1	**Requisito:** Leer archivos de FreeCAD.
Programador: Melissa Cordero Alvarez	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 2 días
Riesgo en Desarrollo: Bajo	**Tiempo Real:** 8 días
Descripción: El usuario selecciona un modelo de FreeCAD de un dispositivo y lo visualiza con una herramienta para estar seguro que es lo que él necesita.	
Observaciones: aquí se observa la estructura geométrica al inicio echa en CAD y observa la representación en forma de partículas coloidales al final después de la transformación y compara esa dos instancias para ver si coinciden.	
Prototipo elemental de interfaz gráfica de usuario:	

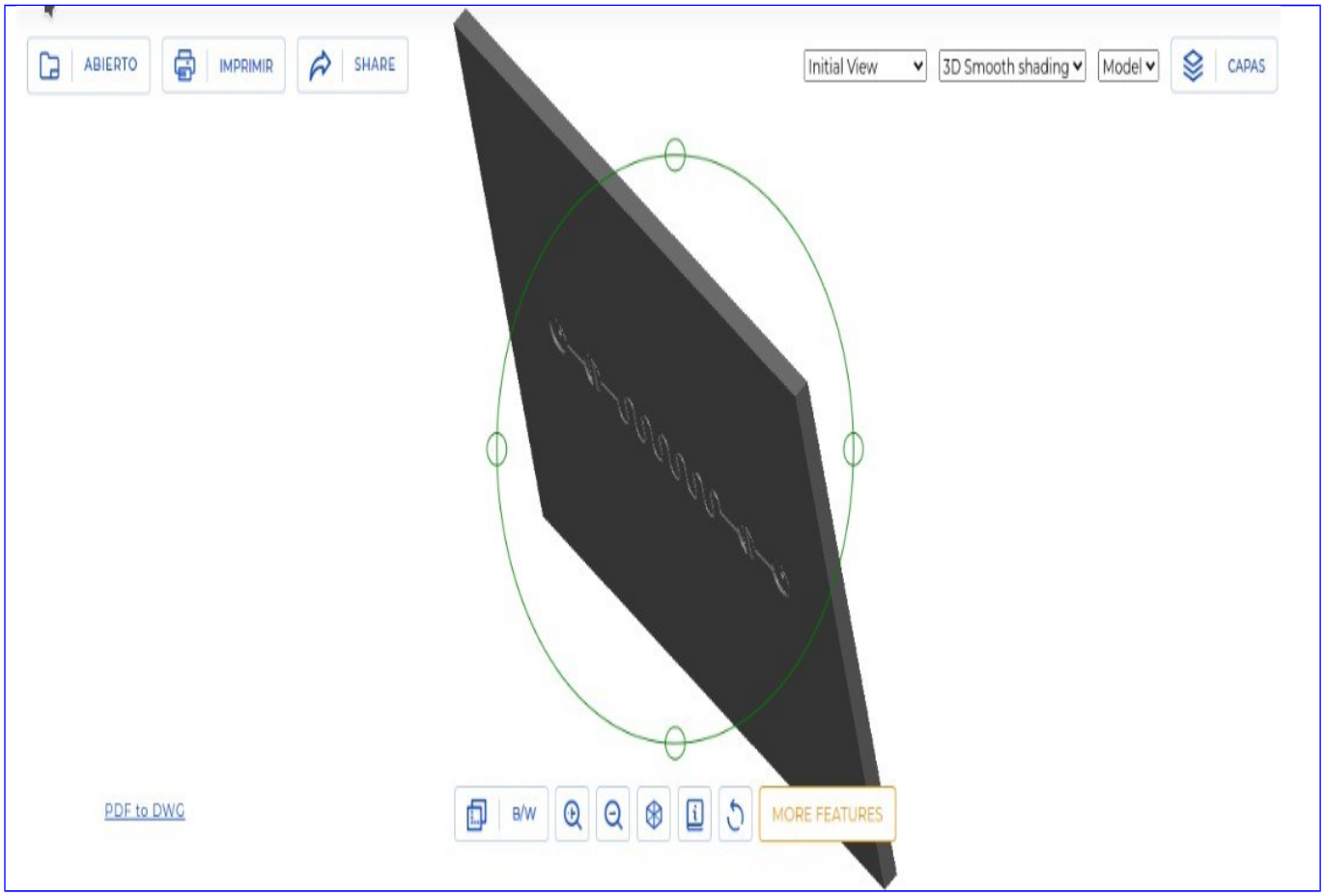


Tabla 7: Historia de Usuario 2

Número: HU_2	**Requisito:** Leer los modelos de superficie
Programador: Melissa Cordero Alvarez	**Iteración Asignada:** 2
Prioridad: Baja	**Tiempo Estimado:** 3 días
Riesgo en Desarrollo: Alto	**Tiempo Real:** 10 días
Descripción: se leen todas las estructuras espaciales que están dentro del archivo y de cada una de esas estructuras se leen las superficies de cada una y a partir de las operaciones booleanas de CAD se obtiene la superficie más externa; que es la superficie a representar del software.	
Observaciones: tiene un alto nivel de complejidad porque hay que leer las estructuras, optimizar el código para su rápida ejecución; pues hacer las operaciones booleanas es complejo.	

Prototipo elemental de interfaz gráfica de usuario:

