Universidad de las Ciencias Informáticas

Facultad 1

“Herramienta de simulación para la asignatura Sistemas Operativos”

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

Autor: Thais Mel Soteras Fong

Tutores: M.Sc. Arianna Rodríguez Jiménez

M.Sc. Mónica María Albo Castro

La Habana, 2021

Declaración de autoría

Declaro por este medio que yo **Thais Mel Soteras Fong**, soy la autora principal del trabajo titulado **"Herramienta de simulación para la asignatura Sistemas Operativos"** y autorizo a la Universidad de las Ciencias Informáticas a hacer uso del misma en su beneficio, así como los derechos patrimoniales con carácter exclusivo.

Para que así conste firmamos la presente a los ____ días del mes de _____ del _____

Autor: _____

Tutor: _____ Tutor: _____

Agradecimientos

- Agradecimiento especial a mi tutora por la confianza depositada en mí.
- A mis padres por darme confianza y seguridad.
- A mi esposo y su familia por todo el apoyo que me han dado.
- A mis amigos, por ser los mejores.
- A Dios por guiar mis pasos.
- A todos los que me han ayudado en este trabajo.

Resumen

La capacidad de comprender el funcionamiento de un Sistema Operativo, sus características, conceptos relacionados, interacción con el hardware y la relación entre sus componentes independientemente de su diseño e implementación es uno de los pilares de esta asignatura en los programas curriculares de las carreras afines a las ciencias de la computación e informática. La enseñanza de los conceptos asociados a la asignatura se basa en fundamentos mayormente teóricos, a pesar de existir diversas herramientas para que los estudiantes puedan visualizar desde la práctica el funcionamiento de los componentes del sistema debido a que no se adaptan a los objetivos de la asignatura ni abordan la totalidad de los contenidos que resultan necesarios para apoyar el proceso de asimilación de los estudiantes. Por tal motivo el objetivo de la presente investigación es desarrollar una herramienta que permita simular el funcionamiento de un conjunto de componentes del Sistema Operativo que se estudian en esta asignatura, para facilitar la comprensión de estos por parte de los estudiantes de la carrera Ingeniería en Ciencias Informáticas. Para ello se realizó el análisis de las herramientas más utilizadas para el apoyo a la impartición de los contenido de la asignatura, y se definieron la técnicas, herramientas y artefactos necesarios para el desarrollo de la solución.

Palabras clave: herramienta de simulación, Sistemas Operativos, proceso de enseñanza-aprendizaje de los Sistemas Operativos

Índice

Índice de figuras

Índice de tablas

Introducción

El desarrollo de las Tecnologías de la Información y las Comunicaciones (TIC) y su introducción en el proceso docente-educativo ha favorecido los métodos de enseñanza-aprendizaje, al permitir la introducción de nuevas formas de enseñanza que se caracterizan por la innovación y la interacción permanente. El impacto de la informatización en la sociedad y el hecho de que los estudiantes que hoy están en las aulas son nativos digitales, brindan a los docentes la posibilidad de abordar desde una perspectiva práctica, cuestiones que hasta el momento solo era posible enseñar de forma teórica.

En aras de formar profesionales capaces de enfrentar los retos que impone esta sociedad de la información, donde las tecnologías ocupan un papel relevante en todas las esferas del trabajo humano, se impone contribuir al desarrollo de habilidades en los estudiantes, el uso de técnicas para el procesamiento y uso de la información, junto a la creación de mejores herramientas que favorezcan el trabajo, permitiendo transmitir experiencias y conocimientos mediante el aprendizaje.

Con el fin de fortalecer la industria cubana del software y de acuerdo con la política del país de informatización de la sociedad, se han puesto en marcha diversas estrategias, entre ellas la creación de la Universidad de las Ciencias Informáticas (UCI) en el año 2002. Esta tiene como misión formar profesionales comprometidos con su patria y altamente calificados en la rama de la Informática, producir aplicaciones y servicios informáticos a partir del vínculo estudio-trabajo como modelo de formación-investigación-producción, sirviendo de soporte a la industria cubana de la Informática (1).

El perfil del Ingeniero en Ciencias Informáticas contempla entre sus modos de actuación el diagnóstico y transformación de procesos en las organizaciones para su informatización, el desarrollo y mantenimiento de sistemas, productos y servicios informáticos y la adopción de tecnologías de la información desde una perspectiva de soberanía tecnológica y Ciberseguridad (2). Por tal motivo las asignaturas impartidas como parte del currículo docente de la carrera, están encaminadas al desarrollo de habilidades, que permitan a los estudiantes desempeñarse en estas áreas.

A partir de su relevancia dentro de la informática, prácticamente desde sus inicios los Sistemas Operativos (SO) se han incluido en los programas de estudio de esta carrera como parte esencial de la misma. Con el paso de los años, esta importancia se ha ido acentuando dado que los SO proporcionan la visión del sistema de la mayoría de los usuarios y en mayor o menor medida, todos los usuarios interactúan con el mismo. Como resultado del desarrollo dinámico en el campo de la informática en general, incluyendo el hardware,

la cada vez mayor flexibilidad y versatilidad de los SO e igualmente la creciente interconexión de servicios y computadoras, hace necesario que los objetivos y contenidos de la misma estén en correspondencia con los requerimientos del ámbito de actuación del ingeniero informático, sin soslayar la necesidad de dotar a los estudiantes de conocimientos básicos que les permitan adaptarse a los constantes cambios que ocurren en el entorno.

Esta asignatura está concebida dentro de la disciplina Sistemas Digitales respondiendo al objetivo de contribuir a la explotación eficiente de sistemas de cómputo, y a la gestión adecuada de los servicios telemáticos y la seguridad informática, por lo que está estrechamente relacionada con los modos de actuación del profesional mencionados con anterioridad.

Consiste básicamente en la caracterización del funcionamiento de los SO y profundizar en las herramientas que brinda el SO para desarrollar y optimizar un sistema informático. Se encuentra estructurada en cuatro temas: Gestión de procesos, Gestión de memoria, Gestión de entrada/salida y Arquitecturas y tendencias actuales. Sin embargo, un análisis de los resultados académicos de los estudiantes en los últimos años refleja que el aprendizaje de estos contenidos es complejo debido a diferentes causas, entre las que destacan las siguientes:

- Los contenidos implican un alto nivel de abstracción dado que precisamente se estudia el funcionamiento de las capas de software que permiten abstraer al usuario de las complejidades del hardware.
- No se dispone de medios y recursos adecuados que permitan apreciar el funcionamiento de sus mecanismos, dificultando su comprensión.
- No se potencia la utilización de herramientas informáticas que permitan a los estudiantes aplicar los conocimientos teóricos adquiridos durante las clases, o que permitan la simulación de los mecanismos estudiados.

Esto repercute en que las principales carencias en cuanto al aprendizaje estén relacionadas con:

- La implementación de los mecanismos de comunicación y sincronización entre procesos.
- La identificación de las relaciones entre los componentes de los SO y la caracterización de las arquitecturas.

- La aplicación de las técnicas para la gestión de memoria virtual.
- Las nociones principales asociadas al manejo de los sistemas de archivos.

Como parte de una estrategia de integración de medios en la asignatura se han identificado varios contenidos de la asignatura como la planificación del procesador, los algoritmos de planificación de discos, entre otros, que se pueden apoyar en herramientas de simulación para elevar el nivel de asimilación por parte de los estudiantes. Sin embargo, su utilización en la UCI es prácticamente nula, a pesar de existir trabajos precedentes que no han logrado la generalización por diversas causas.

Entre estas causas se puede mencionar: el diseño de actividades que integren estas herramientas, problemas en el ajuste a los objetivos de la asignatura y la baja calidad de las herramientas previamente desarrolladas. Por otro lado, los contenidos relacionados con los mecanismos de comunicación y sincronización entre procesos, la gestión de memoria y el manejo de sistemas de archivo no están representados en ninguna de estas herramientas.

Ante la situación antes expuesta se define como **problema de investigación**: La ausencia de medios de enseñanza para visualizar el funcionamiento de determinados componentes del Sistema Operativo, dificulta la comprensión de los contenidos de esta asignatura a los estudiantes de la carrera Ingeniería en Ciencias Informáticas.

La presente investigación tiene como **objeto de estudio** las Herramientas de simulación en el proceso de enseñanza-aprendizaje. Como **objetivo general** se plantea: Desarrollar una herramienta que permita simular el funcionamiento de un conjunto de componentes del Sistema Operativo que se estudian en esta asignatura, para facilitar la comprensión de estos por parte de los estudiantes de la carrera Ingeniería en Ciencias Informáticas.

El **campo de acción** de la investigación se enmarca en las Herramientas de simulación en el proceso de enseñanza-aprendizaje de los Sistemas Operativos. Para lograr el cumplimiento del objetivo se definen las siguientes **tareas de investigación**:

1. Análisis de los principales conceptos relacionados con el objeto de estudio.

2. Investigación sobre los sistemas homólogos y las herramientas a utilizar para el desarrollo de la solución.
3. Selección de la metodología de desarrollo, las tecnologías y la arquitectura para la implementación del sistema.
4. Definición de los requisitos funcionales y no funcionales de la solución propuesta.
5. Diseño de la solución. Generación de diagramas y artefactos requeridos por la metodología de desarrollo a utilizar.
6. Implementación de las funcionalidades de la herramienta de simulación propuesta.
7. Diseño y ejecución de las pruebas a la herramienta desarrollada.

Como **hipótesis de la investigación** se plantea que el desarrollo de una herramienta que permita simular el funcionamiento de un conjunto de componentes del Sistema Operativo, facilitará la comprensión de estos por parte de los estudiantes de la carrera Ingeniería en Ciencias Informáticas.

Se identifica como **variable independiente**: Herramienta de simulación para la enseñanza de Sistemas Operativos, y como **variable dependiente**: la comprensión de los contenidos por parte de los estudiantes.

Para el desarrollo de la investigación se utilizaron los **métodos de investigación** descritos a continuación:

Métodos teóricos:

- Histórico - Lógico: para profundizar en los antecedentes de la utilización de las Tecnologías de la Información y las Comunicaciones en los procesos de enseñanza - aprendizaje y sus tendencias actuales.
- Análisis documental: para analizar, clasificar, verificar, seleccionar los contenidos en la bibliografía e informes de investigación, referentes a las herramientas de simulación, por lo que se será necesario recurrir a la revisión de documentos, la búsqueda bibliográfica y teorías y análisis de criterios de autores.
- Analítico - Sintético: en la revisión bibliográfica, de trabajos de diploma concernientes a la temática, planes de estudio de carreras donde se estudie la asignatura en cuestión y las herramientas que utilizan para sus actividades prácticas y de autoaprendizaje, de los libros de texto de la asignatura Sistemas Operativos que se utilizan en la Universidad.

- Inductivo-Deductivo: permitió llegar al planteamiento del objetivo, además de la extracción de las ideas fundamentales para la elaboración y fundamentación teórica del trabajo de diploma, fue utilizado para el razonamiento de la información consultada, llegando a un grupo de conocimientos particulares y generales sobre las herramientas de simulación para la enseñanza de sistemas operativos.
- La modelación: empleada la construcción de modelos como parte de la metodología de desarrollo de software y además como instrumento de apoyo a la investigación basándose en el lenguaje UML.

Métodos empíricos:

- Observación: para la recopilación de información de las características y comportamiento de los estudiantes al utilizar las TIC en la enseñanza y el aprendizaje con el uso de los Objetos de Aprendizaje en la asignatura de SO.

La presente investigación cuenta con la siguiente **estructura capitular**:

Capítulo 1. Fundamentación Teórica: Se expone el marco teórico de la investigación y se precisa lo referente al estudio de homólogos, así como las definiciones y aspectos relacionados con la temática planteada, la metodología de desarrollo de software a utilizar y las herramientas y tecnologías necesarias para la implementación del sistema.

Capítulo 2. Análisis y diseño de la solución: Se presenta la propuesta de solución, así como los requisitos y funcionalidades del sistema, se define la arquitectura de software y los patrones de diseño. Se describen las historias de usuario y los procesos necesarios para la implementación de la solución.

Capítulo 3. Implementación y Pruebas: Se describe la etapa de implementación mediante la cual será posible la obtención del sistema. Se describen y documentan las pruebas realizadas a las funcionalidades desarrolladas.

Capítulo 1: Fundamentación Teórica

Introducción

En el presente capítulo se realiza un estudio de los conceptos claves de la investigación, entre ellos los sistemas operativos, la enseñanza de la asignatura y algunas herramientas de simulación dentro y fuera del ámbito educacional, así como las vinculadas con el proceso de enseñanza-aprendizaje de los sistemas operativos. Se hace un análisis del estado del arte a nivel internacional y nacional, reflejando resultados de investigaciones y proyectos desarrollados vinculados con la temática de la investigación, y se determina la metodología de desarrollo y las técnicas y herramientas a emplear para la confección de la propuesta de solución.

En la presente investigación solo se tendrá en cuenta la simulación de los contenidos relacionados con la planificación de procesos, las estrategias de interbloqueo, la gestión de memoria y la planificación de acceso al disco. Las cuestiones relacionadas con la concurrencia y sincronización de procesos no serán abordadas debido a que se requiere la implementación de un compilador para interpretar el código escrito por los usuarios, correspondiente a la programación de los semáforos y demás mecanismos de comunicación entre procesos. Tampoco se analizarán los contenidos relacionados con el sistema de archivos debido a que la profundidad de los contenidos que se imparten actualmente no requiere de uso de simuladores para facilitar la comprensión por parte de los estudiantes.

La enseñanza en cualquier materia tiene como fin el aprendizaje eficaz por parte de los estudiantes. Hoy en día el empleo de tecnologías, como herramientas de software, infografías, plantillas, multimedias, foros, etc., permiten elaborar nuevas técnicas de aprendizaje en beneficio de una educación más extendida y de mayor calidad (3). Las técnicas o instrumentos desarrollados deben facilitar una mejor comprensión de los contenidos a los estudiantes. Para desarrollarlas, el profesor, en su el rol de guía y regulador del proceso de enseñanza-aprendizaje, debe llevar a cabo la actividad docente e investigativa, así como incorporar sus conocimientos de investigación a la docencia. Por ello, es fundamental que hagan uso de los recursos didácticos para lograr el aprendizaje del estudiante, para que estos puedan definir claramente la teoría, práctica, trabajos, exámenes, etc. (4).

En la actualidad el método más usado en la asignatura SO, produce una actitud pasiva en los estudiantes, dado que la metodología didáctica consiste en la explicación de los algoritmos y, posteriormente, en su aplicación a un caso muy sencillo para determinar los resultados que producen y realizar una comparación entre ellos. Su enseñanza es una tarea que se torna profundamente compleja debido al alto nivel de abstracción que requiere por parte de los estudiantes para lograr un buen entendimiento acerca del funcionamiento interno de los sistemas (5). Cabe destacar que en las universidades cubanas no se utilizan herramientas de simulación para la impartición de la sincronización de procesos, por ello es de gran necesidad encontrar una alternativa que cumpla con todos los requisitos propuestos en las tareas de investigación.

Diversas universidades han creado herramientas que simulan parte de los contenidos de la asignatura SO, ejemplo de ello son:

- Herramienta gráfica de simulación y exploración interactiva para la enseñanza de SO.
- jBACI.
- Sistema Abierto para La Enseñanza de Sistemas Operativos mediante un Simulador.
- Herramienta Simulador de Gestión de Memoria.
- Operating System Concepts Simulator (OS Sim).
- Herramienta de simulación para la Planificación de Procesos.
- La herramienta VMFS.
- Simulador del Módulo de Reemplazo de Páginas (SMRP).

En epígrafes posteriores se evidencian los estudios realizados a cada una de estas herramientas.

1.1. Herramientas de simulación

La simulación no es un concepto nuevo; siempre se ha buscado la manera de evaluar sistemas complejos, la simulación es la ejecución de un modelo representado por un programa de computadora que permite recrear entornos de red, ahorrando tiempo y dinero (6).

Hoy en día, las herramientas de simulación son un componente fundamental para el diseño, la implementación y el monitoreo de redes de comunicación, porque permiten predecir el comportamiento de

diferentes eventos que pueden afectar el desempeño de la red y degradar la calidad de las aplicaciones y los servicios. El uso de estas herramientas también puede funcionar como parte de un método pedagógico de enseñanza que permite a los estudiantes entender los diferentes conceptos de una manera más clara, sencilla y representada de forma tangible (7).

Las universidades juegan un papel fundamental como instituciones generadoras de conocimiento. Para adaptarse a los cambios tecnológicos, buscan actualizar sus contenidos implementando las tecnologías que envuelven los procesos de generación de conocimiento científico, para alcanzar un alto nivel de calidad y pertinencia en su entorno, puesto que las TIC han modificado la manera de ver y hacer las cosas. Existen diversas herramientas de simulación que apoyan el proceso educativo, ejemplo de ellas: Matlab que simula procesos matemáticos, Karel que simula procesos lógicos, Circuit Maker cuya función es simular la realización de circuitos electrónicos, Catt Acoustics para simular parámetros acústicos (8) y Logisim que es un simulador lógico que permite diseñar y simular circuitos electrónicos digitales mediante una interfaz gráfica de usuario.

El uso de las herramientas de simulación, como parte de los métodos pedagógicos de enseñanza, permiten a los estudiantes entender los diferentes contenidos de manera más clara, sencilla y representada, basándose en la premisa de que los estudiantes aprendan con mayor facilidad si disfrutan de una herramienta educativa (7).

1.1.1 Herramientas de simulación vinculadas a la enseñanza

La revista Actualidades Investigativas en Educación, de la Universidad de Costa Rica, realizó un estudio que exponía la importancia de la simulación como herramienta de aprendizaje en Física. Las situaciones problemáticas consideradas en asignaturas como Física, son eventos en contextos estáticos que al momento de resolverlos el estudiante necesita de un gran poder de abstracción para visualizar mentalmente el hecho presentado. Estos procesos mentales se pueden facilitar con el uso de las simulaciones, las cuales permiten la secuencia dinámica de los aspectos que son tratados en dichas situaciones problemáticas. (9)

También existen diversas herramientas de aprendizajes relacionadas con la programación, La Facultad de Eléctrica y Electrónica de Quito, Ecuador, planteó en un informe investigativo las diferentes herramientas relacionadas con el tema:

- Herramientas de calificación automática: Son aquellas herramientas orientadas a la automatización del proceso de calificación de ejercicios de programación. Ayudan a que el estudiante pueda realizar una mayor cantidad de ejercicios y pueda recibir una rápida retroalimentación, y también a que el profesor no se vea agobiado por un excesivo número de tareas a calificar.

