Universidad de las Ciencias Informáticas

Facultad 10

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE

"INGENIERO EN CIENCIAS INFORMÁTICAS"

TÍTULO: ANÁLISIS Y DISEÑO DE UN SIMULADOR DE ALGORITMOS DE REEMPLAZO DE PÁGINAS PARA LA ASIGNATURA DE SISTEMAS OPERATIVOS EN LA UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS.

Autor:

Yamilé García Hernández

Tutores:

Ing. Joelsy Porven Rubier

MSc. Tomás López Jiménez

Ciudad de la Habana

Junio/ 2008

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo al departamento de Sistemas Digitales de la Universidad de las Ciencias Informáticas; así como a dicho centro para que hagan el uso que estimen pertinente con este trabajo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año 2008

Yamilé García Hernández

_____

Firma del Autor

Joelsy Porven Rubier

_____

Firma del Tutor

Tomás López Jiménez

_____

Firma del Co-Tutor

DATOS DEL CONTACTO:

Joelsy Porven Rubier. Pinar del Río, Cuba. 1979. Graduado de Ingeniero en Automática en el ISPJAE en el 2003. Se desempeña como Asesor Metodológico del Departamento de Sistemas Digitales atendiendo directamente la asignatura de Sistemas Operativos la cual imparte en pregrado, además a impartido las asignaturas de Maquinas Computadoras, Seguridad Informática y Teleinformatica. Ha cursado numerosos postgrados del área de las telecomunicaciones y la administración de servicios en redes.

E-mail: jporven@uci.cu

Tomás López Jiménez. Guantánamo, Cuba. 1943. Graduado de Ingeniero Electricista en la Universidad de la Habana en 1974. MSc en Informática Aplicada en el ISPJAE en el 2007. Investigador Titular desde el año 1985. Profesor Auxiliar en el 2007. Asesor del Rector desde el 2005 y Director de Estrategia y Calidad de la Universidad de las Ciencias Informáticas desde el 2008, con una larga trayectoria laboral, de ella 42 años en la Informática. Su actividad docente la desarrolla en la UCI, impartiendo las asignaturas de Historia de la Informática y Metodología de la Investigación Científica en la Facultad 10 en pregrado, así como profesor de postgrado y colaborador de los Departamentos Docentes Centrales de Práctica Profesional y Sistemas Digitales. Ha cursado numerosos postgrados de la especialidad y de Técnicas de Dirección, incluyendo un diplomado en Dirección y Gestión Empresarial. Ha sido miembro de numerosas organizaciones profesionales y científicas, siendo actualmente integrante de la Sección 9.7 sobre Historia de la Informática de la IFIP.

Correo Electrónico: tlopezj@uci.cu

La forma más antigua de almacenar la información a través del tiempo sigue siendo una gran fuente de la cual podemos extraer todo aquello que necesitamos conocer……

José Carlos Cortizo Pérez

AGRADECIMENTOS

Gracias a mis padres por abrir mis ojos a la luz del mundo….

Gracias en especial a Yavier por soportarme y darme las fuerzas….Gracias…. por existir.

Gracias mis amigas Mónica, Susel, Anaisa, y las que han compartido estos 5 años junto a mí, gracias a mi papá Maikel y a mi familia Hernández por ser ellos mismos.

Mil gracias a todos aquellos que me han brindado su apoyo…

Gracias a la revolución por este sueño que se hace realidad….

Yami.

DEDICATORIA

A mis padres Gladis y Carlos, por su amor incondicional, por ser este su sueño y por todas las cosas lindas que hemos disfrutado juntos.

A mi novio Yavier por soportarme estos años, por enseñarme lo que significa ser amar, por ser siempre él mismo y estar cuando lo necesito.

A mi familia que siempre confió en mí y me brindo su ayuda cuando la necesite.

A mis amigas de toda la vida.

A todos aquellos que amo.

Yami.

RESUMEN

La investigación propone el análisis y diseño de una herramienta de simulación para los algoritmos de reemplazo de página en la asignatura de Sistema Operativo en la Universidad de las Ciencias Informáticas (UCI). El desarrollo de está aplicación favorecerá el estudio y comprensión del tema Administración de Memoria y los estudiantes que reciben estos conocimientos podrán visualizar las operaciones que realiza el Sistema Operativo para optimizar el uso de sus recursos. En el mundo actual existen varias herramientas que apoyan el estudio de este tema sin embargo en la asignatura Sistemas Operativos en la UCI aun no cuenta con una aplicación similar, de ahí que en la investigación se proponga un prototipo de que permita a los estudiantes comprender y estudiar dichos algoritmos.

Se abordan los conceptos para el desarrollo de una aplicación de simulación de los algoritmos de reemplazo de página, se analizan los TDA que pueden ser utilizados para implementar dicho sistema así como las técnicas de reemplazo a simular.

La investigación muestra los resultados del análisis y diseño de la herramienta en cuestión usando eXtreming Programing, se realiza una propuesta del prototipo de interfaz de usuario y además se hacen algunas recomendaciones para la continuidad y desarrollo de la aplicación.

Palabras Claves: Algoritmo de Reemplazo de Página, Fallo de Página, Memoria Virtual**,** Paginación, y Simulador.

TABLA DE CONTENIDO

INTRODUCCIÓN

La evolución de las computadoras ha favorecido el desarrollo de las condiciones de vida de la mayoría de las personas a nivel mundial, por la generalización del uso de estas para el almacenamiento y la manipulación de los datos en las ramas de la economía y la sociedad. Estas revolucionarias máquinas están compuestas por dos partes fundamentales, el hardware (motherboard, tarjetas de red, puertos de E/S, etc) y el software donde encontramos a los Sistemas Operativos (SO) como programa principal.

Un Sistema Operativo está compuesto por un conjunto de programas que actúan como interfaces entre el usuario de la computadora, su hardware y las demás componentes del software. El objetivo principal de un SO es facilitar y optimizar el uso de los recursos del sistema, tanto los de hardware como los de software, por lo que este puede verse como un administrador de los recursos de hardware y el encargado de presentar al usuario una capa de abstracción de software que lo aísla de la complejidad del hardware. Para su estudio suelen dividirse en los cuatro administradores fundamentales siguientes:

- Administrador de procesos
- Administrador de dispositivos
- Administrador de información
- Administrador de memoria

La memoria es un valioso recurso de las computadoras, por ello debe ser asignada y utilizada de forma muy cuidadosa, por lo que existen diversas maneras de administrarla, lo que depende en gran medida de las posibilidades que brinde el SO. Una de las vías más simples es que no exista la administración de memoria, pues así los programas son cargados completos a la memoria y se utilizan todos los recursos del sistema. De esta manera el usuario tiene un dominio total de la memoria y no se necesita un hardware especial, este debe ocuparse del tratamiento de interrupciones y del control del sistema [1].

En la actualidad cuando se habla de memoria puede referirse a una forma de almacenamiento de estado sólido conocido como Memoria RAM (por sus siglas en inglés Random Access Memory) y otras veces se refiere a las formas de guardar y recuperar la información de manera rápida pero temporal (Cache). Por otra parte, estas se describen también como formas de almacenamiento masivo en medios magnéticos

como los discos duros, medios ópticos como los discos compactos y otros tipos de almacenamiento como la ROM (Read Only Memory) y las memorias USB como las flash.

Las primeras ideas de utilizar un dispositivo para guardar la información, surgieron con el francés Joseph Jacquard en 1801 que consistía en un telar que producía automáticamente ropa modelada, pues la información podía codificarse en tarjetas perforadas, lo que podía considerarse como una serie de instrucciones. Más tarde el inglés Charles Babbage en 1830 consideró la realización de una máquina analítica que tendría una memoria interna para guardar instrucciones y los resultados intermedios de los cálculos realizados.

En 1946 fue introducido el concepto de almacenamiento de los programas por John Von Neumann además de la idea de usar un equipo para un mayor número de programas, lo cual hizo posible la lectura de los programas cargados en la memoria y después la ejecución de sus instrucciones sin necesidad de reescribirlas. La primera computadora en usar esto fue Electronic Discrete Variable (EDVAC), creada por Neumann y otros desarrolladores. Estos programas almacenados dieron a las computadoras una mayor velocidad, gran flexibilidad, confiabilidad y la existencia de menos errores [2] .

Durante la segunda generación de computadoras (1959-1964), en 1961 exactamente nace una nueva tecnología dentro de la administración de memoria conocida como memoria virtual diseñada por Fotheringham, esta consiste en que los procesos cuentan con un espacio de memoria mayor que el que realmente tienen, es la utilizada muchos sistemas operativos actuales. Esta utiliza técnicas o algoritmos especiales para cumplir con sus implementaciones y especificaciones.

El desarrollo de estas técnicas es muy importante para el continuo avance de las Ciencias de la Información, pues cada día que pasa se almacenan muchísimos más datos en dispositivos de memoria de dimensiones físicas mas pequeñas, lo que puede observarse claramente en la evolución del hardware de las computadoras, ya que al comienzo estas ocupaban un edificio completo a la vez que se necesitaban varias personas para manejarlas y actualmente hasta en un bolsillo podemos tener acceso a la información. Es por ello que es tan importante analizarlas, y estudiarlas.

En las Universidades del país donde se cursa la carrera de Ingeniero en Informática y otras ingenierías se imparte la asignatura de Sistemas Operativos en las cuales se estudia el tema de la administración de

memoria, la memoria virtual y las técnicas que esta utiliza. El proceso de estudio, comprensión y aprendizaje de este tema les resulta difícil a los estudiantes, pues la asimilación de estos conocimientos no está favorecida por una presentación que ayude a estos a captar la información desde varios puntos de vista. A su vez la práctica de forma manual de dichos conocimientos que demuestren a los alumnos lo que han aprendido es muy trabajosa por que los algoritmos son extensos y complejos, a la vez dependen mucho de las habilidades del docente y estos pierden mucho tiempo tratando de entender cómo es que los esquemas funcionan [3].

Este problema está bien presente en la Universidad de Ciencias Informáticas (UCI) cuando se imparten estos conocimientos, resultan muy ambiguos y la mayoría de los alumnos no pueden interiorizar exactamente el contenido y los objetivos de la clase, no son capaces de reconocer la importancia de la información que reciben, como no existe una herramienta que apoye la actividad docente y que permita a los estudiantes apropiarse del contenido de manera consciente y reflexiva, que les permita valorar, comparar, y observar las transformaciones que ocurren con los algoritmos de administración de memoria, y que tenga una función instructiva, motivadora y creadora en el estudiantado [4].

Por lo anterior esta investigación se plantea como problema científico:

¿Cómo lograr la comprensión de los algoritmos de administración de memoria en la asignatura de Sistemas Operativos?

En el cual el objeto de la investigación es el proceso de enseñanza-aprendizaje de la asignatura de Sistemas Operativos en la carrera de Ingeniería en Ciencias Informáticas, de donde se define el campo de acción como los algoritmos de la administración de memoria en la asignatura.

La idea a defender plantea que:”la existencia de una herramienta de simulación contribuirá al aprendizaje de los algoritmos de reemplazo de página en la asignatura de Sistemas Operativos”.

Partiendo de esta temática el objetivo de la investigación estará centrado realizar al análisis y diseño de una herramienta que simule los algoritmos de administración de memoria para la asignatura de sistemas operativos.

Y para alcanzarlo se han propuesto como tareas las siguientes:

- Investigar el funcionamiento de los algoritmos de administración de memoria en los sistemas operativos.
- Analizar los métodos de implementación para simular los algoritmos de administración de memoria.
- Realizar un estudio a profundidad de las herramientas a utilizar para el desarrollo la aplicación.

Como métodos teóricos para esta investigación se utilizarán:

- Métodos Histórico-Lógicos, pues a los estudiantes les resulta difícil de analizar y asimilar los contenidos de este tema, en el cual desde el surgimiento de la UCI existen año tras año muchos suspensos.
- Métodos Análisis –Síntesis, se aplicó este método para el análisis de los elementos bibliográficos, definiciones, de los diferentes autores sobre la administración de memoria y los Sistemas Operativos.
- Método de Inducción – Deducción para conocer las concepciones acerca de los temas de Administración de Memoria, Sistema Operativo y como estos influyen en los métodos de enseñanza actuales.

Estructuración del contenido por capítulos:

- Capítulo 1: Fundamentación teórica, se definen los principales conceptos relacionados con el objeto de estudio así como se describen las principales herramientas y metodologías utilizadas durante la investigación.
- Capítulo 2: Análisis del sistema, se determina y describe un modelo para simular los algoritmos de administración de memoria, así como los de describe las historias de usuario.
- Capítulo 3: Simulador de Memoria Virtual, se determina el patrón de arquitectura a utilizar y se definen las clases del diseño así como se muestra el prototipo de interfaz de usuario.

CAPÍTULO 1 FUNDAMENTACIÓN TEÓRICA.

1.1 INTRODUCCIÓN

En este capítulo se definen las bases conceptuales de la investigación. Se describen los conceptos fundamentales relacionados con la administración de memoria, paginación, segmentación y memoria virtual. Se analizan las principales características de los algoritmos de reemplazo de páginas y las herramientas existentes que los simulan. Se detallan además, las diferentes herramientas; metodologías utilizadas y consideradas en el desarrollo de la investigación.

1.2 ADMINISTRACIÓN DE MEMORIA

La administración de memoria se define como los distintos métodos y operaciones que se encargan de obtener la máxima utilidad de la memoria, organizando los procesos y programas que se ejecutan de manera tal que se aproveche de la mejor manera posible el espacio disponible [5].

La administración de memoria es una tarea realizada por el Sistema Operativo con el apoyo del hardware de gestión de memoria, estos deben administrar el espacio existente de memoria para cada proceso. La Unidad de Administración de Memoria (MMU por sus siglas en ingles) que no es más que un dispositivo hardware que transforma las direcciones lógicas en físicas. Donde las direcciones lógicas o direcciones virtuales son aquellas generadas por los procesos y que conforman el espacio de direcciones virtuales y las direcciones físicas o direcciones reales como su nombre lo indica son las que conforman el mapa de memoria física.

Primeramente surgen los sistemas Monoprogramdos donde el procesador está mucho tiempo ocioso producto de las operaciones de entrada y salida (E/S), mientras el procesador este desocupado se hace necesario gestionar la memoria de manera eficiente de manera que carguen tantos procesos como sea posible [6].

La multiprogramación, es la técnica que permite que dos o más programas ocupen la misma unidad de memoria principal y que sean ejecutados de manera eficiente. Esta asigna de forma segura los recursos del sistema entre los distintos procesos existentes.

Entre las razones por la que se usa la multiprogramación es para facilitar la programación de una aplicación dividiéndola en dos o más procesos [7]. Lo que aumenta la utilización de los recursos de hardware. Para realizar un correcto funcionamiento de los diversos mecanismos y políticas relacionados con la administración de memoria se proponen cinco requisitos que se deben satisfacer [5]:

- Reubicación.
- Compartición.
- Protección.
- Organización lógica.
- Organización física.

Cada uno de estos requisitos tiene su función; lo que lo hace únicos e importantes. Las técnicas gestión de memoria tratan de integrar el mayor número de estos requisitos para alcanzar la mayor eficiencia posible.

1.3 TÉCNICAS DE GESTIÓN DE MEMORIA

La tarea principal de cualquier sistema de gestión de memoria es cargar programas en la memoria principal para que sean ejecutados, existen diversos formas de realizar esta tarea, algunos de los esquemas utilizados no satisfacen las necesidades y otros son sofisticados y complejos como la memoria virtual [8]

1.3.1 TÉCNICAS BASADAS EN ASIGNACIÓN CONTIGUA

Estas técnicas fueron las primeras utilizadas en sistemas Multiprogramados, eran sencillas y fáciles de implementar, pero no lograron satisfacer las exigencias impuestas a los sistemas computacionales, sobre todo de eficiencia.

Entre las técnicas basadas en la asignación continua se encuentra la técnica de particiones fijas de memoria y la de particiones variables. En la primera de estas prácticas la memoria principal es dividida en regiones con límites fijos. Existen dos posibilidades para esto: dividir la memoria en particiones de igual tamaño o bien de diferentes tamaños.

Con la primera posibilidad surgen dos dificultades:

- Un programa puede ser demasiado grande para ser asignado en una partición.
- Se malgasta el espacio interno a cada partición cuando el bloque cargado es más pequeño, esto es lo que se conoce como fragmentación interna, o sea, cualquier proceso por pequeño que sea, ocupará una partición completa [8].

En el esquema de particiones variables, estos fraccionamientos son distintos en número y longitud pues cuando un proceso se carga en memoria se le asigna exactamente la cantidad que necesita. Este método, aunque más efectivo que el de particiones fijas, también genera inconvenientes como la fragmentación externa; que surge producto a la entrada y la salida de procesos en la memoria, ya que se generan pequeñas segmentos de memoria sin utilizar. Como una solución a este problema se debe eliminar los espacios (huecos) entre procesos [8].

1.3.2 TÉCNICAS BASADAS EN ASIGNACIÓN NO CONTIGUA

Paginación Simple

Tanto las técnicas asignación de memoria de particiones fijas como la de particiones variables, hacen un uso ineficiente de la memoria provocando inconvenientes como la fragmentación ya sea interna o externa. Para solucionar este problema surge lo que conocemos como paginación que consiste en dividir dinámicamente la memoria en unidades de tamaño fijo, llamados marcos y donde los programas se dividen en unidades lógicas, denominadas páginas, estas tienen el mismo tamaño que los marcos de páginas. De está forma, se puede cargar una página de información en cualquier marco de página [9].

Las páginas sirven como unidad de almacenamiento de información y de transferencia entre memoria principal y memoria auxiliar o secundaria. Cada marco se identifica por la dirección de marco, que está en la posición física de la primera palabra en el marco de página.

Para tener el registro de dónde están almacenados los procesos en la memoria el sistema operativo crea y mantiene una tabla de páginas. Donde cada página tiene un número que se utiliza como índice en la tabla de páginas, lo que da por resultado el número del marco correspondiente a esa página virtual [9]. Cada entrada de la tabla de páginas contiene el número de marco de la página correspondiente en la memoria principal. En un proceso o programa en ejecución cada dirección lógica está formada por un número de página y un desplazamiento o sea la dirección lógica es igual al número de página + desplazamiento, y la dirección física es igual a el número de marco + desplazamiento [10]. Ver Figura 1.1

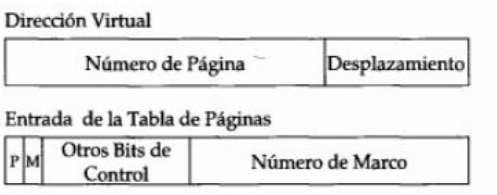


Figura 1.1 Representación de la dirección virtual

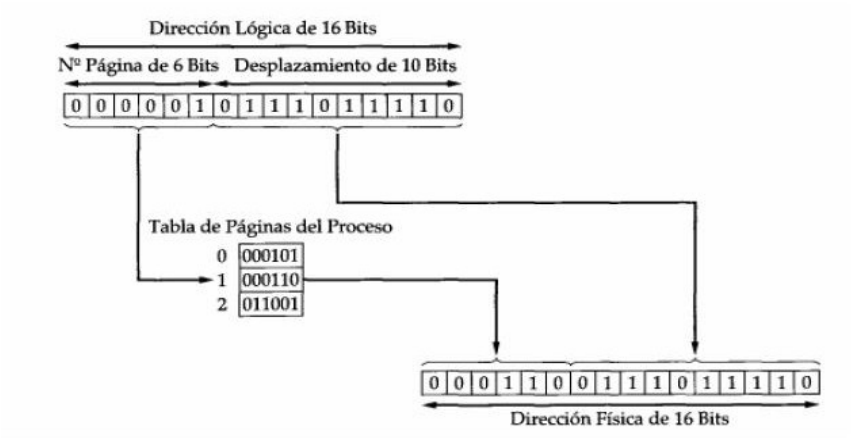


Figura 1.2 División de la dirección lógica para realizar la traducción a dirección física.

Segmentación

En este esquema el programa y sus datos son divididos en segmentos, no es necesario que todos sean del mismo tamaño. La dirección lógica segmentada consta de dos partes, un número de segmento, más un desplazamiento. La segmentación hace uso de una tabla de segmentos para cada proceso y una tabla de segmentos en la memoria principal. La tabla de segmentos tiene la dirección física del inicio del segmento y la longitud máxima de éste.

La segmentación proporciona al programador comodidad a la hora de organizar los programas y datos; el principal inconveniente es que programador debe estar consiente de la limitación del tamaño máximo de los segmentos, otra de las consecuencias del tamaño desigual de los segmentos es que no existe una correspondencia simple entre las direcciones lógicas y las direcciones físicas. Ver figura 1.2

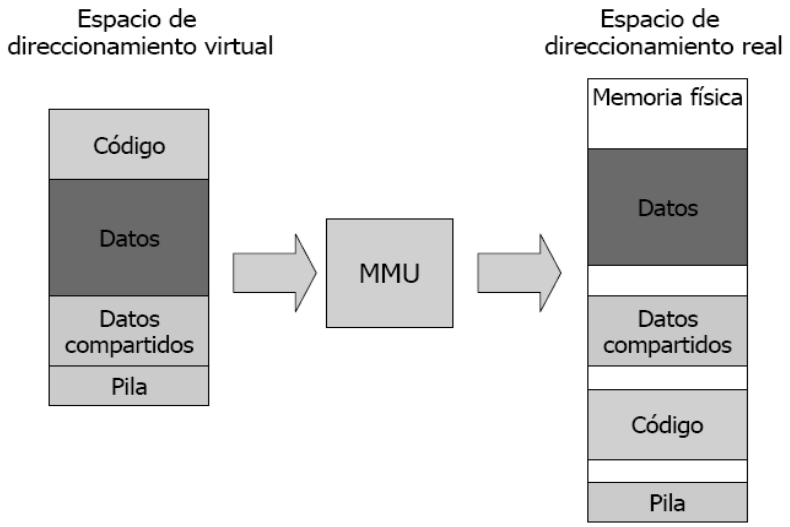


Figura 1.2: División de la memoria utilizando la técnica de Segmentación

Esquemas Combinados

Es posible combinar los esquemas de paginación y segmentación, tomando lo mejor de ambas partes. La paginación elimina la fragmentación externa y así se aprovecha la memoria principal de forma eficiente. Además, puesto que los fragmentos que se cargan y descargan de la memoria principal son de tamaño fijo e iguales. La segmentación tiene incluida la ventaja de encargar y gestionar estructuras de datos que puedan crecer. Este esquema es el utilizado en los Sistemas Operativos actuales [11].

Memoria Virtual

Se define la memoria virtual como una técnica para proporcionar la simulación de un espacio de memoria mucho mayor que la memoria física de una máquina. Esta "ilusión" permite que los programas se hagan sin tener en cuenta el tamaño exacto de la memoria física [11].También se pude definir como una técnica que permite ejecutar procesos que no caben totalmente en memoria Random Access Memory (RAM) [12].

Por lo que se concluye que la memoria virtual es una técnica que permite a los procesos cargarse o descargarse en la memoria principal sin importar su tamaño; esta se ha convertido en un componente esencial de la mayoría de los S.O actuales.

Lo que significa que las direcciones que son generadas por los programas hacen referencia a un espacio de memoria mucho mayor del existe en la memoria real. Donde este espacio de direcciones virtual con la ayuda de un mecanismo de traducción genera la dirección física de memoria principal a partir de la virtual.

A su vez la memoria principal y el disco se dividen en páginas del mismo tamaño, y como el número de páginas de la memoria virtual en general es mayor que el número de páginas disponibles de la memoria física; en cada momento sólo las copias de un conjunto de páginas virtuales del programa están presentes en la memoria física, las cuales son intercambiadas con las que se encuentran en el disco cuando estas son referenciadas [10].

Para la implementación de la memoria virtual se puede usar tanto la segmentación como la paginación, pero es más común emplear la paginación porque se trabaja con bloques de tamaño fijo, de este modo, las transferencias desde y hacia el disco se llevan a cabo de un modo mucho más sencillo [10].

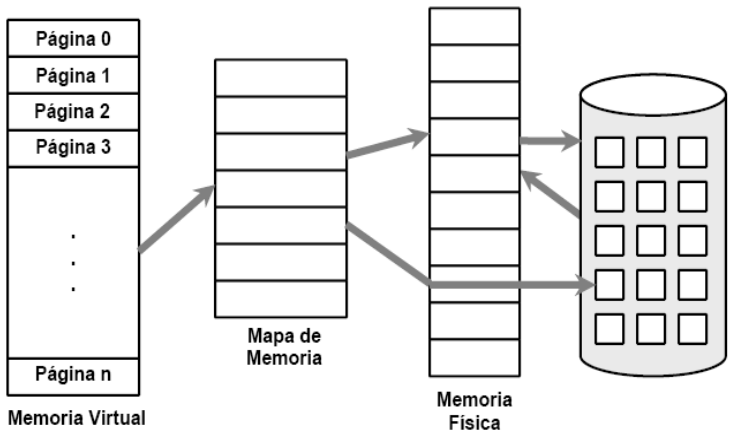


Figura 1.3 Intercambio de páginas entre la memoria principal y el disco.

En la técnica de memoria virtual las direcciones no pasan en forma directa al bus de memoria, sino que van a la MMU. Estas direcciones se llaman direcciones virtuales y conforman el espacio de direcciones virtuales. En este espacio encontramos las páginas y los marcos de página. Si el procesador encuentra una dirección lógica que no está en la memoria principal, genera un fallo de acceso a la memoria.