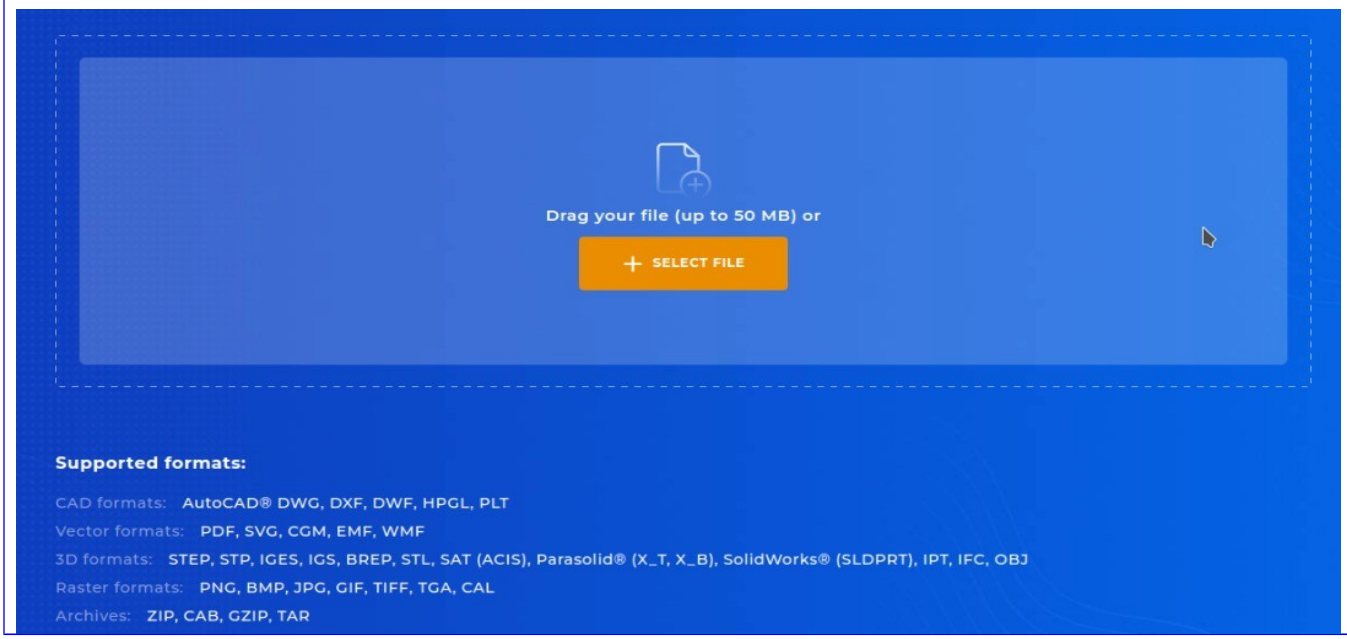
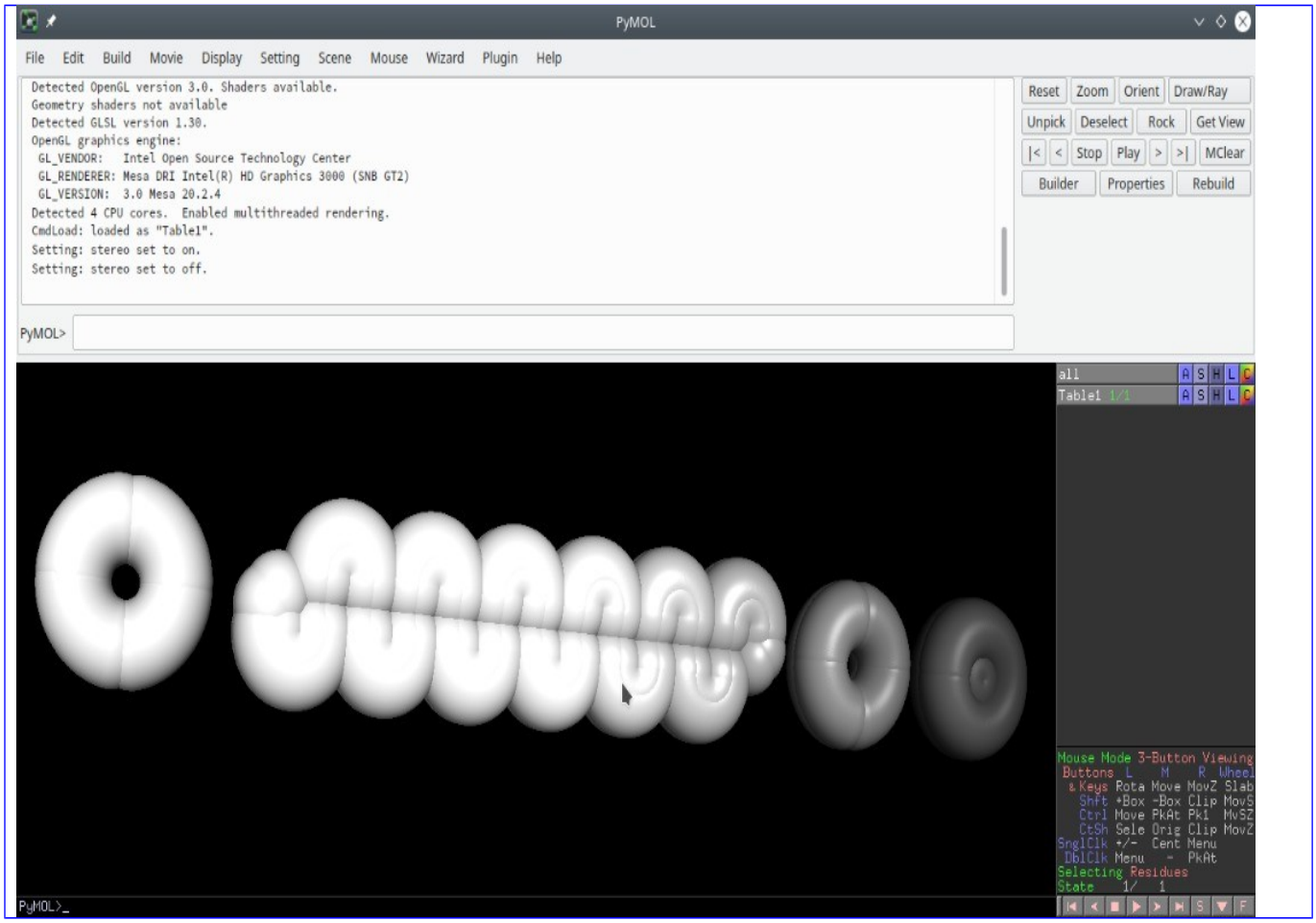


Tabla 8: Historia de Usuario 3

Número: HU_3	**Requisito:** Identificar las superficies a representar
Programador: Melissa Cordero Alvarez	**Iteración Asignada:** 3
Prioridad: Baja	**Tiempo Estimado:** 2 días
Riesgo en Desarrollo: Baja	**Tiempo Real:** 5 días
Descripción: una vez extraída esa superficie exterior se cogen los vértices y en cada uno de los vértices depositar una partícula coloidal con un potencial de interacción.	
Observaciones:	
Prototipo elemental de interfaz gráfica de usuario:	

Tabla 9: Historia de Usuario 4

Número: HU_4	**Requisito:** Mostrar el mayado de particulas en PYMOL
Programador: Melissa Cordero Alvarez	**Iteración Asignada:** 13
Prioridad: Media	**Tiempo Estimado:** *3* días
Riesgo en Desarrollo: Alta	**Tiempo Real:** 10 días
Descripción: cargar el .cell utilizando una herramienta que tiene PYMOL, que es capaz de leer los .cell de LAMMPS, y con esto se puede representar espacialmente cada una de esas partículas coloidales y comprobar si el modelo que estoy visualizando se corresponde con el modelo de entrada del archivo CAD.	
Observaciones: hay que usar una herramienta adicional que tiene PYMOL para leer el .cell de LAMMPS.	
Prototipo elemental de interfaz gráfica de usuario:	



2.6 Diseño de gestión y arquitectura

2.6.1 Arquitectura del sistema

En la construcción de software existen diversos componentes que proporcionan una filosofía de trabajo para los equipos de desarrollo, creando una visión que unifica la forma en la que los miembros del equipo ven el sistema y además impone una transformación del mismo. Entre los mencionados componentes se encuentran los estilos arquitectónicos, patrones arquitectónicos y patrones de diseño, los cuales se encuentran estrechamente relacionados, conformando la base de la arquitectura de un software. *[CITATION Urq14 \l 23562]*

2.6.2 Patrón arquitectónico

Los patrones arquitectónicos, o patrones de arquitectura, también llamados arquetipos ofrecen soluciones a problemas de arquitectura de software en ingeniería de software. Dan un descripción de los elementos y el tipo de relación que tienen junto con un conjunto de restricciones sobre cómo pueden ser usados. Un patrón arquitectónico expresa un esquema de organización estructural esencial para un sistema de software, que consta de subsistemas, sus responsabilidades e interrelaciones. En

comparación con los patrones de diseño, los patrones arquitectónicos tienen un nivel de abstracción mayor. [CITATION Wik \l 23562]

La arquitectura de tres niveles es una arquitectura de software de aplicación bien establecida que separa las aplicaciones en tres niveles de informática lógica y física: el nivel de presentación o la interfaz del usuario, el nivel de aplicación o donde se procesan los datos, y el nivel de datos donde se almacenan y gestionan los datos asociados con la aplicación.
El beneficio principal de la arquitectura de tres niveles es que debido a que cada nivel se ejecuta en su propia infraestructura, cada nivel puede ser desarrollado simultáneamente por un equipo de desarrolladores distinto y se puede actualizar o escalar según sea necesario sin que afecte a los demás niveles.

- Nivel de presentación: es la interfaz de usuario y de comunicación de la aplicación, donde el usuario final interactúa con la aplicación. Su objetivo principal es mostrar información al usuario y recopilar datos de este. Este primer nivel se puede ejecutar con un navegador web como una aplicación de desktop o una interfaz gráfica de usuario (GUI). Los niveles de presentación web se suelen desarrollar utilizando HTLM, CSS Y JavaScript.
- Nivel de aplicación: conocido también como nivel lógico o medio, es el núcleo de la aplicación. En este nivel, se procesa la información recopilada en el nivel de presentación, a veces con otra información en el nivel de datos, mediante la lógica empresarial. Este normalmente se desarrolla utilizando Python, Java, Perl, PHP o Ruby, y se comunica con el nivel de datos mediante llamadas API.
- Nivel de datos: a veces denominado nivel de base de datos, nivel de acceso o backend, es donde se almacena y gestiona la información procesada por la aplicación. Puede ser un sistema de gestión de base de datos relacional como PostgreSQL, MySQL, MariaDB. [CITATION IBM20 \l 23562]