- Herramientas multimedia: Son herramientas que ayudan en el proceso de aprendizaje a través del uso de recursos como textos, imágenes, videos, entre otros.
- Sistemas inteligentes de tutoría: Son herramientas complejas y orientadas al soporte de los estudiantes durante la escritura de sus programas. Incluyen mecanismos por medio de los cuales el estudiante resuelve el ejercicio en base a su nivel de conocimiento.
- Herramientas de aprendizaje visual: Son herramientas orientadas a representar gráficamente un programa. Se puede observar a través de componentes gráficos el algoritmo del programa y la ejecución del mismo. Son herramientas que permiten un alto grado de interacción con el estudiante. (10)

Por otro lado, la cátedra de Humanidades Médicas de la ciudad de Camagüey, desarrolló una investigación sobre la Simulación Educativa: Herramienta Didáctica para la educación de la ciencia tecnología y sociedad en la disciplina Filosofía y Sociedad; esto les permitió comprender lo contradictorio de la Investigación del Genoma Humano y sus consecuencias para el planeta (11).

El empleo de estas herramientas, constituye un método de enseñanza y aprendizaje efectivo para lograr en los estudiantes el desarrollo de un conjunto de habilidades que posibiliten alcanzar modos de actuación superiores; los diversos tipos de simulación disponibles pueden utilizarse para el mejoramiento de las técnicas de diagnóstico, tratamiento y la resolución de problemas, posibilitando de tal forma que los estudiantes se concentren en un determinado objetivo de enseñanza.

1.1.2 Herramientas de simulación en la enseñanza de la informática

La creación de materiales educativos para la enseñanza, diseño y evaluación, ocupa un terreno de significación en el campo de la tecnología educativa. Uno de estos medios de enseñanza son las herramientas de simulación que fueron creados con una finalidad didáctica, estas favorecen el aprendizaje

por descubrimiento, fomentan la creatividad, propician la enseñanza individualizada y facilitan la autoevaluación.

En el ámbito de la Informática, se han producido diferentes simuladores, pudiéndose mencionar:

- Packet Tracer es un programa de simulación de redes que permite a los estudiantes experimentar con el comportamiento d la red.
- Qemu es un emulador de procesadores basado en la traducción dinámica de binarios.
- Kiva es un recurso en la docencia de redes de comunicaciones.
- Network Visualizer sustituye a los laboratorios de routers reales.
- Router Simulator es una aplicación que permite trabajar con las líneas de comandos que ofrece las empresas de routers Cisco Systems.
- Entre otras herramientas.

El objetivo principal de estas herramientas, es permitir que los estudiantes sean capaces tanto de aprender de la experiencia como de tomar decisiones, por ello están centradas en el saber hacer considerando las simulaciones como laboratorios virtuales.

En la informática la simulación promueve el aprendizaje experimental y por descubrimiento, en el cual el diseñador de Software crea ambientes ricos, hasta llegar al conocimiento a partir de una experiencia, creando sus propios modelos de pensamiento, sus propias interpretaciones con el mundo virtual para ser aplicadas posteriormente en el mundo real.

1.2. Herramientas de simulación para la enseñanza de Sistemas Operativos

Existen varias herramientas de simulación y/o sistemas operativos que se han desarrollado con el fin de apoyar la enseñanza de esta asignatura, como son (12):

- Minix: diseñado para impartir la asignatura sistemas operativos; es el clon del sistema Unix.
- NachOs: sistema internacional para la enseñanza de cursos de SO.
- RCOS: diseñado para ayudar a los estudiantes a comprender el funcionamiento interno de un SO.

- PintOS: sistema educativo usado en cursos de para introducir los conceptos de diseño e implementación de sistemas operativos.

Estos sistemas educativos se crearon con el objetivo de entender el funcionamiento interno de un SO y diseñar nuevas arquitecturas. Ellos no forman parte del objeto de estudio de la investigación, pero pueden contribuir de forma positiva al desarrollo de la propuesta de solución.

Los simuladores se conciben como herramientas que permiten ilustrar de forma gráfica una serie de algoritmos y conceptos (13). Uno de los problemas que enfrentan los estudiantes cuando los estudia es la dificultad inherente a su comprensión, ya que debe realizar un esfuerzo de abstracción para poder observar qué es lo que realiza el sistema cuando emplea uno de estos algoritmos (13). Mediante el uso de estas herramientas, el alumno puede observar de forma gráfica el funcionamiento de los algoritmos, observando con detenimiento cómo se comporta el sistema. Esto permite reforzar y aclarar conceptos que previamente han sido adquiridos en las clases teóricas (13).

Las herramientas de software o software educativo que se utilizan de apoyo al proceso de enseñanza-aprendizaje de la asignatura usualmente son poco empleadas por los profesores. Entre las principales causas se encuentran la usabilidad de las mismas, la calidad a nivel funcional y la relación con el programa de la asignatura en aras de satisfacer los objetivos de la misma. Generalmente estas herramientas se caracterizan por simular solo uno de los ejes temáticos de la asignatura, pero una correcta selección puede contribuir a la comprensión del tema por parte de los estudiantes.

A continuación, se realiza un análisis de un grupo de herramientas utilizadas por diversas universidades como apoyo a la enseñanza de SO para determinar cuál o cuáles pueden apoyar el proceso de aprendizaje de los estudiantes en la UCI.

1.2.1 Herramientas de simulación para la enseñanza de Sistemas Operativos utilizadas por universidades extranjeras

Existen diversas herramientas de simulación que fueron creadas por universidades extranjeras para facilitarles a los estudiantes la comprensión de los contenidos de una manera más práctica, pues como se ha mencionado en epígrafes anteriores la asignatura SO es de difícil comprensión por el amplio material de estudio que posee. Entre estas se puede mencionar la Herramienta gráfica de simulación y exploración

interactiva para la enseñanza de SO, diseñada por Joshua W. Buck y Saverio Perugini, profesores del departamento de Ciencias de la Computación en la Universidad de Dayton, Ohio, Estados Unidos.

Esta permite la exploración interactiva procesos o eventos manejados por el SO cuando se ejecuta un programa. Está disponible para su uso en la web y de forma local con fines pedagógicos (14). La aplicación está diseñada para ejecutar una solución a un ejercicio de clases teniendo en cuenta el trabajo del planificador de procesos y el procesamiento de semáforos para cada ejercicio. Muestra el funcionamiento del sistema en función de la cantidad de procesos que se ejecuten teniendo en cuenta parámetros como la memoria, la velocidad de procesamiento y la prioridad de los procesos a ejecutar (14).

La utilización de la herramienta, así como la interpretación de los datos resulta altamente compleja puesto que requiere conocimientos profundos sobre la temática, y vincula aspectos que no se imparten en clases a los estudiantes. Además, sólo está enfocada a este tipo de simulaciones por lo que no resulta abarcadora para la asignatura. Sin embargo, puede ser muy útil en la superación de los profesores y para la preparación de materiales de apoyo a las clases.

jBACI es un entorno de desarrollo integrado para el aprendizaje de la programación concurrente mediante la simulación de la concurrencia. Fue construido por el profesor Mordechai Ben-Ari a partir de los compiladores del BACI (Ben-Ari Concurrency Interpreter), desarrollado por Bill Bynum y Tracy Camp, y del intérprete BACI Debugger, desarrollado por David Strite (15).

Los compiladores BACI traducen programas concurrentes en subconjuntos de Pascal y C a un lenguaje intermedio llamado PCode. Soportan primitivas de sincronización como semáforos y monitores. El BACI está escrito en C, así que hay que construir versiones separadas para cada plataforma. El depurador BACI es un intérprete de PCode escrito en Java para que sea portátil. El depurador permite puntos de ruptura y un solo paso por la declaración de la fuente o la instrucción PCode, y la interfaz gráfica de usuario incluye una completa visualización del código fuente, el PCode, las variables, la pila y el historial de las instrucciones PCode ejecutadas (15).

Entre sus características se destacan:

- Incluye un editor, la invocación de los compiladores y el intérprete.
- La tabla de procesos muestra el estado de cada proceso.

- Amplia configurabilidad: cadenas, teclas y fuentes están en un archivo de configuración en tiempo de compilación para facilitar la localización de la interfaz gráfica de usuario.

El entorno JBACI permite la ejecución paso a paso, pudiendo realizar operaciones de depuración y obtener el valor de las variables tanto globales como las internas de los semáforos y monitores. Esta herramienta es multiplataforma y se encuentra disponible para su descarga desde la web, su última versión (1.4.6) es del año 2017 (16).

Tiene como principal desventaja el hecho de que puede abrumar a los estudiantes con la presentación del código, puesto que requiere la comprensión de los problemas y la interpretación del código que lo soluciona (en lenguaje C o Pascal) de forma simultánea. Esta problemática ocasiona que el tiempo dedicado a la comprensión de un ejercicio sea prolongado y ocupe la mayor parte del planificado para la actividad de laboratorio.

El Sistema Abierto para La Enseñanza de Sistemas Operativos mediante un Simulador, expuesto en la Revista Científica de Tecnología Educativa de la ciudad de Argentina; tiene como principal objetivo que los alumnos de una cátedra universitaria de SO puedan ejecutar simulaciones de algoritmos de administración de recursos de los sistemas operativos en un entorno web; para el desarrollo de esta herramienta utilizaron una serie de applets en el lenguaje Java, que se ejecutan desde un sitio web, o localmente, en donde se podrá observar y estudiar el funcionamiento de algunos algoritmos de administración de recursos de los sistemas operativos.

Esta herramienta simula conjunto de peticiones de operaciones de acceso a los discos, que configuran cantidades de trabajo distintas a ser atendidas, calculando los tiempos de llegada, tiempo de servicio, tiempo de espera y capacidad de cola; también muestra informes detallados de los cálculos efectuados, un análisis estadístico de los resultados y un gráfico de los datos obtenidos (17).

La herramienta Simulador de Gestión de Memoria, se desarrolló en el lenguaje Java, la primera versión de la misma se expuso en el contexto de un proyecto final de la carrera dentro de la Escuela de Informática de la Universidad Politécnica de Valencia; con el tiempo se han incorporaron nuevas versiones con interfaz mejorada, parte del trabajo realizado en el ámbito de un proyecto europeo de educación a distancia. Su principal objetivo es mostrar el funcionamiento de algunos aspectos de la gestión de memoria que intervienen en la ejecución de uno o más procesos en un entorno de multiprogramación.

Concretamente, la herramienta describe la ejecución de un conjunto de procesos mediante el algoritmo de planificación round-robin y como estos procesos generan durante su ejecución accesos a direcciones lógicas que serán traducidas a direcciones de memoria física (18).

Operating System Concepts Simulator (OS Sim) Es una aplicación de propósito educativo para simular gráficamente conceptos relacionados con los SO y complementar así el aprendizaje de los estudiantes. La aplicación se divide en cuatro bloques o simulaciones independientes: planificación de procesos, gestión de memoria, gestión del sistema de ficheros y políticas de planificación de discos (19). Más allá de pretender describir los complejos sistemas reales, se centra en simular el funcionamiento de los componentes básicos fundamentales de los SO, se incluyen algunos de los algoritmos impartidos en clase y sus parámetros más relevantes. Además, dispone de una serie de ejemplos y ejercicios con las simulaciones correspondientes que permiten explorar las posibilidades que ofrece la aplicación (19). Se encuentra en la versión 1.2 desde el año 2011.

La navegación a través de la aplicación puede realizarse de forma intuitiva, sin embargo, las simulaciones no arrojan datos de fácil comprensión si no se domina a fondo el contenido. Además, no representa los mecanismos de gestión de memoria virtual, tiene dificultades en los esquemas de memoria y en particiones fijas hay que definir previamente todas las particiones, por lo que al final resulta engorrosa de utilizar. Aunque no se corresponde en su totalidad con los contenidos abordados en el programa de estudios, por lo que se sugiere que no se utilice para la enseñanza de la asignatura, puede servir como apoyo a la auto preparación de los profesores y para el diseño de actividades evaluativas.

Herramienta de simulación para la Planificación de Procesos. Esta permite simular el trabajo del Planificador de Procesos bajo distintas cargas de trabajo, dando al alumno la posibilidad de definir las características del SO y evaluar su desempeño según parámetros preestablecidos. Muestra la evolución de una planificación previamente configurada con determinados conceptos significativos como: tipos de procesos, ráfagas de CPU necesarias para ejecutar cada proceso y política de planificación (20).

Brinda al alumno la posibilidad de analizar el comportamiento de las distintas políticas de planificación en un SO, proporcionándole así la capacidad de entender, modificar o agregar otra política, con el fin de observar y/o deducir los cambios consecuentes en el sistema, además de visualizar los resultados

estadísticos obtenidos para que los estudiantes puedan elaborar sus propias conclusiones de qué algoritmo se comporta mejor bajo las mismas condiciones del sistema (20).

Es una herramienta multiplataforma, de código abierto por lo que puede ser modificado, además tiene una interfaz de usuario amigable y de navegación intuitiva por lo que puede resultar atractiva a los estudiantes. Sin embargo, no se encuentra disponible un enlace para su descarga desde la web, por lo que no resulta posible su utilización como parte de la solución a la problemática planteada en la presente investigación.

La herramienta VMFS fue creada con fines pedagógicos siendo posible estudiar a través de ella cómo se gestiona en MINIX la asignación del espacio en disco a ficheros, así como las distintas estructuras de datos que se emplean en dicha gestión para implementar los siguientes tipos de ficheros: directorios, ficheros regulares, enlaces físicos y simbólicos (21). Muestra de manera gráfica el sistema de ficheros y permite realizar modificaciones a este sistema mientras se encuentra en ejecución.

Al ser una herramienta creada sobre MINIX puede ser utilizada en sistemas Linux, no así en Windows; además, solo muestra los aspectos relacionados con los i-nodos, no siendo posible el trabajo con la FAT. Debido a esto no se considera adecuada para su utilización puesto que no abarca el contenido a impartir sobre esta temática en la asignatura. A pesar de ser una herramienta libre y de código abierto, no se encuentra disponible para ser descargada desde le web.

1.2.2 Herramientas de simulación para la enseñanza de Sistemas Operativos utilizadas en universidades cubanas

Uno de los algoritmos de planificación más usados para evitar los bloqueos se debe a Edsger Wybe Dijkstra y se conoce con el nombre de: Algoritmo del banquero (22). Este se modela de la forma en que el banquero de un pueblo trabajaría con un grupo de clientes a los que ha otorgado líneas de crédito (23).

En el año 2013 en el Instituto Superior Minero Metalúrgico de Moa se desarrolló un software que permite simular el algoritmo del banquero. Esta herramienta es de gran ayuda para el profesor puesto que, una vez impartida la conferencia sobre este tema, puede ser usada para la realización de actividades independientes de los estudiantes y lleva a estos a una mayor motivación y entendimiento de la asignatura. No necesita de computadoras con elevadas prestaciones para uso y ayuda al estudiante a entender de manera rápida el funcionamiento de este algoritmo una vez ya impartido en clases (24).

La principal desventaja de la herramienta es que está diseñada para ser ejecutada sobre Windows, lo que dificulta su utilización en otros SO. Sin embargo, es una propuesta acertada, que se puede modificar y adaptar a las necesidades actuales y ser utilizada en la impartición de la asignatura.

Simulador del Módulo de Reemplazo de Páginas (SMRP) es una aplicación de escritorio construida con el IDE NetBeans 7.0, usando el lenguaje de programación Java. Permite conocer el estado de la memoria física de un ordenador hipotético que, como parte de su Sistema Operativo, implementa un módulo de reemplazo de páginas de memoria (25). Esta herramienta simula una abstracción de una pequeña parte de todo el proceso que realiza la unidad de administración de memoria para responder ante un fallo de página, a través de tres de estos algoritmos (FIFO, LRU y LFU). El software muestra el estado de la memoria física luego de ser atendida cada solicitud de una lista de peticiones de páginas, además de indicar la cantidad de fallos que generan para esa lista de accesos el algoritmo seleccionado con el número de marcos predefinido (25).

El usuario establece directamente la secuencia de solicitudes, así como la cantidad de marcos en la memoria física y el algoritmo de reemplazo de página (de los tres que se ofrecen) que decide usar para su lista de peticiones. Además, brinda a los estudiantes la posibilidad de confrontar sus propias ejecuciones de las políticas de reemplazo con la ejecución de un sistema que las implementa. Esta aplicación también le permite al profesor comprobar las soluciones de ejercicios sobre este tema que realicen sus estudiantes (25).

A pesar de estas facilidades esta herramienta no abarca todos los contenidos que se imparten relacionados con la gestión de memoria, sólo permite la selección de tres algoritmos, obviando 2 de los que se imparten en clases. Por tal motivo no se recomienda su utilización para la asignatura.

En la Universidad de Ciencias Informáticas, también se planteó en un trabajo de diploma enfocado al diseño de una herramienta didáctica para la enseñanza de los algoritmos de gestión de los sistemas operativos, esta recibe el nombre de AlgSim (26). Esta herramienta tiene como objetivos generales:

- Valorar las características de diferentes SO.
- Establecer las diferencias entre SO distribuidos y no distribuidos.
- Emitir criterios acerca del SO más apropiado a ser utilizado en un ambiente de cómputo.
- Utilizar como eficiencia las facilidades brindadas por determinados SO.

- Configurar SO.
- Asimilar un nuevo sistema de operación a partir del estudio de sus manuales técnicos y de usuarios.
- Hacer uso de las técnicas utilizadas en la elaboración de los SO para la programación de sistemas Informáticos.
- Manejar literatura en idioma extranjero con respecto a los SO y sus técnicas de diseño. (26)

Sin embargo, no satisface las necesidades que originan la presente investigación, pues el usuario a través de ejercicios solo puede ejecutar los diferentes algoritmos de planificación de procesador, reemplazo de páginas y planificación de las peticiones de SO, dejando sin implementar funcionalidades que permitan la simulación de los contenidos relacionados con el interbloqueo. Además, cuenta con errores en su programación por lo que ocasionalmente devuelve resultados erróneos luego de la ejecución de los algoritmos.