El S.O tendrá acceso a la mayor cantidad de páginas posibles, y cuando este transfiera de la memoria una página o un conjunto de páginas deberá expulsar otra. Si esto ocurre justo antes de ser usada, tendrá que traerla nuevamente. Demasiados intercambios de página se conoce como **hiperpaginación**; donde el procesador consume más tiempo intercambiando páginas o fragmentos que ejecutando instrucciones [10].

Cada proceso tiene su propia tabla de páginas, y cada entrada a la tabla de páginas tiene un número de marco de la página correspondiente, ya que sólo algunas de las páginas pueden estar en la memoria principal, por lo se necesita un bit en cada entrada a la tabla para indicar si la página está presente (P) en la memoria principal o no. Otro bit de control necesario es el bit de modificación (M), para indicar si el contenido de la página se ha alterado desde que se cargó la página en la memoria. También encontramos el bit presente (1) / ausente (0), si es 0 se provoca un señalamiento al sistema operativo y si es 1 el número de marco de la tabla de páginas se copia en los bits de mayor orden del registro de salida, también puede haber otros bits de control, por ejemplo si la compartición o protección se gestiona a nivel de página se necesitará otro bit con ese propósito [11].

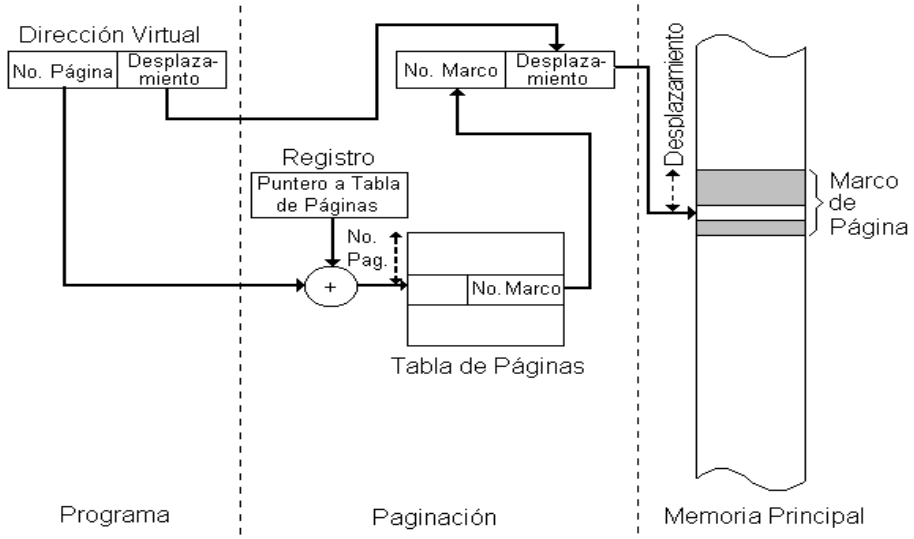


Figura 1.4.Proceso de traducción de direcciones lógicas.

La mayoría de los esquemas de memoria virtual hacen uso de una cache especial para las entradas de la tabla de páginas, llamada buffer de traducción adelantada (TLB, Translation Lookaside Buffer) que contiene las entradas de la tabla de páginas usadas recientemente [11]. Ver Figura 1.5

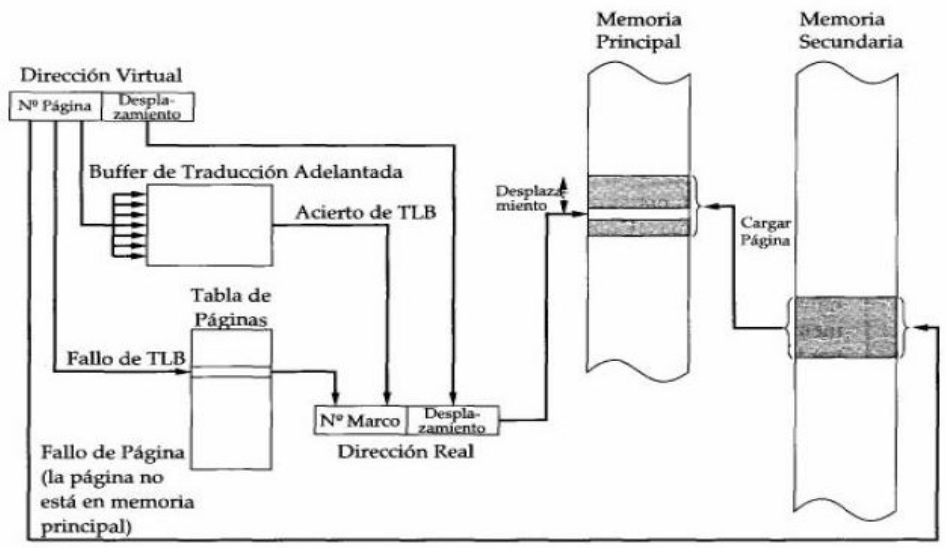


Figura 1.5 Empleo de Buffer de traducción adelantada (TLB).

La TLB contiene sólo algunas de las entradas de la tabla de páginas, el procesador puede consultar al mismo tiempo varias entradas de la TLB para determinar si hay coincidencia en el número de página, esto se conoce como correspondencia asociativa [10].

Dada una dirección virtual el procesador examinará la TLB. Si la entrada de la tabla de páginas buscada está presente se obtiene el número de marco y se forma la dirección real. Si no se encuentra, el procesador emplea el número de página para buscar en la tabla de páginas y examinar la entrada de la tabla de páginas. Si se encuentra activo el bit de presencia, la página está en la memoria principal y el procesador obtiene el número de marco para formar la dirección real. Luego el procesador actualiza la TLB con esta nueva entrada de la tabla de páginas. Si el bit de presencia no está activo, es que la página buscada no está en la memoria principal y se produce un fallo en el acceso a la memoria, llamado fallo de página. Ver Figura 1.6

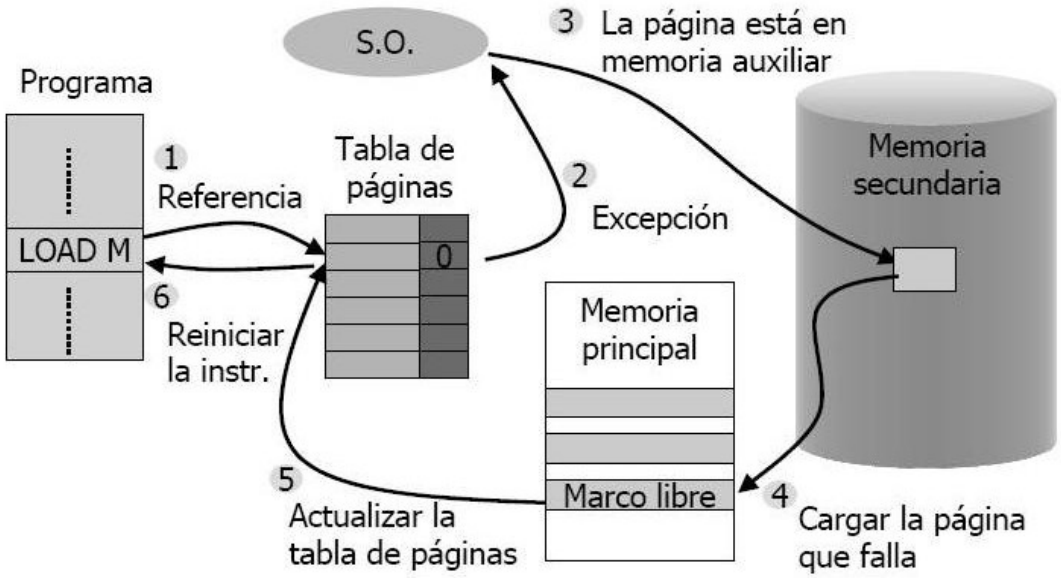


Figura 1.6 Acciones que ocurre con un fallo de página.

1.4 ALGORITMOS DE ADMINISTRACIÓN DE MEMORIA

Cuando ocurre un fallo de página, el sistema operativo debe elegir una página para retirarla de la memoria y traer la página que se necesita para que el sistema siga trabajando. Si la página por eliminar de la memoria fue modificada, se debe volver a escribir al disco para mantener actualizada la información; y si la página no fue modificada, la página que se carga se sobrescribe sobre la página a borrar. Para elegir la página a eliminar de la memoria el sistema operativo utiliza los algoritmos de reemplazo de página.

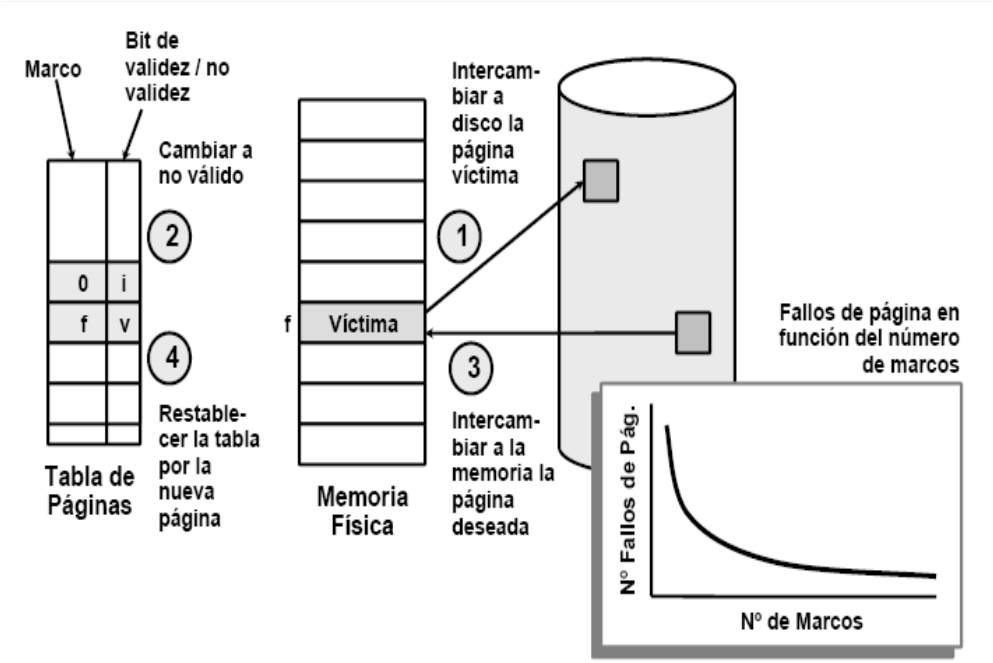


Figura 1.7 Proceso de intercambio de páginas.

1.4.1 ALGORITMO ALEATORIO

Este algoritmo consiste simplemente en reemplazar aleatoriamente cualquier página de la memoria principal, sin hacer ningún esfuerzo de predicción. Es el algoritmo más sencillo pues no se necesita ninguna información, sin embargo no puede lograr un buen desempeño.

1.4.2 ALGORITMO DE REEMPLAZO DE PÁGINAS ÓPTIMO

Este algoritmo debe de tener el menor índice de fallos de página de todos los algoritmos. En teoría, este algoritmo debe de reemplazar la página que no va a ser usada por el periodo más largo de tiempo. Desafortunadamente, el algoritmo de reemplazo óptimo es fácil en teoría, pero es imposible de implementar, pues requiere conocer las futuras necesidades del sistema.

Tal algoritmo existe y ha sido llamado OPT o MIN, pero se usa únicamente para estudios de comparaciones. Por ejemplo, puede resultar muy útil saber que aunque algún nuevo algoritmo no sea óptimo, está entre el 12.3% del óptimo y entre el 4.7% en promedio.

1.4.3 ALGORITMO DE REEMPLAZO "PRIMERO EN ENTRAR, PRIMERO EN SALIR" (FIFO)

Es el algoritmo más sencillo para el remplazo de páginas, consiste en asociar a cada página el momento en que ésta entra en la memoria y cuando una página debe ser reemplazada se selecciona la que más tiempo lleva cargada, se puede implementar en forma de cola a la que se le van agregando las páginas conforme van llegando y al momento de eliminar una página, se selecciona la que está al frente de la lista y la que entra se coloca al final [10]. Figura1.8

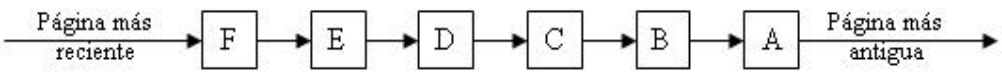


Figura 1.8 Reemplazo de páginas según FIFO

Este algoritmo es fácil de comprender pero no es eficiente pues la página reemplazada puede tener un uso muy frecuente y puede que se tenga necesidad de ella en un corto periodo de tiempo.

1.4.4 ALGORITMO DE REEMPLAZO DE PÁGINAS DE SEGUNDA OPORTUNIDAD

Este algoritmo es una modificación del FIFO; en el se hace uso del bit de referencia de la página y cuando la página va a ser reemplazada se revisa el bit de referencia. Si tiene valor de 0 se produce el

reemplazo y si por el contrario, el bit de referencia es 1 se le da a la página una segunda oportunidad y se le cambia el bit de referencia a 0 [10]. Figura1.9.

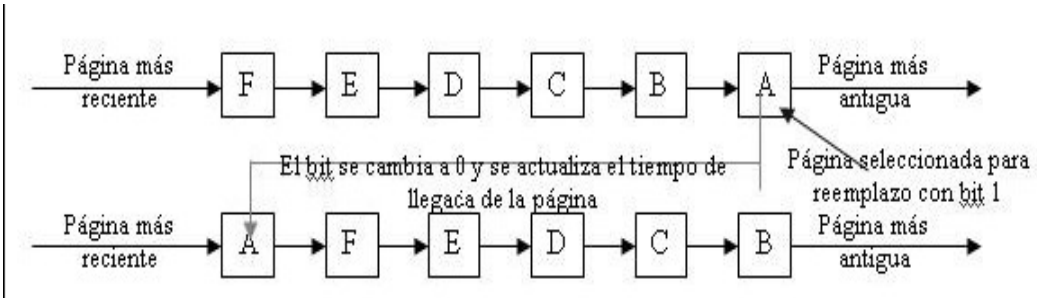


Figura 1.9 Reemplazo de páginas según el algoritmo de Segunda Oportunidad

Luego se actualiza su tiempo de llegada y es colocada al final de la cola. De esta manera, la página espera todo un ciclo completo de páginas para ser entonces reemplazada. Entonces si la página tiene un uso muy frecuente, el bit de referencia se mantendría constantemente en 1y la página no sería reemplazada [11].

1.4.5 ALGORITMO DE REEMPLAZO DE PÁGINAS DEL RELOJ

Este algoritmo se implementa modificando el algoritmo de la segunda oportunidad, en él se organizan las páginas en una lista circular; se usa un apuntador (o manecilla) que señala la página más antigua [10].

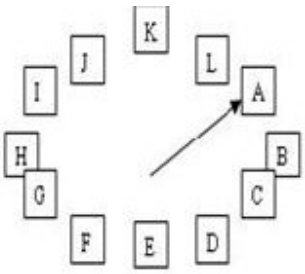


Figura 1.10 Algoritmo de páginas de reloj

1.4.6 ALGORITMO DE REEMPLAZO DE PÁGINAS “LA DE MENOR USO RECIENTE” (LRU)

Este algoritmo se basa en la observación de las páginas que tienen un uso más frecuente en las últimas instrucciones y que se puedan utilizar con cierta probabilidad en las siguientes. De la misma manera, es probable que las páginas que no han sido utilizadas permanezcan sin uso algún tiempo. Entonces al ocurrir un fallo de página, se elimina la página que más tiempo tenga sin ser utilizada. El LRU tiene un mejor rendimiento en el tiempo de aprovechamiento del CPU y del uso de la memoria. Sin embargo, la implementación de este algoritmo es muy cara, ya que requiere de una asistencia de hardware. Otro problema es el de determinar un orden para los marcos definido por el tiempo de menor uso. Para lo cual hay dos posibles implementaciones:

Contador: Es el caso más sencillo, se asocia cada entrada tabla-página un campo de tiempo-de-uso y se le agrega al CPU un reloj lógico o contador. Este reloj es incrementado en cada referencia de memoria. Siempre que se hace referencia a una página, el contenido del registro del reloj es copiado al campo de tiempo-de-uso en la tabla de páginas para esa página. De esta forma, siempre se dispone del “tiempo” de la última referencia a cada página. La página que se reemplaza es la del menor valor de tiempo. Este esquema requiere de una búsqueda en toda la tabla de páginas para encontrar la página LRU, y una escritura en memoria al campo de tiempo-de-uso en la tabla de páginas por cada acceso a memoria. Los tiempos también se deben de mantener cuando las tablas de páginas son alteradas.

Pilas: Otra aproximación para implementar el reemplazo LRU es la de tener una pila con los números de páginas. Siempre que se hace referencia a una página, se quita de la pila y se pone en la parte superior. De esta manera, la parte superior de la pila es la página de uso más reciente y la de abajo es la LRU [10]. Ver Figura 1.11

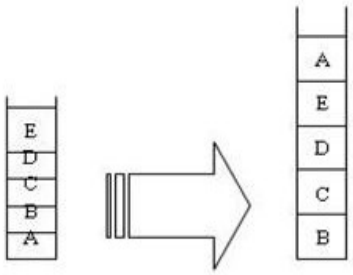


Figura 1.11 Uso de pilas para el reemplazo de página según LRU

1.4.7 ALGORITMO DE REEMPLAZO DE PÁGINAS SEGÚN EL USO NO TAN RECIENTE. (NRU)

Este algoritmo hace uso de los dos bits de estado que están asociados a cada página. Estos bits son: R, el cual se activa cuando se hace referencia (lectura /escritura) a la página asociada; y M, que se activa cuando la página asociada es modificada (escritura). Estos bits deben de ser actualizado cada vez que se haga referencia a la memoria, por esto es de suma importancia que sean activados por el hardware. Una vez activado el bit, permanece en ese estado hasta que el sistema operativo, mediante software, modifica su estado.

Estos bits pueden ser utilizados para desarrollar un algoritmo de reemplazo que cuando inicie el proceso, el sistema operativo asigne un valor de 0 a ambos bits en todas las páginas. En cada interrupción de reloj, se limpia el bit R para distinguir cuáles páginas tuvieron referencia y cuáles no. Cuando ocurre un fallo de página, el sistema operativo revisa ambos bits en todas las páginas y las clasifica de la siguiente manera:

- Clase 0: La página no ha sido referenciada, ni modificada.
- Clase 1: La página no ha sido referenciada, pero ha sido modificada.
- Clase 2: La página ha sido referenciada, pero no ha sido modificada.
- Clase 3: La página ha sido referenciada y también modificada.

Después se elimina una página de manera aleatoria de la primera clase no vacía con el número más pequeño. Para el algoritmo es mejor eliminar una página modificada sin referencias en al menos un intervalo de reloj, que una página en blanco de uso frecuente [10].

1.4.8 COMPARACIÓN ENTRE LOS ALGORITMOS DE REEMPLAZO DE PÁGINA.

Este es un ejemplo de la comparación entre 4 de los algoritmos utilizados donde se supone que el numero de marcos asignados a un proceso es fijo, usando páginas 256 palabras, se realizaron las pruebas con 6, 8,10, 12 y 14 marcos, donde las diferencias son mas marcadas con asignaciones pequeñas, también se observa que con el algoritmo FIFO es casi 2 veces peor que el óptimo. Este experimento, fue realizado por J.Bear en 1980 [11].

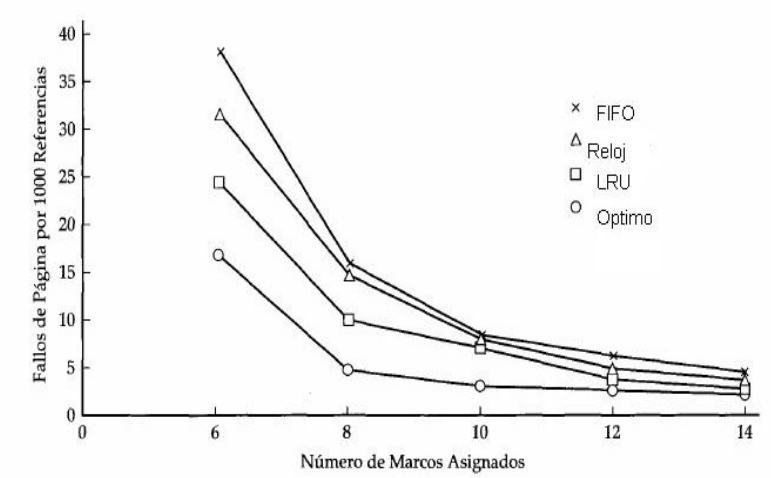


Figura 1.12 Comparación entre 4 de los algoritmos.

1.5 SIMULACIÓN DE LOS ALGORITMOS DE ADMINISTRACIÓN DE MEMORIA

Los algoritmos de administración de memoria son fundamentales para obtener un mejor de los uso de los recursos por el sistema operativo; por lo que crear herramientas de simulación para su estudio y análisis posibilita el avance de la gestión de memoria. También favorece el continuo desarrollo de los sistemas informáticos por que cada día el tamaño de la memoria aumenta y los equipos son más complejos, por esto se necesitan aplicaciones que ayuden a tener una visión más clara del funcionamiento de estas técnicas, lo que permite crear futuros investigadores mas capacitados y mejores preparados [12].

Existen diversas maneras de implementar aplicaciones como esta, pues se pueden utilizar estructuras de datos (TDA por sus siglas en inglés) como listas y arboles; por ejemplo, a través de la utilización de colas para algoritmos como FIFO y pilas para el LRU. Para la implementación del algoritmo LRU se utiliza un contador adicional, también una pila para llevar cuenta de qué páginas llevan más tiempo sin ser utilizadas, a su vez se puede utilizar las listas enlazadas para la tabla de páginas y la tabla de marcos, del mismo modo se debe desarrollar una aplicación con una interfaz amigable, dinámica y flexible que permita apreciar los cambios que ocurren en los distintos algoritmos de reemplazo de página [10].

1.5.1 CONSIDERACIONES SOBRE LOS SIMULADORES EXISTENTES

Se ha implementado aplicaciones donde se representan los diversos temas que comprende la administración de memoria; fundamentalmente con fines docentes, una mayoría de estas tratan temas como la memoria Cache, sin embargo a su vez existen herramientas que reproducen el comportamiento de estos sistema de administración de memoria bajo determinadas condiciones, las cuales se conocen actualmente como simuladores; estos se basan en los algoritmos de reemplazo existentes pero poseen la desventaja de no tener una interfaz gráfica flexible y compresible para los estudiantes, del mismo modo son implementados de manera que resulta muy difícil de interactuar con la aplicación.

1.5.2 MOSS (MEMORY MANAGEMENT SIMULATOR)

Esta aplicación muestra el comportamiento de los fallos de página, el programa lee el estado inicial de la tabla de páginas y una secuencia de instrucciones de la memoria virtual y escribe una traza de registro que indica el efecto de cada instrucción, incluye una interfaz gráfica de usuario de modo que se pueda observar los algoritmos de sustitución de la página. Se implementó en Java Development Kit (JDK) 1.0 .Este tiene versiones que son ejecutadas tanto en el SO Windows como en GNU/Linux. Este programa se comercializa bajo los términos del Software Libre y sin embargo no se garantiza su utilización por que puede presentar problemas de instalación. Fue implementado por Alex Reeder [13].

1.5.3 VISUALOS

Es una herramienta gráfica que permita el estudio y comprensión de como funciona un sistema operativo moderno, con sus aspectos más relevantes, así como permite ver cómo se relacionan tres de sistemas más importantes: planificación de procesos, gestión de memoria y Entrada/Salida. Esta se implementó con fines docentes. Para desarrollarlo se han utilizado el sistema documentación: **SGML** (DocBook v3.1)/gtk-doc-tools v0.3, también para la programación se utilizó GNOME v1.2.4/GTK+ v1.2.8/Windows v3.3.6. La aplicación tiene un código modular lo que le permite hacer cambios sin necesidad de tener en cuenta todo el contexto y le es muy fácil añadir nuevos algoritmos. Para este simulador existen versiones que pueden ser ejecutadas tanto para el Sistema Operativo Windows como para Linux. Entre sus ventajas se encuentran la facilidad que brinda al profesor en su trabajo con las gráficas y otras representaciones que expliquen algunos de los conceptos importantes, y permite que el alumno los vea más claramente; sin embargo el sistema es poco accesible y difícil de trabajar por la cantidad de ventanas que se despliegan, lo que hace complejo su configuración y manejo [14].

1.5.4 PRÁCTICA 9: GESTÍON DE MEMORIA

El objetivo de este simulador desarrollado en Java es mostrar algunos aspectos del funcionamiento de la memoria virtual, en el que intervienen uno o más procesos en un entorno de multiprogramación. Estos procesos generan durante su ejecución accesos a direcciones lógicas que serán traducidos a direcciones en memoria física. El esquema de traducción de direcciones se basa en una técnica de paginación combinada con algoritmos de reemplazo, lo que se conoce globalmente como memoria virtual mediante paginación bajo demanda. Ha sido desarrollado con motivo de un proyecto final de carrera realizado en la Escuela Universitaria de Informática (UPV). Puede ejecutado en algunas versiones de Windows [15].