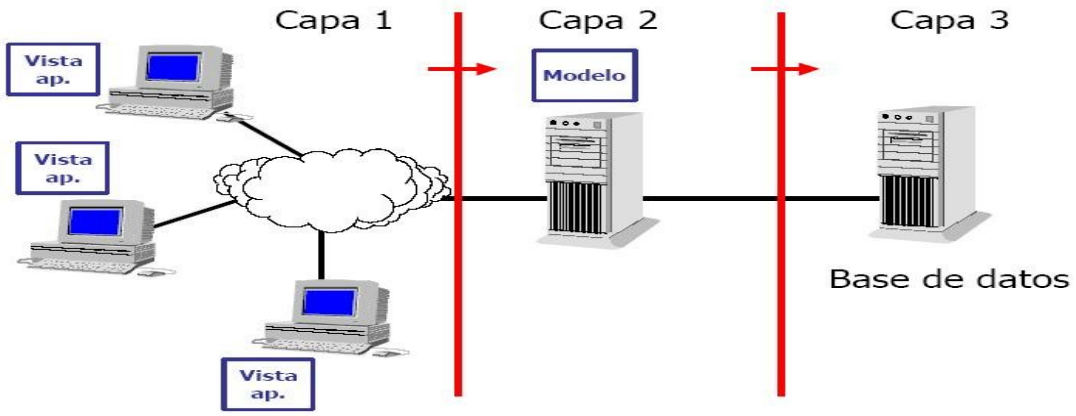
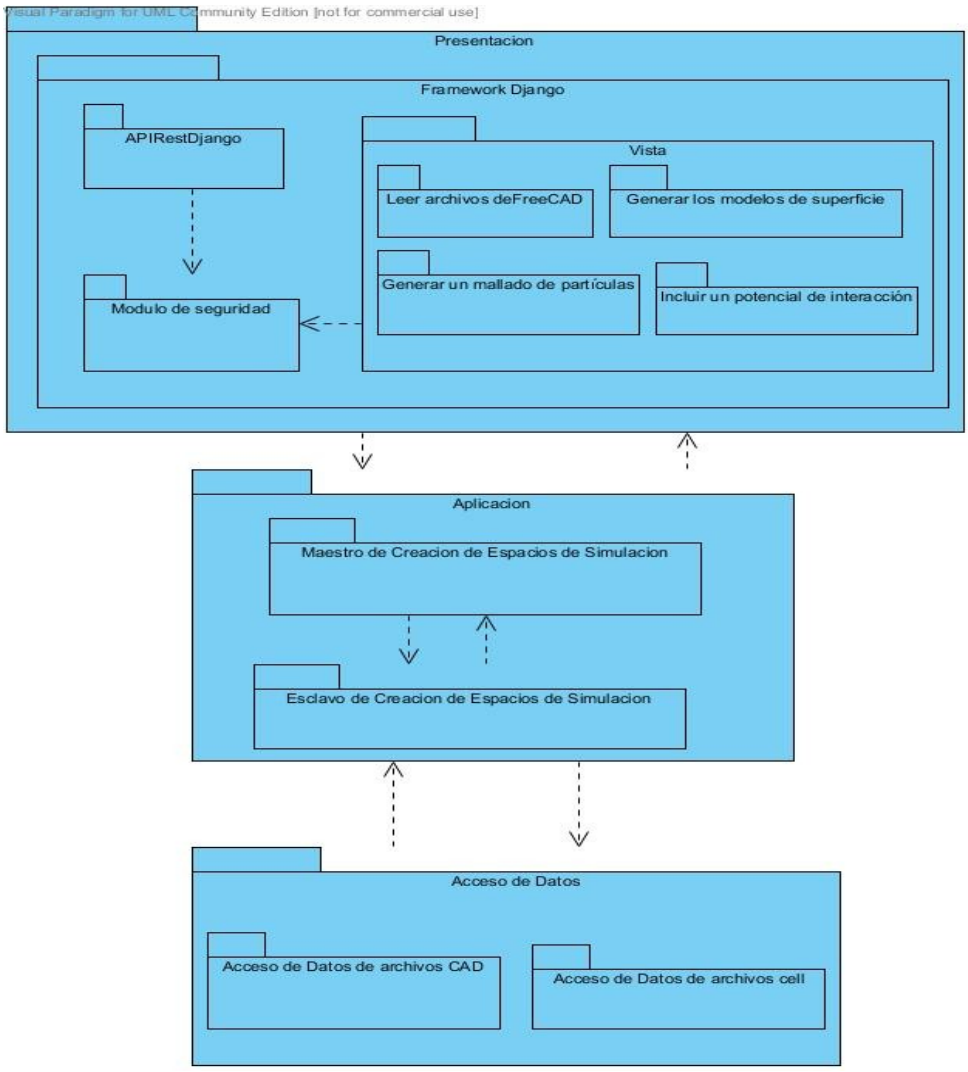


Figura 11: Forma general del Patrón Arquitectónico Tres Capas [CITATION Car04 \l 23562]



Figuras 12: Arquitectura de desarrollo del módulo.

2.6.3 Descripción de la arquitectura

En la presente solución, en la capa Presentación se localiza el Framework Django, el cual brinda servicio APIRest y los paquetes de la Vista. En la capa Aplicación se encuentra la lógica del módulo. La estructura de esta capa sigue la arquitectura Maestro-Esclavo, donde el Maestro Creador de Espacios de Simulación crea y controla instancias locales y remotas de Esclavos de Creación de Espacios de Simulación. La capa de Acceso a Datos permite una interfaz de comunicación de

datos de archivos CAD para su reutilización o extensión, también contiene la interfaz de conexión de datos de archivos .cell.

2.7 Diagrama de Clases de Diseño

Un Diagrama de Clases de Diseño muestra la especificación para las clases software de una aplicación. Incluye la siguiente información:

- Clases, asociaciones y atributos.
- Interfaces, con sus operaciones y constantes.
- Métodos.
- Navegabilidad.
- Dependencias.

A diferencia del Modelo Conceptual, un Diagrama de Clases de Diseño muestra definiciones de entidades software más que conceptos del mundo real. (*Diagrama de Clases de Diseño | LENGUAJE DE MODELADO UNIFICADO - UML*, s. f.)

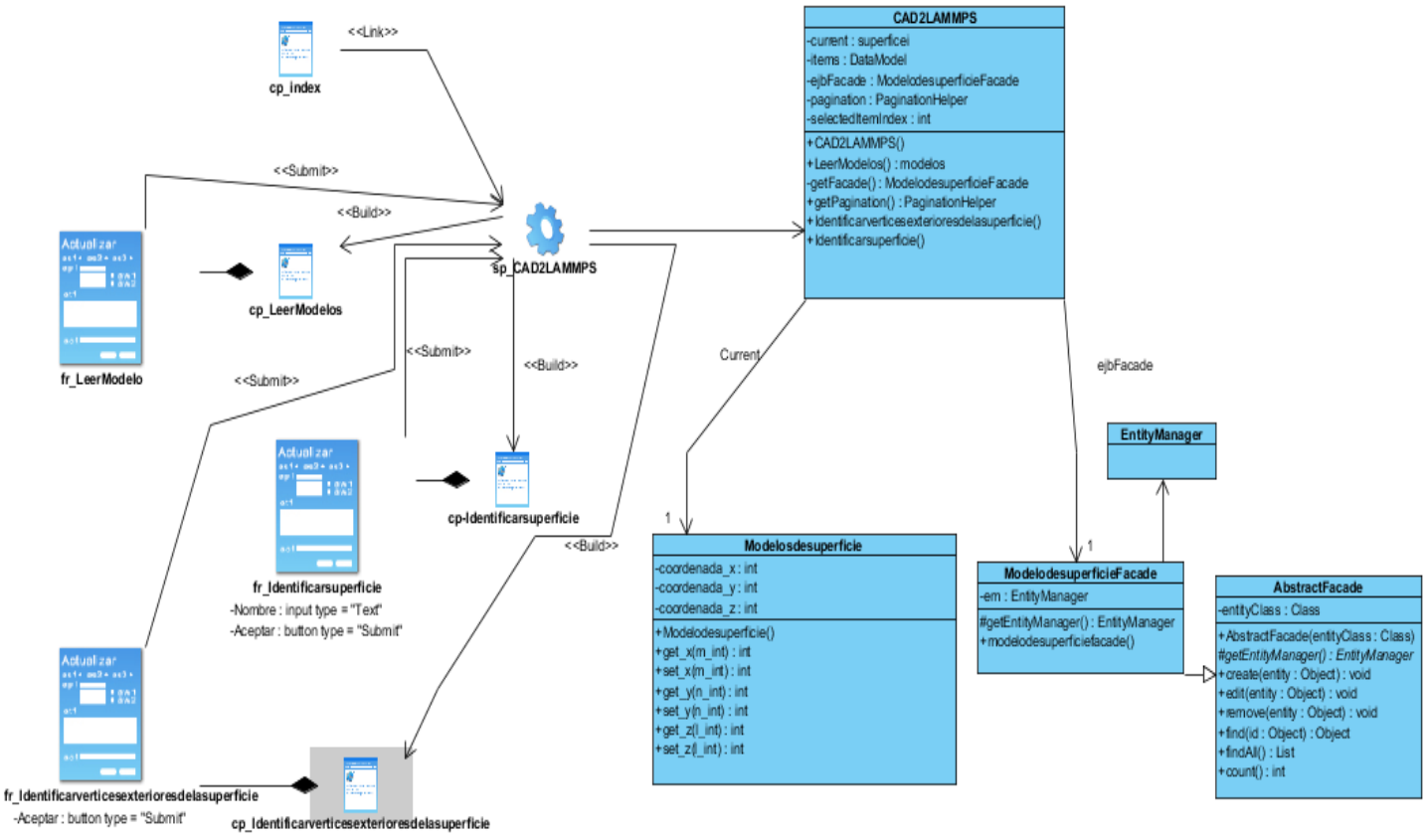


Figura 13: Diagrama de Clases- Generar modelos de superficie.