1.2.3 Análisis de las herramientas de simulación estudiadas

A continuación, se muestra en la Tabla 1 un resumen las características de las herramientas estudiadas como parte del estado del arte:

Tabla 1. Análisis de las características de las herramientas de simulación estudiadas

Herramienta	**Año de lanzamiento de la última versión**	**Disponible para su descarga**	**Sistema operativo sobre el que se ejecuta**
Herramienta gráfica de simulación y exploración interactiva para la enseñanza de SO	2019	Uso Online. La versión local no es gratuita.	-
jBACI	2017	Si	Multiplataforma
Sistema Abierto para La Enseñanza de Sistemas Operativos mediante un Simulador	2013	Uso Online	-
Simulador de Gestión de Memoria	2012	Si	Multiplataforma

Operating System Concepts Simulator (OS Sim)	2011	Si	Multiplataforma
Herramienta de simulación para la Planificación de Procesos	2013	No	Multiplataforma
VMFS	2006	No	Linux
Algoritmo del banquero	2013	No	Windows
Simulador del Módulo de Reemplazo de Páginas (SMRP)	2013	Si	Multiplataforma
AlgSim	2016	Si	Multiplataforma

A partir del análisis de estas características se desechan aquellas que no están disponibles para su descarga, que no pueden ser ejecutadas en sistemas operativos libres, o cuya última versión fue realizada hace más de 5 años. A continuación, se muestran en la Tabla 2 las que no cumplen con los requisitos anteriores para determinar su correspondencia con los contenidos que son posibles de simular, impartidos en la asignatura y verificar si dan solución a la problemática de la investigación:

Tabla 2. Compatibilidad de las herramientas con los contenidos de la asignatura

Herramientas	**Contenidos de la asignatura que aborda**			
	Planificación de procesos	**Interbloqueo**	**Gestión de memoria**	**Gestión de Entrada/Salida**
Herramienta gráfica de simulación y exploración interactiva para la enseñanza de SO	X			
jBaci	x	x		
AlgSim	x		x	x

Como se puede apreciar en la tabla, ninguna de las herramientas estudiadas cubre todo el contenido de la asignatura SO, algunas tratan temas o temáticas en específico, se apoyan de materiales ya obsoletos con corta información o son diseñadas para ser ejecutadas sobre Windows. Por tal motivo, ninguna de ellas

soluciona la problemática que da origen a esta investigación, lo que corrobora la necesidad de desarrollar una nueva herramienta que permita dar solución a la misma.

1.3. Metodología de desarrollo

La metodología de desarrollo de software es un marco de trabajo usado para estructurar, planificar y controlar el proceso de desarrollo en sistemas de información; estas se clasifican en dos clases: tradicionales o robustas y ágiles o ligeras (27). Para el desarrollo de la solución se utilizará una metodología ágil, ya que permite adaptar la forma de trabajo a las condiciones del proyecto, consiguiendo flexibilidad e inmediatez en la respuesta y su desarrollo a las circunstancias específicas del entorno. En esencia este tipo de metodología acorta los ciclos de producción y minimiza los tiempos de reacción y toma de decisiones. Existen diferentes opciones agiles, pero las más usadas son:

Scrum

Se basa en una estructura de desarrollo incremental, es la llamada metodología del caos, pues se realiza a través de entregas parciales de los resultados del proyecto. Esta metodología requiere una exhaustiva definición de las tareas y sus plazos, además, exige que quienes la utilicen cuenten con una alta formación: por ello esta específicamente indicada para proyectos en entornos complejos. En Scrum un proyecto se ejecuta en bloques temporales cortos y fijos; cada iteración tiene que proporcionar un resultado completo (28).

Programación Extrema (XP)

Se basa en un conjunto de reglas y buenas prácticas para el desarrollo de software en ambientes muy cambiantes con requisitos imprecisos. Esta metodología requiere de un control de las versiones, un mayor esfuerzo de trabajo y tiempo, además se considera relativamente cara. En el contexto de XP se pueden identificar algunas irregularidades, como son; cliente bien definido y que los requisitos pueden cambiar a través del proceso de desarrollo (29).

AUP-UCI

La metodología de desarrollo AUP-UCI tiene como objetivo aumentar la calidad del software que se produce, para ello se apoya en CMMI-DEV (Capability Maturity Model Integration for Development, por sus siglas en inglés) v1.3. Este modelo constituye una guía para aplicar las mejores prácticas en una entidad desarrolladora de software. Estas prácticas se centran en el desarrollo de productos y servicios de calidad. En el caso de la variación de la metodología AUP definida para la actividad productiva de la UCI, la misma entre las especificaciones que realiza propone para el ciclo de vida de los proyectos las fases: Inicio, Ejecución y Cierre. De igual manera propone 7 disciplinas: Modelado de negocio, Requisitos, Análisis y diseño, Implementación, Pruebas internas, Pruebas de liberación y Pruebas de aceptación. A partir de que el Modelado de negocio propone tres variantes a utilizar en los proyectos (Casos de Uso del Negocio, Descripción de Proceso de Negocio o Modelo Conceptual) y existen tres formas de encapsular los requisitos (Casos de Uso del Sistema, Historias de usuario, Descripción de requisitos por proceso), surgen cuatro escenarios para modelar el sistema en los proyectos (30):

- Escenario 1: proyectos que modelan el negocio con casos de uso del negocio (CUN) solo pueden modelar el sistema con casos de uso del sistema (CUS).
- Escenario 2: proyectos que modelan el negocio con modelo conceptual (MC) solo pueden modelar el sistema con casos de uso del sistema (CUS).
- Escenario 3: proyectos que modelan el negocio con descripción de proceso de negocio (DPN) solo pueden modelar el sistema con descripción de requisitos de procesos (DRP).
- Escenario 4: proyectos que no modelen negocio solo pueden modelar el sistema con historias de usuario (HU).

A partir de la experiencia de la investigadora utilizando esta metodología y teniendo en cuenta sus características antes mencionadas, se selecciona para desarrollo de la solución propuesta AUP-UCI en su escenario 4. Esta es una metodología ágil y se adapta al ciclo de vida definido para la actividad productiva de la UCI, además se basa en la retroalimentación continua entre el cliente y el equipo de desarrollo lo que permite convenir los detalles de los requisitos y así poder implementarlos, probarlos y validarlos. Esta metodología potencia la comunicación fluida entre todos los actores involucrados en el proceso de desarrollo, la simplicidad en las soluciones implementadas y las facilidades para enfrentar los cambios.

1.4 Tecnologías y herramientas

1.4.1 Modelado del sistema

El lenguaje Unificado de Modelado (UML) es un lenguaje grafico para visualizar, especificar, construir y documentar un sistema; además, capta la información sobre la estructura estática y el comportamiento dinámico de un sistema. También contiene construcciones organizativas para agrupar los modelos en paquetes, lo que permite a los equipos de software dividir grandes sistemas en piezas de trabajo, para entender y controlar las dependencias entre paquetes, y para gestionar las versiones de las unidades del modelo, en un entorno de desarrollo complejo (31). Las funciones de UML se pueden sintetizar en:

- Visualizar: Permite expresar de una forma gráfica un sistema, de manera que otra persona lo puedan entender.
- Especificar: Permite especificar cuáles son las características de un sistema antes de su construcción.
- Construir: A partir de los modelos especificados se pueden construir los sistemas diseñados.
- Documentar: Los propios elementos gráficos sirven como documentación del sistema desarrollado que pueden servir para su futura revisión.

Debido a estas características se empleará UML en su versión 2.5.1 para el modelado de la solución.

Herramientas CASE (Ingeniería de Software Asistida por Computadoras)

Son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y dinero. Estas herramientas nos pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del software en tareas como el diseño de proyectos, cálculo de costes, implementación de una parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores, entre otras (32).

Visual Paradigm for UML es una herramienta CASE. La misma propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación de un código fuente de programas y documentación. Es una herramienta que soporta un

conjunto de lenguajes, tanto en generación de código e ingeniería inversa en: Java, C++, PHP y Python. Además, ofrece un entorno de creación de diagramas de UML, usa un lenguaje estándar, dispone de múltiples versiones y múltiples plataformas. Esta herramienta permite la realización de una amplia gama de diagramas como: casos de uso, de actividades, de despliegue, entre otros, así como la generación de código fuente desde los mismos y la documentación asociada al proceso que esté siendo modelado (31). Por tal motivo es la seleccionada en su versión 1.6.3 para la realización de los diagramas que componen la solución.

1.4.2 Lenguaje de programación

Un lenguaje de programación es un lenguaje artificial que le proporciona al programador, la capacidad de escribir o programar una serie de instrucciones o secuencias de órdenes en forma de algoritmos con el fin de controlar el comportamiento físico o lógico de un sistema informático, de manera que se puedan obtener diversas clases de datos o ejecutar determinadas tareas. Existen diversos lenguajes de programación: PHP, C++, Phyton y Java, estos están formados por un conjunto de símbolos, reglas gramaticales y semánticas.

PHP (Hypertext Preprocessor)

Es un lenguaje de programación de uso general que se adapta especialmente al desarrollo de web; es considerado un lenguaje libre, por lo que se presenta como una alternativa de fácil acceso para todos; el código PHP es procesado a través de un servidor web implementado con modulo, programa residente o como un ejecutable de interfaz de entrada común; este puede ser ejecutado por varios tipos de datos, como: HTML y HTTP. El gran inconveniente de este tipo de lenguaje es que el código fuente no pueda ser ocultado de una manera eficiente y solo se ejecuta en un servidor y necesita un servidor web para que funcione (33).

Phyton

Es un lenguaje de programación interpretado, su principal filosofía es que sea legible por cualquier persona con conocimientos básicos de programación (34). Además, posee unas series de características que lo hacen muy particular y que sin duda, le aportan muchas ventajas; es un lenguaje multiparadigma, ya que combina las propiedades de diferentes paradigmas de programación, lo que permite que sea flexible y fácil

de aprender de manera independiente. Phyton es apto para todas las plataformas, podemos ejecutarlos en diferentes sistemas operativos como Windows o Linux (34).

Los lenguajes escritos con códigos Phyton se consideran un poco lentos, ya que al ser un lenguaje dinámico e interpretado, su código se ejecuta línea por línea, lo que conlleva a una ejecución más lenta. Este también es usado generalmente para la programación del lado del servidor debido a su lento procesamiento (34).

Java

Es un lenguaje de multiplataforma orientada a objetos; cuenta como máquina virtual que permite ejecutar cualquier código compilado, además su uso es gratuito, y es un código abierto (35). Java ofrece la funcionalidad de un lenguaje potente, derivado de C y C++ y proporciona una gran biblioteca estándar y herramientas para que los programas puedan ser distribuidos; una de sus principales ventajas es que logra llevar a cabo varias tareas simultáneamente dentro del mismo programa, esto permite mejorar el rendimiento la velocidad de ejecución (35).

La característica principal (orientado a objeto), se refiere a un método de programación y al diseño del lenguaje; un objeto puede verse como un paquete que contiene el código y los datos; objetivamente es para que el desarrollo de grandes proyectos sean fáciles de gestionar y manejar, mejorando como consecuencia la calidad y reduciendo el número de proyectos fallidos (36).

Debido a que es multiplataforma, presenta una amplia documentación lo cual facilita el trabajo al equipo de desarrollo y basado en la experiencia del equipo en la utilización de este lenguaje, se elige Java en su versión 8 para el desarrollo de la propuesta de solución.

1.4.3 Entorno integrado de desarrollo

Un IDE (Entorno de Desarrollo Integrado), es un sistema informático para el diseño de aplicaciones que combina herramientas del desarrollador con una sola interfaz gráfica de usuario. Los IDE permiten que los desarrolladores comiencen a programar nuevas aplicaciones con mayor rapidez.

Netbeans

Es un entorno de desarrollo integrado libre, hecho principalmente para el lenguaje de programación Java; fue fundada por Sun Microsystems en junio 2000 (37). Esta plataforma permite que las aplicaciones sean desarrolladas a partir de un componente de software llamado módulos, cada uno de estos módulos son un archivo Java que contiene un conjunto de clases que interactúan con las APIs (37). El objetivo de esta arquitectura es favorecer el desarrollo de funcionalidades de forma independiente y la reutilización de componentes. Entre sus características se encuentra un sistema de proyectos basados en Ant, control de versiones y refactoring. El neatbens permite el desarrollo de todos los tipos de aplicación Java (37), por ello fue elegido en la versión 12.5 para solucionar la problemática planteada.

Conclusiones del capítulo

Sistemas Operativos, es una asignatura con alto contenido que presenta una alta complejidad, por ende, necesita recursos y herramientas que apoyen la comprensión del funcionamiento interno de los sistemas por parte de los estudiantes y así puedan aprender a interactuar con ellos y modificarlos. Esto fue demostrado en los ejemplos mencionados en el estado del arte, donde muchas universidades y centros de desarrollo planteaban que era de gran necesidad hacer uso de herramientas de simulación en la asignatura que trata los sistemas.

Sin embargo, este estudio demostró que, aunque existe un grupo de herramientas de simulación creadas con la finalidad de apoyar el proceso docente-educativo, la utilización de las mismas no es una práctica común dentro de las universidades extranjeras y nacionales. Este fenómeno tiene su origen en la variedad de las herramientas, el grado en que se adaptan a los objetivos particulares de los programas de estudio de la asignatura en cada una de estas universidades y la capacidad de integración de diferentes subprocesos de los sistemas operativos que presentan. Este análisis, plasmado como parte del capítulo, reafirmó la necesidad de la creación de una nueva herramienta que solucione la problemática planteada al inicio de la investigación.

El análisis de las herramientas de simulación abordadas en la investigación, unido a las características del área donde impactará la solución y el público al que va dirigida, permitió determinar la metodología de desarrollo y las herramientas y tecnologías a emplear para el desarrollo de la propuesta de solución, teniendo como resultado AUP-UCI como metodología de desarrollo en su escenario 4, UML 2.5.1 para el

modelado de la solución, Visual Paradigm en su versión 1.6.3 para la realización de los diagramas, como lenguaje de programación Java 8 y Net Beans 12.5 como IDE.

Capítulo 2: Análisis y diseño de la solución

Introducción

En este capítulo se muestran los artefactos correspondientes a las fases de inicio y ejecución de la metodología escogida en el capítulo anterior. Se describe la solución propuesta al problema de investigación, además se hace un análisis sobre las características y cualidades de la herramienta descritas mediante la utilización de las Historias de Usuarios (HU). Se realiza la definición de los elementos de análisis y diseño identificando la arquitectura de software a utilizar.

2.1 Descripción de la solución

Como solución a la problemática planteada al inicio de la investigación se propone la creación de una herramienta para la simulación de los algoritmos que se estudian en la asignatura Sistemas Operativos. Este software permitirá visualizar de forma gráfica el comportamiento de los algoritmos para diferentes juegos de datos ya sea de forma integral o paso a paso, realizar comparaciones entre ellos, analizar las características de cada uno y graficar su comportamiento.

Se podrán simular los algoritmos empleados por el planificador para la atención a procesos, así como cálculos asociados a la utilización del CPU y la probabilidad de ocurrencia de bloqueo, técnicas para el tratamiento del bloqueo, algoritmos para la paginación y realización de cálculos asociados a la memoria virtual y la memoria física, algoritmos de planificación para el acceso al disco y duración de la ejecución de estos algoritmos. Para el desarrollo de la propuesta de solución se utilizarán las tecnologías y herramientas seleccionadas en el capítulo anterior.

El sistema estará compuesto por 5 módulos: uno para cada temática y otro para el programa principal; esto posibilita independencia para los temas y mejor adaptación al cambio sin afectar la aplicación. Además, permite que la aplicación este implementada de forma modular, pues el simulador puede implementarse construyendo un programa principal que utilice los módulos ya programados, estos se programan y se compilan por separados permitiendo integrarlos al sistema según se les necesite. En consecuencia, ofrece la posibilidad de reutilizar partes de código ya desarrolladas en el diseño de nuevos módulos sin necesidad

de volverlos a diseñar, lo que facilita la organización del código, por consiguiente, luego de que se implemente el simulador (programa principal) se añaden los módulos, sin que uno afecte al otro.

2.2 Requerimientos del sistema

Unos de los puntos fundamentales en el desarrollo de sistemas son la correcta obtención de los requisitos o ingeniería de requerimientos, la cual trata de establecer las funcionalidades fundamentales que debe cumplir el sistema, así como sus restricciones. Los requerimientos para un sistema son la descripción de los servicios proporcionados por el sistema y sus restricciones operativas. Estos requerimientos reflejan las necesidades de los clientes de realizar un sistema que ayude a resolver algún problema como el control de un dispositivo, hacer un pedido o encontrar información (38). Los requisitos se pueden clasificar en funcionales y no funcionales. Los requerimientos o requisitos funcionales son capacidades o condiciones que el sistema debe cumplir, mientras que los no funcionales describen las características y aspectos de carácter técnicos que el sistema debe poseer (27).

2.2.1 Requisitos funcionales

Los requisitos funcionales explican en detalle las tareas que el sistema debe ser capaz de realizar y expresan la naturaleza de su funcionamiento (cómo interacciona el sistema con su entorno y cuáles van a ser su estado y funcionamiento). Teniendo en cuenta esta definición el sistema debe:

RF 1. Administrar datos de procesos.

RF 2. Planificar los procesos existentes a través del algoritmo seleccionado.

RF 3. Visualizar a través de una simulación la planificación de procesos.

RF 4. Emitir estadísticas de la simulación de procesos.

RF 5. Administrar recursos.

RF 6. Visualizar los recursos asignados y solicitados por cada proceso.

RF 7. Ejecutar el algoritmo del banquero para la prevención del interbloqueo.

RF 8. Calcular la disponibilidad mínima necesaria de cada recurso para ejecutar una secuencia segura.

RF 9. Administrar datos de la memoria.

RF 10. Simular el funcionamiento de los algoritmos de reemplazo de páginas en sus políticas globales y locales.

RF 11. Visualizar la simulación de forma gráfica.

RF 12. Realizar análisis, comparativas y cálculos a partir de los resultados obtenidos en la simulación del algoritmo.

RF 13. Administrar cola de peticiones.

RF 14. Planificar el acceso al disco a través del algoritmo seleccionado.

RF 15. Visualizar a través de una simulación la planificación de acceso al disco.

RF 16. Realizar análisis, comparativas y cálculos a partir de los resultados obtenidos en la simulación del algoritmo teniendo en cuenta los instantes de tiempo en que se realizan las peticiones, tiempo de búsqueda, latencia y transferencia.

2.2.2 Requisitos no funcionales

Un requisito no funcional especifica criterios que pueden usarse para juzgar la operación de un sistema en lugar de sus comportamientos específicos, ya que estos corresponden a los requisitos funcionales. Son restricciones de los servicios o funciones ofrecidos por el sistema. Incluyen restricciones de tiempo sobre el proceso de desarrollo y estándares.

1. **Apariencia o interfaz externa:**
    - RNF1: El diseño de la interfaz debe ser sencillo y fácil de usar con reconocimiento visual a través de elementos visibles que identifiquen cada una de sus acciones.
    - RNF2: La combinación de colores debe ser agradable a la vista del usuario.
    - RNF3: Debe contar con un vínculo a la ayuda en cada ventana de trabajo.
2. **Usabilidad:**
    - RNF4: El sistema puede ser usado por cualquier estudiante o profesor que posea conocimientos básicos sobre el funcionamiento interno de un Sistemas Operativos.

3. **Portabilidad:**
    - RNF5: Debe poder ejecutarse en Windows y en cualquier distribución de Linux.

2.2.3 Validación de requisitos

La validación de los requisitos tiene como objetivo comprobar que estos son correctos. Esta fase debe realizarse o de lo contrario se corre el riesgo de implementar una mala especificación. Los parámetros a validar en los requisitos son:

- **Validez:** no basta con preguntar a un usuario, todos los potenciales usuarios pueden tener puntos de vista distintos y necesitar otros requisitos.
- **Consistencia:** no debe haber contradicciones entre unos requisitos y otros.
- **Completitud:** deben estar todos los requisitos. Esto es imposible en un desarrollo iterativo, pero, al menos, deben estar disponibles todos los requisitos de la iteración en curso.
- **Realismo:** se pueden implementar con la tecnología actual.
- **Verificabilidad:** tiene que existir alguna forma de comprobar que cada requisito se cumple.