1.5.5 SIMULADOR DE LA GESTIÓN DE MEMORIA

Se implementó en Turbo C mediante la paginación para un sistema en el que la memoria virtual consta de ocho módulos de RAM entrelazada y una pequeña cache, describiendo el modo en que permite solucionar, de forma transparente al programador, los problemas de espacio, aprovechamiento y uso en multiprogramación de la memoria principal. Esta aplicación tiene la desventaja la brindar escasos

resultados luego de la simulación entre los que se encuentran la serie de direcciones físicas calculadas por el simulador y la indicación de la cantidad de fallos de páginas producidos [16].

1.5.6 PROYECTO #1 SIMULACIÓN DE MANEJO DE MEMORIA

Este proyecto consiste en un simulador programado para investigar las diferencias relativas de los algoritmos de asignación de memoria y las políticas de reemplazo de páginas (FIFO, LRU, LFU y OPT). El programa debe generar estadísticas apropiadas, fue implementado en Java y desarrollado como proyecto de fin de semestre por la Universidad Tecnológica Centroamericana. Solo se ha implementado en este proyecto, la generación bit de referencia a memoria, y el manejador de memoria contigua que simula la asignación de memoria en sistemas operativos no paginados [17].

1.6 TENDENCIAS Y TECNOLOGÍAS ACTUALES

Para lograr el objetivo de esta investigación se hace necesario el uso de una **metodología** que guíe el proceso de desarrollo así como de herramientas y lenguajes que permitan el diseño de la misma, de forma que se cumplan las tareas antes propuestas.

1.6.1 METODOLOGÍA DE DESARROLLO.

El desarrollo de software es una tarea compleja; por lo que existen numerosas propuestas metodológicas que inciden en los proceso de desarrollo. Por un lado se encuentran las propuestas tradicionales que se centran en el control del proceso, estableciendo rigurosamente las actividades involucradas, los artefactos que se deben producir, las herramientas y notaciones que se usarán. Estas propuestas han demostrado ser efectivas y necesarias en proyectos muy grandes formados por muchos integrantes, donde los requisitos sean bien específicos, sin embargo en otros más sencillos han presentado problemas. En estas metodologías los clientes interactúan con el equipo de desarrollo mediante reuniones.

Precisamente utilizar el factor humano consiste la filosofía de las metodologías ágiles, estas dan un mayor valor a la colaboración con el cliente y al desarrollo incremental del software con iteraciones muy cortas. Este enfoque se usa en proyectos con requisitos muy cambiantes, manteniendo una alta calidad en el producto. Estas metodologías están revolucionando la manera de producir software en el mundo, pues

aunque muchos están en desacuerdo cada día aumentan el número de personas que las utilizan [18]. En la siguiente tabla se muestran las diferencias entre estos distintos procesos de desarrollo.

Metodologías Ágiles	Metodologías Tradicionales
Basadas en heurísticas provenientes de prácticas de producción de código	Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo
Especialmente preparados para cambios durante el proyecto	Cierta resistencia a los cambios
Impuestas internamente (por el equipo de desarrollo)	Impuestas externamente
Proceso menos controlado, con pocos principios	Proceso mucho más controlado, con numerosas políticas/normas
No existe contrato tradicional o al menos es bastante flexible	Existe un contrato prefijado
El cliente es parte del equipo de desarrollo	El cliente interactúa con el equipo de desarrollo mediante reuniones
Grupos pequeños (<10 integrantes) y trabajando en el mismo sitio	Grupos grandes y posiblemente distribuidos
Pocos artefactos	Más artefactos
Pocos roles	Más roles
Menos énfasis en la arquitectura del software	La arquitectura del software es esencial y se expresa mediante modelos

Tabla 1.Comparación entre las metodologías ágiles y las metodologías tradicionales [18].

1.6.2 METODOLOGÍA TRADICIONAL

Proceso Unificado de Desarrollo de software (RUP).

Es una metodología basada en un grupo de principios claves: el equipo de un proyecto de software debe planificar el desarrollo; debe conocer hacia donde se dirige; debe documentar el proyecto de una manera perdurable y extensible, está definido por cuatro características fundamentales:

- Dirigido por casos de uso.
- Centrado en la arquitectura.
- Iterativo e incremental.
- Controlado.

Estas características van a permitir una planificación ajustada al avance que vaya presentando el producto, además se irá verificando que las funcionalidades que van siendo implementadas en el software sean exactamente las que desea el usuario. El Proceso Unificado del desarrollo de software aporta además un enfoque disciplinado a la asignación de tareas y responsabilidades en proyectos. También permite dirigir las tareas de desarrolladores individuales y equipos de trabajo como una sola, incluso ofrece criterios para monitorear y medir los productos y actividades del proyecto. Una característica importante es que permite corregir errores en cada iteración y es flexible a cambios en los requerimientos [19].

1.6.3 METODOLOGÍA ÁGIL

Varias metodologías se ajustan al nombre de ágil. Todas ellas comparten muchas características, también hay algunas diferencias significativas. Entre las más usadas se encuentran eXtreming Porgraming, Agile Modeling, y Scrum, existen además Crystal Methodologies, Dynamic Systems Development Method (DSDM), Adaptive Software Development (ASD), entre otras [20].

eXtreme Programing (XP)

Es una metodología muy difundida entre los desarrolladores de software, se nutre de la comunicación directa entre las personas, se basa en la simplicidad, la comunicación y el reciclado continuo de código. Los objetivos de XP son muy simples: la satisfacción del cliente, se trata de dar al cliente el software que él necesita y cuando lo necesita, el segundo objetivo es potenciar al máximo el trabajo en grupo. Actualmente, XP es el método ágil más documentado [21].

Ventajas de XP

- Puede ser implementado en forma parcial (elegir sólo algunas de las prácticas).
- Puede ser implementado en forma gradual.
- Puede adaptarse a las necesidades de cualquier equipo de desarrollo.
- Exige que se establezca una comunicación más fluida con el cliente y que este tenga mayor participación en el proceso de desarrollo. La consecuencia de esto es que el cliente se involucre más en el desarrollo del producto.
- Se realizan pruebas constantemente del sistema [21].

Es una metodología flexible; basada en la retroalimentación del cliente en los requerimientos del sistema en cualquier momento del desarrollo. Desde el punto de vista de la ingeniería es importante detenerse en el desarrollo de software con el fin de realizarlo de forma correcta y entendible; este proceso comprende en su ciclo de vida las siguientes fases: Planificación, Diseño, Implementación y Pruebas [22].

Agile Modeling (AM)

Es una práctica basada en la metodología para modelado efectivo de sistemas de software. Consta de un conjunto de valores, principios y prácticas para el modelado, estas se pueden aplicar en un proyecto de desarrollo de software en forma efectiva y ligera. Puede adaptarse a otras metodologías de desarrollo como XP o RUP, lo que le permite desarrollar un proceso de software que realmente satisfaga sus necesidades. Las técnicas de AM, en particular Agile Modeling Driven Development, permite modelar situaciones muy complejas en un desarrollo ágil [23]. Uno de los objetivos de AM mostrar como aplicar las técnicas de modelado de software de los proyectos que tienen un enfoque ágil como XP y Scrum.

Esta filosofía recomienda para el modelado utilizar instrumentos sencillos, como pizarras y papel; pues las herramientas simples son fáciles de aprender, fácil de usar, y fácil de compartir con los demás, o algunas herramientas de modelado para aquellos modelos muy sofisticados por que tienen como desventajas el tiempo perdido en el estudio de la herramienta más el tiempo que se desperdicia, realizando los diagramas ajenos a los que necesita [24].

Scrum

Es un proceso ágil y liviano que sirve para administrar y controlar el desarrollo de software. Este desarrollo se realiza en forma iterativa e incremental donde cada ciclo o iteración termina con una pieza de software ejecutable que incorpora nuevas funcionalidades. Las iteraciones en general tienen una duración entre 2 y 4 semanas. Scrum se utiliza como marco para otras prácticas de ingeniería de software como RUP o Extreme Programming. Esta centrado en priorizar el trabajo en función del negocio, maximizando la utilidad de lo que se construye y el retorno de inversión. De la misma manera es diseñado para adaptarse a los cambios de los requerimientos [25].

Por las ventajas mencionadas anteriormente, se utilizan para el análisis y el diseño de esta investigación las metodologías ágiles; eXtreme Programing (XP), como guía para el proceso de software, Agile Modeling, en la realización del modelado y para la gestión de proyecto Scrum.

1.7 LENGUAJE DE MODELADO (UML)

El Proceso Unificado de Modelado (UML) es un lenguaje gráfico para visualizar y documentar los elementos de los sistemas orientados a objetos. Dicho lenguaje es una notación unificada con la que se permite lograr un entendimiento que facilite el intercambio entre los usuarios y los desarrolladores. Se ha convertido en un estándar de la industria del software, debido a que ha sido impulsado por los autores de los tres métodos más usados de orientación a objetos Grady Booch, Ivar Jacobson y Jim Rumbaugh. El UML estándar está compuesto por tres partes: bloques de construcción, relaciones entre los bloques y diagramas [26].

1.7.1 HERRAMIENTAS CASE

Rational Rose

Rational Rose es una herramienta para el modelado visual de sistemas mediante UML. Permite especificar, analizar y diseñar el sistema antes de codificarlo. Se desarrolló por los creadores de UML (Booch, Rumbaugh y Jacobson), esta cubre todo el ciclo de vida de un proyecto: concepción y formalización del modelo, construcción de los componentes, transición a los usuarios y certificación de las distintas fases. Entre las características Rational Rose se encuentran:

- "Mantiene la consistencia de los modelos del sistema software".
- Chequeo de la sintaxis UML.
- Generación documentación automáticamente.
- Generación de código a partir de los modelos.
- Ingeniería Inversa (crear modelo a partir código) [19].

Visual Paradigm

Es una potente herramienta **CASE** que permite visualizar y diseñar elementos de software, para ello utiliza el lenguaje UML, proporciona a los desarrolladores una plataforma que les permita diseñar un producto con calidad de una forma rápida. Facilita la **interoperabilidad** con otras herramientas CASE como el Rational Rose y se integra con las siguientes herramientas Java: Eclipse/IBM WebSphere, Jbuilder, NetBeans IDE, Oracle Jdeveloper, BEA Weblogic. Está disponible en varias ediciones: Enterprise, Professional, Community, Standard, Modeler y Personal. [25] Genera código y realiza ingeniería inversa para diez lenguajes de programación, donde los fundamentales son, Java, C++, CORBA IDL, PHP, XML Schema y ADA. En adición se genera código para C#, Visual Basic.net, Object Definition Lenguage (ODL), Flasch Action Script, Delphi, Perl y Phyton. Se integra con el Visio para importar imágenes del mismo para realizar los diagramas de despliegue. Genera documentación para el proyecto en HTML, MS Word y PDF. Además exporta e importa los diagramas en el estándar XML y como imágenes (ya sea con extensiones jpg o png). Esta herramienta es soportada por varios sistemas operativos [27].

¿Por que Visual Paradigm?

Se decide utilizar Visual Paradigm para visualizar y diseñar los elementos de software porque es multiplataforma de manera que se ajusta a las exigencias de la facultad. De igual forma por las facilidades que brinda para el diseño de los diagramas necesarios y su documentación. Aunque la filosofía AM no recomienda el uso de herramientas CASE para el modelado, esta apoya el análisis de los requisitos para el diseño de código, prueba la coherencia y la validez de sus modelos además se obtienen diferentes puntos de vista y posibles soluciones a un problema. De igual manera se conoce con anterioridad esta herramienta por lo que el consumo del tiempo en el aprendizaje seria mínimo [27].

1.8 LENGUAJES DE PROGRAMACIÓN

C#

Es un lenguaje de programación diseñado para crear una amplia gama de aplicaciones que se ejecutan en .**NET Framework**. C# es simple, eficaz, con seguridad de tipos y orientado a objetos. Con sus diversas innovaciones, permite desarrollar aplicaciones rápidamente, manteniendo la expresividad y elegancia de los lenguajes de tipo C [28] .

Visual Studio admite Visual C# con un editor de código completo, plantillas de proyecto, diseñadores, asistentes para código, un depurador eficaz y fácil de usar, además de otras herramientas. La biblioteca de clases .NET Framework ofrece acceso a una amplia gama de servicios de sistema operativo y a otras clases útiles y adecuadamente diseñadas que aceleran el ciclo de desarrollo de manera significativa [28].

Las características más importantes de C# son:

- Herencia múltiple
- Sobrecarga de operadores y funciones
- Derivación
- Funciones virtuales
- Plantillas
- Gestión de excepciones

Java

Java supone un significativo avance en el mundo de los entornos software, y esto viene dado por tres elementos claves que diferencian este lenguaje desde un punto de vista tecnológico:

- Es un lenguaje de programación que ofrece la potencia del diseño orientado a objetos con una sintaxis fácilmente accesible con un entorno robusto e interesante.
- Proporciona un conjunto de clases potente y flexible.
- Pone al alcance de cualquiera la utilización de aplicaciones que se pueden incluir directamente en páginas Web [29].

Cuenta con determinadas características como:

- Orientación a objetos
- Riqueza semántica
- Robusto
- Fácil aprendizaje
- Interactivo y animado
- Multiplataforma [30].

¿Por qué Java?

Se decide utilizar Java por que es un lenguaje sencillo y a la vez potente, con diversas funcionalidades y características que lo hacen muy popular, es multiplataforma, donde se pueden crear aplicaciones independientes o **applets**, que se puedan integrar a otros componentes, además de que se pueden crear sistemas robustos de manera interactivas. De la misma manera incorpora facilidades para la creación y manipulación de gráficos. También tiene un modelo de objetos más simple y los programas se construyen a partir de módulos independientes, estos módulos se pueden transformar o ampliar fácilmente. Lo cual es muy útil para desarrollar la herramienta propuesta a modo aplicación de escritorio pues Java como lenguaje multiplataforma es idea para poder ejecutarla en varios sistemas operativos. Este lenguaje aplica todos los paradigmas de la Programación Orientada a Objetos [30].

1.9 ENTORNO INTEGRADO DE DESARROLLO (IDE)

NetBeans

NetBeans es un entorno de desarrollo o **IDE** (Integrated Development Environment). Tiene editor de código sensible al contenido con soporte para autocompletar el código, con muchas de etiquetas, autotabulación y uso de abreviaturas para varios lenguajes de programación [31].

Incluye además:

- Soporte para Java, C, C++, **XML** y **lenguajes HTML**.
- Soporte para **JSP**, XML, **RMI**, **CORBA**, **JINI**, **JDBC** y **tecnologías Servlet**
- Incluye CVS (control de versiones) y compilación avanzada
- Posibilidad de utilizar otras versiones de compiladores, depuradores,...
- Creación visual de componentes gráficos
- Herramientas con asistentes para facilitar la escritura de código
- Dispone de todo un entorno para crear documentación javadoc.

El IDE NetBeans es un producto gratuito sin restricciones de uso y pensado para escribir, compilar, depurar y ejecutar programas. Soporta el desarrollo de todos los tipos de aplicación en Java [31].

Empresas independientes, asociadas, especializadas en desarrollo de software, proporcionan extensiones adicionales que se integran fácilmente en la plataforma y que pueden también utilizarse para desarrollar sus propias herramientas y soluciones. Entre las características de la plataforma están:

- Administración de las interfaces de usuario.
- Administración de las configuraciones del usuario.
- Administración del almacenamiento.
- Administración de ventanas.
- Plataforma basada en asistentes [31].

Eclipse

El entorno integrado de desarrollo (IDE) de Eclipse emplea módulos (**plugin**) para proporcionar su funcionalidad. Estos módulos le permiten a Eclipse extenderse usando otros lenguajes de programación como son C/C++ y Phyton, lo que le da la posibilidad de trabajar con lenguajes para procesado de texto. La arquitectura plugin permite escribir cualquier extensión deseada en el ambiente. Se provee soporte para Java y CVS, es el **SDK** de Eclipse. Puede soportar otros lenguajes de programación.

De la misma manera incluye las herramientas de desarrollo de Java, ofreciendo un IDE con un compilador de Java interno y un modelo completo de los archivos fuente de Java. Esto permite técnicas avanzadas de **refactorización** y análisis de código. El IDE también hace uso de un espacio de trabajo, permitiendo modificaciones externas a los archivos en tanto se actualice el espacio de trabajo correspondiente [32].

La versión actual de Eclipse dispone de las siguientes características:

- Editor de texto.
- Resaltado de sintaxis.
- Compilación en tiempo real.
- Pruebas unitarias con JUnit.
- Control de versiones con CVS.
- Compilación avanzada.
- Asistentes para creación de proyectos, clases, tests, etc.
- Refactorización [32].

¿Por qué NetBeans 6.0?

Las herramienta Eclipse y Netbeans son muy similares sin embargo sus características los hacen únicos, se determinó utilizar NetBeans porque posee una interfaz gráfica robusta y consume menos recursos en el proceso de compilación. El este IDE proporciona un completador de código eficiente, además ofrece servicios comunes a las aplicaciones de escritorio, permite enfocarse en la lógica específica de la aplicación. Se basa en el control de versiones y refactoring. Tiene una base modular y extensible que

puede ser usada como una estructura de integración para crear aplicaciones de escritorio grandes. Está compuesto por varios frameworks en el que se destaca **Swings**, que es utilizado en aplicaciones de gran tamaño, además proporciona utilidades para facilitar la creación de aplicaciones gráficas, del mismo modo se apoya sobre **AWT** y añade JComponets, utiliza también componentes ligeros que le proporcionan facilidades a la hora de trabajar con este IDE [33].

1.10 CONCLUSIONES PARCIALES

En este capítulo se definieron conceptos significativos relacionados con la investigación y se describieron tendencias y tecnologías que se utilizan en el análisis y diseño de la aplicación. Se presentaron las principales características de las metodologías y las herramientas a utilizar. Luego de dicho estudio se determinó aplicar la metodología de desarrollo XP; y como lenguaje de programación Java y NeatBeans como entorno de desarrollo.

CAPÍTULO # 2 ANALÍSIS DEL SISTEMA

2.1 INTRODUCCIÓN

En este capítulo se describe el problema y la situación problémica; se realiza un análisis de los procesos que se desean implementar, del mismo modo se realiza la propuesta del sistema y se efectúa la planificación del proyecto junto al análisis de las historias de usuario, clasificándolas en función del riesgo y el tiempo que llevará implementar cada historia. Se describe el plan de entrega, así como las iteraciones que se llevarán a cabo.

2.2 PROBLEMA Y SITUACIÓN PROBLÉMICA

En la actualidad la mayoría de las universidades de nuestro país imparten carreras técnicas, uno de estos centros de estudios es la Universidad de Ciencias Informáticas. Esta, en su programa de estudio incluye a la asignatura de Sistemas Operativos donde unos de los temas fundamentales a impartir es la Administración de Memoria; en la que se hace imprescindible la asimilación de una gran cantidad de información y conocimientos. Esta asignatura no cuenta con herramientas didácticas que apoyen la actividad docente y ayuden al estudiantado comprender los conocimientos que recibe, a su vez los alumnos no pueden apropiarse del contenido en su totalidad.

En las universidades los profesores que imparten está asignatura poseen escasos medios didácticos y en temas como este los estudiantes dependen mucho de su imaginación, además aunque en las clases se cuente con el apoyo de una presentación la información que reciben no es vista desde varios sentidos por los alumnos, por consiguiente no permite un alto rendimiento escolar, lo que es motivo de preocupación para profesores y estudiantes.

Una solución sencilla y práctica consiste en la realización de una aplicación que constituya un apoyo a los profesores que imparten sus clases y que logre simular las técnicas utilizadas de memoria virtual para realizar el intercambio de páginas, así como para facilitar al estudiante la asimilación del contenido de que recibe.

2.3 ALCANCE DEL SISTEMA

Con el desarrollo de la investigación se propone realizar el análisis y el diseño de una aplicación que permita simular el comportamiento de la memoria virtual usando para su implementación el esquema de paginación simple con algoritmos de sustitución de páginas. Esta herramienta debe dar la posibilidad de ilustrar de manera gráfica el funcionamiento del mecanismo teniendo presente los componentes de: TLB, dirección lógica, tabla de páginas y memoria física. Debe contar con un sistema de generación de datos que serían las direcciones lógicas, las cuales tendrán un tamaño fijo de 16 bytes; debe dar la posibilidad de entrar datos manualmente en valores binarios (1,0). Las páginas se cargarán bajo demanda y la simulación se centrará en los algoritmos de reemplazo de página. El sistema se ejecutará para un solo proceso por lo que el reemplazo de páginas será local, y en consecuencia incluye la utilización de una sola tabla de página, en la cual las páginas se cargarán aleatoriamente. Luego de terminar las simulaciones se obtienen los resultados e indicadores que determinan la eficiencia del algoritmo utilizado, de la misma manera se pueden utilizar estos para realizar las comparaciones. No se incluye los mecanismos de pre-paginación, tabla multinivel así como la implementación de la memoria virtual mediante el esquema de segmentación o el esquema combinado, por no incluirse en el plan de estudio de la asignatura.

2.4 PROCESOS A AUTOMATIZAR

Para el desarrollo del simulador de memoria virtual anteriormente descrito es necesario automatizar el comportamiento de los algoritmos de administración de memoria usando la técnica de paginación, lo que incluye los algoritmos de reemplazo de página conocidos como FIFO, LRU, Aleatorio, Reloj, Segundo Chance, NRU, Óptimo, además de permitir o no el uso de la TLB, de la misma manera debe brindar la posibilidad de ilustrar de manera gráfica el funcionamientos de los mismos. Del mismo modo se desea que esta aplicación pueda realizar las traducciones de direcciones virtuales a direcciones físicas.

Teniendo en cuenta estos argumentos, se propone realizar el análisis y diseño de una aplicación de simulación como solución a los problemas planteados, es decir se plantea como propuesta del sistema una herramienta que permitirá reproducir el comportamiento de algoritmos de administración de memoria

bajo ciertas condiciones además brindar a los estudiantes la posibilidad de comprender de una forma más práctica como el Sistema Operativo gestiona unos de los recursos más importantes; la memoria.

En la aplicación se deben incluir los parámetros generales de simulación; donde se muestren las opciones que se implementen en el sistema, el cual tendrá un método de generación arbitraria de datos o direcciones lógicas, así como la opción al usuario de entrar manualmente los datos.

El sistema está compuesto por 8 módulos; uno para cada algoritmo de reemplazo de página y otro compuesto por el programa principal. Lo que aporta la ventaja de que exista la posibilidad de independencia para los algoritmos 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 que 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 (algoritmos de reemplazo), sin que uno afecte al otro. De la igual manera por la estructuración y organización que brinda sirve de apoyo al diseño de la arquitectura.

2.5 PLANIFICACIÓN DEL DESARROLLO DE LA PROPUESTA

Para realizar el diseño del simulador descrito se utiliza la metodología eXtreming Programing dentro de esta hay diversas prácticas en cada uno de los ciclos de desarrollo de un proyecto conocidas como fases las cuales son: Planificación, Diseño, Desarrollo, Prueba; la primera de estas la Planificación comienza en esta sección. En esta fase se desarrolla un diálogo con el cliente, en el que se escriben las historias de usuarios, se decide el alcance, la prioridad, organización de las iteraciones y la fecha de las mismas [34].

Uno de los principales artefactos del ciclo de vida de XP son las historias de usuarios (HU). Estas guían todo el proceso de desarrollo pues son utilizadas para especificar los requisitos del software. Son escritas por los propios clientes, tal y como ven ellos las necesidades del sistema [35]. Las historias de usuario

solamente proporcionarán los detalles sobre cuánto tiempo llevará la implementación de dicha historia de usuario. Este nivel de detalle debe ser mínimo [36].

2.6 HISTORIAS DE USUARIO

Para diseño de la herramienta propuesta y su futura implementación se tienen en cuenta los procesos fundamentales que necesitan ser implementados, con el objetivo de que los estudiantes puedan observar las transformaciones que ocurren en la memoria al utilizar un algoritmo de reemplazo de página. Luego de conversar con el cliente se realiza la redacción de las historias de usuario. Teniendo en cuentas las necesidades del estudiantado se redactaron tres historias de usuario que guarán el desarrollo del sistema [34].

La primera historia de usuario se llama “Introducción de datos al sistema”, describe como se entran las direcciones virtuales y los datos específicos que definen las características de cada simulación.

Historia de Usuario	
Fecha:7/3/2008	
Número: 1	**Dependencia**:-------
Nombre historia: Introducción de datos al sistema	
Prioridad en negocio: Alta Alta /media /baja	**Riesgo en desarrollo**: Alto Alta/medio/baja
Descripción: Se introducen los datos (direcciones virtuales) de forma manual o los genera la aplicación, 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 2.1 Historia de Usuario “Introducción de datos al sistema”.

La segunda historia se llama “Simulación de Memoria Virtual”. El usuario podrá realizar la traducción de las direcciones virtuales y seleccionar la opción de utilizar o no la TLB, así como podrá observar las

transformaciones que ocurren en la memoria física. En las observaciones se aclara que la gestión de las páginas se realiza utilizando la paginación simple.

Historia de Usuario	
Fecha:7/3/2008	
Número: 2	**Dependencia**:1
Nombre historia: ”Simulación de Memoria Virtual”	
Prioridad en negocio: Alta Alta /media /baja	**Riesgo en desarrollo**: Alto Alta/medio/baja
Descripción: El usuario podrá realizar la traducción de las direcciones virtuales y seleccionar la opción de utilizar o no la TLB, así como podrá observar las transformaciones que ocurren en la memoria física y las estadísticas que se producen en cada simulación. Se utilizan botones que determinan la velocidad de la simulación.	
Observaciones: La gestión de páginas se realizará mediante el esquema de paginación simple.	