En los anexos está el resto de los diagramas.

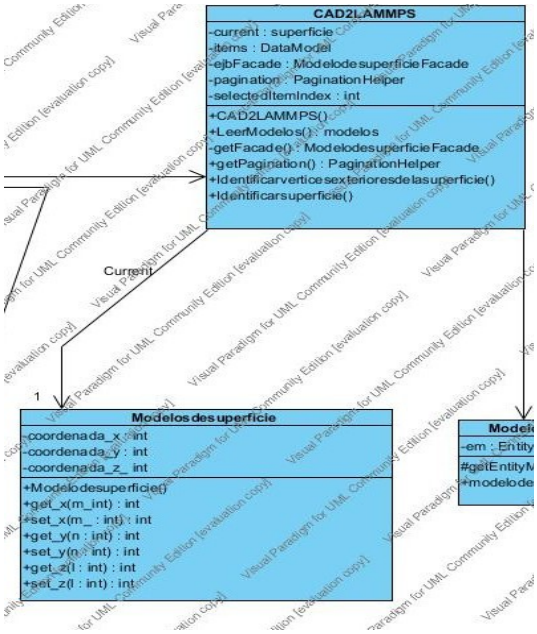
2.8 Patrones del diseño

Los patrones de diseño representan soluciones a problemas que surgen cuando se desarrolla un software en un contexto particular. ´´Un patrón de diseño es una descripción de clases y objetos comunicándose entre sí, adaptada para resolver un problema de diseño general en un contexto particular´´ [CITATION Urq14 \l 23562]. Para realizar el diseño del módulo se han utilizado algunos de los patrones existentes, los cuales se mencionan a continuación:

Patrones GRASP

Los Patrones Generales de Software para Asignar Responsabilidades (GRASP, por sus siglas en inglés), son parejas de problema y solución, que codifican buenos principios y sugerencias relacionados frecuentemente con la asignación de responsabilidades. Describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. Los patrones GRASP utilizados en la investigación son:

Creador: Este patrón guía la asignación de responsabilidades relacionadas con la creación de objetos. El propósito fundamental de este patrón es encontrar un creador que se debe conectar con el objeto producido en cualquier evento. Se evidencia en las clases Controller porque contienen la información necesaria para crear instancias de una entidad. Esto se evidencia en la clase CAD2LAMMPS que es encargada de los modelos de superficie.



Figuras 14: Patrón creador.

Controlador: sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es el controlador quien recibe los datos del usuario y quien los envía a las distintas clases según el método llamado. Para el caso de generar los modelos de superficie es la sp_CAD2LAMMPS.

Figuras 15: Patrón controlador

Experto: permite asignar una responsabilidad al experto en información. Indica que la responsabilidad de la creación de un objeto o la implementación de un método, debe recaer sobre la clase que conoce toda la información necesaria para crearlo. De este modo se obtendrá un diseño con mayor cohesión y así la información se mantiene encapsulada[CITATION Urq14 \l 23562]. Este patrón se evidencia en las entidades que son las que conforman el modelo y se encuentran en la carpeta Entity de cada Bundle en el Controlador. Son expertas en la información que manejan, por ejemplo: la entidad Identificar_Superficie que posee métodos y atributos asociados a una asignatura como el método getFacade.

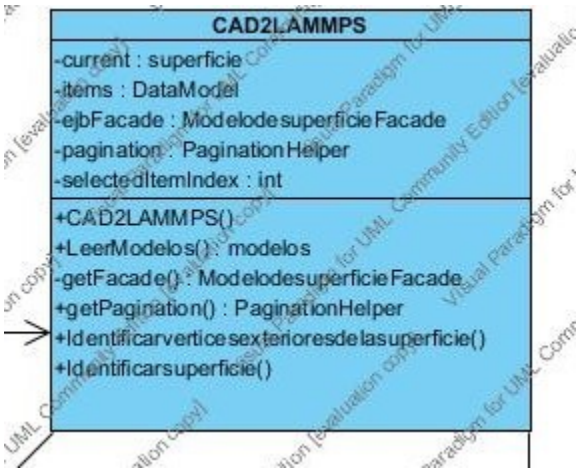


Figura 16: Patrón Experto

Conclusiones del capítulo

Durante la fase de Análisis y Diseño propuesto por la metodología APU – UCI, se crearon como principal artefacto las Historias de Usuarios descritas en el presente capítulo. Se obtuvieron los diagramas correspondientes al modelo del análisis y del diseño, se define la arquitectura Tres Capas como arquitectura para aplicar por las facilidades que proporciona, se utilizaron varios patrones para el diseño que facilitan la reutilización y mejor organización del módulo como Experto, Controlador y Creador.

Capítulo III: Pruebas y análisis de resultados

En el presente capítulo muestran los estándares de codificación empleados en la implementación del subsistema, los cuales garantizan que el código resultante siga las buenas prácticas de implementación y mantenga la seguridad y sostenibilidad requeridas. Posteriormente, se realizan las validaciones de la solución mediante los métodos de pruebas de software, con el fin de garantizar la mayor calidad posible en el desarrollo del módulo. Las pruebas de validación aplicadas al módulo se encontrarán divididas en pruebas de caja blanca y pruebas de caja negra.

3.1 Diagrama de despliegue

La vista de despliegue establece una correspondencia entre la arquitectura software y la arquitectura hardware del sistema. Permite modelar la disposición física o la topología de un sistema, muestra las conexiones físicas entre el hardware y las relaciones entre componentes. Muestra el hardware usado y los componentes instalados en el hardware. [CITATION Ecu19 \l 23562]

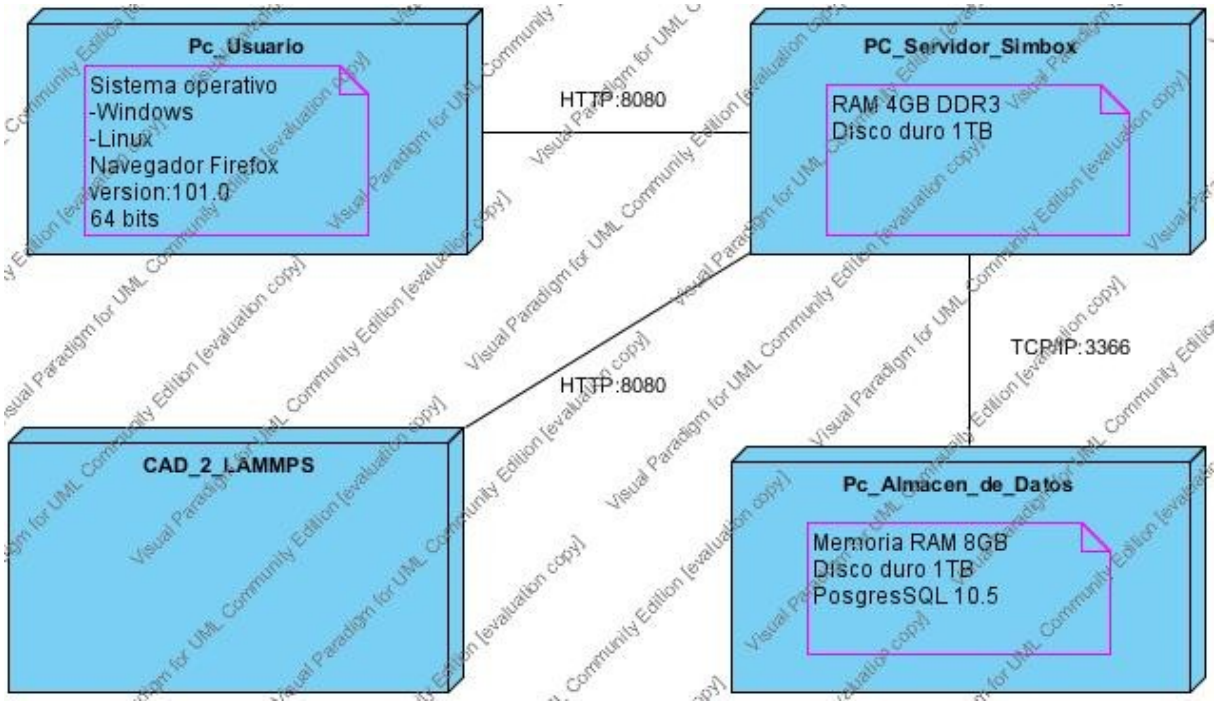


Figura 17. Modelo de despliegue.