Se escogió para la validación de los requisitos de la herramienta la técnica prototipo de interfaz de usuario debido a que la opinión del cliente es necesaria desde las fases iniciales del desarrollo. Según (39), los prototipos son medios de comunicación entre analistas, clientes o usuarios, que muestran las decisiones tomadas, con el fin de validarlas y permitir la resolución de los problemas de comprensión presentes en la etapa de levantamiento de requisitos. Los prototipos se pueden considerar como pequeñas implementaciones de un sistema de software, que ayudan al diseño y que pueden ser usadas como una técnica de determinación y validación de requerimientos. Los dos tipos principales de prototipos de interfaz de usuario son:

- **Desechables:** se utilizan sólo para la validación de los requisitos y posteriormente se desechan. Pueden ser prototipos en papel o en software.
- **Evolutivos:** una vez utilizados para la validación de los requisitos, se mejora su calidad y se convierten progresivamente en el producto final.

Se decidió emplear prototipos de interfaz desechables, ya que según (39) esta técnica facilita la evaluación del comportamiento y la descripción de las propiedades de un sistema. La técnica sugiere el uso de un

esquema preliminar de la interfaz que ayude a comprender mejor las necesidades del entorno, pues, por lo general, los usuarios no comunican correctamente los requerimientos. Este esquema se puede mostrar a través de ventanas o widgets, en este caso se optó por la variante de las ventanas.

A continuación, se muestra la Figura 1 con el prototipo de interfaz de usuario que permite validar los RF 1, 2, 3 y 4.

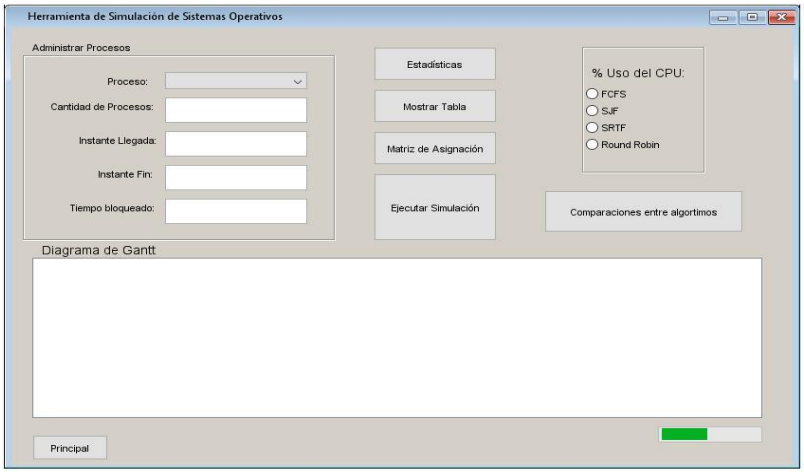

Figura 1. Prototipo de interfaz de usuario para la validación de los requisitos funcionales 1, 2, 3 y 4.

La Figura 2 muestra el prototipo para la validación de los requisitos 5, 6, 7 y 8:

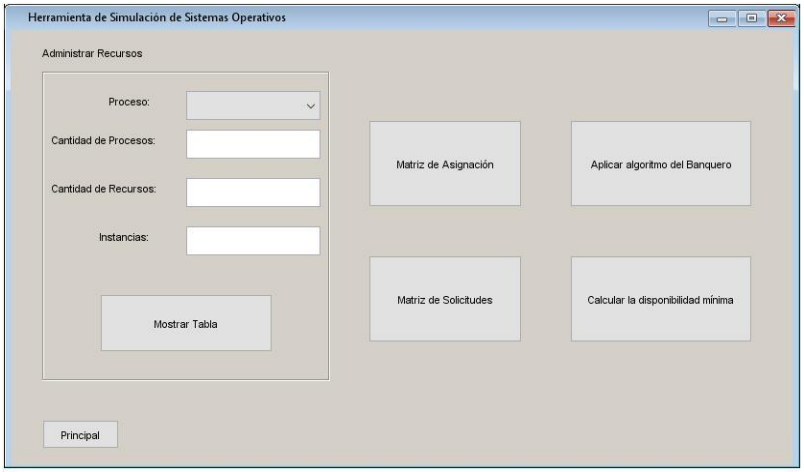

Figura 2. Prototipo de interfaz de usuario para la validación de los requisitos funcionales 5, 6, 7 y 8.

En la Figura 3 se muestra el prototipo de interfaz para la validación de los requisitos 9, 10, 11 y 12:

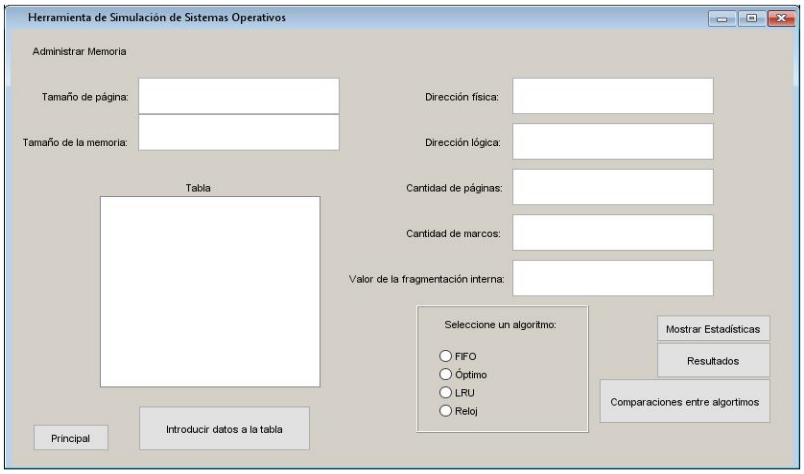

Figura 3. Prototipo de interfaz de usuario para la validación de los requisitos funcionales 9, 10, 11 y 12.

Por último, la Figura 4 muestra el prototipo desechable para la validación de los restantes requisitos funcionales (13, 14, 15 y 16):

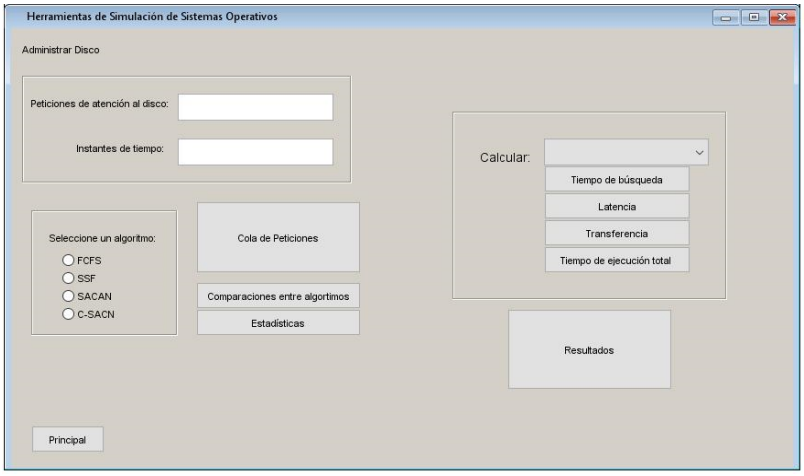

Figura 4. Prototipo de interfaz de usuario para la validación de los requisitos funcionales 9, 10, 11 y 12.

2.3 Historias de usuario

Las historias de usuarios conforman la parte central del Escenario 4 de la metodología AUP-UCI pues definen lo que se debe construir en el proyecto de software. Tienen una prioridad (Alta, Media, Baja)

asociada, definida por el cliente y la importancia que tiene para el funcionamiento de la herramienta las funcionalidades antes mencionadas.

El riesgo de desarrollo está dado por la ausencia del desarrollador por enfermedad o por pérdida de información imprescindible. El tiempo de cada funcionalidad será estimado por el desarrollador. La estimación del tiempo está dada de acuerdo al tiempo que toma la tarea en ser desarrollada. A continuación, se muestran las historias de usuario de mayor prioridad, definidas para el desarrollo de la herramienta, las restantes se encuentran relacionadas en el apartado Anexo 1: Historias de usuario.

Tabla 3. Historia de Usuario 2. Planificar los procesos existentes utilizando un algoritmo de planificación.

Historia de Usuario	
Número: 2	**Nombre:** Planificar los procesos existentes utilizando un algoritmo de planificación.
Usuario: Programador	**RF 2:** Planificar los procesos existentes a través del algoritmo seleccionado
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Alto
Puntos Estimados: 2	**Iteración Asignada:** 1
Programador Responsable: Thais Mel Soteras Fong	
Descripción: El usuario podrá el algoritmo que desea utilizar (FCFS, SJF, SRTF, Round Robin) además podrá observar las transformaciones que ocurren en el planificador a partir de los datos proporcionados.	
Observaciones:	

Tabla 4. Historia de Usuario 10. Aplicar el algoritmo del banquero para la prevención del interbloqueo.

Historia de Usuario	
Número: 10	**Nombre:** Aplicar el algoritmo del banquero para la prevención del interbloqueo
Usuario: Programador	**RF 7:** Ejecutar el algoritmo del banquero para la prevención del interbloqueo

Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Alta
Puntos Estimados: 2	**Iteración Asignada:** 2
Programador Responsable: Thais Mel Soteras Fong	
Descripción: Determinar una secuencia segura de ejecución de los procesos a partir de los datos insertados por el usuario aplicando el Algoritmo del Banquero.	
Observaciones: Esta funcionalidad solo se ejecutará en caso de no haberse insertado una secuencia segura de ejecución.	

Tabla 5. Historia de Usuario 11. Determinar la disponibilidad mínima necesaria de cada recurso para ejecutar una secuencia.

Historia de Usuario	
Número: 11	**Nombre:** Determinar la disponibilidad mínima necesaria de cada recurso para ejecutar una secuencia.
Usuario: Programador	**RF 8:** Calcular la disponibilidad mínima necesaria de cada recurso para ejecutar una secuencia segura
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Alto
Puntos Estimados: 2	**Iteración Asignada:** 2
Programador Responsable: Thais Mel Soteras Fong	
Descripción: El usuario podrá determinar la disponibilidad mínima necesaria para atender las solicitudes de cada proceso teniendo en cuenta la secuencia segura insertada por el usuario.	
Observaciones: Esta funcionalidad solo se ejecutará en caso de no haberse insertado una disponibilidad de recursos y una existencia.	

Tabla 6. Historia de Usuario 14. Simulación de algoritmos de reemplazo de páginas.

Historia de Usuario	
Número: 14	**Nombre:** Simulación de algoritmos de reemplazo de páginas.
Usuario: Programador	**RF 10:** Simular el funcionamiento de los algoritmos de reemplazo de páginas en sus políticas globales y locales.
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Alto
Puntos Estimados: 2	**Iteración Asignada:** 3
Programador Responsable: Thais Mel Soteras Fong	
Descripción: El usuario podrá realizar la traducción de las direcciones virtuales, seleccionar el algoritmo que desea utilizar (FIFO, Óptimo, LRU, Reloj) e indicar su política (global o local), además podrá observar las transformaciones que ocurren en la memoria física y las estadísticas que se producen en cada simulación.	
Observaciones: La gestión de páginas se realizará mediante el esquema de paginación simple.	

Tabla 7. Historia de Usuario 20. Simulación de algoritmos de planificación de acceso al disco.

Historia de Usuario	
Número: 20	**Nombre:** Simulación de algoritmos de planificación de acceso al disco.
Usuario: Programador	**RF 14:** Planificar el acceso al disco a través del algoritmo seleccionado
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Alto
Puntos Estimados: 1.5	**Iteración Asignada:** 4
Programador Responsable: Thais Mel Soteras Fong	
Descripción:	

Se realiza la simulación correspondiente a los algoritmos FIFO (primero en llegar, primero en salir), SSF (primero la búsqueda más cercana), SCAN (algoritmo del ascensor), CSCAN (variante circular del SCAN), LOOK y CLOOK (variante circular del LOOK), en dependencia de la selección de los usuarios y teniendo en cuenta la cola de solicitudes.
Observaciones:

2.4 Estimación de esfuerzo por historias de usuario

Las historias de usuarios seleccionadas para entregar en cada iteración, son desarrolladas y probadas en su ciclo correspondiente, teniendo en cuenta el orden preestablecido. Al comienzo de cada ciclo, se realiza una reunión de planificación de la iteración. A continuación, en la Tabla 3 se resume la estimación del esfuerzo realizada.

Tabla 8 Estimación de esfuerzo por historias de usuario

Historias de Usuario		Puntos Estimados (Semanas)
1	Administrar datos de procesos	**0.5**
2	Planificar los procesos existentes utilizando un algoritmo de planificación.	**2**
3	Visualizar la simulación instante a instante o de forma automática a través de un diagrama de Gantt.	**1.5**
4	Mostrar los datos de cada proceso, en cuanto a instante de llegada y de fin, tiempo que estuvo en estado listo, tiempo que estuvo en estado bloqueado, y tiempo que estuvo en ejecución.	**1**
5	Calcular y mostrar estadísticas relacionadas con: tiempo de espera medio, tiempo de retorno promedio y por ciento de uso del CPU.	**0.5**
6	Mostrar las gráficas que representan el tiempo de espera medio y tiempo de retorno medio de cada proceso.	**1**
7	Realizar comparaciones entre algoritmos de planificación.	**1**
8	Administrar datos de recursos	**1**
9	Elaborar las matrices de asignación y solicitud a partir de los datos de los recursos	**1.5**
10	Aplicar el algoritmo del banquero para la prevención del interbloqueo (determinar si existe una secuencia segura para la ejecución de los procesos)	**2**

11	Determinar la disponibilidad mínima necesaria de cada recurso para ejecutar una secuencia.	**2**
12	Administrar datos de la memoria y los procesos	**1**
13	Gestionar tabla de páginas de los procesos	**0.5**
14	Simulación de algoritmos de reemplazo de páginas	**2**
15	Visualizar la simulación instante a instante o de forma automática a través de la tabla de memoria	**1.5**
16	Análisis de resultados (Algoritmos de reemplazo de páginas)	**1**
17	Calcular y mostrar estadísticas relacionadas con: dirección física, dirección lógica, cantidad de páginas, cantidad de marcos, valor de la fragmentación interna.	**1**
18	Realizar comparaciones entre algoritmos de reemplazo de páginas.	**1**
19	Administrar cola de peticiones	**0.5**
20	Simulación de algoritmos de planificación de acceso al disco	**1.5**
21	Visualizar la simulación instante a instante o de forma automática a través de un gráfico	**1.5**
22	Calcular y mostrar estadísticas relacionadas con: instantes de tiempo en que se realizan las peticiones, tiempo de búsqueda, latencia y transferencia.	**1**
23	Realizar comparaciones entre algoritmos de planificación de acceso a disco.	**0.5**
	Total	**27**

2.5 Plan de iteraciones

Como resultado de determinar dicho flujo, aparece la plantilla plan de iteraciones (Tabla 4), en la que se recogen las iteraciones a realizar con sus características, además del orden de las historias de usuario con su planificación estimada para ser implementadas. En la tabla plan de iteraciones, se incluyen los siguientes campos: **Iteración:** que contiene el identificador de la iteración que se va a desarrollar; **Historias de usuario:** se enuncian las historias de usuario a desarrollar en cada iteración y **Duración total:** cantidad de semanas que durará realizar la iteración, la que depende del tiempo estimado de las HU propuestas.

Tabla 9 Plan de iteraciones

Iteración	Historias de Usuario		Duración (Semanas)
1	**1**	Administrar datos de procesos	**7.5**
	2	Planificar los procesos existentes utilizando un algoritmo de planificación.	

	3	Visualizar la simulación instante a instante o de forma automática a través de un diagrama de Gantt.	
	4	Mostrar los datos de cada proceso, en cuanto a instante de llegada y de fin, tiempo que estuvo en estado listo, tiempo que estuvo en estado bloqueado, y tiempo que estuvo en ejecución.	
	5	Calcular y mostrar estadísticas relacionadas con: tiempo de espera medio, tiempo de retorno promedio y por ciento de uso del CPU.	
	6	Mostrar las gráficas que representan el tiempo de espera medio y tiempo de retorno medio de cada proceso.	
	7	Realizar comparaciones entre algoritmos de planificación.	
2	**8**	Administrar datos de recursos	**6.5**
	9	Elaborar las matrices de asignación y solicitud a partir de los datos de los recursos	
	10	Aplicar el algoritmo del banquero para la prevención del interbloqueo (determinar si existe una secuencia segura para la ejecución de los procesos)	
	11	Determinar la disponibilidad mínima necesaria de cada recurso para ejecutar una secuencia.	
3	**12**	Administrar datos de la memoria y los procesos	**8**
	13	Gestionar tabla de páginas de los procesos	
	14	Simulación de algoritmos de reemplazo de páginas	
	15	Visualizar la simulación instante a instante o de forma automática a través de la tabla de memoria	
	16	Análisis de resultados (Algoritmos de reemplazo de páginas)	
	17	Calcular y mostrar estadísticas relacionadas con: dirección física, dirección lógica, cantidad de páginas, cantidad de marcos, valor de la fragmentación interna.	
	18	Realizar comparaciones entre algoritmos de reemplazo de páginas.	
4	**19**	Administrar cola de peticiones	**5**
	20	Simulación de algoritmos de planificación de acceso al disco	
	21	Visualizar la simulación instante a instante o de forma automática a través de un gráfico	
	22	Calcular y mostrar estadísticas relacionadas con: instantes de tiempo en que se realizan las peticiones, tiempo de búsqueda, latencia y transferencia.	
	23	Realizar comparaciones entre algoritmos de planificación de acceso a disco.	
Total			**27**

Iteración 1: Esta iteración tiene como objetivo la implementación de las historias de usuario de la 1 a la 7, las cuales abarcan la simulaciones de las cuestiones relativas a la planificación de procesos.
Iteración 2: Esta iteración tiene como objetivo la implementación de las historias de usuario de la 8 a la 11, las cuales abarcan la simulaciones de las cuestiones relativas al interbloqueo.
Iteración 3: Esta iteración tiene como objetivo la implementación de las historias de usuario de la 12 a la 19, las cuales abarcan la simulaciones de las cuestiones relativas a la memoria virtual y algoritmos de reemplazo de página.
Iteración 4: Esta iteración tiene como objetivo la implementación de las historias de usuario de la 20 a la 23, las cuales abarcan la simulaciones de las cuestiones relativas a la planificación del acceso a disco.

2.6 Descripción de la arquitectura del sistema

La arquitectura de software puede ser vista como la estructura del sistema en función de la definición de los componentes y sus interacciones. Teniendo en cuenta que en el sistema que se desea desarrollar la transformación de los datos se realizará en pasos sucesivos y a través de una serie de componentes para el cálculo de los mismos, se puede decir que posee un funcionamiento basado en flujo de datos.