Tabla 2.2 Historia de Usuario “Generación de algoritmos de reemplazo de páginas”.

La tercera de estas historias de usuario es la “Gestión de los Resultados”, la cual gestiona los resultados de la simulación así como los indicadores que muestra la eficiencia de cada algoritmo, además de establecer las comparaciones entre las simulaciones y generar un informe final sobre las simulaciones generadas.

Historia de Usuario	
Fecha:7/3/2008	Versión:1.1
Número: 3	**Dependencia**: 3
Nombre historia: Gestión de Resultados	
Prioridad en negocio: **Media**	Riesgo en desarrollo: **Medio**

Alta/Medio/Baja	Alta/Medio/Baja
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 2.3 Historia de Usuario "Gestión de Resultados".

2.7 PLANIFICACIÓN DE LAS HISTORIAS DE USUARIO (PLAN DE ENTREGAS)

Para realizar una correcta implementación de estas historias de usuarios escritas es necesario planificar el tiempo justo que se le dedicará a cada una de estas y el orden en que serán codificadas, para obtener la herramienta en el menor tiempo posible. Esta tarea debe hacerse con eficiencia pues no se puede descuidar la calidad requerida del simulador.

Por lo tanto para realizar una correcta planificación es necesario hacer una estimación de cuánto tiempo se necesita para implementar la historia de usuario sin otras distracciones. Como resultado se debe obtener un periodo ideal de una a tres semanas [37]. En la planificación se define cuales HU son más importantes para el cliente, lo que permite saber en que **iteración** debe ser desarrollada la historia, y así elaborar el plan de entrega estimado según el tiempo ideal [38].

No	Nombre de HU	Prioridad	Riesgo	Esfuerzo	Iteración
1	Introducción de datos al sistema	Alta	Alto	3	1
2	Simulación de Memoria Virtual	Alta	Alto	3	1
3	Gestión los resultados	Media	Medio	2	2

Tabla 2.3 Plan de Entrega

El plan de entrega estimado se usa para definir los objetivos que serán cumplidos en cada iteración, esto se concreta en función de dos parámetros: tiempo ideal y grado de importancia para el cliente. El objetivo

de la planificación consiste en seleccionar las historias de usuario que según el plan de entregas, corresponderán a cada una de las iteraciones individuales [37].

2.7.1 PLAN DE ENTREGA ESTIMADO

Por lo que se decide realizar:

Iteración 1: Se trata de tener preparadas las funcionalidades básicas de la aplicación, aquellas que constituyen imprescindibles para el cliente, lo que significa que los datos sean introducidos correctamente así como que el simulador reproduzca el comportamiento de la memoria virtual, incluyendo la imitación de los algoritmos de reemplazo de página donde se utilice para la gestión de estas la técnica de paginación. Las historias de usuario que se codifican en esta iteración son la historia número 1 “Introducción de datos al sistema” y la número 2 “Simulación de Memoria Virtual” .El tiempo que demoraría puede llegar a las tres semanas. Para evaluar la calidad de esta primera versión se diseñan las pruebas que evalúen correcta entrada de datos al sistema y la efectividad de los algoritmos de simulación.

Iteración 2: Siendo esta la última iteración se pretende entregar el producto acabado con todas las funcionalidades propuestas o sea un simulador que imite el comportamiento de la memoria virtual donde los resultados obtenidos puedan ser guardados y se puedan realizar comparaciones entre las simulaciones producidas. Se terminaría de codificar la última historia de usuario “Gestión los resultados” que tienen por número 3. El tiempo que necesitará esta será de aproximadamente dos semanas y para la evaluación de la calidad se diseñan pruebas acreditarán los resultados obtenidos luego haber realizado las simulaciones.

Al terminar este plan de entrega quedan definidas las historias que se deben desarrollar en cada iteración, el tiempo que tardarán estas en codificarse y cómo se evaluará la calidad del trabajo realizado [38].

2.7.2 MODELACIÓN DE LAS HISTORIAS DE USUARIO

Estas historias de usuarios rigen todo el proceso de desarrollo, incluyendo una descripción de sus características esenciales sin embargo no son lo suficientemente explicitas, pues las historias de usuario son escritas de la manera sencilla, por es necesario modelar el funcionamiento de las mismas para

realizar un análisis más profundo de los procesos que incluye la memoria virtual, de manera que permita obtener una idea más detallada de lo que se desea implementar [39].

Para el diseño de la herramienta que se propone en la investigación se utilizará para modelar las historias de usuarios la filosofía de Modelado Ágil a través del lenguaje de modelado UML, utilizando los diagramas de actividad y los modelos de datos.

Modelo de dominio

El modelado de datos permite explorar el comportamiento de los datos de un sistema, con este es posible identificar entidades y clases. Es posible utilizar estos para diversos fines, o diferentes tipos de modelos para un propósito similar, existen tres estilos de modelos de datos:

- El modelo lógico de datos, se utiliza para explorar los conceptos del dominio, puede describir la lógica de los diferentes tipos de entidades y rara vez se utiliza en los proyectos donde se usan las metodologías ágiles aunque estos modelos se encuentran en los proyectos que se guían por las metodologías tradicionales.
- El modelos de datos físicos que se utiliza para diseñar el esquema interno de una base de datos, el cual ha demostrado ser útiles en proyectos donde se empleen ambas metodologías.
- El modelo de datos conceptuales es el que se utilizará en esta investigación, estos modelos son conocido como modelos de dominio, se utilizan normalmente para explorar los conceptos del dominio de un proyecto y en equipos que trabajan con las metodologías ágiles estos modelos son creados a menudo como parte de sus necesidades iníciales [40].

Diagrama de Actividad

Un diagrama de actividad muestra una serie de acciones que deben ser realizadas en un orden específico, estos pueden modelar la lógica interna de una compleja operación, son equivalentes a los diagramas de flujo de datos [41].

Diagrama del Modelo de Dominio

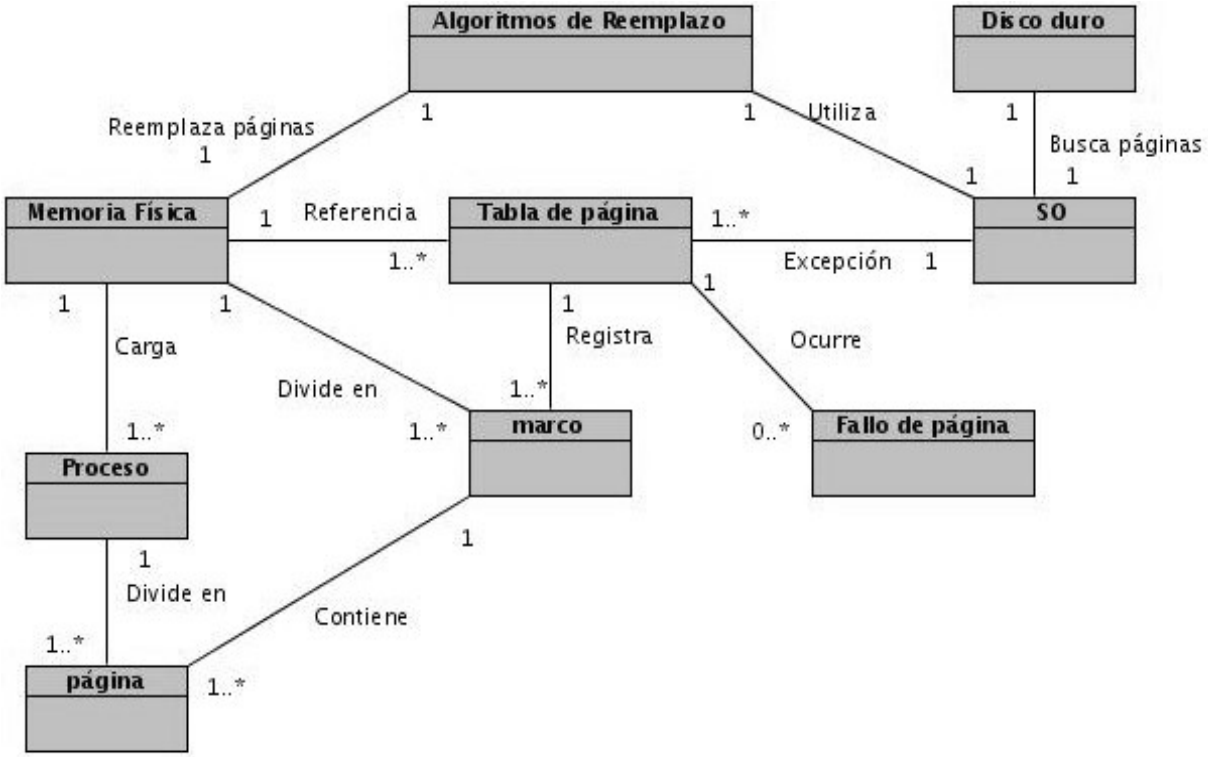


Diagrama 1.1 Modelo de Dominio del Sistema

Conceptos Fundamentales

- Marco: Consiste en pequeñas porciones de memoria de igual tamaño. A los cuales son asignadas las páginas.
- Página: Consiste en pequeñas unidades en que se dividen los procesos, estas tienen igual tamaño que los marcos.
- Fallo de página: Cuando se hace referencia a una página que no se encuentra en la memoria principal, se lanza una excepción al sistema operativo que debe ir al almacenamiento destinado por la memoria virtual a traer la página necesitada.

- Algoritmos de Reemplazo: Cuando ocurre un fallo de una página, el S.O tiene que escoger una página y retirarla de la memoria principal con el fin de dejar espacio para la página que tiene que traerse. Para elegir la página víctima utiliza un algoritmo de reemplazo.
- S.O: Es un conjunto de programas de computadora que a permite una administración eficaz de sus recursos y abstraen al usuario de la complejidad del hardware.
- Disco Duro: Dispositivo de almacenamiento.
- Procesos: Son las instrucciones de un programa que es una imagen de memoria que puede o no estar cargado en la memoria.
- Tabla de página: Registro con los marcos utilizados por un proceso determinado por lo que existe una tabla por cada proceso.
- Memoria Física: Es un recurso del sistema, referenciando a la memoria principal o memoria RAM.

Este modelo de dominio representa los conceptos fundamentales interactúan en el simulador que diseña; como se muestra la memoria física se encuentra dividida en marcos, y los procesos divididos en páginas, donde cada marco contiene una página; luego que en la memoria física se cargan los procesos, estos hacen referencias a las páginas que necesitan, y cuando llaman a una página que no se encuentra en la tabla de página, ocurre un fallo de página, por lo que el SO lanza una excepción y utiliza un algoritmo de reemplazo para seleccionar una página victima y colocar en su lugar la página que ha sido transferida del disco duro .

2.8 ITERACIÓN 1

Hay dos historias de usuario que codificar en esta iteración: “Introducción de datos al sistema” y “Simulación de Memoria Virtual”. Primeramente se realiza un modelo de dominio donde se integren estas las dos HU, luego se analizan por separado y son representadas sus características esenciales.

En esta primera iteración se muestran las funcionalidades básicas del simulador de memoria virtual, o sea la introducción ya sea manual o no de los datos y la utilización de los algoritmos de reemplazo de página [34].

Modelo de dominio de la iteracíon1

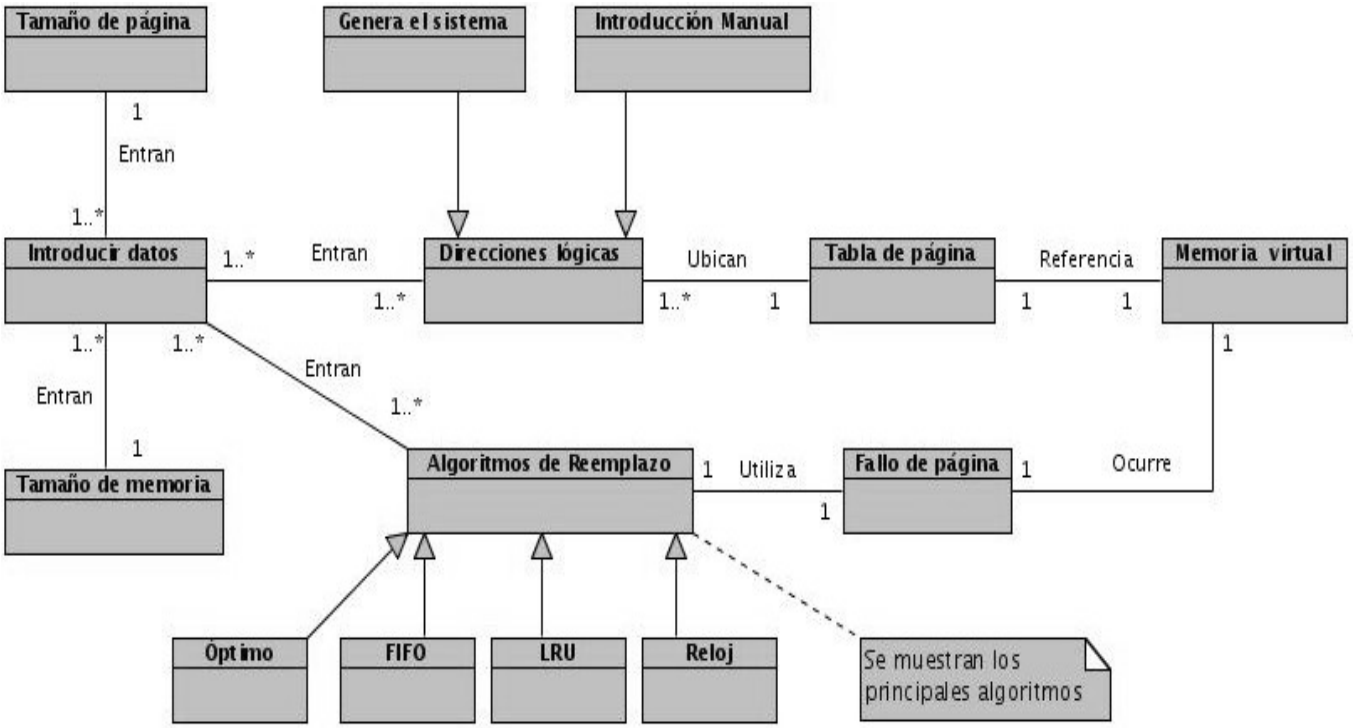


Diagrama 2.2 Modelo de dominio Iteracion1.

Este modelo representa los diferentes datos que pueden ser introducidos como tamaño de página, tamaño de memoria, el algoritmo de reemplazo deseado y las direcciones lógicas, a su vez como estas se incorporan a la tabla de página y son traducidas. Estas direcciones se cargan en la memoria y cuando se hace referencia una página que no se encuentra en ella, ocurre un fallo de página, en el cual se utiliza él algoritmo de reemplazo introducido para elegir una página víctima y colocar en su lugar la página necesitada.

Introducción de datos al sistema.

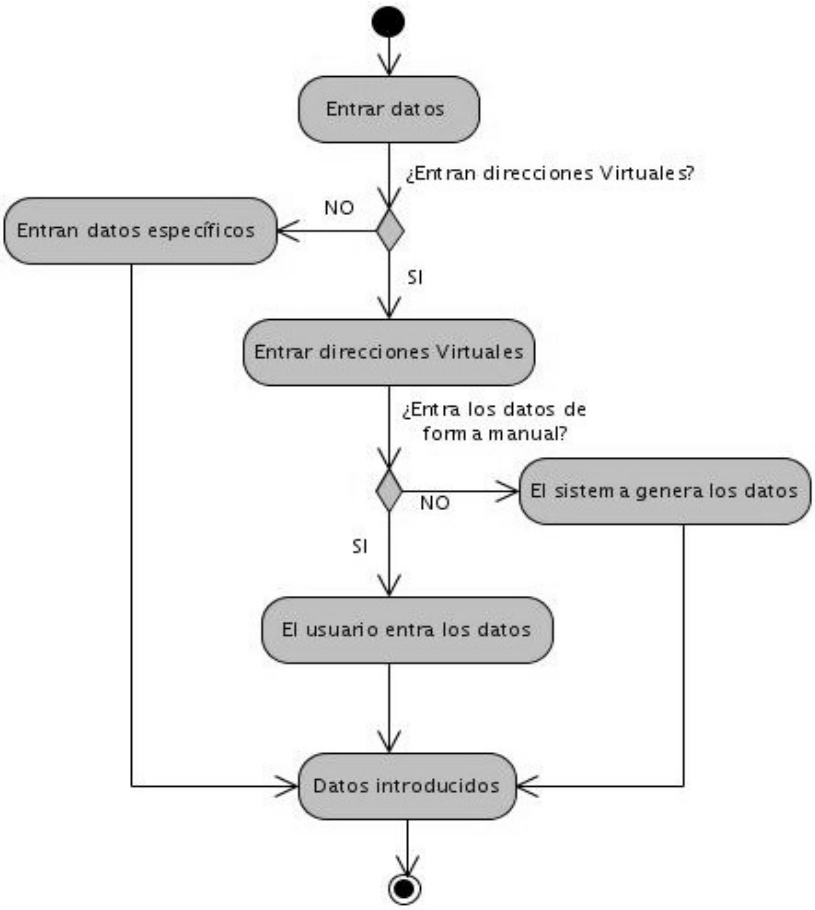
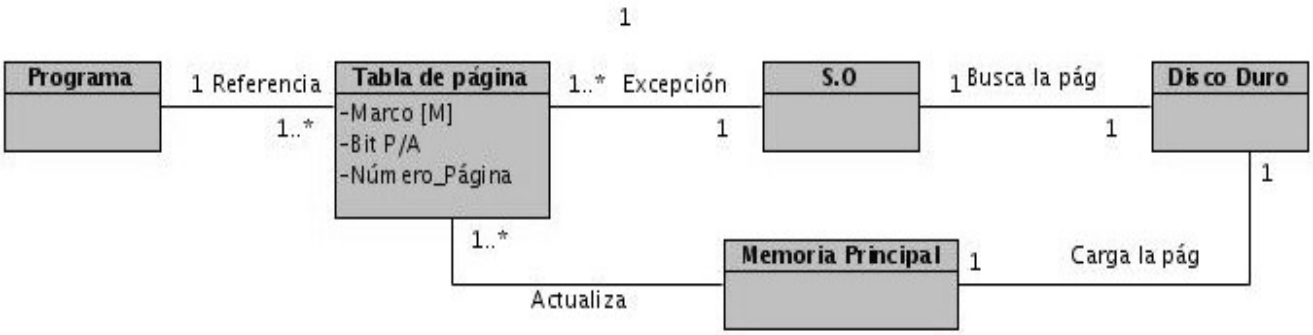


Diagrama de actividad 2.3 Introducir datos al sistema.

Se muestra mediante un diagrama de actividad los datos que pueden ser entrados al sistema, pues se pueden entrar datos que especifican las características del simulador como el tamaño de memoria, tamaño de página y el algoritmo seleccionado, a su vez se entran las direcciones virtuales de forma automática o sencillamente el usuario puede entrar las direcciones deseadas, estas direcciones tienen un tamaño fijo de 16 bytes.

Gestión de página mediante paginación simple



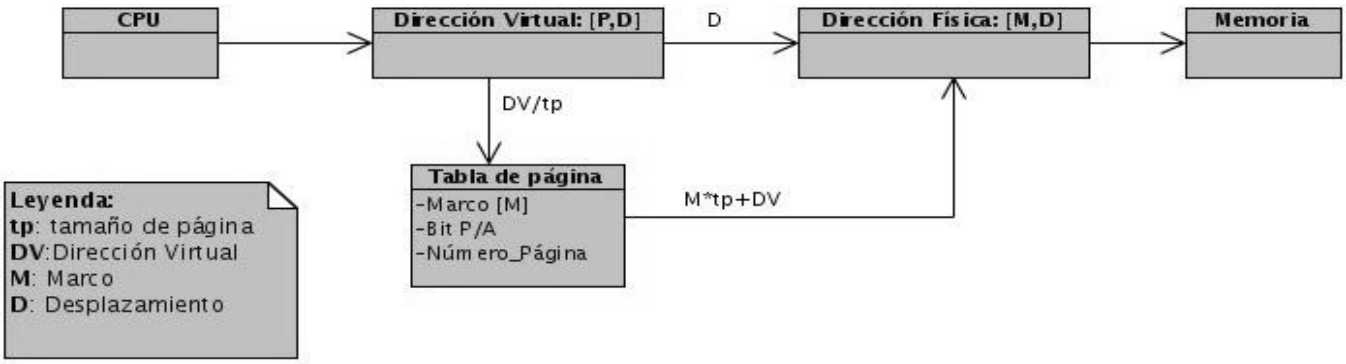
Modelo de Dominio 2.4 Gestión de páginas mediante la paginación simple.

En este se representa la gestión de página mediante el esquema de paginación simple, ocurre cuando un proceso se introduce en la memoria y se cargan sus páginas en los marco libres, en consecuencia se rellena su tabla de página. Cada entrada en la tabla contiene el número de marco de la página correspondiente en la memoria principal y como solo algunas páginas pueden estar en la memoria principal se necesita un Bit P/A (Presente /Ausente) para indicar si la página está en la memoria o no. Cuando el Bit de está en 0 indica que la página no se encuentra en la memoria, el S.O lanza un excepción y va al disco duro a traer la página que se necesita, se ubica en a la memoria principal si existen marcos libres sino utiliza un algoritmo de reemplazo para liberar un marco y colocarla en él, luego se actualiza la tabla página y se cambia el bit de 0 a 1.

Traducción de direcciones virtuales a direcciones físicas

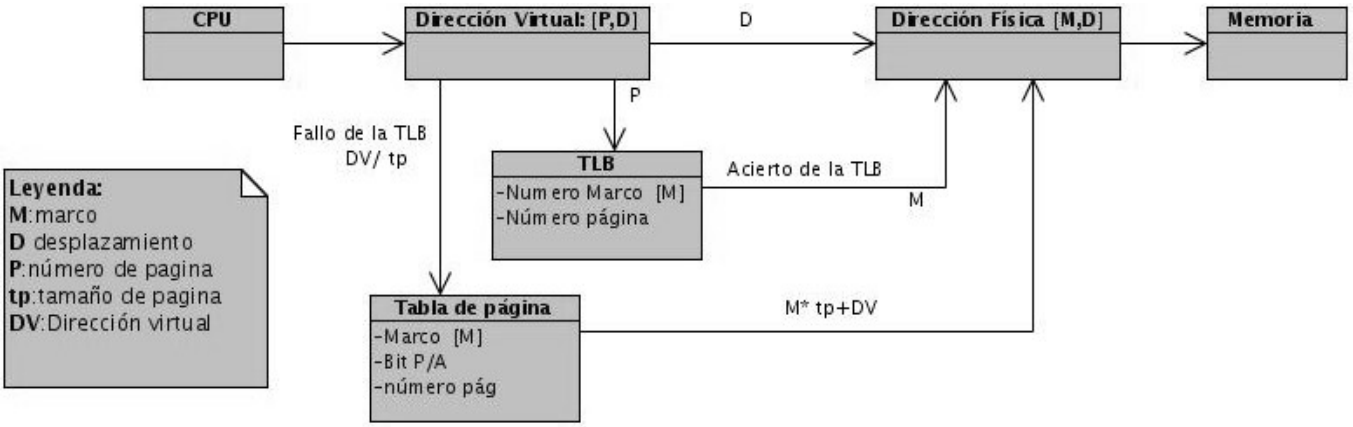
En el siguiente diagrama se representa la traducción de direcciones lógicas o virtuales a direcciones físicas, este proceso lo realiza la CPU que está integrado junto con otros componentes con un hardware especial encargado de realizar la traducción de las direcciones, conocido como MMU, está utiliza la tabla de página para realizar la traducción. La dirección virtual se divide en número de página (P) y desplazamiento (D). De la división entre la dirección virtual y el tamaño de página se obtiene como

resultado el número de página, el cual se asocia en la tabla de página al marco correspondiente que sumado al desplazamiento, es igual a la dirección física de la memoria principal.



Modelo de dominio 2.5 Traducción de direcciones virtuales.

Traducción de direcciones usando la TLB.



Modelo de dominio 2.6 Traducción de Direcciones usando la TLB.

En este diagrama la MMU encargada de la traducción envía la dirección virtual a la TLB que funciona como una cache especial; si se encuentra el número de página de la dirección correspondiente se produce un acierto de la TLB y la dirección es traducida, sin embargo si no se encuentra ocurre un fallo de

la TLB y se manda el número de página a la tabla de páginas, la cual se ocupa de la traducción y se ocurre el proceso descrito con anterioridad.