El primer nodo de la vista de despliegue correspondiente al software lo constituye la computadora del usuario; la cual puede tener sistema operativo tanto Windows como Linux, el navegador web sería Firefox en su versión 101.1. Esta Pc debe conectarse con el servidor de SIMBOX mediante el puerto 8080 donde puede hacer peticiones REST; contando con 4GB de RAM y un disco duro de 1TB. El nodo CAD_2_LAMMPS tiene acceso al servidor de SIMBOX a través del puerto 8080; donde los datos se almacenan en el almacén de datos de SIMBOX mediante TCCP y puerto 3366.

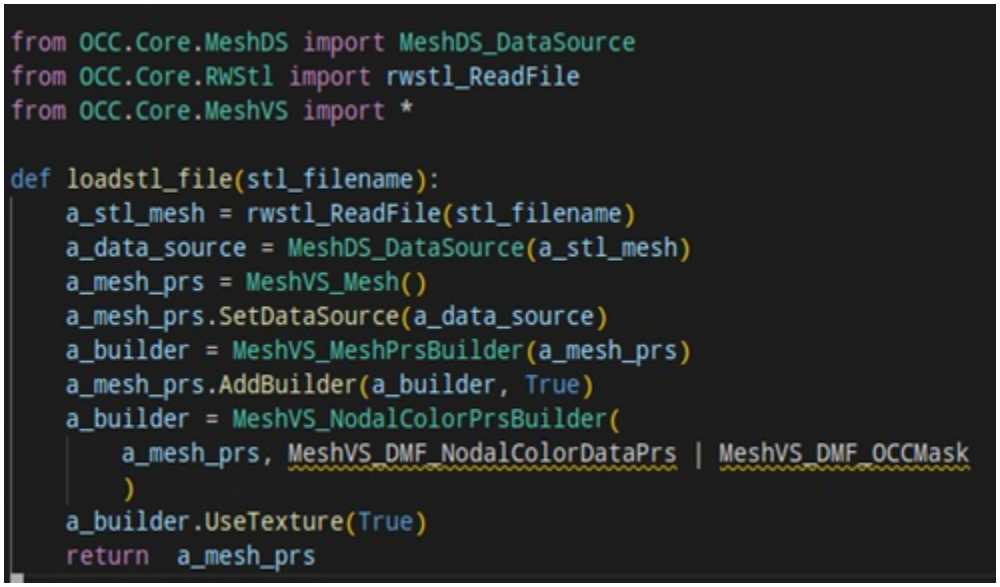
3.2 Estándar de codificación

Un estándar de codificación completo comprende todos los aspectos de la generación de código. Al inicio de un desarrollo de software, es necesario establecer un estándar de codificación para que todos los desarrolladores trabajen sobre un código similar. Si fuera necesario incorporar código fuente previo, o bien realizar mantenimiento a un sistema de software creado anteriormente, el estándar de codificación debería operar con el código ya existente. Incorporan principios de ingeniería sólidos que componen la base de cualquier enfoque preventivo.[CITATION Azc17 \l 23562]

A continuación se presentan los estándares de codificación definidos para el proyecto:

- Todas las instancias y variables de clases o métodos empezarán con minúscula. Las palabras internas que lo forman, si son compuestas, empiezan con su primera letra en mayúsculas.

- Se debe dar un espacio en blanco entre una palabra clave del lenguaje y un paréntesis.
- Se debe utilizar como idioma el español, las palabras no se acentuarán.
- Las líneas en blanco mejoran la facilidad de lectura separando secciones de código que están lógicamente relacionadas. Se deben usar siempre dos líneas en blanco en las siguientes circunstancias:
    - Entre las secciones de un fichero fuente.
    - Entre las definiciones de clases e interfaces.
- Se debe usar siempre una línea en blanco en las siguientes circunstancias:
    - Entre métodos.
    - Entre las variables locales de un método y su primera sentencia.
    - Antes de un comentario de bloque o de un comentario de una línea.



```
from OCC.Core.MeshDS import MeshDS_DataSource
from OCC.Core.RWStl import rwstl_ReadFile
from OCC.Core.MeshVS import *

def loadstl_file(stl_filename):
    a_stl_mesh = rwstl_ReadFile(stl_filename)
    a_data_source = MeshDS_DataSource(a_stl_mesh)
    a_mesh_prs = MeshVS_Mesh()
    a_mesh_prs.SetDataSource(a_data_source)
    a_builder = MeshVS_MeshPrsBuilder(a_mesh_prs)
    a_mesh_prs.AddBuilder(a_builder, True)
    a_builder = MeshVS_NodalColorPrsBuilder(
        a_mesh_prs, MeshVS_DMF_NodalColorDataPrs | MeshVS_DMF_OCCMask
        )
    a_builder.UseTexture(True)
    return  a_mesh_prs
```

Figura 18. Demostración de estándares de codificación.

3.3 Validación de requisitos

La validación de requisitos es un proceso continuo en el proyecto de desarrollo de software con el fin de asegurar que los requerimientos elicitados sean representaciones exactas de las necesidades y expectativas de los usuarios. Esta actividad contribuye a mejorar la calidad de los requerimientos, a reducir costos, tiempos y riesgos en el desarrollo de software. [CITATION Son20 \l 23562]

3.3.1 Técnicas de validación de requisitos

Existen varias técnicas para la validación de los requisitos con el objetivo fundamental de obtener una mayor calidad y demostrar realmente que los requisitos obtenidos describen al software que se necesita desarrollar. La técnica que se utilizó es:

- Prototipado de interfaz: permite al cliente entender la propuesta de solución fácilmente, pues brinda la representación aproximada de la interfaz de usuario que tendrá el módulo. Existen dos tipos de prototipos de interfaz: el primero es el desechable que se utiliza sólo para la validación de los requisitos y luego se desecha; pues pueden ser prototipos en papel. Por otro lado están los evolutivos que una vez utilizados para la validación de los requisitos, se mejora su calidad y se convierten posteriormente en el producto final. Este último fue el utilizado en la propuesta de solución ya que de esta forma el prototipo diseñado en todo el proceso del software se puede reutilizar.

3.4 Pruebas realizadas a la solución

"Pruebas de software: es la ejecución del código usando combinaciones de entradas, en un determinado estado, para revelar defectos. Proceso de ejecutar un programa con el fin de encontrar errores" *(Martínez, 2016).*

Las pruebas son un elemento importante dentro del ciclo de vida de un proyecto que proporciona una medida de la calidad del software. Estas son actividades en las cuales un sistema o componente es ejecutado bajo condiciones o requerimientos especificados, los resultados son observados y registrados, y una evaluación es hecha de algún aspecto del sistema o componente.[CITATION Azc17 \l 23562]

Se aplicaron un conjunto de pruebas definidas por Pressman para evaluar la calidad del módulo que se está desarrollando y verificar el cumplimiento de los objetivos trazados.

Tabla 10: Pruebas realizadas.

Pruebas	Método	Técnica
Prueba de Unidad	Caja Blanca	Camino básico
Prueba de Funcionales	Caja Negra	Partición de equivalencia

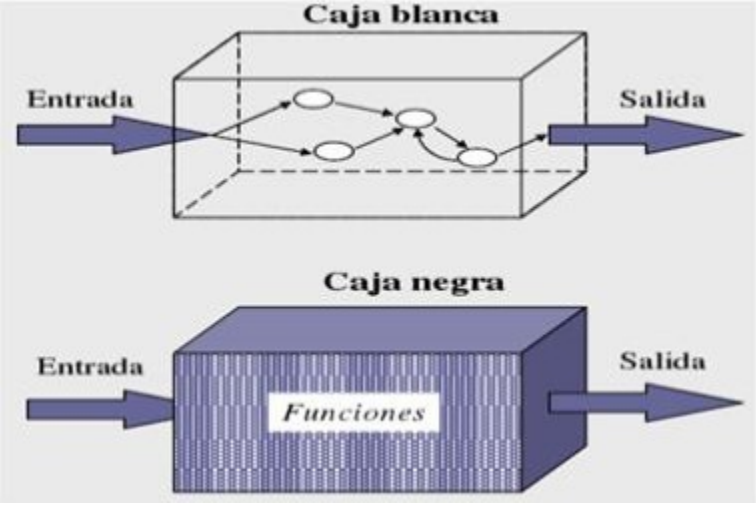


Figura 17. Representación de los tipos de métodos [CITATION Vis \l 23562].