El flujo de datos se basa en un patrón tuberías y filtros, este consiste en ir transformando un flujo de datos en un proceso comprendido por varias fases secuenciales, siendo la entrada de cada una la salida de la anterior. Consta de un conjunto de componentes denominados filtros que conectados entre sí por tuberías transmiten los datos desde un componente al siguiente. Cada filtro trabaja de manera independiente de los componentes que se encuentren situados antes o después de él. Los filtros se encuentran diseñados para recibir la entrada de datos de una forma y producir la salida de datos en una forma específica. Si el flujo de datos genera una línea de transformaciones se denomina secuencial por lotes (27):

- Tuberías y filtros: una tubería (pipeline) es una popular arquitectura que conecta componentes computacionales (filtros) a través de conectores (pipes), de modo que las computaciones se ejecutan a la manera de un flujo. Los datos se transportan a través de las tuberías entre los filtros, transformando gradualmente las entradas en salidas. El sistema tubería-filtros se percibe como una serie de transformaciones sobre sucesivas piezas de los datos de entrada. Los datos entran al sistema y fluyen a través de los componentes (40).

- Secuencial por lotes: los componentes son programas independientes; el supuesto es que cada paso se ejecuta hasta completarse antes que se inicie el paso siguiente. Varios autores sostienen que la variante por lotes es un caso degenerado del estilo, en el cual las tuberías se han vuelto residuales (41).

En la herramienta a desarrollar se evidencia el uso del patrón Secuencial por lotes porque, específicamente en los módulos, los datos fluyen a través de las diferentes condiciones de ejecución del algoritmo que se esté simulando, devolviendo una salida específica: en este caso, la simulación gráfica del comportamiento del algoritmo.

2.6.1 Patrones

Un patrón de diseño provee un esquema para refinar los subsistemas o componentes de un sistema de software, o las relaciones entre ellos. Describe la estructura comúnmente recurrente de los componentes en comunicación, que resuelve un problema general de diseño en un contexto particular (38). Para el desarrollo de la propuesta de solución se utilizarán los siguientes patrones:

- Experto en información: las responsabilidades deben ser asignadas a las clases que poseen la información para realizar dicha responsabilidad.
- Bajo acoplamiento: es la idea de tener las clases lo menos ligadas entre sí que se pueda. De tal forma que, en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de clases, potenciando la reutilización, y disminuyendo la dependencia entre las clases.
- Alta cohesión: una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme. Significa que las clases del sistema tienen asignadas solo las responsabilidades que les corresponde y mantienen una estrecha relación con el resto de las clases.
- Creador: el patrón creador ayuda a identificar quién debe ser el responsable de la creación (o instanciación) de nuevos objetos o clases. La nueva instancia deberá ser creada por la clase que tiene la información necesaria para realizar la creación del objeto, usa directamente las instancias creadas del objeto, almacena o maneja varias instancias de la clase o contiene o agrega la clase.

Una de las consecuencias de usar este patrón es la visibilidad entre la clase creada y la clase creador.

- Controlador: es el encargado de asignar la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase que represente una de las siguientes opciones. Se evidencia el uso de este patrón en la aplicación, ya que para cada petición o evento que se genere en el mismo, existe un controlador con la responsabilidad de obtenerla y devolver una respuesta. La respuesta puede ser mostrar una vista, ejecutar un método, devolver un mensaje, etc.
- Polimorfismo: es útil cuando se identifican variaciones en un comportamiento para asignar la clase (interfaz) al comportamiento y utilizar polimorfismo para implementar los comportamientos alternativos.

Conclusiones del capítulo

En el transcurso del presente capítulo, se presentó la propuesta de solución para darle respuesta a la problemática planteada al inicio de la investigación. Como parte del proceso de análisis y diseño de la solución propuesta se realizó la extracción de 16 requisitos funcionales que fueron posteriormente validados a través de la técnica de prototipo de interfaz de usuario desechable y descritos en 23 historias de usuarios y se precisaron 5 requisitos no funcionales de la herramienta a desarrollar. Se definió la arquitectura sobre la cual se construye el sistema, además, se obtuvieron los artefactos generados por la metodología empleada, los cuales permitien una mejor comprensión de las funcionalidades, elementos y guía de trabajo durante el desarrollo. El estilo arquitectónico definido, así como los patrones a utilizar resultan una forma de representación acertada en relación al funcionamiento del sistema. Al concluir el presente capítulo, se han creado las condiciones para efectuar la implementación de la herramienta propuesta.

Capítulo 3: Implementación y pruebas

Introducción

En el actual capítulo se describen los resultados obtenidos durante la ejecución de las pruebas teniendo en cuenta el flujo de implementación del sistema, con el objetivo de definir la organización del código, la implementación de los elementos de diseño y la integración de los diferentes componentes, generando un ejecutable entregable o producto final. Cabe señalar que luego de su culminación, este debe cumplir con todos los requisitos funcionales y no funcionales definidos en el capítulo anterior.

3.1 Implementación

La implementación constituye una de las fases más importantes del desarrollo de software. En ella se toman como punto de partida los resultados obtenidos en el diseño, implementándose el sistema en términos de componentes como ficheros de código binario, código fuente, scripts y ejecutables. Su importancia reside en que se obtiene como consecuencia un sistema ejecutable, siendo esto uno de los principales objetivos en el desarrollo de software.

Para describir las tareas llevadas a cabo en la fase de implementación, se emplea un lenguaje técnico, el cual no necesariamente debe ser entendible por el cliente. Dichas tareas son asignadas al equipo o programador responsable, normalmente la codificación se lleva a cabo por una pareja de programadores. Esta labor se lleva a cabo con el objetivo de detallar mejor las historias de usuario, lo cual facilita el entendimiento en el proceso de implementación.

3.1.1 Tareas de ingeniería o programación

Para alcanzar los objetivos de una iteración es necesario completar las HU que están presentes en estas, por lo que se hace necesario saber cuáles son las tareas de ingeniería que componen cada una, las cuales harán posible el cumplimiento de los objetivos de cada HU. En ellas están presentes los siguientes campos:

- **Número de tarea:** que contiene un número consecutivo en base a la historia de usuario correspondiente
- **Número historia de usuario:** que contiene el identificador de la HU a la que pertenece esta tarea

- **Nombre tarea:** contiene un nombre que identifica a la tarea
- **Tipo de tarea:** contiene el tipo de tarea, que puedes ser de desarrollo, corrección, mejora, o la especificación de otra
- **Puntos estimados:** estimación en semanas de la duración de la tarea
- **Responsable:** programador responsable de ejecutar la tarea
- **Descripción:** contiene la descripción de la tarea.

A continuación, se muestran las tareas correspondientes a las historias de usuario de mayor prioridad, las restantes se encuentran relacionadas en el apartado Anexo 2: Tareas de ingeniería.

Tabla 10. Tarea de ingeniería 2. Planificar los procesos existentes utilizando un algoritmo de planificación.

Tarea	
Número de tarea: 2	**Número de historia de usuario:** 2
Nombre de la tarea: Planificar los procesos existentes utilizando un algoritmo de planificación.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 2
Programador Responsable: Thais Mel Soteras Fong.	
Descripción: Con el desarrollo de la presente tarea, se realiza la funcionalidad que permite al usuario usar el algoritmo que desee (FCFS, SJF, SRTF, Round Robin) además podrá observar las transformaciones que ocurren en el planificador a partir de los datos proporcionados.	

Tabla 11. Tarea de ingeniería 10. Aplicar el algoritmo del banquero para la prevención del interbloqueo.

Tarea	
Número de tarea: 10	**Número de historia de usuario:** 10
Nombre de la tarea: Aplicar el algoritmo del banquero para la prevención del interbloqueo	
Tipo de tarea: Desarrollo	**Puntos estimados: 2**
Programador Responsable: Thais Mel Soteras Fong.	

Descripción: Con el desarrollo de la presente tarea, se realiza la funcionalidad que permite al sistema determinar una secuencia segura de ejecución de los procesos a partir de los datos insertados por el usuario aplicando el Algoritmo del Banquero.

Tabla 12. Tarea de ingeniería 11. Determinar la disponibilidad mínima necesaria de cada recurso para ejecutar una secuencia.

Tarea	
Número de tarea: 11	**Número de historia de usuario:** 11
Nombre de la tarea: Determinar la disponibilidad mínima necesaria de cada recurso para ejecutar una secuencia.	
Tipo de tarea: Desarrollo	**Puntos estimados: 2**
Programador Responsable: Thais Mel Soteras Fong.	
Descripción: Con el desarrollo de la presente tarea, se realiza la funcionalidad que permite al sistema determinar la disponibilidad mínima necesaria para atender las solicitudes de cada proceso teniendo en cuenta la secuencia segura insertada por el usuario.	

Tabla 13. Tarea de ingeniería 14. Simular algoritmo de reemplazo de páginas.

Tarea	
Número de tarea: 14	**Número de historia de usuario:** 14
Nombre de la tarea: Simular algoritmo de reemplazo de páginas.	
Tipo de tarea: Desarrollo	**Puntos estimados: 2**
Programador Responsable: Thais Mel Soteras Fong.	
Descripción: Con el desarrollo de la presente tarea, se realiza la funcionalidad que permite al sistema realizar la traducción de las direcciones virtuales, seleccionar el algoritmo que desea utilizar (FIFO, Óptimo, LRU, Reloj) e indicar su política (global o local), además podrá observar las transformaciones que ocurren en la memoria física y las estadísticas que se producen en cada simulación.	

Tabla 14. Tarea de ingeniería 20. Simular algoritmos de planificación de acceso al disco.

Tarea	
Número de tarea: 20	**Número de historia de usuario:** 20
Nombre de la tarea: Simular algoritmos de planificación de acceso al disco.	
Tipo de tarea: Desarrollo	**Puntos estimados: 1.5**
Programador Responsable: Thais Mel Soteras Fong.	
Descripción: Con el desarrollo de la presente tarea, se realiza la funcionalidad que permite al sistema realizar la simulación correspondiente a los algoritmos FIFO (primero en llegar, primero en salir), SSF (primero la búsqueda más cercana), SCAN (algoritmo del ascensor), CSCAN (variante circular del SCAN), LOOK y CLOOK (variante circular del LOOK), en dependencia de la selección de los usuarios y teniendo en cuenta la cola de solicitudes.	

3.1.2 Estándar de código

Un estándar de programación es una forma de "normalizar" la programación para que, al trabajar en un proyecto cualquiera, las personas involucradas en el mismo tengan un acceso rápido y una correcta comprensión del código. Un estándar define la escritura y organización del código fuente de un programa. Ventajas que brinda:

- Facilitan el mantenimiento de una aplicación. Dicho mantenimiento constituye el 80% del coste del ciclo de vida de la aplicación.
- Permite que cualquier programador entienda y pueda mantener la aplicación. En muy raras ocasiones una misma aplicación es mantenida por su autor original.
- Los estándares de programación mejoran la legibilidad del código, al mismo tiempo que permiten su compresión rápida.

Los estándares de codificación propuestos se enfocan en la legibilidad del código ya que esto repercute de forma directa en la comprensión del sistema por parte del programador. Además, el mantenimiento del sistema es más fácil y puede modificarse para añadirle nuevas características, modificar las ya existentes, depurar errores o mejorar el rendimiento.

Nombre de clases: Los nombres deben ser mezclas de mayúsculas y minúsculas, con la primera letra de cada palabra interna en mayúsculas.

Ejemplo:

class NombreClase{

Métodos: Los métodos deberán ser verbos (en infinitivo), en mayúscula y minúsculas con la primera letra del nombre en minúsculas, y con la primera letra de cada palabra interna en mayúsculas.

Ejemplo:

tipo de método(parámetros){

//Cuerpo del método

Variables: Los nombres de las variables tanto de instancia como estáticas reciben el mismo tratamiento que para los métodos, con la salvedad de que aquí sí importa más la relación entre la regla mnemónica y la longitud del nombre.

Ejemplo:

Correctos: simSistemas, resultados.

Declaración de las variables: Se utiliza una declaración de cada vez y no se permiten dejar variables locales sin inicializar salvo en el caso de que sean propiedades de un objeto bean.

Ejemplo:

public static Integer entero = new Integer(0);

La declaración de las variables locales de una clase, método o bloque de código se realizan al principio del mismo y no justo antes de necesitarse la utilización de la variable. La única excepción a esta regla son las variables que gestionan los bucles for.

Las variables de avance de bucles for no podrán ser modificadas de ninguna manera fuera de la propia sentencia del bucle. La duplicidad de los nombres de variables en diferentes niveles dentro de la misma clase se tiene que evitar.

Constantes: Los nombres de constantes de clases deberían escribirse todo en mayúsculas con las palabras separadas por subrayados ("__"). Todas serán declaradas como public static.

Ejemplo:

public static sistemas String PROPERTY_URL_SERVICIO = "urlServicio";

Sentencias: Normas básicas son:

- Una sentencia por línea de código.
- Todo bloque de sentencias entre llaves, aunque sea una sola sentencia después de un if.

La declaración de los bucles for :

for (int i = 0; i < condición; i++){

Son obligatorias las tres condiciones del bucle for:

- Inicialización,
- Condición de finalización,
- Actualización del valor de la variable de avance.

La variable de avance del bucle nunca podrá ser modificada dentro del propio bucle.

3.2 Pruebas

La prueba es un proceso de ejecución de un programa con la intención de descubrir un error. No puede asegurar la ausencia de defectos, sino demostrar la presencia de los mismos en el software, por lo que constituye el único instrumento adecuado para determinar el status de la calidad de un producto.

Un correcto proceso de pruebas permite aumentar la calidad de los sistemas disminuyendo el número de posibles errores futuros y reduciendo el tiempo de detección de los mismos. También permite mantener la estabilidad del software, aumentando la seguridad de evitar efectos colaterales no deseados a la hora de realizar modificaciones y refactorizaciones.

Según (19) y (38), las pruebas deben reunir un conjunto de características para que se obtengan buenos resultados:

- Crear la prueba abstrayéndose del futuro código, de esta forma se logra la independencia de esta respecto al código que evalúa.
- Observar la refactorización. La prueba permite verificar que un cambio en la estructura del código no tiene por qué cambiar su funcionamiento.
- Realizar pruebas a las funcionalidades generales que debe cumplir el programa especificado en las historias de usuario.

Las pruebas unitarias son creadas por los desarrolladores, son una técnica de caja blanca y consisten en comprobar la lógica interna del software (38). Estas se deben ejecutar luego de concluir cada tarea de implementación para asegurar que no se arrastren errores de codificación a la próxima etapa. Según la metodología de desarrollo escogida, se deben realizar también pruebas funcionales al sistema. En este proceso se ejecutan pruebas dirigidas a componentes del software o al sistema en su totalidad, con el objetivo de medir el grado en que este cumple con los requerimientos. En ellas se usan casos de prueba, especificados de forma estructurada mediante diferentes técnicas. Entre sus objetivos están:

- Verificar que todos los requisitos se han implementado correctamente.
- Detectar defectos en el software.
- Identificar y asegurar que los defectos encontrados se han corregido antes de entregar el software al cliente.
- Diseñar casos de prueba que sistemáticamente saquen a la luz diferentes clases de errores, haciéndolo en el menor tiempo y esfuerzo posible.

Para evaluar la calidad de la solución propuesta se realizarán pruebas unitarias y pruebas de aceptación.

3.2.1 Pruebas de Aceptación

Las pruebas de aceptación son pruebas funcionales, pero vistas directamente desde el cliente, demostrándole que la funcionalidad está terminada y correcta, es decir, se realizan con el fin de verificar si el producto ha sido desarrollado de acuerdo con las normas o criterios establecidos y que cumplen con todos los requisitos especificados por el cliente.

Las pruebas de aceptación son creadas sobre la base de las historias de usuarios en cada ciclo de la iteración del desarrollo. El cliente debe especificar uno o diversos escenarios para comprobar que una HU

ha sido correctamente implementada. Las pruebas de aceptación son consideradas como pruebas de caja negra. Los clientes son responsables de verificar que los resultados de estas pruebas sean correctos. Este tipo de pruebas representan la satisfacción del cliente, es por este motivo que deben ser diseñadas por el mismo. Para el diseño de las pruebas de aceptación se definieron los siguientes elementos:

- Código de la prueba: Representa el caso de prueba, incluye el número de la historia de usuario y de la prueba.
- Nombre de la historia de usuario: Incluye el nombre de la historia de usuario.
- Descripción de la prueba: Acción que debe realizar el sistema.
- Condiciones de ejecución: Características y elementos que debe contener el sistema para ejecutar el caso de prueba.
- Pasos de ejecución: Incluye los pasos necesarios para realizar la prueba.
- Resultados esperados: Respuesta que el sistema debe dar ante la ejecución de la prueba.

A continuación, se presentan los casos pruebas de aceptación diseñados para las historias de usuarios 2, 10, 11, 14 y 20. Los restantes casos de prueba se encuentran en el Anexo 3: Casos de prueba de aceptación.

Tabla 15. Caso de prueba de aceptación HU2-P2. Planificar los procesos existentes utilizando un algoritmo de planificación.

Caso de prueba de aceptación	
Código de la prueba: HU2-P2	**Nombre de la historia de usuario:** Planificar los procesos existentes utilizando un algoritmo de planificación.
Descripción de la prueba: La prueba consiste en verificar que se aplique el algoritmo seleccionado por el usuario de forma correcta a partir de los datos introducidos.	
Condiciones de ejecución: • La aplicación debe estar abierta en la computadora. • Las acciones contempladas en la prueba HU1-P1 deben ejecutarse satisfactoriamente.	

Pasos de ejecución: • Abrir la aplicación. • Seleccionar el algoritmo que se desea utilizar
Resultados esperados: El algoritmo de planificación seleccionado se ejecuta correctamente.

Tabla 16. Caso de prueba de aceptación HU10-P10. Aplicar el algoritmo del banquero para la prevención del interbloqueo.

Caso de prueba de aceptación	
Código de la prueba: HU10-P10	**Nombre de la historia de usuario:** Aplicar el algoritmo del banquero para la prevención del interbloqueo.
Descripción de la prueba: La prueba consiste en verificar si se determina una secuencia segura de ejecución de los procesos a partir de los datos insertados por el usuario aplicando el Algoritmo del Banquero.	
Condiciones de ejecución: • La aplicación debe estar abierta en la computadora. • Las acciones contempladas en las pruebas HU8-P8 y HU9-P9 deben ejecutarse satisfactoriamente.	
Pasos de ejecución: • Abrir la aplicación. • Introducir la disponibilidad • Ejecutar el Algoritmo del Banquero	
Resultados esperados: La aplicación muestra una secuencia segura para la ejecución de los procesos.	

Tabla 17. Caso de prueba de aceptación HU11-P11. Determinar la disponibilidad mínima necesaria de cada recurso para ejecutar una secuencia segura.