Gestión de página usando la TLB

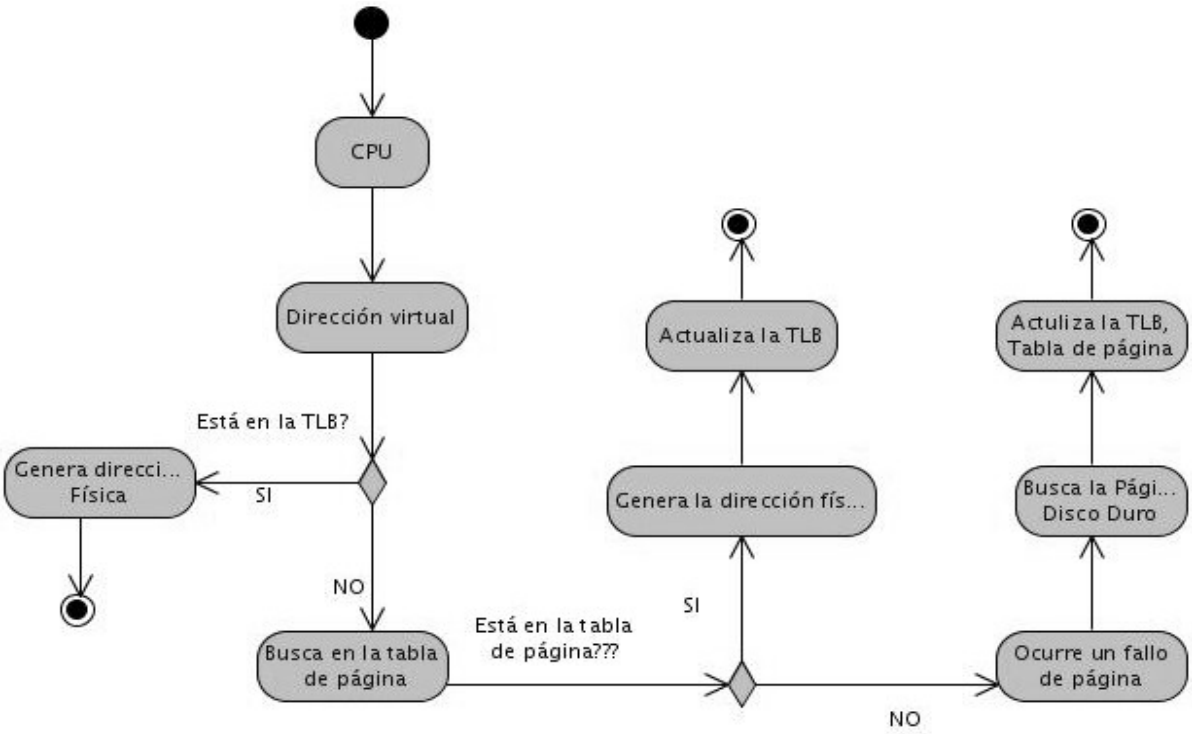


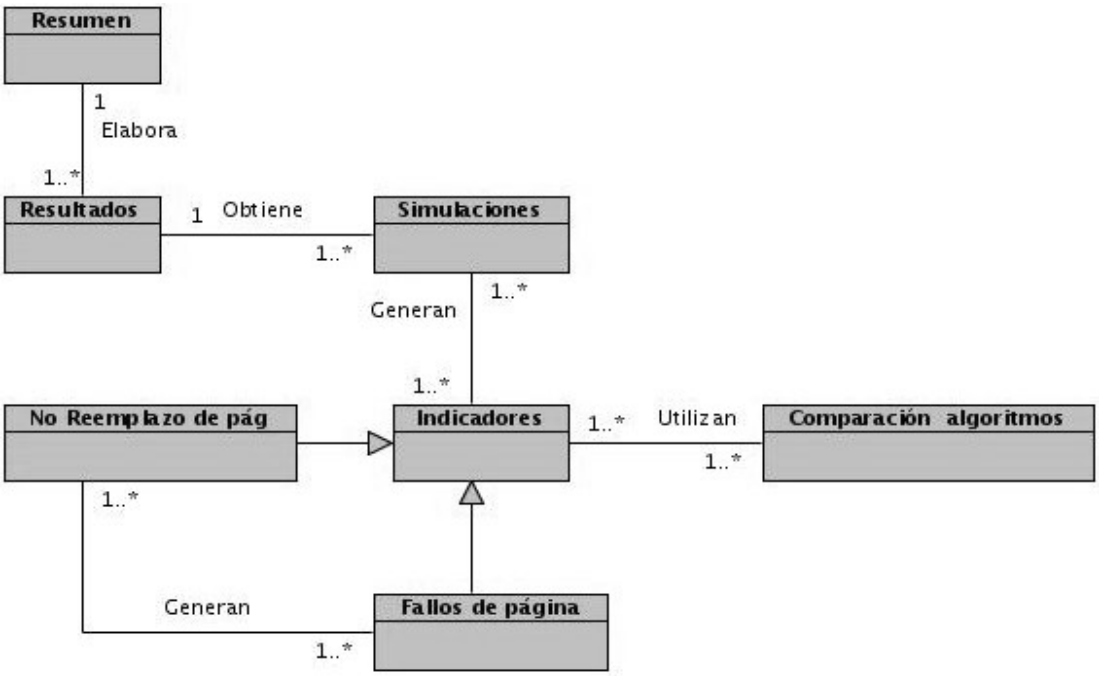
Diagrama de Actividad 2.7 Gestión de páginas usando la TLB.

En este diagrama de actividades se describe el proceso de gestión de páginas cuando se utiliza la TLB en consecuencia si la dirección está, inmediatamente se obtiene la dirección física sino se busca en la tabla de páginas, si se encuentra se traduce la dirección, si no ocurre un fallo de página y el SO va hacia el disco duro a buscar la página necesitada, luego de que es traída se actualiza la TLB y la tabla de página. Como se puede observar es mucho rápido el proceso cuando la página se encuentra en la TLB, lo que significa una mayor eficiencia y un mejor aprovechamiento de los recursos de la máquina.

2.9 ITERACIÓN 2:

En esta última iteración existe una única historia de usuario “Gestión los resultados”, con la cual la aplicación reúne todos los requisitos especificados por el cliente, y se termina el proyecto, en está historia se tratan los resultados obtenidos, las comparaciones entre diferentes los algoritmos aplicados.

Indicadores



Modelo de dominio 2.8 Gestión de los resultados.

Este diagrama representa los indicadores que se obtienen como parte de los resultados en las simulaciones, los cuales se utilizan como parámetros para establecer las comparaciones. Al terminar se puede obtener un resumen de los resultados.

Modelos Complementarios

Se hace necesario para una mayor comprensión de la investigación otro modelo que apoya y explica estos procesos que integran la memoria virtual. En este diagrama de actividades se modela la lógica del tratamiento general de las páginas durante la administración de memoria.

Tratamiento de página

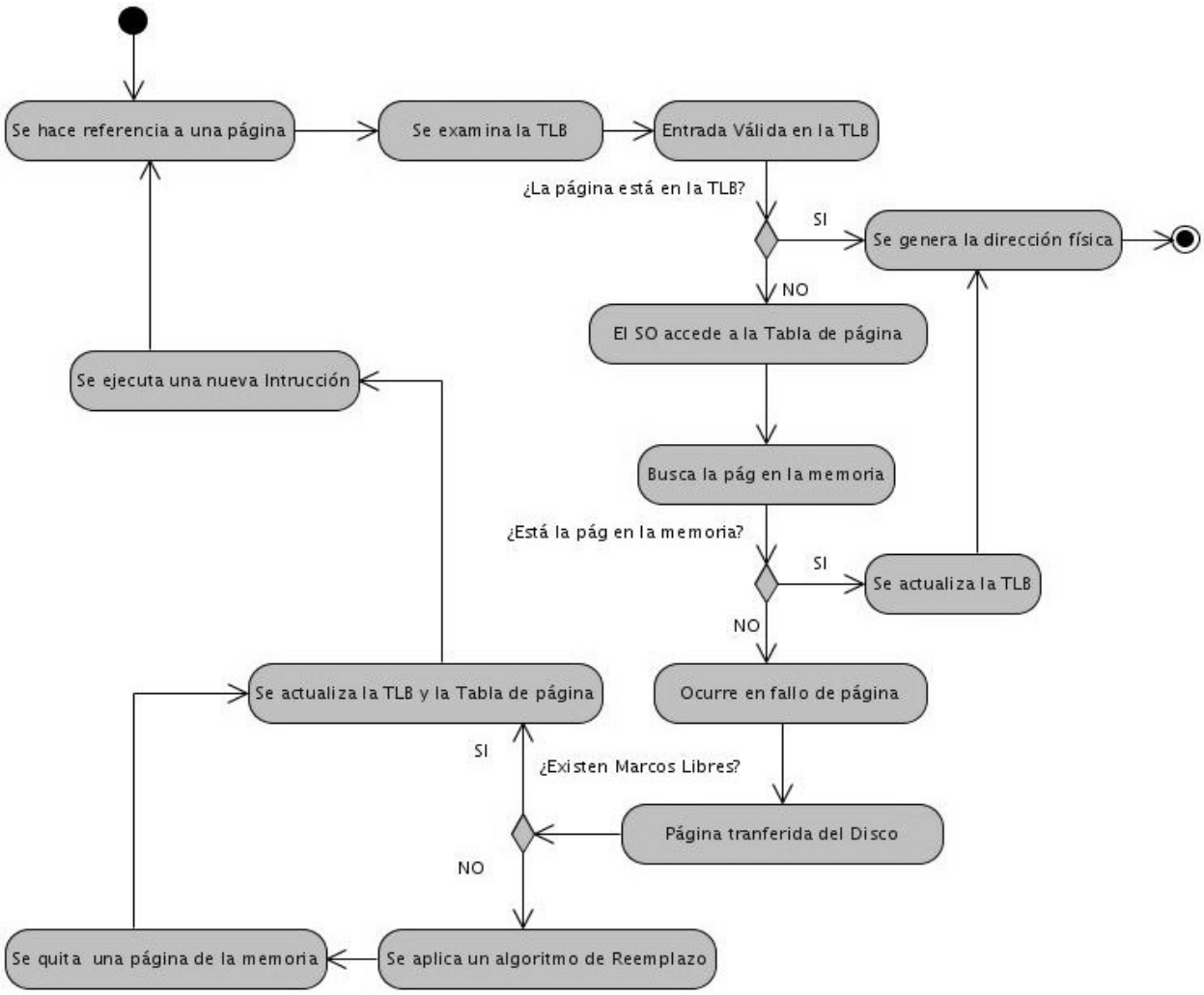


Diagrama de Actividad 2.9 Gestión de páginas

Este diagrama de actividades muestra toda la gestión de páginas que ocurre en los Sistemas Operativos. El esquema comienza cuando se hace referencia a una página, primeramente se examina la TLB si está se genera la dirección física sino se va a la tabla de página, si se encuentra se actualiza la TLB y se genera la dirección física , si no ocurre un fallo de página, mientras la página es transferida del disco, la CPU ejecuta un nuevo proceso, luego se observa si hay espacio en la memoria para colocar la página, y se actualiza tanto la TLB como la tabla de página, sino se utiliza un algoritmo de reemplazo para sacar una página de la memoria . Luego se planifica un nuevo proceso se carga en la memoria que lanza una petición de referencia a una página.

2.10 DESARROLLO DE LAS ITERACIONES

El desarrollo de las iteraciones comienza desde la fase de planificación con el diseño de las pruebas de aceptación y la selección de las tareas de la ingeniería.

Cada historia de usuario se divide en dos o más tareas de ingeniería de esta manera son elaboradas y a cada una le corresponde un periodo ideal de uno a tres días de desarrollo, estas deben ser concretas y mucho más sencilla que las historias de usuario [34].

La composición de las pruebas del sistemas constituyen unos de objetivos fundamentales en XP para lograr la calidad, esta metodología divide las pruebas del sistema en dos grupos: las pruebas unitarias y las pruebas de aceptación, las primeras son las encargadas de verificar el código y diseñada por los programadores. Las pruebas de aceptación o pruebas funcionales son las destinadas a evaluar si al final de una iteración se logro la funcionalidad deseada por el cliente. En la fase de Planificación solo son diseñadas las pruebas de aceptación [42].

Teniendo en cuenta las características modeladas con anterioridad en está primera iteración se trata de tener dispuestas las funcionalidades básicas, o sea aquellas sin las cuales la aplicación no tendría sentido. Las Historias de usuario que se priorizan son la número 1 y la número 2 ; la historia número 3 se analiza durante la segunda iteración [38].

2.10.1 ITERACIÓN 1

Tareas de Ingeniería.

En esta primera iteración se diseñan las tareas asociadas a la historia de usuario número 1 “Introducción de datos al sistema” y la número 2 “Simulación de Memoria Virtual”. Ver Anexos 2.

En la historia número 1 se trata de describir como se realiza todo el proceso de la introducción de datos, de la misma manera se analiza los datos introducidos además de ofrecer especificaciones de la interfaz de usuario, por ello está historia de usuario se divide en tres tareas de ingeniería de las cuales se solo se muestra la tarea de usuario 1 “Diseño de la interfaz “Introducir datos”, para ver las tareas en su totalidad dirigirse a Anexos 2.

Tarea de Ingeniería	
Número tarea:1	Número historia:1
Nombre tarea: Diseño de la interfaz “Introducir datos”	
Tipo de tarea : Desarrollo Desarrollo / Corrección / Mejora / Otra	Puntos estimados:6
Fecha inicio:	Fecha fin:
Programador responsable: Yamilé García Hernández	
Descripción: En el menú de la forma principal se selecciona “Introducir datos”, se diseñará una pestaña donde se especifica si los datos (Direcciones lógicas) son introducidos de forma manual o los genera el sistema, a su vez se entran los datos esenciales como tamaño de memoria y tamaño de página. En la misma ventana existe una tabla que muestra los datos que ya han sido introducidos.	

Tabla 2.4: Tarea de Ingeniería Diseño de la interfaz “Introducir datos

En las tareas asociadas a la historia de usuario número 2 “Simulación de Memoria Virtual” se describe la interfaz usuario correspondiente donde los estudiantes realizarán sus simulaciones; además se ofrecer detalles sobre la traducción de las direcciones virtuales incluyendo si se usa o no la TLB y el comportamiento de la memoria en tiempo real; está historia se divide en cuatro tareas .Ver Anexos 2. Se muestran la Tarea de ingeniaría “Diseño de la interfaz de “Memoria en Tiempo Real”

Tarea de Ingeniería	
Número tarea:4	Número historia:2
Nombre tarea: Diseño de la interfaz de “Memoria en Tiempo Real”	
Tipo de tarea : Desarrollo Desarrollo / Corrección / Mejora / Otra	Puntos estimados:10
Fecha inicio:	Fecha fin:
Programador responsable: Yamilé García Hernández	
Descripción: A partir de seleccionar “Memoria en Tiempo Real” en el menú de la forma principal; se diseñará una pestaña que muestre en una tabla las transformaciones que ocurren en la memoria. En esta pestaña se muestran como disminuye la cantidad de marcos libres y como aumentan el número de fallos y el número de reemplazos	

Tabla 2.7: Tarea de Ingeniería Diseño de la interfaz de “Algoritmos de Reemplazo”

Diseño de las pruebas de aceptación

Las historias de usuario son la principal fuente de información para diseñar las pruebas de aceptación, las cuales se basan en los requisitos que las historias representan, por lo que estas pruebas son hechas a partir de las funcionalidades descritas en ellas. En consecuencia son elaboradas a lo largo de la iteración y conjuntamente con el desarrollo del sistema, adaptándose a los cambios que este pueda tener [42]. Ver Anexos 4.

2.10.2 ITERACIÓN 2

La historia asociada a esta iteración es la historia de usuario “Gestión los resultados” se aborda sobre el diseño de la interfaz de la herramienta en el que el cliente verá los resultado del proceso de simulación. Ver Anexos 3, se divide en dos tareas de las que se muestra la tarea de ingeniería 8,”Diseño de la interfaz de “Gestión de Resultados”"

Tarea de Ingeniería	
Número tarea:8	Número historia:3
Nombre tarea: Diseño de la interfaz de “Gestión de Resultados”	
Tipo de tarea : Desarrollo Desarrollo / Corrección / Mejora / Otra	Puntos estimados:6
Fecha inicio:	Fecha fin:
Programador responsable: Yamilé García Hernández	
Descripción: A partir de seleccionar “Resultados” en el menú de la forma principal; se diseñará una pestaña que muestre la lista de las simulaciones realizadas. Se deberá habilitar una opción para seleccionar la simulación de la cual se desea ver los resultados, otra para procesar las comparaciones entra las simulaciones realizadas. De la misma manera se mostrarán los resultados y las comparaciones entre los algoritmos.	

Tabla 2.10: Tarea de Ingeniería “Diseño de la interfaz de “Gestión de Resultados”

Diseño de las pruebas de aceptación

Las pruebas de aceptación asociadas a la historia de usuario “Gestionar resultados” comprueban la interfaz de esta historia, la manera en que son gestionados los resultados e introducidos los indicadores. Ver Anexos 5

2.11 CONCLUSIONES PARCIALES

En este capítulo se abordaron los temas relacionados con la propuesta del sistema y se dio inicio a la primera fase de desarrollo de software siguiendo la metodología XP, así como se definieron las historias de usuario y se modelaron las principales funciones; también se realizó la planificación de las iteraciones, de la misma manera se comenzó el análisis del sistema con la selección de las tareas de ingeniería y el diseño de las pruebas de aceptación.

CAPÍTULO 3. SIMULADOR DE MEMORIA VIRTUAL

3.1 INTRODUCCIÓN

En este capítulo continúa el análisis y diseño del simulador propuesto según la metodología XP. Se plantea la metáfora de la aplicación como sustitución de la arquitectura inicial, además se realiza el diseño de las clases. De la misma manera se describen los patrones de diseño utilizados durante el desarrollo de la aplicación; se detalla el estilo arquitectónico a utilizar y se realiza la presentación de las interfaces de usuario que necesita la aplicación.

3.2 FASE DE DISEÑO

Se continúa el diseño del simulador de memoria virtual utilizando la metodología XP que propone el Diseño como segunda fase de desarrollo. Una de sus prácticas fundamentales es realizar un diseño sencillo que significa diseñar según las necesidades presentes y otra de sus prácticas más empleadas consiste en la elaboración la metáfora del sistema.

Cuando los desarrolladores se dispongan a codificar las tareas de ingeniería analizadas en el capítulo anterior se pueden plantear algunas dudas a la hora de nombrar los métodos, funciones, variables, lo que no propicia la comprensión entre estos, una solución a este problema es plantear la metáfora del sistema [34].

3.3 METÁFORA DEL SISTEMA

La metáfora es una historia que describe el funcionamiento el sistema con las palabras más sencillas posibles, de la misma manera soluciona los posibles problemas que pueden presentarse en el simulador por no tener una definición de la arquitectura base, ya que en XP la arquitectura de define de manera evolutiva [43].

Otra definición más sencilla consiste en la metáfora es una historia que todo el mundo puede contar a cerca de como el sistema funciona [34].

La tarea de elegir una metáfora para el sistema consiste en seleccionar un sistema de nombres que permita saber la relación entre el objeto y aquello que representa [44]. Una forma simple de representar la metáfora es a través de un diagrama sencillo pues cualquier persona que no ha comprendido bien el sistema puede hacerse una idea más clara de los procesos y definiciones con los que se trabajan.

En la aplicación que se diseña pueden existir definiciones y procesos que causen duda en algunas personas; uno de los conceptos que pude presentar más complejidad en el momento de implementar el sistema es orden en que el simulador debe realizar las acciones, como cargar las direcciones virtuales en la memoria, realizar la traducción de estas direcciones, así en que momento es reemplazada una página y cuando se utiliza un algoritmo de reemplazo. Por lo que se decide elaborar una metáfora que ayude a los desarrolladores a entender como funciona exactamente el sistema.

Teniendo en cuenta que la aplicación que se está diseñando es un simulador, y que este debe realizar las funciones de un modo similar a las acciones que realiza el S.O. en la administración de memoria, las tareas modeladas tienen en cuenta estas semejanzas Las cuales son escritas de manera sencilla para obtener una idea precisa de lo que en el futuro se desea implementar.

Por ello la metáfora del sistema consiste:

La aplicación comienza con la entrada de datos (direcciones lógicas o virtuales, tamaño de página, tamaño de memoria, algoritmo de reemplazo), luego se asigna estas direcciones aleatoriamente al proceso que se ejecuta, lo que significa que estas direcciones serán cargadas en la tabla de página del proceso y cuando se hace referencia a una página se comprueba si esta se encuentra en la memoria, de ser así se realiza la traducción de la dirección virtual de la página para conocer la dirección física o sea donde se localiza esta página en la memoria y se actualiza la tabla de página, de caso contrario el SO trae esa página del disco, se comprueba la existencia de marcos libres, y en uno de estos se coloca la página referenciada, sin embargo si no existen marcos libres entonces el sistema utiliza un algoritmo de reemplazo para hacer en la memoria un espacio y ubicar en él la página necesitada.

Una representación sencilla de está consiste en:

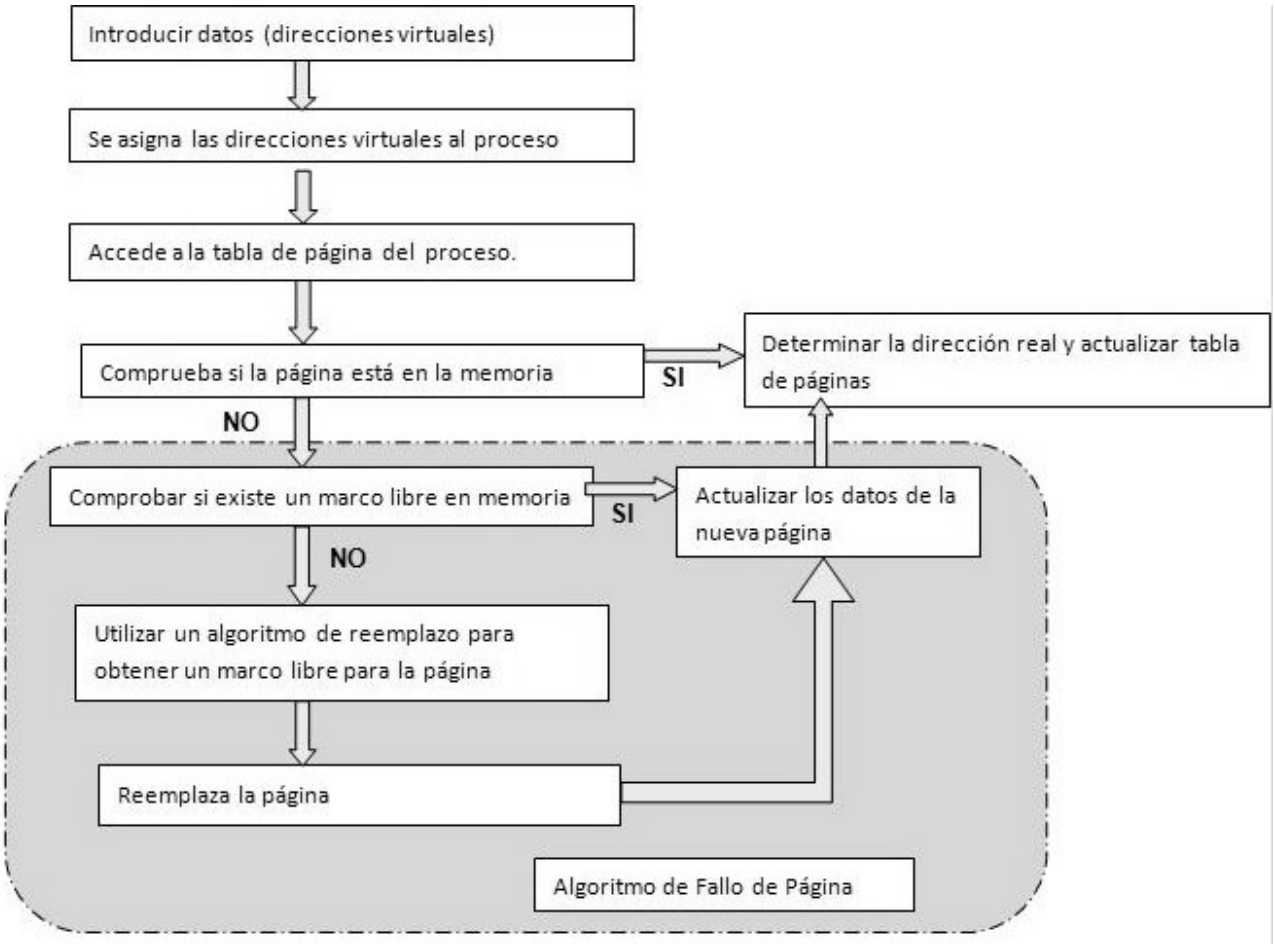


Figura 3.1 Representación de la metáfora del sistema

3.4 TARJETAS CRC

Teniendo en cuenta la metáfora elaborada y los procesos que debe reproducir el simulador se hace el modelo de tarjetas CRC que ayuda en la realización del análisis de las clases. Este modelo no es más que una colección de tarjetas CRC (Clase -Responsabilidad - Colaborador), las cuales se dividen en tres secciones que contienen la información del nombre de la clase, sus responsabilidades y sus colaboradores. Donde una clase es cualquier persona, cosa, evento, concepto, pantalla o reporte. Las responsabilidades de una clase son las acciones que conoce y las que realizan, sus atributos y métodos

[24]. Los colaboradores de una clase son las restantes clases con las que trabaja en conjunto para realizar sus responsabilidades. Los modelos CRC son una herramienta eficiente para el diseño detallado de las clases [45]. Las tarjetas CRC de la aplicación que se propone en esta investigación son:

Clase: CManejadora_Datos	
Responsabilidad: Cargar un fichero donde se guarda una lista de direcciones virtuales. Agregar a la lista las direcciones virtuales que sean introducidas por el usuario.	**Colaborador:**

Clase: CTraducción	
Responsabilidad: Realiza la traducción de las direcciones virtuales.	**Colaborador**: CSimulador

Clase: CTLB	
Responsabilidad: Busca la dirección lógica entre las recientemente traducidas.	**Colaborador:** CSimulador

Clase: CMemoria	
Responsabilidad: Cargar una lista marcos, indicando existen sí marcos libres o no.	**Colaborador:** CTraducción, CSimulador, CTLB

Clase: CResultados	
Responsabilidad: Obtiene los indicadores sobre los cuales se establecen las comparaciones. Gestiona los resultados de la simulación y genera los informes.	**Colaborador:** CSimulador

Clase: CSimulador	
Responsabilidad: Introduce las direcciones virtuales y el algoritmo de reemplazo. Comprueba el estado de la memoria. Establece los pasos de la simulación. Genera los resultados.	**Colaborador:** CAlgoritmo de Reemplazo, CTraducción, CMemoria, CResultados

Clase: CAlgoritmo de reemplazo	
Responsabilidad: Carga el algoritmo de reemplazo seleccionado.	**Colaborador:**CFIFO, CLRU, CNRU, CÓptimo, CReloj, CAleatorio, y CSegunda_Oportunidad

Clase: CMarco	
Responsabilidad: Tiene una lista de páginas.	**Colaborador:** CPágina

Clase: CPágina	
Responsabilidad: Almacena la información referente a las páginas.	**Colaborador:**

Clase: CLRU	
Responsabilidad: Realiza el intercambio de la página que tuvo el menor uso reciente.	**Colaborador:**

Clase: CFIFO	
Responsabilidad: Elimina la página que fue la primera en entrar en la memoria.	**Colaborador:**

Clase: CÓptimo	
Responsabilidad: Debe reemplazar la página que tiene el menor índice de fallos.	**Colaborador:**

Clase: CNRU	
Responsabilidad: Debe reemplazar la página según los bits de referencia y modificación.	**Colaborador:**

Clase: CReloj	
Responsabilidad: Debe reemplazar la página más antigua	**Colaborador:**

Clase: CSegunda_Oportunidad	
Responsabilidad: Realiza el reemplazo de la página que tenga el bit de referencia en 0	**Colaborador:**

Clase: CAleatorio	
Responsabilidad: Se realiza el reemplazo de una página escogida aleatoriamente.	**Colaborador**

3.5 DEFINIENDO LA ARQUITECTURA

La aplicación que se diseña debe utilizar una arquitectura sirva a los programadores de apoyo y que pueda mostrar las funcionalidades y la estructura del sistema.