3.4.1 Pruebas internas

Distribuye las compilaciones de forma rápida a un grupo pequeño de verificadores de confianza sin esperar a que los usuarios envíen sus opiniones. Integra los procesos de calificación de las compilaciones antes de proporcionar el software a más usuarios. Se deben desarrollar artefactos de prueba como listas de chequeo y diseños de casos de prueba.

Pruebas de unidad

En programación, una prueba unitaria o test unitario es una forma efectiva de comprobar el correcto funcionamiento de las unidades individuales más pequeñas de los programas informáticos. Esto sirve para asegurar que cada unidad funcione correcta y eficientemente por separado. Además de verificar que el código hace lo que tiene que hacer, verificamos que sea correcto el nombre, los nombres y tipos de los parámetros, el tipo de lo que se devuelve, que si el estado inicial es válido, entonces el estado final es válido también.

Este tipo de prueba se realiza mediante el método de caja blanca, la cual se basa en un examen minucioso de los detalles procedimentales, comprobando los caminos lógicos del programa, las condiciones y ciclos[CITATION Vis \l 23562]**.** La técnica de prueba de caja blanca utilizada en la investigación es el camino básico o ruta básica.

Técnica de camino básico

- Permite obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución.
- Hace uso de la representación del flujo de control mediante un grafo de flujo.
- Se basa en la representación del código a probar a través de círculos y arcos que indican el flujo de control. Cada círculo representa una o más sentencias[CITATION Vis \l 23562]

Pressman propone realizar un análisis de la complejidad ciclomática de cada procedimiento que componen las clases del módulo, una vez concluido este paso se selecciona el método con valor de contener errores. Luego, se determina la complejidad ciclomática V (G) del grafo resultante, el cual muestra el número de caminos independientes que existen en el grafo. Esta se puede calcular de tres opciones: [CITATION Mor19 \l 23562]

- Restar las aristas menos los nodos y sumar 2:
  V (G) = Aristas – Nodos + 2
  V (G) = (9 – 10) + 2
  V (G) =-1 + 2 = 1

- Sumar 1 al número de nodos predicados (aquellos de los que salen dos flechas):
  V (G) = Nodos predicados + 1
  V (G) = 0 + 1 = 1

- Contar el número de regiones (espacios ‹‹encerrados entre nodos y aristas››, también se tiene en cuenta el espacio ‹‹exterior ›› a todos los nodos y aristas):
  V (G) = Regiones
  V (G) = 1

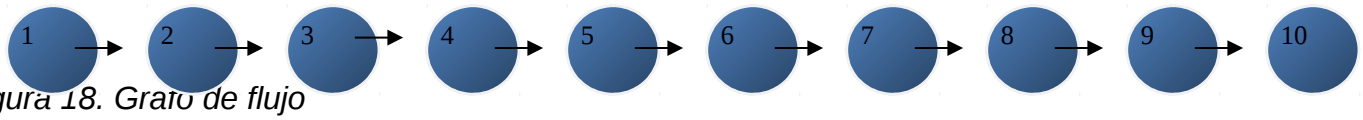


Figura 18. Grafo de flujo

```
from OCC.Core.MeshDS import MeshDS_DataSource
from OCC.Core.RWStl import rwstl_ReadFile
from OCC.Core.MeshVS import *

def loadstl_file(stl_filename):
  a_stl_mesh = rwstl_ReadFile(stl_filename)
  a_data_source = MeshDS_DataSource(a_stl_mesh)
  a_mesh_prs = MeshVS_Mesh()
  a_mesh_prs.SetDataSource(a_data_source)
  a_builder = MeshVS_MeshPrsBuilder(a_mesh_prs)
  a_mesh_prs.AddBuilder(a_builder, True)
  a_builder = MeshVS_NodalColorPrsBuilder(
    a_mesh_prs, MeshVS_DMF_NodalColorDataPrs |
MeshVS_DMF_OCCMask
    )
  a_builder.UseTexture(True)
  return a_mesh_prs
```

Figura 19. Código utilizado para calcular la complejidad ciclomática.

Existe un cierto acuerdo en cuanto a la simplicidad de un código, en función de su complejidad ciclomática. Por lo que se establece que, según el valor obtenido, podemos determinar cómo es:

Tabla 11: Análisis de riesgo de la complejidad ciclomática.

Complejidad ciclomática	Tipo de código
1 – 10	Simple
11 - 20	Algo complejo
21 – 50	Complejo
50	No testeable

El resultado de la complejidad ciclomática coincidió al hacer las tres tipos de fórmulas, dando como resultado 1. Es decir, que existe sólo un posible camino por donde el flujo puede circular.

Tabla 12: Caso de prueba por la trayectoria

Descripción	**Cargar archivo**
Camino	1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9 - 10
Entrada	Se introduce el nombre de un archivo
Resultados	Se debe mostrar que cargó satisfactoriamente el archivo

Condiciones	Que lo llame otro método

A partir de la ejecución del caso de prueba obtenido de la técnica camino básico, se concluye que el código generado no genera códigos infinitos y no existe código innecesario en el módulo desarrollado.

3.4.2 Pruebas funcionales

Son aquellas que se realizan con el objetivo de probar que el sistema cumple con las funciones específicas para las cuales ha sido creado. [CITATION Mor19 \l 23562]

Método de Caja Negra

Las pruebas de caja negra se llevan a cabo sobre la interfaz del software, por lo que los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce una salida correcta, así como que la integridad de la información externa se mantiene. Esta prueba examina algunos aspectos del modelo del sistema sin tener mucho en cuenta la estructura interna del software, lo cual permite encontrar funciones incorrectas o ausentes, errores de interfaz, rendimiento, de inicialización y terminación, de estructuras de datos o en accesos a las bases de datos.[CITATION Mor19 \l 23562]

Dentro de las pruebas de caja negra que se emplea **como técnica la partición equivalente**, la cual consiste en dividir el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software, lo cual permite descubrir de forma inmediata errores que de otro modo requerirían la ejecución de muchos casos antes ser detectados.

Para aplicar esta técnica, se debe primeramente realizar el Diseño de casos de prueba con el objetivo de obtener un conjunto de prueba que tenga la mayor probabilidad de descubrir los defectos del software.

Tabla 14: Caso de prueba del escenario Adicionar potencial de interacción.

Descripción	**Variable 1**	**Respuesta del modulo**	**Flujo central**
Permite adicionar potencial de interac-	V (Introducir datos del potencial)	Lo adiciona satisfactoriamente	-Antes haber Mostrado el ma-

ción	I (Introducir carácter extraño en los campos)	El módulo lanza un mensaje indicando que hay valores incorrectos	yado de partículas en PYMOL -Introducir los datos
	L (Dejar campos vacíos)	El módulo lanza un mensaje que lo campos no deben estar vacíos	-Adicionar potencial