Caso de prueba de aceptación	
Código de la prueba: HU11-P11	**Nombre de la historia de usuario:** Determinar la disponibilidad mínima necesaria de cada recurso para ejecutar una segura.
Descripción de la prueba: La prueba consiste en verificar si se puede determinar una disponibilidad mínima para que los procesos se ejecuten según la secuencia introducida por el usuario.	
Condiciones de ejecución: • La aplicación debe estar abierta en la computadora. • Las acciones contempladas en las pruebas HU8-P8 y HU9-P9 deben ejecutarse satisfactoriamente.	
Pasos de ejecución: • Abrir la aplicación. • Introducir una secuencia de ejecución segura. • Calcular la disponibilidad.	
Resultados esperados: La aplicación muestra la disponibilidad mínima necesaria para atender las solicitudes de cada proceso teniendo en cuenta la secuencia segura insertada por el usuario.	

Tabla 18. Caso de prueba de aceptación HU14-P14. Simulación de algoritmos de reemplazo de páginas.

Caso de prueba de aceptación	
Código de la prueba: HU14-P14	**Nombre de la historia de usuario:** Simulación de algoritmos de reemplazo de páginas.
Descripción de la prueba: La prueba consiste en verificar que se realice la traducción de las direcciones virtuales, seleccionar el algoritmo que desea utilizar (FIFO, Óptimo, LRU, Reloj) e indicar su política	

(global o local), además podrá observar las transformaciones que ocurren en la memoria física y las estadísticas que se producen en cada simulación.
Condiciones de ejecución: • La aplicación debe estar abierta en la computadora. • Las acciones contempladas en la prueba HU12-P12 deben ejecutarse satisfactoriamente.
Pasos de ejecución: • Abrir la aplicación. • Debe seleccionar el algoritmo que va a usar, indicar su política. • Ejecutar la simulación
Resultados esperados: La aplicación muestra la traducción de las direcciones virtuales y simular el funcionamiento del algoritmo seleccionado.

Tabla 19. Caso de prueba de aceptación HU14-P14. Simulación de algoritmos de planificación de acceso al disco.

Caso de prueba de aceptación	
Código de la prueba: HU20-P20	**Nombre de la historia de usuario:** Simulación de algoritmos de planificación de acceso al disco.
Descripción de la prueba: La prueba consiste en ejecutar la simulación en dependencia de la selección del algoritmo realizada por el usuario y teniendo en cuenta la cola de solicitudes.	
Condiciones de ejecución: • La aplicación debe estar abierta en la computadora. • Las acciones contempladas en la prueba HU19-P19 deben ejecutarse satisfactoriamente.	
Pasos de ejecución: • Abrir la aplicación.	

• Seleccionar el algoritmo de atención a la cola de solicitudes que se simulará. • Ejecutar la simulación.
Resultados esperados: La aplicación muestra la simulación del algoritmo de atención a peticiones del disco seleccionado por el usuario.

Conclusiones del capítulo

Debido a la situación de aislamiento impuesta por la pandemia del Covid-19 y a que la autora de la presente investigación no contaba con los medios tecnológicos necesarios para desarrollar la solución propuesta al problema planteado, no fue posible implementar todas las funcionalidades del sistema. Sin embargo, se especificaron las tareas de ingeniería que permiten guiar la implementación del sistema en su totalidad, se definió el estándar de codificación a seguir para obtener claridad y organización en el código fuente de la solución, se indicó cómo debe realizarse el proceso de pruebas una vez que el sistema esté desarrollado en su totalidad y se describieron los casos de prueba de aceptación que se deben emplear para validar la solución propuesta.

Conclusiones

La investigación realizada permitió constatar que, aunque están identificados los principales problemas de asimilación de contenidos de la asignatura Sistemas Operativos por parte de los estudiantes, y que en su mayoría estos pueden ser minimizados o erradicados con la utilización de una herramienta de simulación, la utilización de las mismas no es una práctica común dentro de las universidades extranjeras y nacionales a pesar de existir una amplia variedad de ellas. Se pudo identificar como la causa principal de este fenómeno, el grado de adaptación de las mismas al plan de estudios de la asignatura.

Este análisis permitió determinar que la problemática planteada al inicio de la investigación podía resolverse con la creación de una nueva herramienta que abarcara los principales contenidos de la asignatura y se adaptara a los objetivos de la misma. Se realizó un estudio de las herramientas de simulación abordadas en la investigación, unido a las características del área donde impactará la solución y el público al que va dirigida lo que aportó la base tecnológica para el desarrollo de la solución propuesta. Este análisis permitió identificar las características de dichas herramientas que resultaran de utilidad en el desarrollo de la solución.

Se identificaron las principales funcionalidades y cualidades con las que debía cumplir el sistema, se describieron los artefactos de ingeniería definidos por la metodología de software escogida y a partir de un análisis de las características del mismo se determinó la arquitectura y los patrones que se debían seguir para su desarrollo. Aunque no fue posible implementar el sistema en su totalidad, se describieron todos los artefactos involucrados en el proceso de desarrollo, los estándares de codificación y las pautas a seguir para la realización de las pruebas de software una vez que se complete la implementación de la herramienta.

Recomendaciones

Durante el desarrollo de la investigación surgieron un conjunto de ideas que tienen como objetivo garantizar el el futuro perfeccionamiento del sistema:

1. Desarrollar compilador que permita simular el funcionamiento de los mecanismos de sincronización y comunicación de procesos del Sistema Operativo e integrarlo a la solución propuesta en la presente investigación.
2. Incorporar al sistema funcionalidades que permitan simular los aspectos relacionados con el sistema de archivo para que la herramienta pueda ser utilizada por otras instituciones de educación superior en correspondencia con los contenidos abordados en sus programas de estudios.

Bibliografía

1. **Universidad de las Ciencias Informáticas.** Universidad de las Ciencias Informáticas. [En línea] [Citado el: 29 de junio de 2021.] https://www.uci.cu/universidad/mision.

2. **Informáticas, Universidad de las Ciencias.** *Plan de estudios ¨E¨ de la carrera Ingeniería en Ciencias Informáticas.* La Habana : s.n., 2019.

3. **Estero, Juan José Dominguéz y Antonia.** *Herramienta de Apoyo a la docencia de Sistemas Operativos.* Cádiz, Universidad : s.n.

4. **Jaramillo, Jorge.** *Tecnología Superior en Redes y Telecomunicaciones.* México : s.n., 2020.

5. **Jesús Carretero, Alejandro Calderón, Jóse Daniel García.** *Sistemas Operativos: propuesta de contenidos y técnicas.* Avda. Universidad 30, 28911, Leganés, Madrid : s.n.

6. **Thomas Enge, Phillip J. Reid.** *Introducción a la Fisicoquimica:Termodinamica.* s.l. : Pearson, 2007.

7. **Goldstein, Leisten, Stark, Tickle.** *Using a Network Simulation Tool to Engage Students in Active Learning Enhances their Undertanding of Complex Data Communications Concepts.* 2005.

8. **María Alejandra Garcia, Elí Samuel Gonzalez, Gloria Pedrosa.** *El uso de simuladores como herramienta de apoyo para la enseñanza.* Nuevo León, México : s.n., Abril 2018.

9. *La Simulación como Herramienta de Aprendizaje de Física.* **Luisa Casadei Carniel, Marisol Cuicas Avila, Edie Debel Chourio, Zulma Alvarez Vargas.** Costa Rica : s.n., 2008.

10. **Guerrero M, Guamán D, Caiza J.** *Revisión de Herramientas de Apoyo en el Proceso de Enseñanza-Aprendizaje de Programación.* Quito, Ecuador : s.n.

11. **Graciela López-Chávez, Sura Chávez Hernández.** *Simulación educativa: Herramienta educación Ciencia Tecnología y Soc idedidaádc teinca l ap ara disciplina Filosofía y Sociedad.* Camagüey : s.n., 2013.

12. *Herramientas de apoyo a la docencia de Sistemas Operativos.* **Domínguez Jiménez, Juan José y Estero Botaro, Antonia.** 06, Cádiz. España : s.n., 2000, Departamento de Lenguajes y Sistemas Informáticos. Universidad de Cádiz, Vol. 27, pág. 05.

13. **Botaro, Juan Jose Domínguez Jiménez y Antonia Estero.** *Herramientas de apoyo a la docencia de Sistemas Operativos.* Cádiz,España : s.n., 2000. 06.

14. *An Interactive, Graphical CPU Scheduling Simulator for Teaching Operating Systems.* **Buck, Joshua y Perugini, Saverio.** Ohio : s.n., 2019. Proceedings of the 50th CM Technical Symposium on Computer Science Education. pág. 1290.

15. **Ben-Ari, Mordechai.** Weizmann Insitute of Science. Departament of Science Teaching. *jBACI Concurrency Simulator.* [En línea] https://www.weizmann.ac.il/sci-tea/benari/software-and-learning-materials/jbaci-concurrency-simulator.

16. **Informer Technology, Inc.** mac.informer. [En línea] 30 de Noviembre de 2017. https://macdownload.informer.com/jbaci.

17. **David Luis La Red, Nelson Fabian Rodriguez.** *Sistema Abierto para La Enseñanza de Sistemas Operativos mediante un Simulador.* Argentina : s.n.

18. **Operativos, Uso de simuladores y herramientas Web para la enseñanza de Sistemas.** *Felix Buendia, Juan-Carlos Cano, Julio Sahuquillo.* Valencia : s.n., 2005.

19. *OSSIM - An Operating System Simulator.* **Reese, Richard M.** [ed.] Ruth Robbins. 1, Texas : Association for Computer Educators in Texas and EBSCO Publishing, INC., 2008, ACET Journal of Computer Education and Research, Vol. 5. 1547-3716.

20. *Una herramienta de Simulación para la Planificación de Procesos.* **Barrionuevo, Mercedes, Piccoli, Fabiana y Apolloni, Ruben.** 9, La Plata : Red de Universidades Nacionales con Carrera en Informática ± Universidad Nacional de La Plata (RedUNCI - UNLP), Abril de 2013, Revista Iberoamericana de Educación en Tecnología y Tecnología en Educación Especial, págs. 7-15. 1850-9959.

21. *VMFS: herramienta visual para la enseñanza del funcionamiento de un sistema de ficheros.* **García Fornés, Ana, Espinosa, Agustín y Valero, Soledad.** Bilbao : s.n., 2006. XII Jornadas de Enseñanza Universitaria de Informática (JENUI). págs. 561-566.

22. **Tanenbaum, Andrew Stuart.** *Sistemas Operativos Modernos.* Tercera Edición. Amsterdam : PRENTICE HALL, INC, 2009. 978-607-442-046-3.

23. *Go To Statement Considered Harmful.* **Dijkstra, Edsger Wybe.** 3, Marzo de 1968, Communications of the ACM, Vol. 11, págs. 147-148.

24. *Software de simulación del algoritmo del banquero.* **Faez Cobo, Ivan.** 3, 2013, Ciencia & Futuro, Vol. 3.

25. *SMRP: Una aplicación informática para simular el reemplazo de páginas de memoria en los Sistemas Operativos.* **Velázquez Rodríguez, Carlos Ernesto.** 2, 2013, Ciencia & Futuro, Vol. 3. 2306-823X.

26. **Vega, Eulicer Ernesto Martínez.** *Herramienta didáctica para la enseñanza de los algoritmos de gestión de los Sistemas Operativos.* La Habana : s.n., Abril, 2016.

27. **Pressman, Roger.** *Ingeniería de software. Un enfoque práctico.* Séptima Edición. s.l. : McGraw Hill, 2010. 978-607-15-0314-5.

28. **Villán, Vanessa Roselló.** IEBS. *Las metodologias agiles mas usadas y sus ventajas dentro de la empresa.* [En línea] 15 de Marzo de 2014. https://www.iebschool.com/blog/que-son-metologias-agiles-agile-scrum/.

29. **Penadés, Patricio Letelier y María del Carmen.** Cyta. *Metodologías ágiles para el desarrollo de software: eXtreme Programming (XP).* [En línea] 15 de Enero de 2006. http://www.cyta.com.ar/ta0502/v5n2a1.htm.

30. **Rodríguez Sánchez, Tamara.** *Metodología de desarrollo para la Actividad productiva de la UCI.* La Habana : s.n., 2015.

31. **Visual Paradigm.** Visual Paradigm 16.0 Released. *Visual Paradigm.* [En línea] Julio de 2019. https://www.visual-paradigm.com/features.

32. **Menéndez-Barzanallana Asensio, Rafael.** *Principales herramientas CASE del mercado y su uso.* [En línea] 29 de febrero de 2020. https://www.um.es/docencia/barzana/IAGP/Enlaces/CASE_principales.html.

33. **GeneratePress.** Curso de PHP desde Cero. *Aquí en sistemas.* [En línea] 2021. https://aquiensistemas.com/curso-de-php-desde-cero-gratis-2021/.

34. **School, Tokio New Technology.** ¿Por qué escoger el lenguaje Phyton? *Tokio.* [En línea] 27 de Enero de 2020.

35. **Perry, J Steven.** Conceptos básicos del lenguaje Java. *IBM Developer.* [En línea] 3 de Diciembre de 2012. https://developer.ibm.com/es/languages/java/tutorials/j-introtojava1/.

36. **Conoce el Lenguaje de Programación Java. *Blog SEAS.* [En línea] 17 de Julio de 2019. http://www.seas.es/blog/informatica/conoce-el-lenguaje-de-programación-java/.**

37. The Apache Software Foundation. Apache NetBeans. [En línea] Oracle Corporation, 2021. https://netbeans.apache.org/download/nb125/nb125.html.

38. Sommerville, Ian. *Ingeniería de Software.* Novena. México : Pearson Education, 2011.

39. *Requerimientos de software: prototipado, software heredado y análisis de documentos.* Medina Cruz, Javier, Pineda Ballesteros, Eliécer y Téllez Acuña, Freddy Reynaldo. 2, Barranquilla : s.n., 2019, Ingeniería y Desarrollo, Vol. 37. ISSN 2145-9371.

40. *Indización de grafos RDF desde un SPARQL Endpoint.* Mariño Moleiro, Alejandro Jesús, Hidalgo Delgado, Yusniel y Ortiz Muñoz, Ernesto. 9, La Habana : s.n., 2016, Serie Científica de la Universidad de las Ciencias Informáticas, págs. 1-12.

41. *Evaluación de una Arquitectura de Software.* Riveros Sanabria, Fernando y Valbuena Rodríguez, Santiago. 2, Barranquilla : s.n., 2021, Prospectiva. Una nueva visión para la ingeniería, Vol. 19.

Anexos

Anexo 1: Historias de Usuario

Tabla 20. Historia de Usuario 1. Administrar datos de procesos.

Historia de Usuario	
Número: 1	**Nombre:** Administrar datos de procesos
Usuario: Programador	**RF 1:** Administrar datos de procesos
Prioridad en Negocio: Baja	**Riesgo en Desarrollo:** Bajo
Puntos Estimados: 0.5	**Iteración Asignada:** 1
Programador Responsable: Thais Mel Soteras Fong	
Descripción: Se introducen los datos correspondientes de forma manual o los genera la aplicación a partir de la tabla de procesos, así como los datos imprescindibles para realizar la simulación como los instantes de tiempo en que se generan los procesos y la ocurrencia de bloqueos.	
Observaciones:	

Tabla 21. Historia de Usuario 3. Visualizar la simulación instante a instante o de forma automática a través de un diagrama de Gantt.

Historia de Usuario	
Número: 3	**Nombre:** Visualizar la simulación instante a instante o de forma automática a través de un diagrama de Gantt.
Usuario: Programador	RF 3: Visualizar a través de una simulación la planificación de procesos
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Alto
Puntos Estimados: 1.5	**Iteración Asignada:** 1
Programador Responsable: Thais Mel Soteras Fong	
Descripción:	

El usuario puede visualizar la simulación completa a través de un diagrama de Gantt o generarla instante a instante. También podrá seleccionar rangos de tiempo en los que desea visualizar la simulación.
Observaciones:

Tabla 22. Historia de Usuario 4. Mostrar los datos de cada proceso, en cuanto a instante de llegada y de fin, tiempo que estuvo en estado listo, tiempo que estuvo en estado bloqueado, y tiempo que estuvo en ejecución.

Historia de Usuario	
Número: 4	**Nombre:** Mostrar los datos de cada proceso, en cuanto a instante de llegada y de fin, tiempo que estuvo en estado listo, tiempo que estuvo en estado bloqueado, y tiempo que estuvo en ejecución.
Usuario: Programador	**RF 4:** Emitir estadísticas de la simulación de procesos
Prioridad en Negocio: Media	**Riesgo en Desarrollo:** Medio
Puntos Estimados: 1	**Iteración Asignada:** 1
Programador Responsable: Thais Mel Soteras Fong	
Descripción: Se mostrarán los datos relacionados con los parámetros de los algoritmos de planificación (instante de llegada y de fin, tiempo que estuvo en estado listo, tiempo que estuvo en estado bloqueado, y tiempo que estuvo en ejecución), a través de esquemas o tablas.	
Observaciones:	

Tabla 23. Historia de Usuario 5. Calcular y mostrar estadísticas relacionadas con: tiempo de espera medio, tiempo de retorno promedio y por ciento de uso del CPU.

Historia de Usuario

Número: 5	**Nombre:** Calcular y mostrar estadísticas relacionadas con: tiempo de espera medio, tiempo de retorno promedio y por ciento de uso del CPU.
Usuario: Programador	**RF 4:** Emitir estadísticas de la simulación de procesos
Prioridad en Negocio: Baja	**Riesgo en Desarrollo:** Bajo
Puntos Estimados: 0.5	**Iteración Asignada:** 1
Programador Responsable: Thais Mel Soteras Fong	
Descripción: Se podrá calcular el tiempo de espera medio, tiempo de retorno promedio y por ciento de uso del CPU, a partir de los datos introducidos o de los resultados obtenidos luego de la simulación.	
Observaciones:	

Tabla 24. Historia de Usuario 6. Mostrar las gráficas que representan el tiempo de espera medio y tiempo de retorno medio de cada proceso.

Historia de Usuario	
Número: 6	**Nombre:** Mostrar las gráficas que representan el tiempo de espera medio y tiempo de retorno medio de cada proceso.
Usuario: Programador	**RF 4:** Emitir estadísticas de la simulación de procesos
Prioridad en Negocio: Media	**Riesgo en Desarrollo:** Medio
Puntos Estimados: 1	**Iteración Asignada:** 1
Programador Responsable: Thais Mel Soteras Fong	
Descripción: Se muestran las gráficas que representan el tiempo de espera medio y tiempo de retorno medio de cada proceso y del sistema.	
Observaciones:	

Tabla 25. Historia de Usuario 7. Realizar comparaciones entre algoritmos de planificación.