Existen diversos conceptos que precisan que es la arquitectura; está el que la define como los elementos más importantes de un sistema. Es decir que nos brinda una visión general de las componentes del sistema. Determinar estos elementos es una tarea difícil pero muy importante, pues no responden únicamente a requisitos estructurales del proyecto, sino que están relacionadas con aspectos de rendimiento, usabilidad, reutilización, restricciones económicas [46].

Para concluir se tomará como definición de Arquitectura de Software la siguiente:

La Arquitectura de Software de un programa o sistema es la estructura o estructuras del sistema, la cual comprende los componentes de software, las propiedades externas visibles de estos elementos y las relaciones entre ellos [24].

En el desarrollo de software con metodologías ágiles como XP la arquitectura de software no es tratada con el enfoque que se necesita, pues existe escasa información disponible, tampoco se encuentra una documentación estandarizada para definirla. Esto implica que no existe una guía estándar, para fundamentar las arquitecturas lo que influye en proyectos más riesgosos, donde se aprecia la asignación de tareas de manera incorrecta, y se evidencia en la calidad final del producto [47].

3.6 PATRONES

En el mundo de la informática han existido problemas comunes entre las personas que se dedican a desarrollar software y a las soluciones que se llegan por la práctica y experiencia se conocen como patrones que son utilizados para alcanzar un mejor resultado producción de software [48].

Por lo que un patrón no es más que una solución a un problema en un contexto, codifica conocimiento específico acumulado por la experiencia en un dominio [48].

Entre las prácticas que se propone XP para esta fase se encuentra el diseño simple, lo que significa integración continua y **refactoring**, lo que implica que siguiendo las normas de diseño simple se pueden utilizar los patrones, en este sentido XP trata diferente el uso de los patrones pero no menosprecia su valor [49].

Existen diversas clasificaciones para los patrones entre las que se destacan:

- Patrones de Arquitectura
- Patrones de Diseño
- Patrones de Análisis
- Patrones de Proceso o de Organización
- Idiomas

En consecuencia en el diseño del simulador se utilizan los patrones de arquitectura, para logar una mejor implementación y organización, además de evitar que puedan presentarse algunos problemas comunes en las aplicaciones de este tipo.

3.6.1 PATRÓN UTILIZADO

Los patrones de arquitectura muestran la organización estructural o esquema de software de la aplicación, y se centran en las interacciones que van desde lo más sencillo a los complejo. Existen muchos patrones de arquitectura entre los que están [50] :

- Arquitecturas en Capas
- Modelo 3 capas
- Patrón Broker
- Arquitectura cliente/servidor:
- Arquitectura Orientada a Servicio (SOA)
- Arquitectura cliente grueso
- Modelo-Vista –Controlador (MCV)

Para el desarrollo de la investigación se utiliza este último patrón el Modelo Vista Controlador.

3.7 ESTILO ARQUITECTÓNICO UTILIZADO MODELO- VISTA- CONTROLADOR (MVC)

Modelo Vista Controlador (MVC) es un patrón de arquitectura de software que separa los datos (controlador), la interfaz de usuario (vista), y la lógica de control (modelo) en tres componentes distintos. El patrón MVC se ve frecuentemente en aplicaciones web [51].

Donde:

Modelo: Esta es la representación específica de la información con la cual el sistema opera. La lógica de datos asegura la integridad de estos y permite derivar nuevos datos.

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

Controlador: Este responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista.

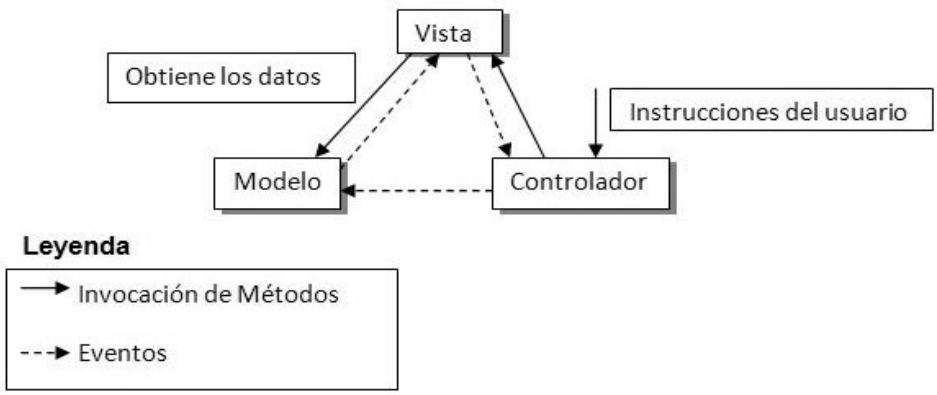


Figura 3.2 Patrón MVC

Se seleccionó este patrón de arquitectura por que es usado en aplicaciones que manejan gran cantidad de datos y transacciones complejas donde se requiere una separación de los conceptos para que el desarrollo esté estructurado de forma más sencilla y simple, facilitando la implementación en capa de

forma paralela e independiente. Además Swings que constituye uno de los framework que usa Netbeans que está basado en este patrón, pues MVC facilita la creación de aplicaciones interactivas, que aprovecha la ventaja de la modularidad, para permitir que las piezas ya desarrolladas para una aplicación sean reutilizadas en una nueva. Además de que el lenguaje de programación Java proporciona un soporte especial para la arquitectura MVC mediante dos clases; una conocida como Observer que no es mas que cualquier objeto que desee ser notificado cuando el estado de otro objeto sea alterado y la clase Observable que consiste en cualquier objeto cuyo estado puede representar interés y sobre el cual otro objeto ha demostrado ese interés [52].

El Modelo es un subtipo de Observable y la Vista es un subtipo de Observer. Esto tiene como ventaja que estas clases manejan adecuadamente la función de informe de cambios que necesita la arquitectura MVC y facilitan el mecanismo por el cual las Vistas pueden ser informadas de los cambios producidos en el Modelo. Del mismo modo permiten acceder a los datos del objeto Modelo [52].

3.7.1 MODELADO DE LA ARQUITECTURA

Está investigación se diseña la arquitectura de componentes que consiste en modelar los componentes funcionales en la arquitectura lógica de la aplicación. Para este modelado se utilizará el diagrama de componentes.

Donde un componente no es mas que una parte modular de un sistema, que encapsula la implementación y un conjunto de interfaces, además contiene clases y puede ser implementado por uno o más artefactos (ficheros ejecutables, binarios, etc.) [53].

Se modela la arquitectura de componente basada en los módulos del sistema, está representa la estructura general de la aplicación, de la misma manera se diseña el otro diagrama que muestra posibles las clases pueden tener este simulador y sus interacciones.

Por lo que el diagrama de componentes basado en los módulos del simulador sería:

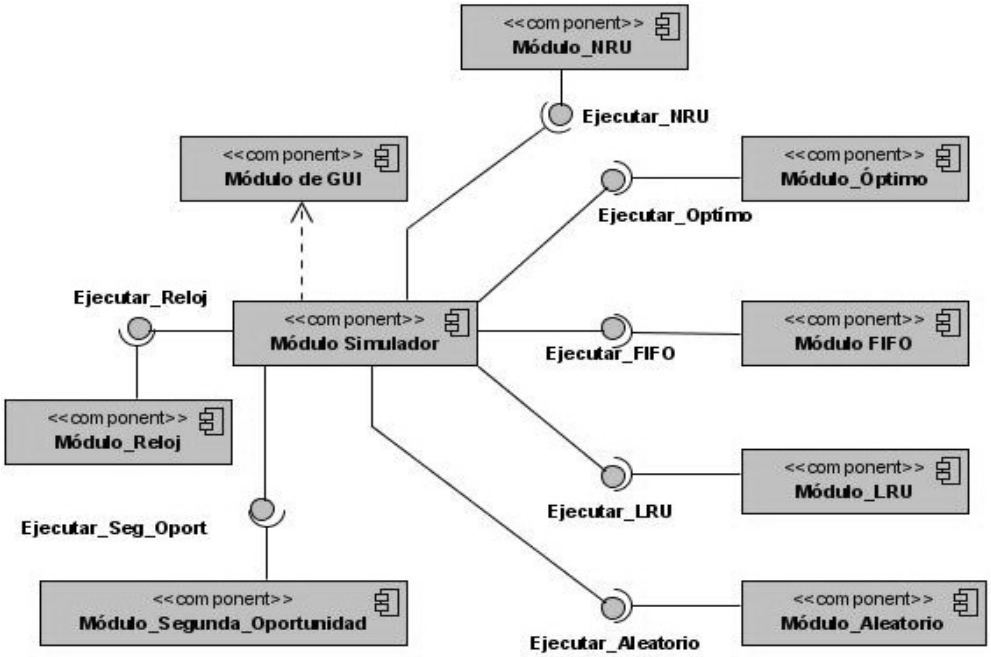


Figura 3.1 Diagrama de componentes por módulos.

Como se muestra, este diagrama representa los 8 módulos que integran la aplicación y un módulo llamado “Modulo de GUI”; las siglas en inglés GUI significan Graphics User Interface o interfaz de usuario gráfica, este módulo como su nombre lo indica contiene las interfaces de la aplicación, de la misma manera se modelaron los módulos correspondientes a los algoritmos de reemplazo que se incorporan al programa principal que pertenece al “Modulo Simulador”.

Diagrama de Componente que representa las posibles clases.

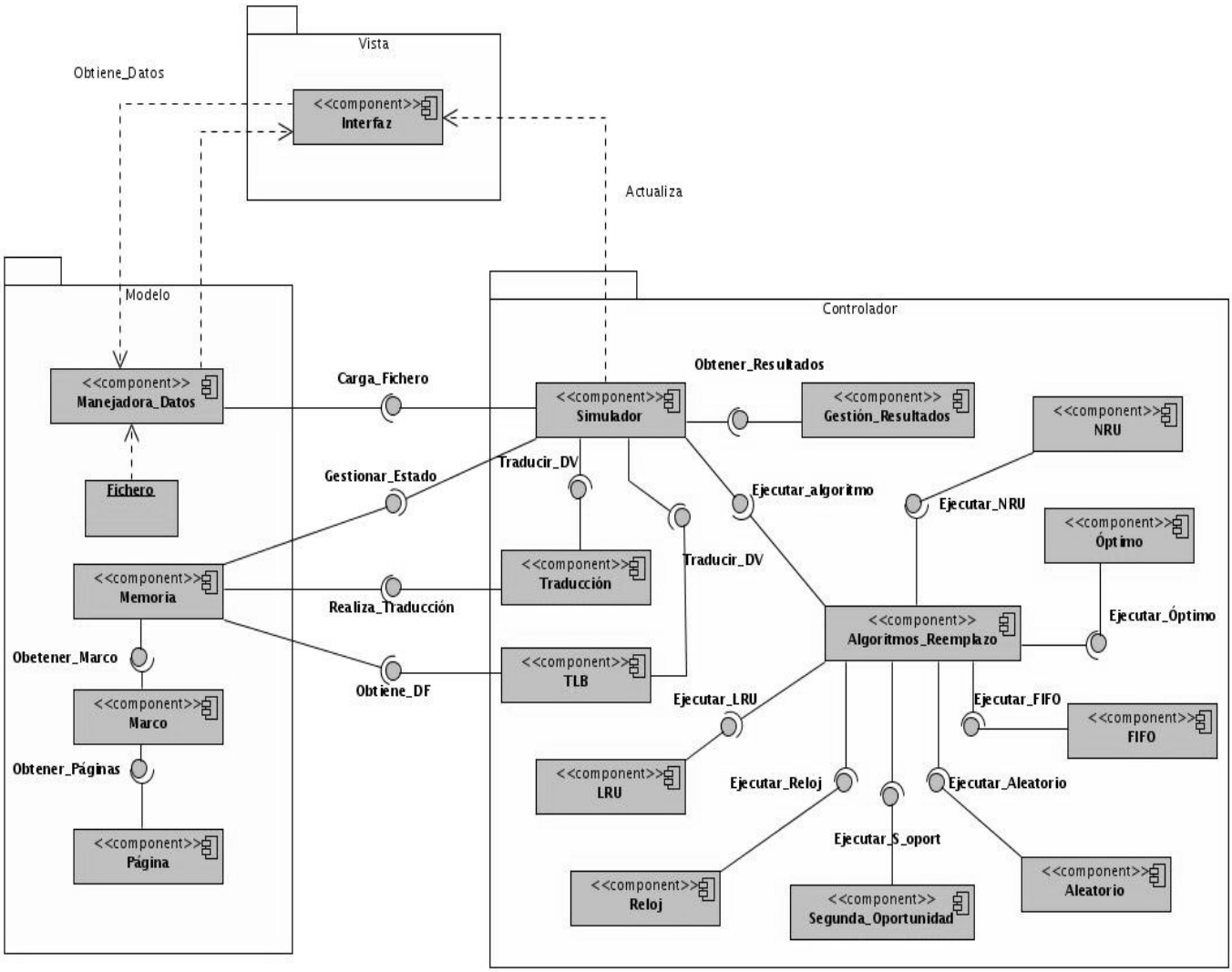


Figura 3.2 Diagrama de componentes por clases.

Donde:

En la figura 3.2 se muestran los componentes organizados en paquetes divididos según el patrón arquitectónico seleccionado, en el cual al Modelo pertenecen el componente memoria con tiene relación con el componente Marco y este se relaciona con el componente Página, de la misma manera se encuentra el componente manejador de datos. Este paquete tiene la responsabilidad de definir la lógica de

la aplicación así como manipular todos los datos que son utilizados en el sistema, lo que quiere decir es que el modelo es responsable de cargar los datos (direcciones virtuales) en la memoria además de saber cuando esta se encuentra llena o no, de la misma manera debe informar a al componente Interfaz en el paquete Vista de cualquier cambio que se produce con la información.

También se encuentra representado el paquete Vista que incluye las clases que representan las interfaces que determinan la representación visual del simulador .La Vista tiene la tarea de mantener actualizada la representación de la simulación, para lo que debe recibir mensajes indirectos del modelo y directos del controlador.

De igual manera encontramos el paquete Controlador que contiene los componentes Resultados, Simulador, Traducción, Algoritmos_Reemplazo y los algoritmos a simular como FIFO, LRU, Aleatorio, Óptimo, Segunda Oportunidad, NRU, Reloj. Este paquete tiene la responsabilidad de determinar las acciones que deben ser ejecutadas. El funcionamiento de este patrón comienza cuando las instrucciones del usuario entran al sistema por medio del componente “Simulador”, rápidamente se actualiza las interfaces relacionadas en el paquete Vista que obtiene los datos del los componentes ubicados en el paquete Modelo. Por ejemplo el usuario desea ver la simulación del algoritmo FIFO, el controlador envía esta instrucción a la Vista que determina que FIFO es el algoritmo seleccionado por el usuario y envía un mensaje al Modelo, el cual se actualiza y lo informa ; luego envía un mensaje a la Vista indicando que debe actualizar la selección.

3.8 DIAGRAMA DE CLASES

El diagrama de clases es uno de los artefactos más utilizados en el modelado de sistemas orientados a objetos. Este muestra un conjunto de clases, y sus relaciones donde una clase es cualquier persona, lugar, cosas, concepto aplicable al sistema; estas tienen atributos y métodos, donde los atributos identifican las características propias de cada clase y los métodos las acciones que se implementan. Estas se relacionan entre sí a través de relaciones (asociación, herencia, agregación); a su vez tienen roles y cardinalidad (multiplicidad) [54]. Para la creación de este diagrama de clase se utilizó el apoyo de las tarjetas CRC.

El diagrama de clase del sistema sería:

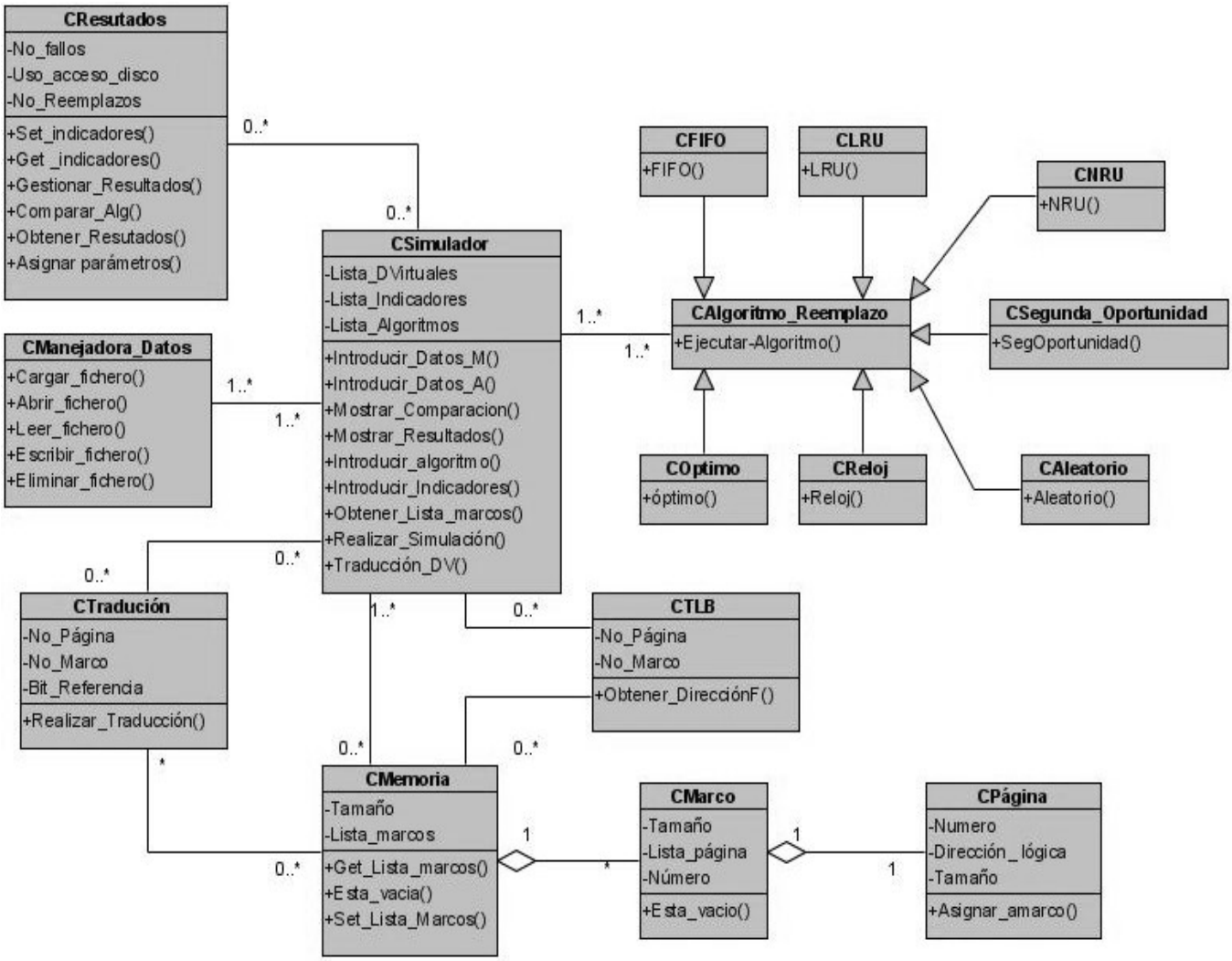


Figura 3.3 Diagrama de clases del diseño.

La aplicación consta de 16 clases; donde la clase CSimulador es la encargada de que las restantes reciban la información que necesita, por medio de esta son introducido los datos y se parte de que la memoria que no se encuentre inicialmente vacía, pues cuando se ejecuta la aplicación la clase CManejadora_Datos carga un fichero con varias direcciones virtuales; luego de que el usuario entre los

datos (direcciones virtuales) se guardan en el mismo fichero, y se espera que el usuario comience el proceso de simulación.

Cuando esta comienza en dependencia de las opciones, utilizar la TLB la cual almacena los últimos accesos a memoria de la clase CTLB, o utilizar la tabla de página que está representada en la clase CTraducción donde se lleva un registro de la equivalencia entre el número de página y el número de marco necesario para el acceso a memoria son traducidas las direcciones virtuales. Luego se pueden especificar los parámetros que pueden influir o no en el proceso de simulación en la clase CResultados, se continua con la elección del algoritmo que se desea simular, al finalizar se guardan los resultados obtenidos en el fichero, y para terminar el sistema muestra opciones comparar dos simulaciones previamente realizadas y mostrar los resultados de la clase CResultados.

3.9 PROTOTIPO DE INTERFAZ DE USUARIO

Diseño del prototipo de interfaz de usuario

La interfaz gráfica de usuario (GUI por sus siglas en inglés) es la parte del software que interactúa directamente con el usuario; que nace a partir del prototipo de interfaz de usuario, este representa los requisitos de interfaz de una manera independiente, los irá evolucionando a través del análisis y el diseño para dar lugar a la interfaz de usuario final del sistema. Estos prototipos permiten explorar y validar el diseño de la interfaz de usuario, comprueban que esta sea factible, cómoda, y de igual manera cumpla con los requerimientos. Unos de los propósitos principales de crear prototipos de interfaz de usuario es comprobar la usabilidad antes que el usuario comience a utilizar la aplicación [55].

Presentación de la aplicación

Como se ha descrito anteriormente el objetivo fundamental de esta investigación es realizar el análisis y el diseño de un simulador que muestre el funcionamiento de la memoria virtual, centrándose en los algoritmos de reemplazo de página los que se implementarán bajo el esquema de paginación simple [56].

3.9.1 PRINCIPALES ELEMENTOS DE LA INTERFAZ DE USUARIO

Unos de los elementos más importantes en el diseño del prototipo de interfaz de usuario es determinar tema de presentación, o sea la pantalla, página HTML o informe, y un elemento de menor importancia lo constituyen los campos de entrada de datos; por todo ello la descripción del prototipo de interfaz de usuario del simulador comenzará por la descripción de la vista principal y luego se extenderá hacia los restantes elementos que lo componen [55]. Al iniciarse la aplicación se obtiene está pantalla o vista principal.

Pantalla Principal.