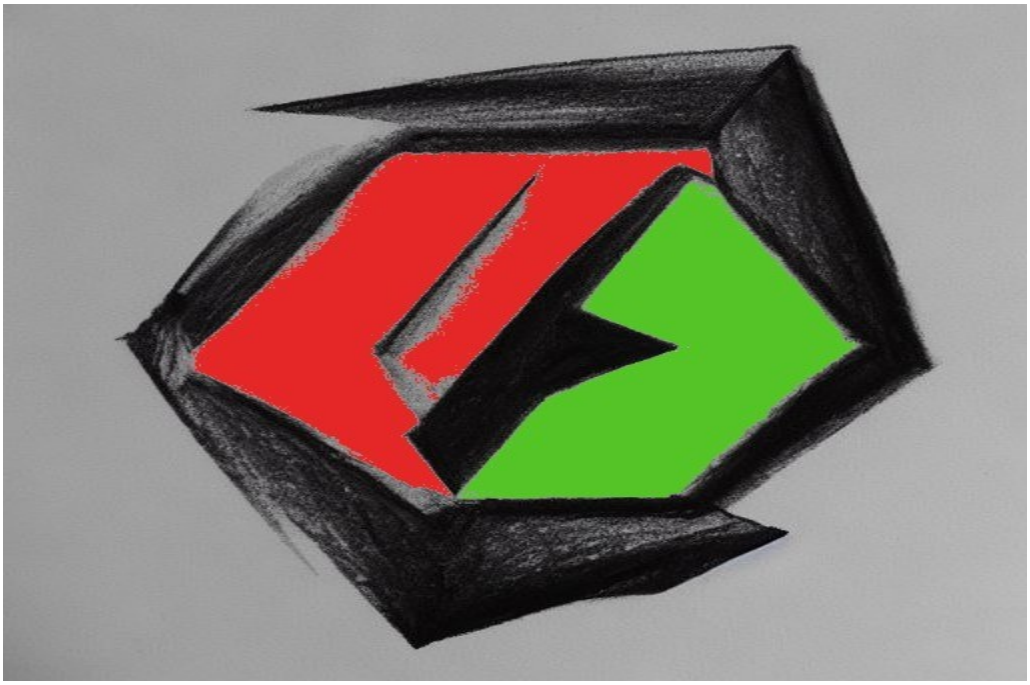


Figura 21. Login del módulo

Conclusiones del capitulo

En el presente capítulo se concluye la etapa de construcción y validación de la investigación, la cual arroja como resultado la primera versión del sistema propuesto para dar solución al problema de investigación. Fueron aplicados los estándares de codificación que garantizan un código seguro y optimiza-

ble. En sistema fue sometido a pruebas de caja negra, que facilitaron la detección y corrección de no conformidades con el objetivo de obtener un producto de mayor calidad.

CONCLUSIONES GENERALES

Luego de realizar el análisis, el diseño, la implementación al módulo desarrollado se arribó a las siguientes conclusiones:

- La construcción del marco teórico referencial permitió establecer las bases del desarrollo de la investigación y garantizó una mejor comprensión del proceso para desarrollar el módulo. Definiendo así los principales conceptos asociados a la investigación, se evidenció la necesidad de crear una funcionalidad en LAMMPS que permitiera cargar los archivos CAD.

- En la fase de análisis y diseño quedó documentado cada uno de los artefactos definidos por la metodología AUP – UCI, haciendo posible el desarrollo de un software que cumpliera con las buenas prácticas requeridas por el negocio. Donde se diseñó el diagrama de clases del diseño, las historias de usuario y el modelo del dominio.

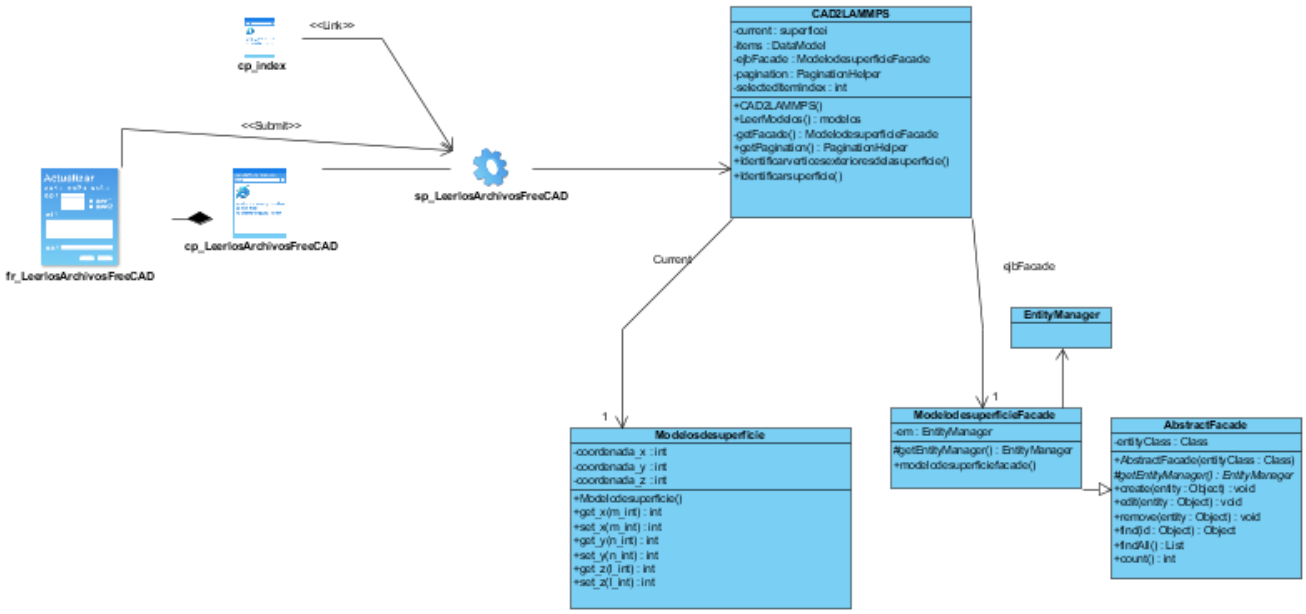
- El empleo de estrategias de validación constató la calidad y el correcto funcionamiento del sistema obtenido, dando como resultado un sistema robusto y confiable para el usuario.

RECOMENDACIONES

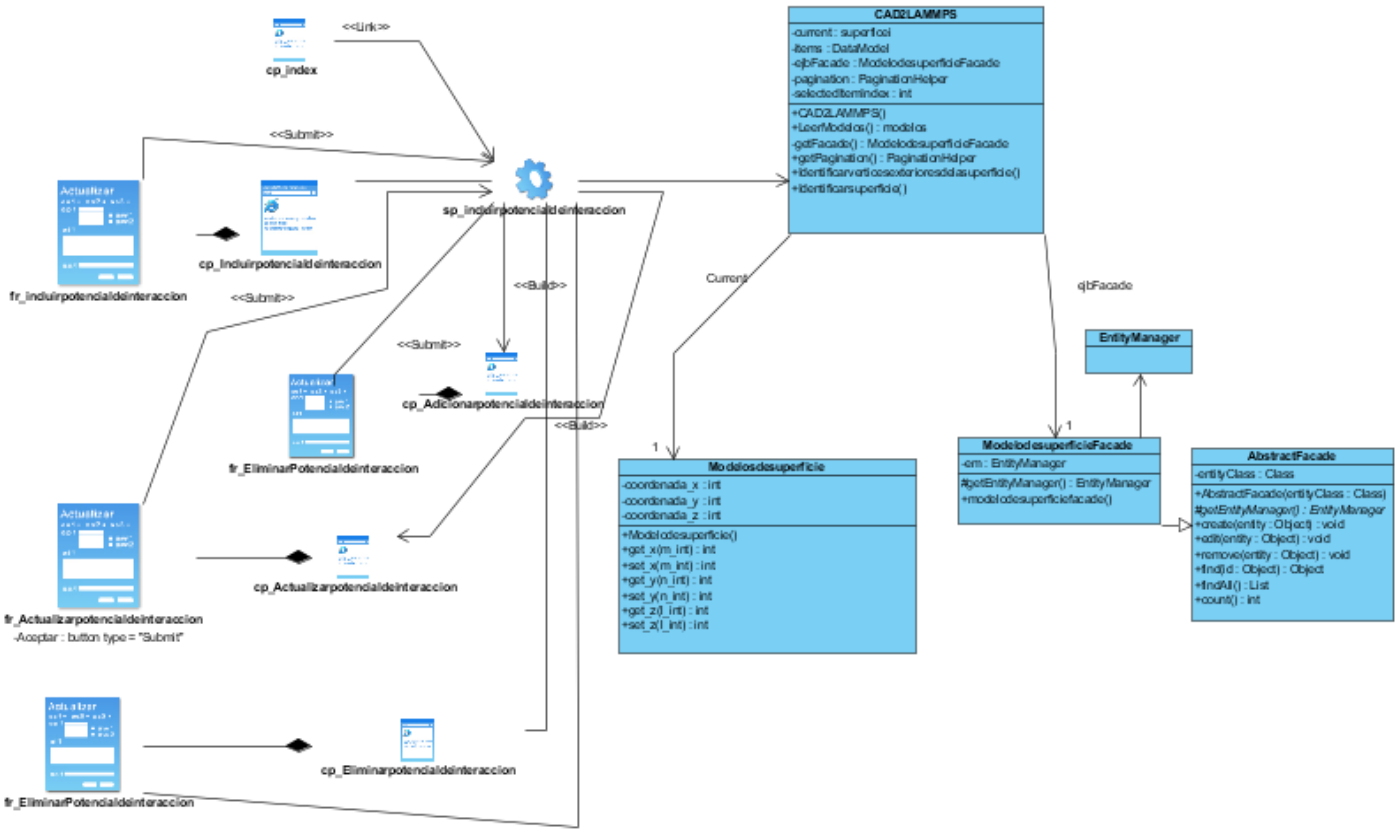
Tras la investigación realizada para dar solución al problema planteado, se obtuvo un módulo para la carga de archivos CAD para la simulación de microfluidos. De este trabajo se hacen las siguientes recomendaciones:

- Darle continuidad al módulo para que este vaya evolucionando de acuerdo a los requerimientos especificados.