Historia de Usuario	
Número: 7	**Nombre:** Realizar comparaciones entre algoritmos de planificación.
Usuario: Programador	**RF 4:** Emitir estadísticas de la simulación de procesos
Prioridad en Negocio: Media	**Riesgo en Desarrollo:** Medio
Puntos Estimados: 1	**Iteración Asignada:** 1
Programador Responsable: Thais Mel Soteras Fong	
Descripción: Se podrán realizar comparaciones entre los resultados obtenidos luego de la simulación con diferentes algoritmos para el mismo juego de datos.	
Observaciones:	

Tabla 26. Historia de Usuario 8. Administrar datos de recursos.

Historia de Usuario	
Número: 8	**Nombre:** Administrar datos de recursos.
Usuario: Programador	**RF 5:** Administrar recursos
Prioridad en Negocio: Media	**Riesgo en Desarrollo:** Medio
Puntos Estimados: 1	**Iteración Asignada:** 2
Programador Responsable: Thais Mel Soteras Fong	
Descripción: Se introducen los datos correspondientes a la cantidad de procesos, cantidad de recursos, los recursos asignados y solicitados por cada proceso, la existencia y la disponibilidad del sistema, así como una secuencia segura de ejecución de los procesos.	
Observaciones: Se debe dar la posibilidad de no introducir todos los datos puesto que el usuario podrá calcular algunos de ellos de forma automática.	

Tabla 27.Historia de Usuario 9. Elaborar las matrices de asignación y solicitud a partir de los datos de los recursos.

Historia de Usuario	
Número: 9	**Nombre:** Elaborar las matrices de asignación y solicitud a partir de los datos de los recursos.
Usuario: Programador	**RF 6:** Visualizar los recursos asignados y solicitados por cada proceso
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Alta
Puntos Estimados: 1.5	**Iteración Asignada:** 2
Programador Responsable: Thais Mel Soteras Fong	
Descripción: Deben mostrarse las matrices de Asignación y Solicitud a partir de los datos introducidos por el usuario.	
Observaciones: Esta funcionalidad solo se ejecutará en caso de no haberse insertado los datos correspondientes a las solicitudes de recursos de los procesos.	

Tabla 28. Historia de Usuario 12. Administrar datos de la memoria y los procesos.

Historia de Usuario	
Número: 12	**Nombre:** Administrar datos de la memoria y los procesos.
Usuario: Programador	**RF 9:** Gestionar datos de la memoria
Prioridad en Negocio: Media	**Riesgo en Desarrollo:** Medio
Puntos Estimados: 1	**Iteración Asignada:**
Programador Responsable: Thais Mel Soteras Fong	
Descripción: Se introducen los datos correspondientes de forma manual o los genera la aplicación a partir de la tabla de páginas, así como los datos imprescindibles para realizar la	

simulación como tamaño de página, tamaño de la memoria y el algoritmo de reemplazo deseado.
Observaciones:

Tabla 29. Historia de Usuario 13. Gestionar tabla de páginas de los procesos.

Historia de Usuario	
Número: 13	**Nombre:** Gestionar tabla de páginas de los procesos.
Usuario: Programador	**RF 9:** Gestionar datos de la memoria.
Prioridad en Negocio: Baja	**Riesgo en Desarrollo:** Bajo
Puntos Estimados: 0.5	**Iteración Asignada:** 3
Programador Responsable: Thais Mel Soteras Fong	
Descripción: Permite introducir los datos asociados a la tabla de páginas de cada proceso.	
Observaciones:	

Tabla 30. Historia de Usuario 15. Visualizar la simulación instante a instante o de forma automática a través de la tabla de memoria.

Historia de Usuario	
Número: 15	**Nombre:** Visualizar la simulación instante a instante o de forma automática a través de la tabla de memoria.
Usuario: Programador	**RF 11:** Visualizar la simulación de forma gráfica
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Alto
Puntos Estimados: 1.5	**Iteración Asignada:** 3
Programador Responsable: Thais Mel Soteras Fong	
Descripción:	

El usuario puede visualizar la simulación completa en la tabla de páginas o generarla instante a instante. También podrá seleccionar rangos de tiempo en los que desea visualizar la simulación.
Observaciones:

Tabla 31. Historia de Usuario 16. Análisis de resultados.

Historia de Usuario	
Número: 16	**Nombre:** Análisis de resultados.
Usuario: Programador	**RF 12:** Realizar análisis, comparativas y cálculos a partir de los resultados obtenidos en la simulación del algoritmo.
Prioridad en Negocio: Media	**Riesgo en Desarrollo:** Medio
Puntos Estimados: 1	**Iteración Asignada:** 3
Programador Responsable: Thais Mel Soteras Fong	
Descripción: Siempre que el sistema realice una simulación, el usuario obtiene los resultados de esta, así como los indicadores que determinan la eficiencia de cada método que se use.	
Observaciones: El usuario puede establecer comparaciones entre distintas simulaciones utilizando como parámetros los indicadores obtenidos.	

Tabla 32. Historia de Usuario 17. Calcular y mostrar estadísticas relacionadas con: dirección física, dirección lógica, cantidad de páginas, cantidad de marcos, valor de la fragmentación interna.

Historia de Usuario	
Número: 17	**Nombre:** Calcular y mostrar estadísticas relacionadas con: dirección física, dirección lógica, cantidad de páginas, cantidad de marcos, valor de la fragmentación interna.

Usuario: Programador	**RF 12:** Realizar análisis, comparativas y cálculos a partir de los resultados obtenidos en la simulación del algoritmo.
Prioridad en Negocio: Media	**Riesgo en Desarrollo:** Medio
Puntos Estimados: 1	**Iteración Asignada:** 3
Programador Responsable: Thais Mel Soteras Fong	
Descripción: El usuario podrá calcular la dirección física, dirección lógica, cantidad de páginas, cantidad de marcos y valor de la fragmentación interna a partir de los datos introducidos o de los resultados obtenidos luego de la simulación.	
Observaciones:	

Tabla 33. Historia de Usuario 18. Realizar comparaciones entre algoritmos de reemplazo de páginas.

Historia de Usuario	
Número: 18	**Nombre:** Realizar comparaciones entre algoritmos de reemplazo de páginas.
Usuario: Programador	**RF 12:** Realizar análisis, comparativas y cálculos a partir de los resultados obtenidos en la simulación del algoritmo.
Prioridad en Negocio: Media	**Riesgo en Desarrollo:** Medio
Puntos Estimados: 1	**Iteración Asignada:** 3
Programador Responsable: Thais Mel Soteras Fong	
Descripción: Se podrán realizar comparaciones entre los resultados obtenidos luego de la simulación con diferentes algoritmos para el mismo juego de datos.	
Observaciones:	

Tabla 34. Historia de Usuario 19. Administrar cola de peticiones.

Historia de Usuario	
Número: 19	**Nombre:** Administrar cola de peticiones
Usuario: Programador	**RF 13:** Administrar cola de peticiones
Prioridad en Negocio: Baja	**Riesgo en Desarrollo:** Bajo
Puntos Estimados: 0.5	**Iteración Asignada:** 4
Programador Responsable: Thais Mel Soteras Fong	
Descripción: Se introducen las peticiones de atención del disco y los instantes de tiempo en que se generan.	
Observaciones:	

Tabla 35. Historia de Usuario 21. Visualizar la simulación instante a instante o de forma automática a través de un gráfico.

Historia de Usuario	
Número: 21	**Nombre:** Visualizar la simulación instante a instante o de forma automática a través de un gráfico.
Usuario: Programador	**RF 15:** Visualizar a través de una simulación la planificación de acceso al disco
Prioridad en Negocio: Media	**Riesgo en Desarrollo:** Medio
Puntos Estimados: 1.5	**Iteración Asignada:**
Programador Responsable: Thais Mel Soteras Fong	
Descripción: Se muestra el resultado de la simulación del algoritmo para la cola de solicitudes. Puede ser la simulación completa o paso a paso.	
Observaciones: El usuario elige como visualizar, si a través de una tabla o un gráfico.	

Tabla 36. Historia de Usuario 22. Calcular y mostrar estadísticas relacionadas con: instantes de tiempo en que se realizan las peticiones, tiempo de búsqueda, latencia y transferencia.

Historia de Usuario	
Número: 22	**Nombre:** Calcular y mostrar estadísticas relacionadas con: instantes de tiempo en que se realizan las peticiones, tiempo de búsqueda, latencia y transferencia.
Usuario: Programador	**RF 16:** Realizar análisis, comparativas y cálculos a partir de los resultados obtenidos en la simulación del algoritmo teniendo en cuenta los instantes de tiempo en que se realizan las peticiones, tiempo de búsqueda, latencia y transferencia
Prioridad en Negocio: Media	**Riesgo en Desarrollo:** Medio
Puntos Estimados: 1	**Iteración Asignada:** 4
Programador Responsable: Thais Mel Soteras Fong	
Descripción: El usuario podrá calcular el tiempo de búsqueda, latencia y transferencia, así como el tiempo de ejecución total del algoritmo a partir de los datos introducidos y de los resultados obtenidos luego de la simulación.	
Observaciones:	

Tabla 37. Historia de Usuario 23. Realizar comparaciones entre algoritmos de planificación de acceso a disco.

Historia de Usuario	
Número: 23	**Nombre:** Realizar comparaciones entre algoritmos de planificación de acceso a disco.
Usuario: Programador	**RF 16:** Realizar análisis, comparativas y cálculos a partir de los resultados obtenidos en la simulación del algoritmo teniendo en cuenta los instantes de

	tiempo en que se realizan las peticiones, tiempo de búsqueda, latencia y transferencia
Prioridad en Negocio: Baja	**Riesgo en Desarrollo:** Bajo
Puntos Estimados: 0.5	**Iteración Asignada:** 4
Programador Responsable: Thais Mel Soteras Fong	
Descripción: Se podrán realizar comparaciones entre los resultados obtenidos luego de la simulación con diferentes algoritmos para el mismo juego de datos.	
Observaciones:	

Anexo 2: Tareas de ingeniería

Tabla 38. Tarea de ingeniería 1. Administrar los datos de procesos.

Tarea	
Número de tarea: 1	**Número de historia de usuario:** 1
Nombre de la tarea: Administrar los datos de procesos.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.5
Programador Responsable: Thais Mel Soteras Fong.	
Descripción: Con el desarrollo de la presente tarea, se realiza la funcionalidad que permite al usuario introducir los datos correspondientes de forma manual al sistema o los genera la aplicación a partir de la tabla de procesos, así como los datos imprescindibles para realizar la simulación como los instantes de tiempo en que se generan los procesos y la ocurrencia de bloqueos.	

Tabla 39. Tarea de ingeniería 3. Visualizar la simulación del diagrama de Gantt.

Tarea	
Número de tarea: 3	**Número de historia de usuario:** 3
Nombre de la tarea: Visualizar la simulación del diagrama de Gantt.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1.5
Programador Responsable: Thais Mel Soteras Fong.	
Descripción: Con el desarrollo de la presente tarea, se realiza la funcionalidad que permite al usuario visualizar la simulación completa a través de un diagrama de Gantt o generarla instante a instante. También podrá seleccionar rangos de tiempo en los que desea visualizar la simulación.	

Tabla 40. Tarea de ingeniería 4. Mostrar los datos de cada proceso.

Tarea	
Número de tarea: 4	**Número de historia de usuario:** 4
Nombre de la tarea: Mostrar los datos de cada proceso.	

Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Programador Responsable: Thais Mel Soteras Fong.	
Descripción: Con el desarrollo de la presente tarea, se realiza la funcionalidad que permite al sistema mostrar los datos relacionados con los parámetros de los algoritmos de planificación (instante de llegada y de fin, tiempo que estuvo en estado listo, tiempo que estuvo en estado bloqueado, y tiempo que estuvo en ejecución), a través de esquemas o tablas.	

Tabla 41. Tarea de ingeniería 5. Actualizar las estadísticas relacionadas con: tiempo de espera medio, tiempo de retorno promedio y por ciento de uso del CPU.

Tarea	
Número de tarea: 5	**Número de historia de usuario:** 5
Nombre de la tarea: Actualizar las estadísticas relacionadas con: tiempo de espera medio, tiempo de retorno promedio y por ciento de uso del CPU.	
Tipo de tarea: Desarrollo	**Puntos estimados: 0.5**
Programador Responsable: Thais Mel Soteras Fong.	
Descripción: Con el desarrollo de la presente tarea, se realiza la funcionalidad que permite al sistema calcular el tiempo de espera medio, tiempo de retorno promedio y por ciento de uso del CPU, a partir de los datos introducidos o de los resultados obtenidos luego de la simulación.	

Tabla 42. Tarea de ingeniería 6. Mostrar las gráficas de la simulación cada proceso.

Tarea	
Número de tarea: 6	**Número de historia de usuario:** 6
Nombre de la tarea: Mostrar las gráficas de la simulación cada proceso.	
Tipo de tarea: Desarrollo	**Puntos estimados: 1**
Programador Responsable: Thais Mel Soteras Fong.	

Descripción: Con el desarrollo de la presente tarea, se realiza la funcionalidad que permite al sistema mostrar las gráficas que representan el tiempo de espera medio y tiempo de retorno medio de cada proceso y del sistema.

Tabla 43. Tarea de ingeniería 7. Comparar los algoritmos de planificación.

Tarea	
Número de tarea: 7	**Número de historia de usuario:** 7
Nombre de la tarea: Comparar los algoritmos de planificación.	
Tipo de tarea: Desarrollo	**Puntos estimados: 1**
Programador Responsable: Thais Mel Soteras Fong.	
Descripción: Con el desarrollo de la presente tarea, se realiza la funcionalidad que permite al sistema realizar comparaciones entre los resultados obtenidos luego de la simulación con diferentes algoritmos para el mismo juego de datos.	

Tabla 44. Tarea de ingeniería 8. Administrar datos de recursos.

Tarea	
Número de tarea: 8	**Número de historia de usuario:** 8
Nombre de la tarea: Administrar datos de recursos.	
Tipo de tarea: Desarrollo	**Puntos estimados: 1**
Programador Responsable: Thais Mel Soteras Fong.	
Descripción: Con el desarrollo de la presente tarea, se realiza la funcionalidad que permite al sistema introducir los datos correspondientes a la cantidad de procesos, cantidad de recursos, los recursos asignados y solicitados por cada proceso, la existencia y la disponibilidad del sistema, así como una secuencia segura de ejecución de los procesos.	

Tabla 45. Tarea de ingeniería 9. Elaborar las matrices de asignación y solicitud a partir de los datos de los recursos.

Tarea	
Número de tarea: 9	**Número de historia de usuario:** 9
Nombre de la tarea: Elaborar las matrices de asignación y solicitud a partir de los datos de los recursos.	
Tipo de tarea: Desarrollo	**Puntos estimados: 1.5**
Programador Responsable: Thais Mel Soteras Fong.	
Descripción: Con el desarrollo de la presente tarea, se realiza la funcionalidad que permite al sistema mostrar las matrices de Asignación y Solicitud a partir de los datos introducidos por el usuario.	

Tabla 46. Tarea de ingeniería 12. Administrar datos de la memoria y los procesos.

Tarea	
Número de tarea: 12	**Número de historia de usuario:** 12
Nombre de la tarea: Administrar datos de la memoria y los procesos.	
Tipo de tarea: Desarrollo	**Puntos estimados: 1**
Programador Responsable: Thais Mel Soteras Fong.	
Descripción: Con el desarrollo de la presente tarea, se realiza la funcionalidad que permite al sistema introducir los datos correspondientes de forma manual o los genera la aplicación a partir de la tabla de páginas, así como los datos imprescindibles para realizar la simulación como tamaño de página, tamaño de la memoria y el algoritmo de reemplazo deseado.	

Tabla 47. Tarea de ingeniería 13. Gestionar tabla de páginas de los procesos.

Tarea	
Número de tarea: 13	**Número de historia de usuario:** 13
Nombre de la tarea: Gestionar tabla de páginas de los procesos.	
Tipo de tarea: Desarrollo	**Puntos estimados: 0.5**
Programador Responsable: Thais Mel Soteras Fong.	

Descripción: Con el desarrollo de la presente tarea, se realiza la funcionalidad que permite al sistema introducir los datos asociados a la tabla de páginas de cada proceso.

Tabla 48. Tarea de ingeniería 15. Visualizar la simulación de la tabla de memoria.

Tarea	
Número de tarea: 15	**Número de historia de usuario:** 15
Nombre de la tarea: Visualizar la simulación de la tabla de memoria.	
Tipo de tarea: Desarrollo	**Puntos estimados: 1.5**
Programador Responsable: Thais Mel Soteras Fong.	
Descripción: Con el desarrollo de la presente tarea, se realiza la funcionalidad que permite al usuario visualizar la simulación completa en la tabla de páginas o generarla instante a instante. También podrá seleccionar rangos de tiempo en los que desea visualizar la simulación.	

Tabla 49. Tarea de ingeniería 16. Analizar los resultados obtenidos en la simulación del algoritmo.

Tarea	
Número de tarea: 16	**Número de historia de usuario:** 16
Nombre de la tarea: Analizar los resultados obtenidos en la simulación del algoritmo.	
Tipo de tarea: Desarrollo	**Puntos estimados: 1**
Programador Responsable: Thais Mel Soteras Fong.	
Descripción: Con el desarrollo de la presente tarea, se realiza la funcionalidad que permite al sistema realizar una simulación, donde el usuario obtiene los resultados de esta, así como los indicadores que determinan la eficiencia de cada método que se use.	

Tabla 50. Tarea de ingeniería 17. Actualizar las estadísticas relacionadas con: dirección física, dirección lógica, cantidad de páginas, cantidad de marcos, valor de la fragmentación interna.

Tarea	
Número de tarea: 17	**Número de historia de usuario:** 17
Nombre de la tarea: Actualizar las estadísticas relacionadas con: dirección física, dirección lógica, cantidad de páginas, cantidad de marcos, valor de la fragmentación interna.	
Tipo de tarea: Desarrollo	**Puntos estimados: 1**
Programador Responsable: Thais Mel Soteras Fong.	
Descripción: Con el desarrollo de la presente tarea, se realiza la funcionalidad que permite al sistema calcular la dirección física, dirección lógica, cantidad de páginas, cantidad de marcos y valor de la fragmentación interna a partir de los datos introducidos o de los resultados obtenidos luego de la simulación.	

Tabla 51. Tarea de ingeniería 18. Comparar los algoritmos de reemplazo de páginas.

Tarea	
Número de tarea: 18	**Número de historia de usuario:** 18
Nombre de la tarea: Comparar los algoritmos de reemplazo de páginas.	
Tipo de tarea: Desarrollo	**Puntos estimados: 1**
Programador Responsable: Thais Mel Soteras Fong.	
Descripción: Con el desarrollo de la presente tarea, se realiza la funcionalidad que permite al sistema realizar comparaciones entre los resultados obtenidos luego de la simulación con diferentes algoritmos para el mismo juego de datos.	

Tabla 52. Tarea de ingeniería 19. Administrar cola de peticiones.