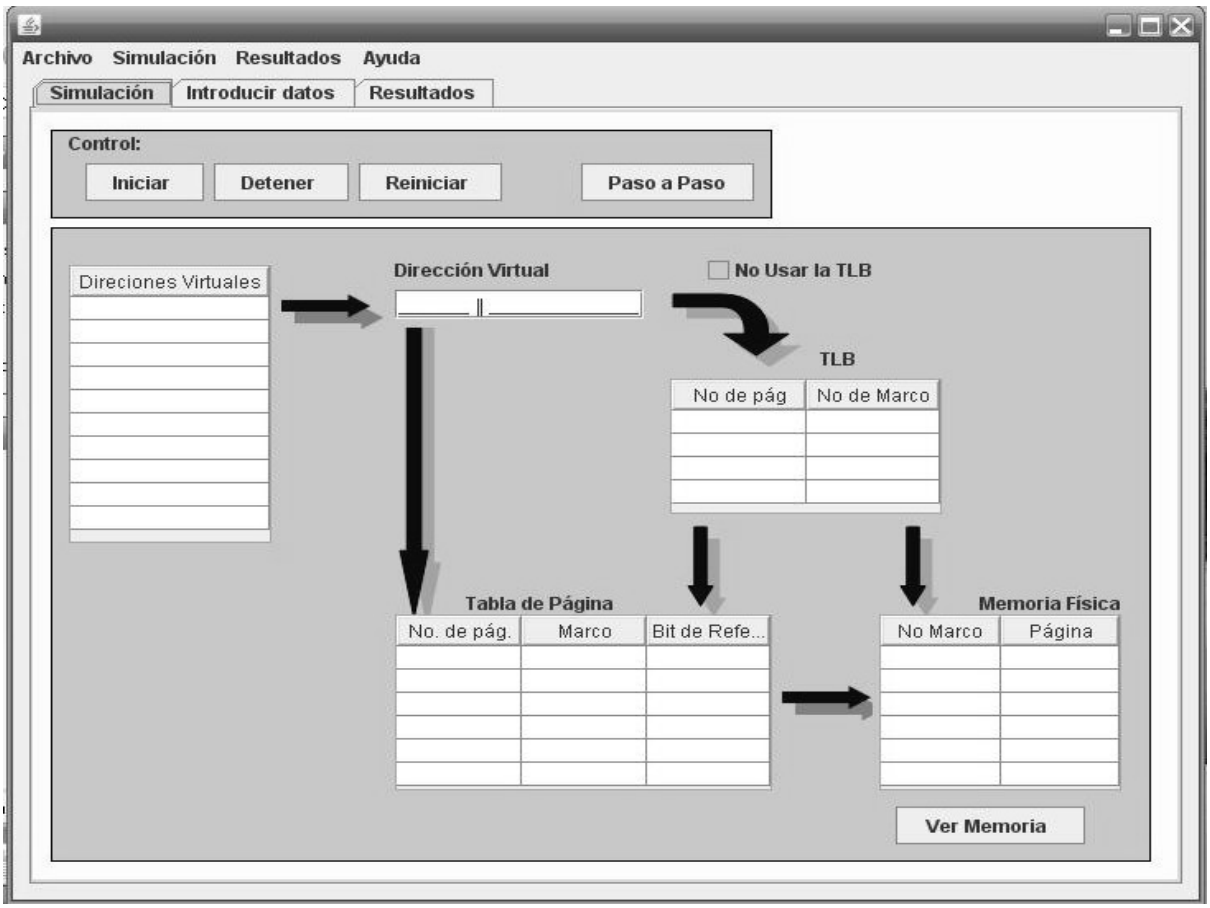
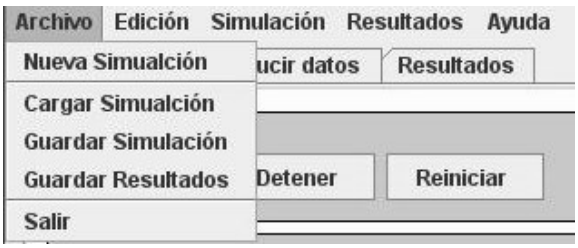


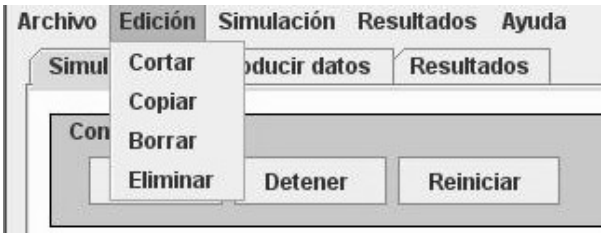
Figura 3.4 Ventana principal

En esta vista se pueden diferenciar varias zonas; la primera de estas se encuentra en la parte superior y es la barra de menú, que tiene las opciones:

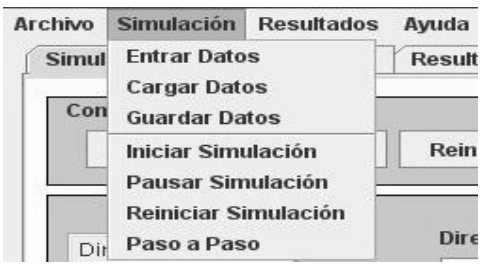
- Archivo: Contiene las opciones básicas del simulador.



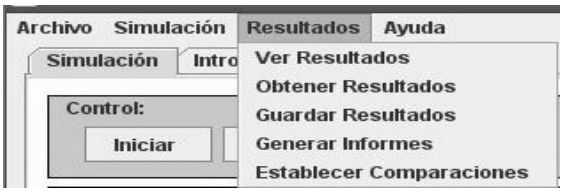
- Edición: Se encuentran las opciones básicas de edición.



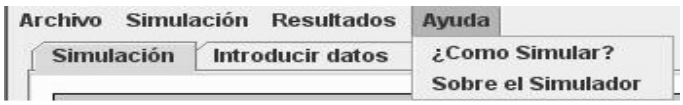
- Simulación: Se encuentran las opciones referidas con la simulación así como las relacionadas con la entrada de datos.



- Resultados: Se encuentran las opciones relacionadas con la obtención de los resultados.



- Ayuda: Se encuentra una guía para los estudiantes realizar sus simulaciones.



La segunda zona una zona donde se puede correr la simulación paso a paso o detenerla cuando así se desee. Luego se observa en la pestaña “Simulación” el orden en que las direcciones son traducidas y como se llena la memoria física. De la misma manera se aprecia el botón “Ver memoria” que abre la siguiente ventana.

Ver memoria en Tiempo Real

En la siguiente ventana en la parte superior se muestra un conjunto de botones que facilitarán al estudiante la observación y análisis de la técnica de memoria virtual, luego se mostrará como se llena la memoria física y como van ocurriendo los fallos de página, cuando esto suceda se marca en rojo la página a reemplazar y se utiliza el algoritmo de reemplazo seleccionado para elegir la página víctima. En la parte inferior se muestran los resultados de la simulación, así como aumenta la cantidad de fallos ocurridos y cantidad de reemplazos; de la misma manera se representa como disminuye la cantidad de marcos libres.

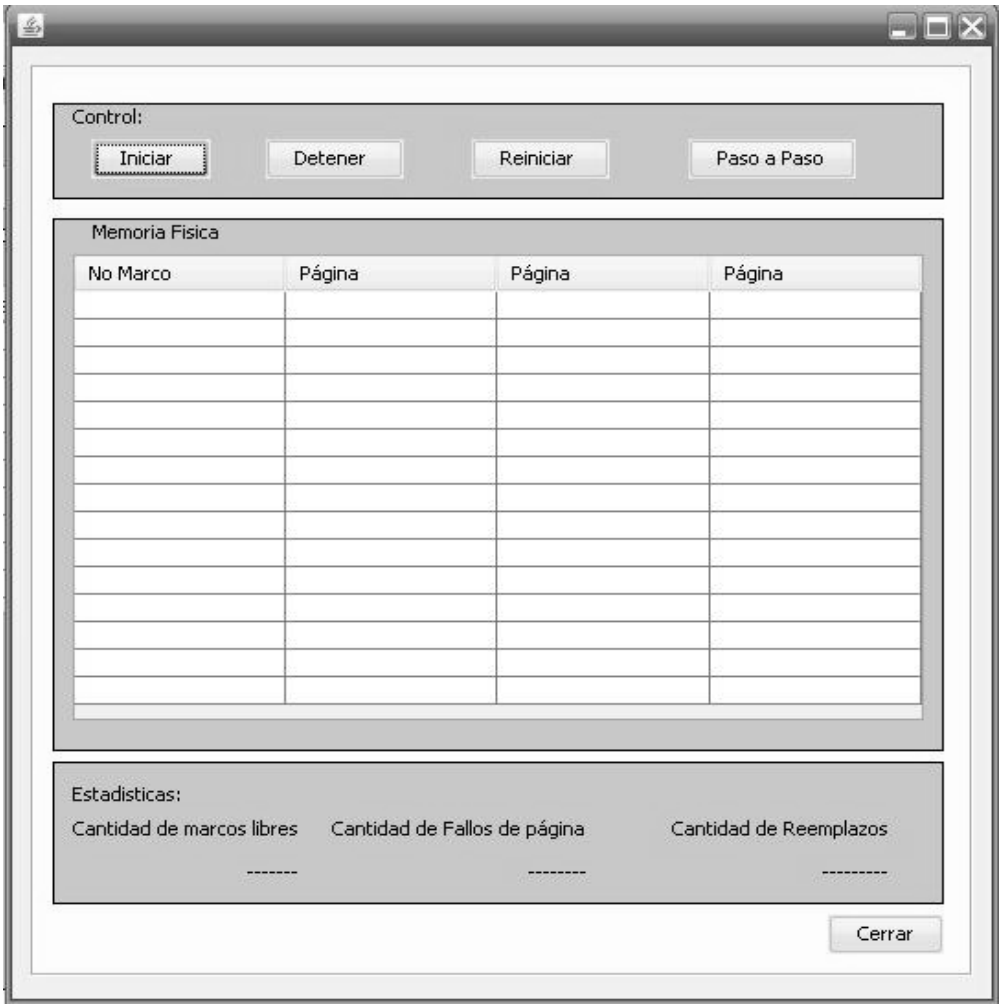


Figura 3.5 Interfaz "Ver memoria en tiempo real".

Para realizar la simulación antes se debe ir a la pestaña introducir datos o seleccionar la opción cargar simulación en el menú archivo.

Introducir datos

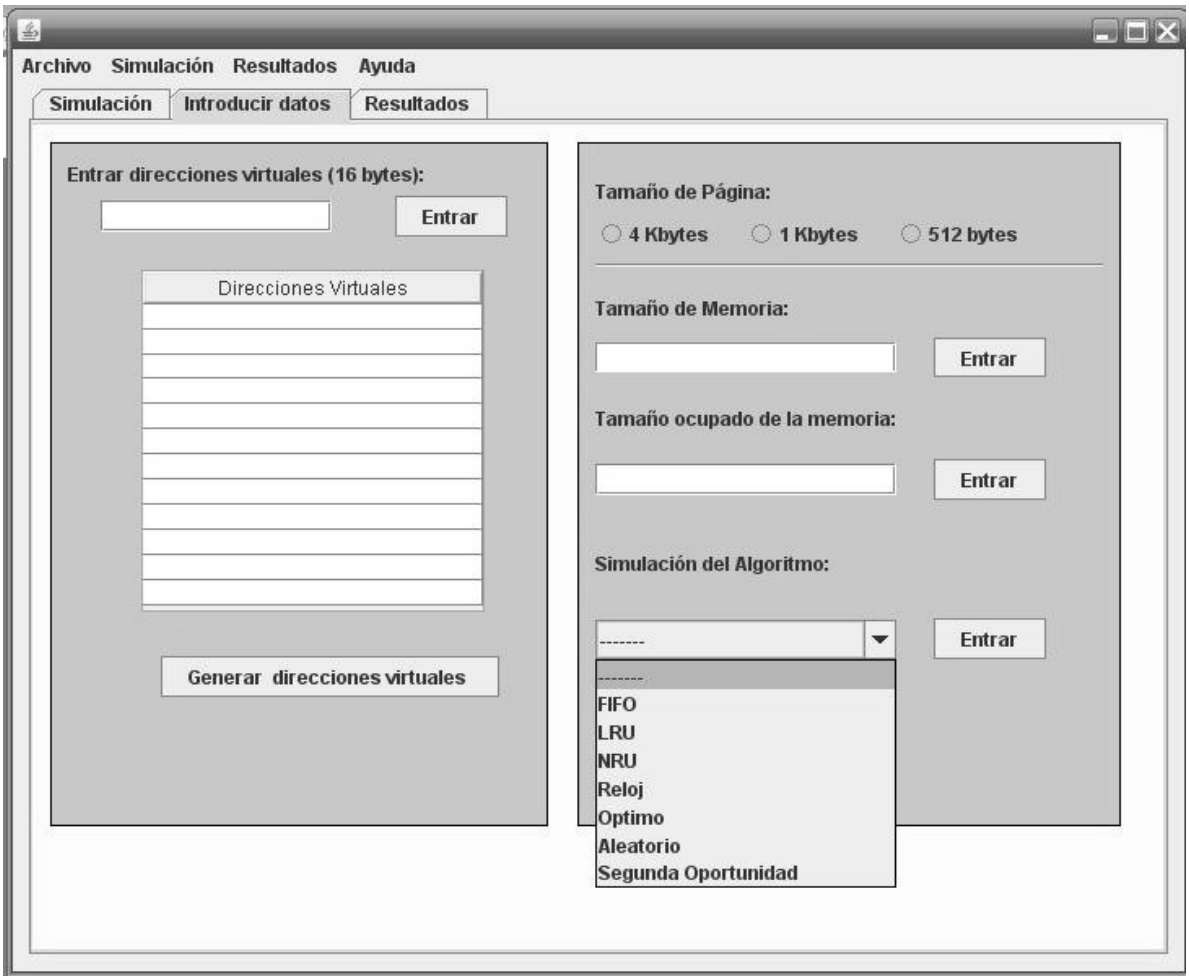


Figura 3.6 Pestaña "Introducción de datos manuales"

En la cual se aprecia en la parte derecha el campo para la entrada de datos (direcciones lógicas), los que serán introducidos por el usuario en el componente JTextFiels en valores binarios (1,0), los cuales serán mostrados en la tabla que aparece en la parte inferior. De igual forma en la parte inferior se muestra un botón que genera las direcciones aleatoriamente, estas direcciones son cargadas del fichero que controla la clase CManejadora_Datos.

En la parte izquierda se encuentran los datos que deben ser introducidos para determinar las características específicas de cada simulación. De la misma manera los datos pueden ser guardados y luego cargados por medio del menú simulación que muestra las opciones “Guardar Datos” y “Cargar Datos”. Los resultados de estas simulaciones pueden ser guardados por medio de la pestaña.

Resultados

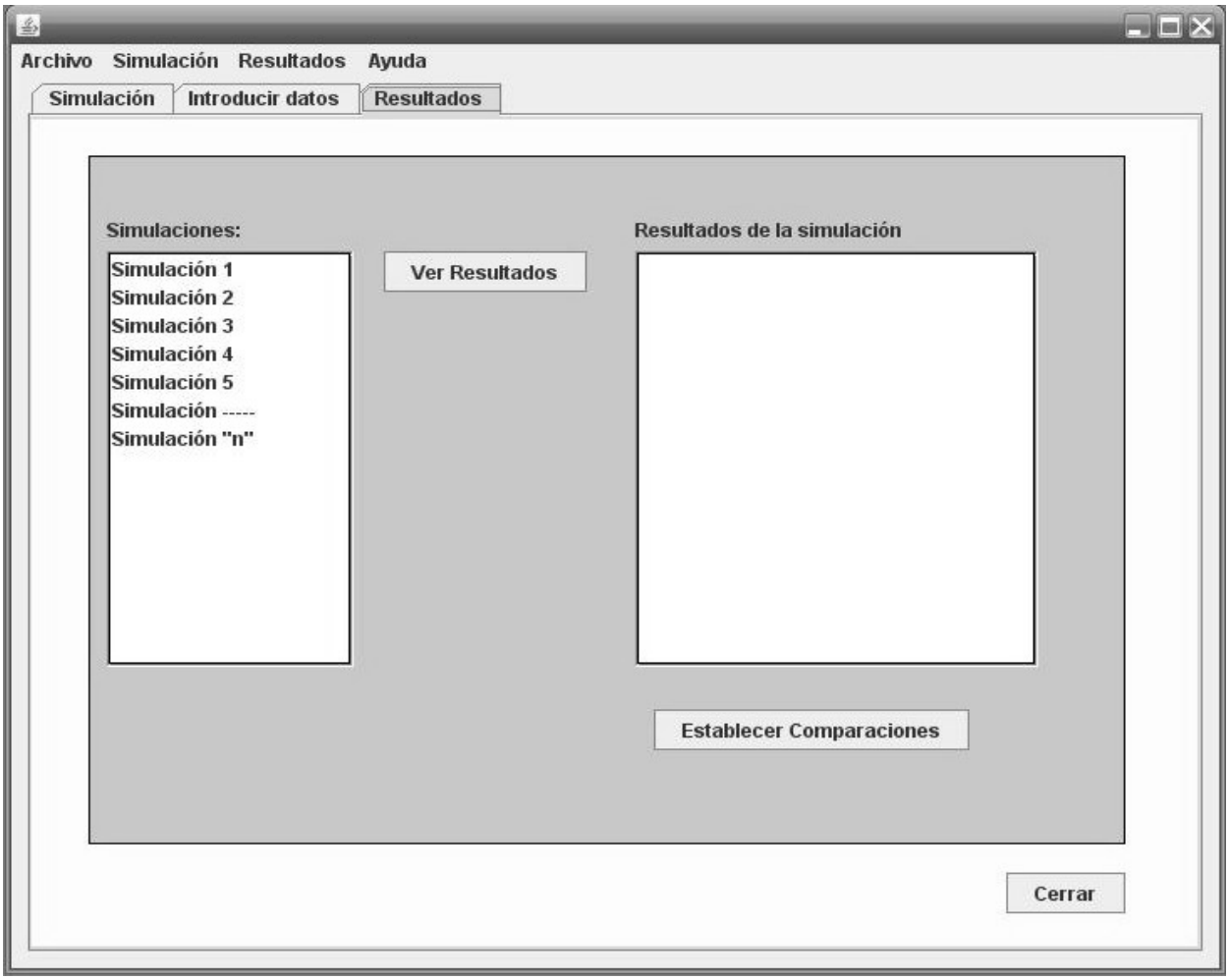


Figura 3.11 Pestaña “Resultados”

Se representa en la parte izquierda una lista de varias simulaciones hechas con anterioridad, de donde se selecciona la simulación deseada y se oprime el botón “Ver Resultados”, luego se muestra en la parte derecha los resultados obtenidos, los cuales se muestran en forma de gráficos. En la parte inferior se encuentra el botón establecer comparaciones que abrirá la ventana que contenga la función de establecer las comparaciones.

3.10 CONCLUSIONES PARCIALES:

En este capítulo se analiza y se define la arquitectura del sistema, de la misma manera se diseña el diagrama de clases y se muestran los modelos correspondientes. Además se describen los prototipos de interfaz de usuario que pueden ser utilizados para realizar la simulación de los algoritmos de reemplazo de página.

CONCLUSIONES GENERALES

La administración de memoria es una de las tareas más significativas realizadas por el Sistema Operativo por lo que desarrollar medios didácticos que permitan lograr una mejor comprensión en general y de este tema en particular constituye un objetivo primordial. Por está razón en la investigación, se dispone del análisis y diseño de una herramienta para la simulación de la Administración de Memoria como apoyo didáctico al aprendizaje de los Sistemas Operativos. Para ello fue necesario:

- Estudiar los diferentes algoritmos de reemplazo.
- Investigar y seleccionar los TDA necesarios para su implementación.
- Definir y diseñar las clases necesarias.
- Proponer un prototipo de Interfaz de usuario

Con este resultado y su implementación la UCI dispondrá de una herramienta propia para mejorar el proceso de enseñanza/aprendizaje de estos tópicos.

RECOMENDACIONES

De acuerdo a los resultados obtenidos en esta investigación se hace recomendable:

- Refinar e implementar el simulador de memoria virtual con las funcionalidades diseñadas.
- Incorporar su utilización al plan calendario de la asignatura de SO.
- Implementarle nuevas funcionalidades
- Continuar las investigaciones de simuladores a manera de laboratorios virtuales para la Disciplina de Sistemas Digitales.

REFERENCIAS BIBLIOGRÁFICAS.

1. MasterMagazine. *Definición de Administración de memoria*. 2006 [cited 2007 septiembre, 25]; Available from: http://www.mastermagazine.info/termino/3780.php.
2. Morelia, I.T.d. *Introducción a las Ciencias Computacionales.* 2006 [cited 2008 noviembre 12]; Available from: antares.itmorelia.edu.mx/~rvargas/intro/introTec01.pdf
3. UCI. *Estrategia de Aprendizaje. Estrategias básicas de Aprendizaje-Enseñanza*. 2005-2006.
4. UCI. *Medios de Enseñanza. Planificación de la Asignatura* 2005-2006.
5. M. Varea, A.G., P. Muñoz, N. Sánchez y G. Mamani. *Arquitectura de Computadoras*. 2007 [cited 2007 diciembre 15]; Available from: www.mauricio.varea.info/publications/notes/memoria.pdf.
6. EUITIO. *Tema 4 Gestión de Memoria*. 2007-2008 [cited 2007 diciembre 15]; Available from: 156.35.81.1/ssoo/doce/teor/t4/t4-4.pdf.
7. Mendieta, E., *Paginacion en Memoria Virtual*. 2008.
8. Valenzuela, M.G., *Gestion de Memoria*. 2002.
9. Romero, D. *Gestión de Memoria*. 2003 [cited 2008 enero 10]; Available from: http://www.elprisma.com/apuntes/ingenieria_de_sistemas/gestiondememoria/default2.asp.
10. Tamenbaum, A.S., *Sistemas Operativos Modernos*. Tercera Edición ed. 2003.
11. Stalling, W., *Sistemas Operativos*. Tercera Edición ed. 2008.
12. UCM. *Tema 7: Memoria Virtual.* 2008 [cited 2008 enero 10]; Available from: www.fdi.ucm.es/profesor/jjruz/WEB2/Temas/EC7.pdf.
13. Ray Ontko, A.R. *MOSS Memory Management Simulator* 2001 [cited 2008 enero 20]; Available from: http://www.ontko.com/moss/memory/user_guide.html.
14. Vinci, I.L.D. *Welcome to the VisualOS Project*. 2002 [cited 2008 enero25]; Available from: http://visualos-project.org/documentation.html.
15. UPV. *Arquitectura de Computadoras y Sistemas Operativos*. 2008 [cited 2008 febrero 2]; Available from: http://personales.alc.upv.es/pabmitor/index_CST_archivos/page0002.htm.
16. Gabia, J.R.P. *Diseño y simulación de una unidad de gestión de memoria virtual*. 2008 [cited 2008 febrero 3]; Available from: http://bib.fi.udc.es/cgi-bin/wwwisis/wwwisis/%5Bin=ficha.in%5D?mfn=000056.
17. es-minix.org. *Proyecto #1 - Simulación de manejo de memoria*. 2006 [cited 2008 enero 23]; Available from: http://es-minix.org/forum/viewforum.php?f=8&sid=8d07b8f1277723f5b002b887832097bb.

18. José H. Canós, P.L.y.M.C.P. *Métodologías Ágiles en el Desarrollo de Software*. [cited 2008 febrero 2]; Available from: www.willydev.net/descargas/prev/TodoAgil.Pdf.
19. James Jacobson , I.B., Grady Rumbaugh, *El proceso unificado de software*. 1999.
20. Fowler, M. *La Nueva Metodología*. 2003. [cited 2008 abril 28]; Available from: http://www.programacionextrema.org/articulos/newMethodology.es.html#tth_sEc5.
21. Castro, C.D.C. *Metodologías ágiles, un nuevo enfoque metodológico*. 2007 [cited 2008 enero 15,]; Available from: http://www.usat.edu.pe/campusvirtual/dai/investigaciones/MetodologiasAgiles.html.
22. Zarzuela, J.F., *Metodologias Agiles*. 2005.
23. Ambler, S.W. *Agile Modeling (AM)*. 2007 [cited 2008 abril, 12]; Available from: http://www.agilemodeling.com/.
24. Ambler, S.W. *Agile Modeling (AM) Home Page*. 2008 [cited 2008 febrero 13]; Available from: http://www.agilemodeling.com/.
25. baufest, *¿Que es Scrum?*
26. Orallo, E.H. *El Lenguaje Unificado de Modelado (UML)*. 2006 [cited 2008 febrero 25]; Available from: www.disca.upv.es/enheror/pdf/ActaUML.PDF.
27. Paradigm, V. *Visual Paradigm*. 2008 [cited 2008 febrero 26]; Available from: http://www.visual-paradigm.com/product/vpuml/.
28. Corporation, M. *The C# Language*. 2008 [cited 2008 febrero 25]; Available from: http://msdn.microsoft.com/en-us/vcsharp/aa336809.aspx.
29. Marañón, G.Á. *¿Que es Java?* 1999 [cited 2008 febrero 26]; Available from: http://www.iec.csic.es/CRIPTONOMICON/java/quesjava.html.
30. Microsystems, S. *The Source for Java Developers*. 2008 [cited 2008 febrero 28]; Available from: http://java.sun.com/.
31. Netbeans. *Bienvenido a NetBeans* 2008 [cited 2008 marzo2]; Available from: http://www.netbeans.org/index_es.html.
32. Foundation, T.E. *Eclipse - an open development platform*. 2008 [cited 2008 marzo 2]; Available from: http://www.eclipse.org/.
33. netbeans.org. *NetBeans Platform*. 2008 [cited 2008 marzo 2]; Available from: http://platform.netbeans.org/.
34. Beck, K., *Extreme Programming Explained*. 2000.