Anexos



Anexo 1: Diagrama de Clases- Leer los archivos de FreeCAD



Anexo 2: Diagrama de Clases- Incluir un potencial de interacción entre las partículas de superficie y los componentes del microfluidos.

BIBLIOGRAFÍA

PAREDES CHICAIZA, J.A. y DT-MELÉNDEZ TAMAYO, C., 2012. “El diseño asistido por computadora (cad) y su incidencia en el proceso de interaprendizaje de la asignatura de Dibujo Técnico en los estudiantes de décimo año de Educación Básica del Instituto Superior Tecnológico Docente Guayaquil de la ciudad de Ambato”. En: Accepted: 2014-04-02T20:49:09Z [en línea], [Consulta: 20 octubre 2022]. Disponible en: https://repositorio.uta.edu.ec:8443/jspui/handle/123456789/7149.

ROJAS LAZO, O., 2014. Diseño asistido por computador. *Industrial Data*, vol. 9, no. 1, pp. 007. ISSN 1810-9993, 1560-9146. DOI 10.15381/idata.v9i1.5709.

CARIBE, C.E. para A.L. y el, 2010. *Impacto de las TIC en los aprendizajes de los estudiantes: estado del arte* [en línea]. S.l.: CEPAL. [Consulta: 26 octubre 2022]. Disponible en: https://www.cepal.org/es/publicaciones/3781-impacto-tic-aprendizajes-estudiantes-estado-arte.

CAÑO, A. del, CRUZ, M.P. de la y SOLANO, L., 2007. Diseño, ingeniería, fabricación y ejecución asistidos por ordenador en la construcción: evolución y desafíos a futuro. *Informes de la Construcción*, vol. 59, no. 505, pp. 53-71. ISSN 1988-3234. DOI 10.3989/ic.2007.v59.i505.500.

Qué es CAD, para qué sirve y qué ventajas tiene. *Integral Innovation Experts Blog* [en línea], 2019. [Consulta: 26 octubre 2022]. Disponible en: https://integralplm.com/blog/2019/08/20/que-es-cad/.

AUTODESK. (2022). Acerca del modelado de objetos 3D.

Gonzales, L. A. (2019). *Subsistema para la Creación de Espacios de Simulación PETRI.* Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas.

Liedekerke, P. V. (2011). *The implementation of Smooth Particle Hydrodynamics in LAMMPS.* Faculty of Bio- Engineering, Freiburg, Germany.

Gomez, G. M. (2021). *Subsistema para la creacion y control de simulaciones (SIMBOX).* Trabajo de Diploma para optar por el titulo de Ingeniero en Ciencias Informaticas, UCI, La Habana.

Solvusoft. (2022). Formatos de archivo CAD. *Solvusoft Corporation*.

Gazarkh, M. (2021). Everything you need to know about CAD file formats. *CAD Exchanger*.

KEEPCODING, R., 2022. ¿Qué son las operaciones booleanas en diseño gráfico? [en línea]. [Consulta: 3 noviembre 2022]. Disponible en: https://keepcoding.io/blog/que-son-operaciones-booleanas-diseno-grafico/.

Scientific, D. (2019). PyMOL. *Wikipedia*.

Mediforum. (2018). Un dispositivo microfluídico revolucionará diagnósticos. *SALUD DIGITAL*.

CLARO, M., 2010. Impacto de las TIC en los aprendizajes de los estudiantes. Estado del arte. *Santiago de Chile: CEPAL* [en línea], [Consulta: 26 octubre 2022]. Disponible en: https://www.academia.edu/86473944/Impacto_de_las_TIC_en_los_aprendizajes_de_los_estudiantes_Estado_del_arte.

Bryan Molina Montero, H. V. (2018). *Metodologias agiles frente a las tradicionales en el proceso de desarrollo de software. .* Espirales revista multidisciplinaria de investigacion.

CADAVID, A.N., 2013. Revisión de metodologías ágiles para el desarrollo de software. *Prospectiva*, vol. 11, no. 2, pp. 30. ISSN 22161368, 16928261. DOI 10.15665/rp.v11i2.36.

contributors, E. (14 de agosto de 2013). Visual Paradigm. *EcuRed*.

contributors, E. (4 de agosto de 2019). Diagrama de despliegue. (EcuRed, Ed.) *EcuRed*.

contributors, E. (29 de agosto de 2019). Metodologia de Desarrollo de Software Variacion de AUP para la UCI. *EcuRed*

VIRTANEN, P., GOMMERS, R., OLIPHANT, T.E., HABERLAND, M., REDDY, T., COURNAPEAU, D., BUROVSKI, E., PETERSON, P., WECKESSER, W., BRIGHT, J., VAN DER WALT, S.J., BRETT, M., WILSON, J., MILLMAN, K.J., MAYOROV, N., NELSON, A.R.J., JONES, E., KERN, R., LARSON, E., CAREY, C.J., POLAT, İ., FENG, Y., MOORE, E.W., VANDERPLAS, J., LAXALDE, D., PERKTOLD, J., CIMRMAN, R., HENRIKSEN, I., QUINTERO, E.A., HARRIS, C.R., ARCHIBALD, A.M., RIBEIRO, A.H., PEDREGOSA, F. y VAN MULBREGT, P., 2020. SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nature Methods*, vol. 17, no. 3, pp. 261-272. ISSN 1548-7105. DOI 10.1038/s41592-019-0686-2.

MOMJIAN, B., 2000. *PostgreSQL: introduction and concepts*. Boston, MA: Addison-Wesley. ISBN 978-0-201-70331-3.

HOLOVATY, A. y KAPLAN-MOSS, J., 2009. *The Definitive Guide to Django: Web Development Done Right*. 2nd ed. edition. Berkeley, Calif: Apress. ISBN 978-1-4302-1936-1.

Diagrama de Clases de Diseño | LENGUAJE DE MODELADO UNIFICADO - UML. [en línea], [sin fecha]. [Consulta: 1 noviembre 2022]. Disponible en: https://unadzsurlab.com/UML/U2/diagrama_de_clases_de_diseo.html.

Azcuy, R. A. (2017). *Modulo para la configuracion y monitorizacion de replicas con la herramienta SymmetricDS para la arquitectura Xalix.* Trabajo de diploma para optar por el titulo de Ingeniero en Ciencias Informaticas.

Moreno, O. (2019). La complejidad ciclomatica y como simplificar tus desarrollos.

(2017). Visiones internas y externas de las pruebas. En *Seleccion de tecnicas de Ingenieria de Software.*

Sonia Santana, L. P. (2020). Evaluacion de tecnicas para la validacion de requerimientos en entornos de trabajo para el desarrollo de software. *Repositorio Institucional de la UNLP*.

Martínez, I. C. (2016). Pruebas de Software.

Urquiaga, P. A. (2014). *Portafolio de Desarrollo Personal para la centralizacion de las evidencias de trabajo.* Trabajo de Diploma para optar por le titulo de Ingeniero en Ciencias Informaticas.

Pressman. (10 de noviembre de 2011). Software Engineering, a practitioners approach. *wikipedia*.

Sanchez, C. (2004). *ONess: un proyecto open source para el negocio textilmayorista desarrollado con tecnologias open source innovadoras.* Universidade da Coruña.

Peñalvo, F. J. (2017 - 2018). *Modelo de Dominio.* Universidad de Salamanca, Departamento de Informatica y Automatica.

Sommerville, I. (2005). *Ingeniería del software*. Pearson Educación.

User, G. (2 de agosto de 2018). Escribiendo Historias de Usuarios. *SCRUM MEXICO*.

FreeCAD. (2022). OpenCASCADE. *FreeCAD Documentation*.

Documentation, F. (29 de agosto de 2021). PythonOCC. *FreeCAD Documentation*.

Mediforum. (2018). Un dispositivo microfluídico revolucionará diagnósticos. *SALUD DIGITAL*.

Hub, I. C. (28 de octubre de 2020). Arquitectura de tres niveles. *Que es la arquitectura de tres niveles*.