Tarea	
Número de tarea: 19	**Número de historia de usuario:** 19
Nombre de la tarea: Administrar cola de peticiones.	
Tipo de tarea: Desarrollo	**Puntos estimados: 0.5**

Programador Responsable: Thais Mel Soteras Fong.
Descripción: Con el desarrollo de la presente tarea, se realiza la funcionalidad que permite al sistema introducir las peticiones de atención del disco y los instantes de tiempo en que se generan.

Tabla 53. Tarea de ingeniería 21. Visualizar la simulación la planificación de acceso al disco.

Tarea	
Número de tarea: 21	**Número de historia de usuario:** 21
Nombre de la tarea: Visualizar la simulación la planificación de acceso al disco.	
Tipo de tarea: Desarrollo	**Puntos estimados: 1.5**
Programador Responsable: Thais Mel Soteras Fong.	
Descripción: Con el desarrollo de la presente tarea, se realiza la funcionalidad que permite al sistema mostrar el resultado de la simulación del algoritmo para la cola de solicitudes. Puede ser la simulación completa o paso a paso.	

Tabla 54. Tarea de ingeniería 22. Actualizar las estadísticas relacionadas con: instantes de tiempo en que se realizan las peticiones, tiempo de búsqueda, latencia y transferencia.

Tarea	
Número de tarea: 22	**Número de historia de usuario:** 22
Nombre de la tarea: Actualizar las estadísticas relacionadas con: instantes de tiempo en que se realizan las peticiones, tiempo de búsqueda, latencia y transferencia.	
Tipo de tarea: Desarrollo	**Puntos estimados: 1**
Programador Responsable: Thais Mel Soteras Fong.	
Descripción: Con el desarrollo de la presente tarea, se realiza la funcionalidad que permite al usuario calcular el tiempo de búsqueda, latencia y transferencia, así como el tiempo de ejecución total del algoritmo a partir de los datos introducidos y de los resultados obtenidos luego de la simulación.	

Tabla 55. Tarea de ingeniería 23. Comparar los algoritmos de planificación de acceso a disco.

Tarea	
Número de tarea: 23	**Número de historia de usuario:** 23
Nombre de la tarea: Comparar los algoritmos de planificación de acceso a disco.	
Tipo de tarea: Desarrollo	**Puntos estimados: 0.5**
Programador Responsable: Thais Mel Soteras Fong.	
Descripción: Con el desarrollo de la presente tarea, se realiza la funcionalidad que permite al sistema realizar comparaciones entre los resultados obtenidos luego de la simulación con diferentes algoritmos para el mismo juego de datos.	

Anexo 3: Casos de prueba de aceptación

Tabla 56. Caso de prueba de aceptación HU1-P1

Caso de prueba de aceptación	
Código de la prueba: HU1-P1	**Nombre de la historia de usuario:** Administrar datos de procesos
Descripción de la prueba: La prueba consiste en verificar que el sistema gestiona los datos de los procesos que fueron introducidos.	
Condiciones de ejecución: • La aplicación debe estar abierta en la computadora. • Los datos se introducen de forma manual o los genera la aplicación.	
Pasos de ejecución: • Abrir la aplicación. • Introducir la cantidad de procesos • Introducir el instante de llegada y la duración de cada proceso. • Introducir la cantidad de bloqueos de cada proceso y la duración de cada uno.	
Resultados esperados: La aplicación muestra una ventana para introducir los datos al sistema y debe permitir la modificación o eliminación de estos.	

Tabla 57. Caso de prueba de aceptación HU3-P3

Caso de prueba de aceptación	
Código de la prueba: HU3-P3	**Nombre de la historia de usuario:** Visualizar la simulación instante a instante o de forma automática a través de un diagrama de Gantt.
Descripción de la prueba: La prueba consiste en comprobar que se muestran los resultados obtenidos en la prueba HU3-P3 a través de un diagrama de Gantt.	

Condiciones de ejecución: • La aplicación debe estar abierta en la computadora. • Las acciones contempladas en la prueba HU2-P2 deben ejecutarse satisfactoriamente.
Pasos de ejecución: • Abrir la aplicación. • Ejecutar la simulación
Resultados esperados: Se visualiza la simulación del algoritmo seleccionado a través de un Diagrama de Gantt.

Tabla 58. Caso de prueba de aceptación HU4-P4

Caso de prueba de aceptación	
Código de la prueba: HU4-P4	**Nombre de la historia de usuario:** Mostrar los datos de cada proceso, en cuanto a instante de llegada y de fin, tiempo que estuvo en estado listo, tiempo que estuvo en estado bloqueado, y tiempo que estuvo en ejecución.
Descripción de la prueba: La prueba consiste en verificar que se muestre el instante de llegada y de fin, tiempo que estuvo en estado listo, tiempo que estuvo en estado bloqueado, y tiempo que estuvo en ejecución para cada proceso.	
Condiciones de ejecución: • La aplicación debe estar abierta en la computadora. • Las acciones contempladas en la prueba HU2-P2 deben ejecutarse satisfactoriamente.	
Pasos de ejecución: • Abrir la aplicación. • Ejecutar la simulación. • Seleccionar los datos que se desea mostrar.	

Resultados esperados:
La aplicación muestra en una tabla los datos seleccionados de cada proceso.

Tabla 59. Caso de prueba de aceptación HU5-P5

Caso de prueba de aceptación	
Código de la prueba: HU5-P5	**Nombre de la historia de usuario:** Calcular y mostrar estadísticas relacionadas con: tiempo de espera medio, tiempo de retorno promedio y por ciento de uso del CPU.
Descripción de la prueba: La prueba consiste en verificar que se efectúen los cálculos del tiempo de espera medio, tiempo de retorno promedio y por ciento de uso del CPU de forma correcta.	
Condiciones de ejecución: • La aplicación debe estar abierta en la computadora. • Las acciones contempladas en la prueba HU2-P2 deben ejecutarse satisfactoriamente.	
Pasos de ejecución: • Abrir la aplicación. • Ejecutar la simulación. • Introducir la duración de los cambios de contexto. • Efectuar cálculos.	
Resultados esperados: La aplicación muestra los cálculos relacionados con el uso del CPU y las estadísticas correspondientes a tiempo de espera y tiempo de retorno.	

Tabla 60. Caso de prueba de aceptación HU6-P6

Caso de prueba de aceptación	
Código de la prueba: HU6-P6	**Nombre de la historia de usuario:**

	Mostrar las gráficas que representan el tiempo de espera medio y tiempo de retorno medio de cada proceso.
Descripción de la prueba: La prueba consiste en verificar que se muestren las gráficas que representan el tiempo de espera medio y tiempo de retorno medio de cada proceso.	
Condiciones de ejecución: • La aplicación debe estar abierta en la computadora. • Las acciones contempladas en las pruebas HU2-P2 y HU5-P5 deben ejecutarse satisfactoriamente.	
Pasos de ejecución: • Abrir la aplicación. • Ejecutar la simulación. • Introducir la duración de los cambios de contexto. • Efectuar cálculos. • Seleccionar gráficos.	
Resultados esperados: La aplicación muestra las gráficas que representan el tiempo de espera medio y tiempo de retorno medio de cada proceso y del sistema.	

Tabla 61. Caso de prueba de aceptación HU7-P7

Caso de prueba de aceptación	
Código de la prueba: HU7-P7	**Nombre de la historia de usuario:** Realizar comparaciones entre algoritmos de planificación.
Descripción de la prueba: La prueba consiste en verificar que el sistema permita realizar varias simulaciones a partir de los datos introducidos utilizando diferentes algoritmos para determinar cuál resulta más óptimo para los datos empleados.	

Condiciones de ejecución: • La aplicación debe estar abierta en la computadora. • Las acciones contempladas en la prueba HU1-P1 deben ejecutarse satisfactoriamente.
Pasos de ejecución: • Abrir la aplicación. • Ejecutar la simulación. • Introducir la duración de los cambios de contexto. • Efectuar cálculos. • Ejecutar nueva simulación • Efectuar cálculos. • Comparar simulaciones.
Resultados esperados: La aplicación muestra las comparaciones entre los resultados obtenidos luego de la simulación con diferentes algoritmos para el mismo juego de datos

Tabla 62. Caso de prueba de aceptación HU8-P8

Caso de prueba de aceptación	
Código de la prueba: HU8-P8	**Nombre de la historia de usuario:** Administrar datos de recursos.
Descripción de la prueba: La prueba consiste en verificar que el sistema gestiona los datos correspondientes a la cantidad de procesos, cantidad de recursos, los recursos asignados y solicitados por cada proceso, la existencia y la disponibilidad del sistema, así como una secuencia segura de ejecución de los procesos introducidos por el usuario.	
Condiciones de ejecución: • La aplicación debe estar abierta en la computadora.	

Pasos de ejecución: • Abrir la aplicación. • Introducir cantidad de procesos • Introducir cantidad de recursos y sus instancias • Introducir las asignaciones de recursos a cada proceso
Resultados esperados: • Debe mostrar los datos correspondientes a la cantidad de procesos, cantidad de recursos, los recursos asignados y solicitados por cada proceso, la existencia y la disponibilidad del sistema, así como una secuencia segura de ejecución de los procesos. • Debe dar la posibilidad de no introducir todos los datos puesto que el usuario podrá calcular algunos de ellos de forma automática.

Tabla 63. Caso de prueba de aceptación HU9-P9

Caso de prueba de aceptación	
Código de la prueba: HU9-P9	**Nombre de la historia de usuario:** Elaborar las matrices de asignación y solicitud a partir de los datos de los recursos.
Descripción de la prueba: La prueba consiste en verificar si se elaboran las matrices de asignación y solución a partir de los datos de los recursos.	
Condiciones de ejecución: • La aplicación debe estar abierta en la computadora. • Las acciones contempladas en la prueba HU8-P8 deben ejecutarse satisfactoriamente.	
Pasos de ejecución: • Abrir la aplicación. • Introducir cantidad de procesos • Introducir cantidad de recursos y sus instancias	

• Introducir las asignaciones de recursos a cada proceso • Introducir las solicitudes de recursos de cada proceso
Resultados esperados: La aplicación muestra las matrices de Asignación y Solicitud a partir de los datos introducidos por el usuario.

Tabla 64. Caso de prueba de aceptación HU12-P12

Caso de prueba de aceptación	
Código de la prueba: HU12-P12	**Nombre de la historia de usuario:** Administrar datos de la memoria y los procesos.
Descripción de la prueba: La prueba consiste en verificar que el sistema gestiona los datos de los procesos y la memoria que fueron introducidos por el usuario.	
Condiciones de ejecución: • La aplicación debe estar abierta en la computadora. • Los datos se introducen de forma manual o los genera la aplicación.	
Pasos de ejecución: • Abrir la aplicación. • Introducir la cantidad de procesos • Introducir la cantidad de páginas • Introducir la cantidad de marcos • Introducir tamaño de página y tamaño de la memoria. • Introducir tamaño de cada proceso.	
Resultados esperados: La aplicación muestra una ventana para introducir los datos al sistema y debe permitir la modificación o eliminación de estos.	

Tabla 65. Caso de prueba de aceptación HU13-P13

Caso de prueba de aceptación

Código de la prueba: HU13-P13	**Nombre de la historia de usuario:** Gestionar tabla de páginas de los procesos.
Descripción de la prueba: La prueba consiste en verificar la gestión de la tabla de páginas de los procesos a partir de los datos introducidos en la HU12-P12.	
Condiciones de ejecución: • La aplicación debe estar abierta en la computadora. • Las acciones contempladas en la prueba HU12-P12 deben ejecutarse satisfactoriamente.	
Pasos de ejecución: • Abrir la aplicación. • Generar tabla de páginas.	
Resultados esperados: La aplicación muestra la tabla de páginas de cada proceso.	

Tabla 66. Caso de prueba de aceptación HU15-P15

Caso de prueba de aceptación	
Código de la prueba: HU15-P15	**Nombre de la historia de usuario:** Visualizar la simulación instante a instante o de forma automática a través de la tabla de memoria.
Descripción de la prueba: La prueba consiste en verificar que se visualice la simulación instante a instante o de forma automática a través de la tabla de memoria.	
Condiciones de ejecución: • La aplicación debe estar abierta en la computadora. • Las acciones contempladas en la prueba HU14-P14 deben ejecutarse satisfactoriamente.	
Pasos de ejecución: • Abrir la aplicación.	

• Seleccionar el algoritmo y su política. • Ejecutar la simulación. • Seleccionar la visualización (automática o instante a instante)
Resultados esperados: La aplicación muestra la simulación correctamente a partir de los parámetros seleccionados.

Tabla 67. Caso de prueba de aceptación HU16-P16

Caso de prueba de aceptación	
Código de la prueba: HU16-P16	**Nombre de la historia de usuario:** Análisis de resultados.
Descripción de la prueba: La prueba consiste en verificar que el análisis de los resultados obtenidos luego de la simulación es correcto.	
Condiciones de ejecución: • La aplicación debe estar abierta en la computadora. • Las acciones contempladas en la prueba HU14-P14 deben ejecutarse satisfactoriamente.	
Pasos de ejecución: • Abrir la aplicación. • Seleccionar el algoritmo y su política. • Ejecutar la simulación. • Analizar resultados.	
Resultados esperados: La aplicación muestra los resultados obtenidos luego de que el usuario haga la simulación de los algoritmos.	

Tabla 68. Caso de prueba de aceptación HU17-P17

Caso de prueba de aceptación

Código de la prueba: HU17-P17	**Nombre de la historia de usuario:** Calcular y mostrar estadísticas relacionadas con: dirección física, dirección lógica, cantidad de páginas, cantidad de marcos, valor de la fragmentación interna.
Descripción de la prueba: La prueba consiste en verificar que se efectúen los cálculos de dirección física, dirección lógica, cantidad de páginas, cantidad de marcos, valor de la fragmentación interna de forma correcta.	
Condiciones de ejecución: • La aplicación debe estar abierta en la computadora. • Las acciones contempladas en la prueba HU14-P14 deben ejecutarse satisfactoriamente.	
Pasos de ejecución: • Abrir la aplicación. • Abrir la aplicación. • Seleccionar el algoritmo y su política. • Ejecutar la simulación. • Efectuar cálculos.	
Resultados esperados: La aplicación muestra los cálculos y estadísticas de la dirección física, dirección lógica, cantidad de páginas, cantidad de marcos, valor de la fragmentación interna.	

Tabla 69. Caso de prueba de aceptación HU18-P18

Caso de prueba de aceptación	
Código de la prueba: HU18-P18	**Nombre de la historia de usuario:** Realizar comparaciones entre algoritmos de reemplazo de páginas.

Descripción de la prueba: La prueba consiste en verificar que el sistema realice comparaciones entre los resultados obtenidos luego de la simulación con diferentes algoritmos de reemplazo de páginas para el mismo juego de datos.
Condiciones de ejecución: • La aplicación debe estar abierta en la computadora. • Las acciones contempladas en la prueba HU14-P14 deben ejecutarse satisfactoriamente.
Pasos de ejecución: • Abrir la aplicación. • Seleccionar el algoritmo y su política. • Ejecutar la simulación. • Ejecutar nueva simulación. • Comparar simulaciones.
Resultados esperados: La aplicación muestra las comparaciones entre los resultados obtenidos luego de la simulación con diferentes algoritmos de reemplazo de páginas para un mismo juego de datos.

Tabla 70. Caso de prueba de aceptación HU19-P19

Caso de prueba de aceptación	
Código de la prueba: HU19-P19	**Nombre de la historia de usuario:** Administrar cola de peticiones.
Descripción de la prueba: La prueba consiste en verificar que el sistema gestiona los datos de las peticiones que fueron introducidas por el usuario.	
Condiciones de ejecución: • La aplicación debe estar abierta en la computadora.	

Pasos de ejecución: • Abrir la aplicación. • Introducir cantidad de pistas. • Introducir las peticiones una a una. • De manera opcional introducir el instante de tiempo en que se realiza cada petición. • Introducir el tiempo de búsqueda. • Introducir el tiempo de Latencia-Transferencia.
Resultados esperados: La aplicación crea una cola con las peticiones generadas por el usuario.

Tabla 71. Caso de prueba de aceptación HU21-P21

Caso de prueba de aceptación	
Código de la prueba: HU21-P21	**Nombre de la historia de usuario:** Visualizar la simulación instante a instante o de forma automática a través de un gráfico.
Descripción de la prueba: La prueba consiste en verificar si se mostró el resultado de la simulación del algoritmo para la cola de solicitudes. Puede ser la simulación completa o paso a paso.	
Condiciones de ejecución: • La aplicación debe estar abierta en la computadora. • Las acciones contempladas en la prueba HU20-P20 deben ejecutarse satisfactoriamente.	
Pasos de ejecución: • Abrir la aplicación. • Seleccionar el algoritmo. • Ejecutar la simulación. • Seleccionar la visualización (automática o instante a instante)	

Resultados esperados:
La aplicación muestra la simulación correctamente a partir de los parámetros seleccionados.

Tabla 72. Caso de prueba de aceptación HU22-P22

Caso de prueba de aceptación	
Código de la prueba: HU22-P22	**Nombre de la historia de usuario:** Calcular y mostrar estadísticas relacionadas con: instantes de tiempo en que se realizan las peticiones, tiempo de búsqueda, latencia y transferencia.
Descripción de la prueba: La prueba consiste en verificar que se efectúen los cálculos de instantes de tiempo en que se realizan las peticiones, tiempo de búsqueda, latencia y transferencia correctamente.	
Condiciones de ejecución: • La aplicación debe estar abierta en la computadora. • Las acciones contempladas en la prueba HU20-P20 deben ejecutarse satisfactoriamente.	
Pasos de ejecución: • Abrir la aplicación. • Abrir la aplicación. • Seleccionar el algoritmo. • Ejecutar la simulación. • Efectuar cálculos.	
Resultados esperados: La aplicación muestra los cálculos y estadísticas de los instantes de tiempo en que se realizan las peticiones, tiempo de búsqueda, latencia y transferencia.	

Tabla 73. Caso de prueba de aceptación HU23-P23

Caso de prueba de aceptación	
Código de la prueba: HU23-P23	**Nombre de la historia de usuario:** Realizar comparaciones entre algoritmos de planificación de acceso a disco.
Descripción de la prueba: La prueba consiste en verificar que el sistema realice comparaciones entre los resultados obtenidos luego de la simulación con diferentes algoritmos de atención a peticiones del disco para el mismo juego de datos.	
Condiciones de ejecución: • La aplicación debe estar abierta en la computadora. • Las acciones contempladas en la prueba HU20-P20 deben ejecutarse satisfactoriamente.	
Pasos de ejecución: • Abrir la aplicación. • Seleccionar el algoritmo. • Ejecutar la simulación. • Ejecutar nueva simulación. • Comparar simulaciones.	
Resultados esperados: La aplicación muestra las comparaciones entre los resultados obtenidos luego de la simulación con diferentes algoritmos para un mismo juego de datos.	