35. Ariel Erlijman Piwen, A.G.F. *PROBLEMAS Y SOLUCIONES EN LA IMPLEMENTACIÓN DE EXTREME PROGRAMMING*. 2001 [cited 2008 marzo 12]; Available from: www.alejandrogoyen.com/MemoriaDeGradoXP.pdf.
36. Solís, M.C. *Una explicación de la programación extrema (XP*. 2003 [cited 2008 marzo 13]; Available from: www.willydev.net/InsiteCreation/v1.0/descargas/prev/explicaxp.pdf.
37. José Carlos Cortizo Pérez, D.E.G.y.M.R.L. *eXtreme Programming*. [cited 2008 marzo 15]; Available from: www.esp.uem.es/jccortizo/xp.pdf.
38. Escribano., G.F. *Introducción a Extreme Programming.* 2002 [cited 2008 marzo 15]; Available from: www.info-ab.uclm.es/asignaturas/42551/trabajosAnteriores/Presentacion-XP.pdf
39. Ambler, S. *Agile Modeling and eXtreme Programming (XP)*. 2007 [cited 2008 marzo 21]; Available from: http://www.agilemodeling.com/essays/agileModelingXP.htm.
40. Ambler, S. *Agile/Evolutionary Data Modeling: From Domain Modeling to Physical Modeling.* 2006 [cited 2008 abril 2]; Available from: http://www.agiledata.org/essays/agileDataModeling.html.
41. Ambler, S. *UML 2 Diagramas de actividad*. 2006 [cited 2008 marzo 20]; Available from: http://www.agilemodeling.com/artifacts/activityDiagram.htm.
42. J. J. Gutiérrez, M.J.E., M. Mejías, J. Torres. *PRUEBAS DEL SISTEMA EN PROGRAMACIÓN EXTREMA.* 2006 [cited 2008 abril 5]; Available from: www.lsi.us.es/~javierj/investigacion_ficheros/PSISEXTREMA.pdf.
43. Fowler, M. *¿Ha muerto el diseño?* 2004 [cited 2008 marzo 12]; Available from: http://www.programacionextrema.org/articulos/designdead.es.html.
44. Ron Jeffries, A.A., Chet Hendrickson *Extreme Programming Installed.* Primera Edición ed. 2008.
45. Mestras, J.P. *Tarjetas CRC.* 2004 [cited 2008 abril 24]; Available from: www.fdi.ucm.es/profesor/jpavon/is2/05TarjetasCRC.pdf.
46. Toledano., M.D.D. *Cómo Desarrollar una Arquitectura Software.* 2004 [cited 2008 abril 25]; Available from: www.willydev.net/InsiteCreation/V1.0/descargas/ComoArqui.pdf
47. Anacleto, V.A. *"El rol de la arquitectura de software en las metodologías ágiles."* 2005 [cited 2008 25 mayo]; Available from: http://www.epidataconsulting.com/tikiwiki/tikiread_article.php?articleId=28#_Las_pr_cticas_de_Arquitectura_de_Software_el_dise_o_y_las_metodolog_as_giles. .
48. Larman, C., *UML y Patrones*. Primera Edición ed. 1999.

49. Díaz, M.Á. *Introducción al Diseño con Patrones, UNIVERSIDADE DE CORUÑA*. 2006 [cited 2008 mayo 2]; Available from: www.tic.udc.es/~mad/teaching/2006-07/pfc3/IntroDisenoConPatrones.pdf
50. UCI. *Arquitecura*. 2007- 2008.
51. Carlos Reynoso, N.K., *Estilos y Patrones en la Estrategia de Arquitectura de Microsoft*. 2004, UNIVERSIDAD DE BUENOS AIRES.
52. MVC. *ARQUITECTURA Modelo/Vista/Controlador*. 2007 [cited 2008 mayo 13]; Available from: http://www.ulpgc.es/otros/tutoriales/java/Apendice/mvc.html.
53. UCI. *Flujo de Implementación*. 2007-2008.
54. Ambler, S.W. *"UML 2 Class Diagrams."* 2006 [cited 2008 mayo 26]; Available from: http://www.agilemodeling.com/essays/umlDiagrams.htm. .
55. Ambler, S. *Artifacts*. 2007 [cited 2008 mayo 30]; Available from: http://www.agilemodeling.com/artifacts.
56. Sainz, M.E. *VisualOS: Manual de Usuario*. 2002 [cited 2008 mayo 20].
57. Webopedia. *Webopedia*. 2008 [cited 2008 mayo 29]; Available from: http://www.webopedia.com.

BIBLIOGRAFÍA

Libros:

1. TANEMBAUN, A.S. : *Modern Operating Systems*. 2008.Third Edition, Pearson - Prentice Hall, New Jersey.
2. STALLINGS, W.: *OPERATING SYSTEMS - Internals and Design Principles.*2005.Fifth Edition - Pearson International Edition, Pearson - Prentice Hall, New Jersey.
3. BECK,K .*Extreme Programming Expleined*.2000.Primera Edición
4. LARMAN, C*. UML y Patrones*.1999. Primera Edición.

Conferencias:

5. UCI. *Estrategia de Aprendizaje. Estrategias básicas de Aprendizaje-Enseñanza* .2005-2006
6. UCI. *Medios de Enseñanza. Planificación de la Asignatura* .2005-2006
7. UCI. *Arquitectura.*2007-2008
8. UCI*. Flujo de Implementación.* 2007-2008

Otras fuentes bibliográficas:

9. AMBLER, Scott. *Agile Modeling (AM) Home Page.*2008. [Consulta: febrero 12,2008].Disponible en: http://agilemodeling.com/.
10. NETBEANS. *Bienvenido a NetBeans*. 2008. [Consulta: marzo 2,2008]. Disponible en: http://www.netbeans.org/index_es.html.
11. UCM. *Tema 7: Memoria Virtual*. 2008. [Consulta enero 10, 2008]. Disponible en: www.fdi.ucm.es/profesor/jjruz/WEB2/Temas/EC7.pdf.
12. VINCI, I.L.D. *Welcome to the VisualOS Project*. 2002. [Consulta enero 25,2008]; Disponible en: http://visualos-project.org/documentation.html

GLOSARIO DE TÉRMINOS

- **Applets**: Un programa diseñado para ser ejecutado dentro de otra aplicación. A diferencia de una solicitud, los applets no pueden ser ejecutados directamente desde el sistema operativo. Con la creciente popularidad de OLE (Object Linking and Embedding), applets son cada vez más frecuente. Una bien diseñada applet puede ser invocado de muchas aplicaciones diferentes [57].
- **AWT:** Es parte de los framework que utiliza Java, la API estándar para proporcionar interfaces gráficas de usuario (GUIs) para programas Java [57].
- **CASE**: **C**omputer **A**ided **S**oftware **E**ngineering, una categoría de software que proporciona un entorno de desarrollo para los equipos. CASE sistemas ofrecen herramientas para automatizar, gestionar y simplificar el proceso de desarrollo [57]. Estos pueden incluir herramientas para:

    - Resumiendo necesidades iníciales.
    - Desarrollo de diagramas de flujo.
    - Programación de tareas de desarrollo.
    - Preparación de la documentación.
    - El control de versiones de software.
    - Desarrollo de código de programa.

- **Corba**: **C**ommon **O**bject **R**equest **B**roker **A**rchitecture, es una arquitectura que permite a los objetos comunicarse unos con otros independientemente de qué lenguaje de programación que fueron escritos o en qué sistema operativo se está ejecutando. CORBA fue desarrollado por un consorcio de la industria conocido como el Object Management Group (OMG) [57].
- **CVS: V**ersion **C**ontrol **S**ystem o **C**oncurrent **V**ersions **S**ystem, un código abierto, transparente red-programa que permite a los desarrolladores realizar un seguimiento de las diferentes versiones de desarrollo de código fuente. CVS no mantiene múltiples versiones de archivos de código fuente, pero mantiene una sola copia y registros de todos los cambios que se realizan. Cuando un desarrollador quiere una particular versión de desarrollo de un archivo, CVS tendrá que reconstruir la versión sobre la base de sus registro [57].

- **Hiperpaginación:** En los sistemas que utilizan la memoria virtual, generalmente causado por demasiados procesos que compiten por los escasos recursos de memoria. Para detenerlo temporalmente, es necesario poner fin a una o más aplicaciones [57].
- **HTML**: **H**yper **T**ext **M**arkup **L**anguage, lenguaje de autoría de los utilizados para crear documentos en la World Wide Web. Define la estructura y el diseño de un documento Web utilizando una variedad de etiquetas y atributos [57].
- **IDE (Integrated Development Environment):** un entorno de programación integrado en una aplicación de software que proporciona un constructor de GUI, un texto o editor de código, un compilador o intérprete y un depurador. Visual Studio, Delphi, JBuilder, FrontPage y Dreamweaver son todos ejemplos de IDEs [57].
- **Interoperabilidad:** Es la condición mediante la cual sistemas heterogéneos pueden intercambiar procesos o datos, se le suele llamar también como neutralidad tecnológica [57].
- **JDBC**: **J**ava **D**ata**b**ase **C**onnectivity, permite ejecutar programas de sentencias SQL. Esto permite que los programas de Java interactuar con cualquier sentencia de SQL compatible con la base de datos. Dado que casi todas las base de datos relacional de gestión de sistemas (DBMS), JDBC hace posible escribir una sola base de datos de aplicación que puede ejecutarse en diferentes plataformas e interactuar con diferentes DBMS [57].
- **JSP**: Abreviatura de **J**ava **S**erver **P**age, son una extensión de la tecnología Java Servlets que fue desarrollado por Sun. Tienen capacidad dinámica de secuencias de comandos que funcionan en conjunto con código HTML, separando la página lógica de los elementos estáticos, la realidad, el diseño y la visualización de la página, para ayudar a hacer el HTML más funcional (es decir, consultas a base de datos dinámicas) [57].
- **Metodología**: Conjunto de métodos que se siguen en una investigación [57].
- **.NET Framework:** Una Infraestructura de programación creado por Microsoft para construir, desplegar, ejecutar aplicaciones y servicios que utilizan. NET, tales como aplicaciones de escritorio y servicios Web [57].
- **Refactoring:** Mejorar el diseño de software de código existente. Refactoring no cambia el comportamiento observable del software, sino que mejora su estructura interna. Por ejemplo, si un programador quiere añadir nuevas funcionalidades a un programa, él puede decidir refactoriza el programa primero para simplificar la adición de nuevas funcionalidades [57].

- **RMI**: **R**emote **M**ethod **I**nvocation, es un conjunto de protocolos que se están desarrollando por parte de la división Sun JavaSoft que permite a objetos Java para comunicarse a distancia con otros objetos Java. RMI es un protocolo relativamente simple, pero a diferencia de protocolos más complejos, tales como CORBA y DCOM, que sólo funciona con los objetos Java. CORBA y DCOM están destinadas a apoyar objetos creados en cualquier idioma [57].
- **SDK: K**it de **D**esarrollo de **S**oftware, un paquete de programación que permite a un programador para desarrollar aplicaciones para una plataforma concreta. Típicamente un SDK incluye una o más APIs, herramientas de programación y documentación [57]
- **Servlet technologies:** Un pequeño programa que se ejecuta en un servidor. El término generalmente se refiere a un applet de Java que se ejecuta dentro de un entorno de servidor Web. Esto es análogo a un applet de Java que se ejecuta dentro de un navegador Web [57].
- **SGML: S**tandard **G**eneralized **M**arkup **L**anguage, un sistema para organizar y etiquetar los elementos de un documento. SGML fue desarrollado y estandarizado por la Organización Internacional de Estándares (ISO) en 1986. SGML no especifica de por sí cualquier formato, sino que especifica las normas de etiquetado de elementos. Estas etiquetas pueden ser interpretados para dar formato a elementos de diferentes maneras [57].
- **Swing:** Una biblioteca de controles GUI (por ejemplo, botones, deslizadores, casillas de verificación) que sustituye a la débil y poco inflexible AWT controles. Swing es parte de la Fundación de las clases de Java (JFC). Swing es a menudo visto como escrito Swing (Java) [57].
- **XML**: **E**xtensible **M**arkup **L**anguage, es una versión reducida de SGML, diseñado especialmente para documentos web. Permite a los diseñadores crear sus propias etiquetas personalizadas, lo que permite la definición, transmisión, la validación y la interpretación de los datos entre aplicaciones y entre organizaciones [57].

ANEXOS

1. ANEXOS 1: HISTORIAS DE USUARIOS

Historia de Usuario	
Fecha:7/3/2008	
Número: 1	**Dependencia**:-------
Nombre historia: Introducción de datos al sistema	
Prioridad en negocio: Alta Alta /media /baja	**Riesgo en desarrollo**: Alto Alta/medio/baja
Descripción: Se introducen los datos (direcciones virtuales) de forma manual o los genera la aplicación, 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 1: Historia de Usuario" Introducción de datos al sistema"

Historia de Usuario	
Fecha:7/3/2008	
Número: 2	**Dependencia**:1
Nombre historia: "Simulación de Memoria Virtual"	
Prioridad en negocio: Alta Alta /media /baja	**Riesgo en desarrollo**: Alto Alta/medio/baja
Descripción: El usuario podrá realizar la traducción de las direcciones virtuales y seleccionar la opción de utilizar o no la TLB, así como podrá observar las transformaciones que ocurren en la memoria física y las estadísticas que se producen en cada simulación. Se utilizan botones que determinan la velocidad de la simulación.	
Observaciones: La gestión de páginas se realizará mediante el esquema de paginación simple.	

Tabla 2: Historia de Usuario "Simulación de Memoria Virtual"

Historia de Usuario	
Fecha:7/3/2008	
Número: 3	**Dependencia**: 3
Nombre historia: Gestión de Resultados	
Prioridad en negocio: Media **Alta/Medio/Baja**	**Riesgo en desarrollo:** Medio **Alta/Medio/Baja**
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 3: Historia de Usuario "Gestión de Resultados"

2. ANEXOS 2: TAREAS DE INGENIERÍA:"ITERACIÓN 1"

Tarea de Ingeniería	
Número tarea:1	Número historia:1
Nombre tarea: Diseño de la interfaz "Introducir datos"	
Tipo de tarea : Desarrollo Desarrollo / Corrección / Mejora / Otra	Puntos estimados:6
Fecha inicio:	Fecha fin:
Programador responsable: Yamilé García Hernández	
Descripción: En el menú de la forma principal se selecciona "Introducir datos", se diseñará una pestaña donde se especifica si los datos (Direcciones lógicas) son introducidos de forma manual o los genera el sistema, a su vez se entran los datos esenciales como tamaño de memoria y tamaño de página. En la misma ventana existe una tabla que muestra los datos que ya han sido introducidos.	

Tabla 2.4: Tarea de Ingeniería Diseño de la interfaz "Introducir datos

Tarea de Ingeniería	
Número tarea:2	Número historia:1
Nombre tarea: Introducción de direcciones lógicas	
Tipo de tarea : Desarrollo Desarrollo / Corrección / Mejora / Otra (especificar)	Puntos estimados:5
Fecha inicio:	Fecha fin:
Programador responsable: Yamilé García Hernández	
Descripción: Según la opción del usuario se introducen los datos y se muestran los que actualmente se	

encuentren en el sistema; debe comprobar si los datos (direcciones lógicas) se generan de manera aleatoria o si el usuario los va entrar de forma manual.

Tabla 2.5: Tarea de Ingeniería "Introducción de direcciones lógicas"

Tarea de Ingeniería	
Número tarea:2	Número historia:1
Nombre tarea: Introducción datos específicos	
Tipo de tarea : Desarrollo Desarrollo / Corrección / Mejora / Otra (especificar)	Puntos estimados:5
Fecha inicio:	Fecha fin:
Programador responsable: Yamilé García Hernández	
Descripción: Según la opción del usuario se introducen los datos que determinan las características de la simulación como el tamaño de página, tamaño de memoria y el algoritmo de reemplazo deseado.	

Tabla 2.6: Tarea de Ingeniería "Introducción datos específicos"

En las tareas asociadas a la historia de usuario número 2 : "Simulación de Memoria Virtual".

Tarea de Ingeniería	
Número tarea:4	Número historia:2
Nombre tarea: Diseño de la interfaz de "Memoria en Tiempo Real"	
Tipo de tarea : Desarrollo Desarrollo / Corrección / Mejora / Otra	Puntos estimados:10
Fecha inicio:	Fecha fin:
Programador responsable: Yamilé García Hernández	

Descripción: A partir de seleccionar “Memoria en Tiempo Real” en el menú de la forma principal; se diseñará una pestaña que muestre en una tabla las transformaciones que ocurren en la memoria. En esta pestaña se muestran como disminuye la cantidad de marcos libres y como aumentan el número de fallos y el número de reemplazos

Tabla 2.7: Tarea de Ingeniería Diseño de la interfaz de “Memoria en Tiempo Real”

Tarea de Ingeniería	
Número tarea:3	Número historia:2
Nombre tarea: Traducción de las direcciones Lógicas a direcciones Reales	
Tipo de tarea : Desarrollo Desarrollo / Corrección / Mejora / Otra (especificar)	Puntos estimados:10
Fecha inicio:	Fecha fin:
Programador responsable: Yamilé García Hernández	
Descripción: Una vez introducidos los datos (direcciones lógicas) al sistema, comprobar si el usuario desea utilizar o no la TLB y se debe realizar la traducción de las direcciones.	

Tabla 2.7: Tarea de Ingeniería “Traducción de las direcciones Lógicas a direcciones Reales”

Tarea de Ingeniería	
Número tarea:5	Número historia:2
Nombre tarea: Ver Memoria en Tiempo Real	
Tipo de tarea : Desarrollo Desarrollo / Corrección / Mejora / Otra	Puntos estimados:6

Fecha inicio:	Fecha fin:
Programador responsable: Yamilé García Hernández	
Descripción: Se muestra las transformaciones que ocurren en la memoria, así como ocurren los fallos de página y el empleo de los algoritmos de reemplazo de página.	

Tabla 2.8: Tarea de Ingeniería "Ver Memoria en Tiempo Real"

Tarea de Ingeniería	
Número tarea:6	Número historia:2
Nombre tarea: Gestión de página mediante la paginación simple	
Tipo de tarea : Desarrollo Desarrollo / Corrección / Mejora / Otra	Puntos estimados:6
Fecha inicio:	Fecha fin:
Programador responsable: Yamilé García Hernández	
Descripción: Se implementan los algoritmos utilizando para la gestión de páginas la paginación simple. Las páginas se cargan bajo demanda.	

Tabla 2.9: Tarea de Ingeniería "Gestión de página mediante la paginación simple"

3. ANEXOS 3: TAREAS DE INGENIERÍA:"ITERACIÓN 2"

La historia asociada a esta iteración es la historia de usuario "Gestión los resultados" y las tareas asociadas a esta historia son:

Tarea de Ingeniería	
Número tarea:8	Número historia:3
Nombre tarea: Diseño de la interfaz de "Gestión de Resultados"	
Tipo de tarea : Desarrollo Desarrollo / Corrección / Mejora / Otra	Puntos estimados:6
Fecha inicio:	Fecha fin:
Programador responsable: Yamilé García Hernández	
Descripción: A partir de seleccionar "Resultados" en el menú de la forma principal; se diseñará una pestaña que muestre la lista de las simulaciones realizadas. Se deberá habilitar una opción para seleccionar la simulación de la cual se desea ver los resultados, otra para procesar las comparaciones entra las simulaciones realizadas. De la misma manera se mostrarán los resultados y las comparaciones entre los algoritmos.	

Tabla 2.11: Tarea de Ingeniería Diseño de la interfaz de "Gestión de Resultados"

Tarea de Ingeniería	
Número tarea:9	Número historia:3
Nombre tarea: Obtención de los resultados	
Tipo de tarea : Desarrollo Desarrollo / Corrección / Mejora / Otra	Puntos estimados:6

Fecha inicio:	Fecha fin:
Programador responsable: Yamilé García Hernández	
Descripción: Luego de ser realizadas las simulaciones se obtendrán los datos obtenidos en el proceso, se elaboraran los informes, y se establecerán las comparaciones las cuales tomarán como parámetros los indicadores seleccionados durante la simulación.	

Tabla 2.11: Tarea de Ingeniería "Obtención de los resultados"

4. ANEXOS 4: PLAN DE RELEASE

Release	Descripción de la Iteración	Orden HU a Implementar	Duración total
1	Los datos deben ser introducidos correctamente así como que el simulador reproduzca el comportamiento de la memoria virtual, incluyendo la imitación de los algoritmos de reemplazo de página donde se utilice para la gestión de estas la técnica de paginación.	Historia número 1 "Introducción de datos al sistema". Historia número 2 "Simulación de Memoria Virtual".	3 semanas
2	Se pretende entregar el simulador que imite el comportamiento de la memoria virtual donde los resultados obtenidos puedan ser guardados y se puedan realizar comparaciones entre las simulaciones producidas.	Historia número 3 "Gestión los resultados".	2 semanas

5. ANEXOS 5: PRUEBA DE ACEPTACIÓN “ITERACIÓN 1”

Pruebas de Aceptación: Historia de Usuario 1

Prueba de Aceptación	
Código Caso de Prueba:1	Número Historia de Usuario:1
Nombre del Caso de Prueba: Selección de direcciones válidas de 16 bytes.	
Entrada: El usuario entra los datos.	
Resultado Esperado: El sistema tomará estos datos como direcciones lógicas, y continuará su ejecución.	
Evaluación de la Prueba:	

Prueba de Aceptación	
Código Caso de Prueba:2	Número Historia de Usuario:1
Nombre del Caso de Prueba: Verificación de direcciones válidas.	
Entrada: El usuario entra los datos arbitrarios.	
Resultado Esperado: El sistema dará error y le pedirá entrar valores válidos.	
Evaluación de la Prueba:	

Prueba de Aceptación	
Código Caso de Prueba:3	Número Historia de Usuario:1
Nombre del Caso de Prueba: Datos específicos.	

Entrada: El usuario entrará correctamente el tamaño de memoria, el de página y el algoritmo deseado.
Resultado Esperado: El sistema debe adaptarse a esas características.
Evaluación de la Prueba:

Prueba de Aceptación	
Código Caso de Prueba:3	Número Historia de Usuario:1
Nombre del Caso de Prueba: Verificar la entrada de datos específicos.	
Entrada: El usuario entra un tamaño de memoria menor al tamaño de página.	
Resultado Esperado: El sistema debe lanzar un mensaje de error y pedir que se entre un mayor tamaño de memoria.	
Evaluación de la Prueba:	

Pruebas de Aceptación: Historia de Usuario 2

Prueba de Aceptación	
Código Caso de Prueba:4	Número Historia de Usuario:2
Nombre del Caso de Prueba: Realizar traducción de las direcciones.	
Entrada: El usuario desea iniciar la simulación.	
Resultado Esperado: El sistema debe realizar la simulación de la traducción de las direcciones lógicas	
Evaluación de la Prueba:	

Prueba de Aceptación	
Código Caso de Prueba:5	Número Historia de Usuario:2
Nombre del Caso de Prueba: Verificar traducción de direcciones lógicas.	
Entrada: Direcciones lógicas menores o mayores a 16 bytes.	
Resultado Esperado: El sistema lanza un mensaje de error y pide que se entre las direcciones lógicas de 16 bytes para realizar su traducción.	
Evaluación de la Prueba:	

Prueba de Aceptación	
Código Caso de Prueba:6	Número Historia de Usuario:2
Nombre del Caso de Prueba: Ver memoria en tiempo real	
Entrada: Se carga la memoria física.	
Resultado Esperado: Dada la traducción de las direcciones lógicas estas son cargadas en la memoria fisca y se muestra las transformaciones que ocurren, en conjunto con la imitación de los algoritmos de reemplazo.	
Evaluación de la Prueba:	

Prueba de Aceptación	
Código Caso de Prueba:7	Número Historia de Usuario:2
Nombre del Caso de Prueba: Verificación de la memoria física.	

Entrada: Se carga la memoria física.
Resultado Esperado: No se observan las transformaciones que ocurren en la memoria física, nila imitación de los algoritmos de reemplazo.
Evaluación de la Prueba:

6. ANEXOS 6: PRUEBAS DE ACEPTACIÓN”ITERACIÓN 2”

Pruebas de Aceptación: Historia de Usuario 3

Prueba de Aceptación	
Código Caso de Prueba:8	Número Historia de Usuario:3
Nombre del Caso de Prueba: Obtención de Indicadores.	
Entrada: Se obtienen los indicadores y los resultados de la simulación	
Resultado Esperado: Se obtienen los resultados de la simulación junto a los indicadores.	
Evaluación de la Prueba:	

Prueba de Aceptación	
Código Caso de Prueba:8	Número Historia de Usuario:3
Nombre del Caso de Prueba: Verificación de los resultados	
Entrada: Los indicadores que se obtiene no se corresponden con los resultados de la simulación.	
Resultado Esperado: Mensaje de error, el sistema pide que se repita la simulación.	
Evaluación de la Prueba:	

Prueba de Aceptación	
Código Caso de Prueba:8	Número Historia de Usuario:3
Nombre del Caso de Prueba: Comparaciones entre algoritmos	

Entrada: Dados los indicadores ,especificar la obtención de comparaciones entre los algoritmos
Resultado Esperado: Se obtienen las comparaciones entre los algoritmos
Evaluación de la Prueba:

Prueba de Aceptación	
Código Caso de Prueba:8	Número Historia de Usuario:8
Nombre del Caso de Prueba: Las comparaciones obtenidas	
Entrada: Se pide obtener comparación y no se especifica los indicadores que se utilizan como parámetros	
Resultado Esperado: El sistema lanza una error y pide especificación de los parámetros	
Evaluación de la Prueba:	