ISTRED
INSTITUTO SUPERIOR TÉCNICO REY DAVID

PERIODO I - 2024 - 2025

GUÍA DEL APRENDIZAJE DEL ESTUDIANTE

TEORÍA DE SISTEMAS

ING. JUSSEN FACUY DELGADO

GESTIÓN DE BASE DE DATOS
1ER. SEMESTRE

Contenido

I. DATOS GENERALES DE LA ASIGNATURA 8

Información General 8

Requisitos 8

Descripción de la asignatura 8

Objetivos de la asignatura: 8

Introducción de la Asignatura 8

I. DATOS GENERALES DE LA UNIDAD 1 10

Información General 10

Resultado de Aprendizaje 10

Metodología 10

DESARROLLO 11

Unidad 1: Partes y funciones de computadores y servidores 11

Tema 1: Funcionalidades de un computador 11

Importancia del Computador: 11

Hardware-Software: 12

Memoria, Disco, Tarjeta Madre, Tarjetas Gráficas, Unidades de Almacenamiento: ... 12

Características 13

Beneficios de la memoria RAM 14

Consejos para elegir la memoria RAM adecuada 15

Tipos de discos duros 16

Beneficios de los discos duros 17

Procesador y Buses: 19
ACTIVIDADES PRÁCTICAS 21
FORO DE DEBATE 22
PRÁCTICA INVESTIGATIVA 22
PRÁCTICA TEST 23
Tema 2: Definición de Servidor 24
Definición de servidor 24
Aspectos Hardware-Software 24
Aspectos Clave en Hardware: 24
ACTIVIDADES PRÁCTICAS 27
FORO DE DEBATE 28
PRÁCTICA INVESTIGATIVA 28
PRÁCTICA TEST 29
II. DATOS GENERALES DE LA UNIDAD 2 30
Información General 30
Resultado de Aprendizaje 30
Metodología 30
DESARROLLO 31
Unidad 2: Sistemas operativos 31
Tema 1: Sistemas operativos 31
Definición Sistema Operativo 31
Funciones y objetivos del sistema operativo 31

Gestión de Procesos: 32
Gestión de Memoria: 32
Sistema de Archivos: 32
Gestión de Dispositivos de Entrada/Salida: 32
Gestión de Recursos del Sistema: 32
Objetivos del sistema operativo 33
Tipos y Características de los sistemas operativos 34
Tipos 34
Características de los Sistemas Operativos: 35
Evolución de los sistemas operativos 36
Sin Sistemas Operativos (1940-1950): 36
Lote por Lote (Batch Processing) (1950-1960): 36
Sistemas de Tiempo Compartido (1960-1970): 36
Sistemas Multitarea (1970-1980): 36
Sistemas Operativos de Red (1980-1990): 36
Sistemas Operativos para Computadoras Personales (1980-1990): 37
Sistemas Operativos Móviles (2000 en adelante): 37
ACTIVIDADES PRÁCTICAS 38
FORO DE DEBATE 39
PRÁCTICA INVESTIGATIVA 39
PRÁCTICA TEST 40
Tema 2: Software libre y propietario 41

Definición 41
Ventajas y desventajas entre software libre y Propietario 42
Ventajas del Software Libre: 42
Desventajas del Software Libre: 43
Ventajas del Software Propietario: 43
Desventajas del Software Propietario: 44
Patentes 44
Visión histórica del software 45
Década de 1940 - Primeros Pasos: 45
Década de 1970 - Revolución de los Microprocesadores: 46
Década de 1980 - Computadoras Personales y Software Comercial: 46
Década de 1990 - Internet y Software Distribuido: 46
Década de 2000 - Software de Código Abierto y Móvil: 46
Década de 2010 - Computación en la Nube y Software como Servicio (SaaS): 46
Década de 2020 - Inteligencia Artificial y Desarrollo Ágil: 46
ACTIVIDADES PRÁCTICAS 47
FORO DE DEBATE 48
PRÁCTICA INVESTIGATIVA 48
PRÁCTICA TEST 49
III. DATOS GENERALES DE LA UNIDAD 3 50
Información General 50
Resultado de Aprendizaje 50

Metodología 50

DESARROLLO 51

Unidad 3: Paradigmas y lenguajes de programación 51

Tema 1: Paradigmas 51

Ejemplos de Paradigmas: 51

Tipos de Paradigmas en Teoría de Sistemas: 52

Tendencias en Paradigmas de Programación: 53

ACTIVIDADES PRÁCTICAS 54

FORO DE DEBATE 55

PRÁCTICA INVESTIGATIVA 55

PRÁCTICA TEST 56

Tema 2: Lenguaje de programación 57

Lenguaje de programación en Teoría de Sistemas: 57

Definición 57

Tipos de Paradigmas en Teoría de Sistemas: 58

Lenguajes Orientados a Objetos: 59

Lenguajes Funcionales: 59

Lenguajes de Scripting: 59

Tendencias de los lenguajes de programación: 60

ACTIVIDADES PRÁCTICAS 61

FORO DE DEBATE 62

PRÁCTICA INVESTIGATIVA 62

PRÁCTICA TEST 63
IV. DATOS GENERALES DE LA UNIDAD 4 64
Información General 64
Resultado de Aprendizaje 64
Metodología 64
DESARROLLO 65
Unidad 4: Enfoques de Programación 65
Tema 1: La programación 65
La Programación 65
Clasificación de la Programación 65
Programación Imperativa 65
Programación Estructurada 66
Programación Orientada a Objetos 67
ACTIVIDADES PRÁCTICAS 68
FORO DE DEBATE 69
PRÁCTICA INVESTIGATIVA 69
PRÁCTICA TEST 70
Tema 2: Enfoques 71
Tendencia de Desarrollo 71
Clouds (SAAS, PAAS, IAS): 72
Tendencia de Desarrollo ERP 72
ACTIVIDADES PRÁCTICAS 74

FORO DE DEBATE 75

PRÁCTICA INVESTIGATIVA 75

PRÁCTICA TEST 76

V. Bibliografía 77

Bibliografía básica 77

Bibliografía de consulta 77

Anexo 1: Modelo de Foro 78

Anexo 2: Esquema de Ensayo 79

Anexo 3: Esquema de Taller/Investigación 80

Anexo 4: Rúbrica de Foro 81

Anexo 5: Rubrica de Ensayo 82

Anexo 6: Rúbrica de Taller/Investigación 83

I. DATOS GENERALES DE LA ASIGNATURA

Información General

1.1.	Asignatura	Teoría de Sistemas					
1.2.	Carrera	Gestión de Base de Datos					
1.3.	Código de asignatura	GBD1236					
1.4.	Créditos	1,2					
1.5.	Nivel	Primero					
1.6.	Detalle de horas	ACD	24	AA	16	APE	20

Requisitos

Prerrequisitos		Correquisitos	
Asignatura	Código	Asignatura	Código
N/A	N/A	Base de datos	GBD2239

Descripción de la asignatura

La Teoría de Sistemas es una disciplina interdisciplinaria que se enfoca en el estudio de sistemas complejos y su comportamiento. Un sistema se define como un conjunto de elementos interconectados que interactúan entre sí para lograr un objetivo común. Esta asignatura introduce a los estudiantes a los principios fundamentales de la teoría de sistemas, sus aplicaciones en diversos campos y cómo se pueden utilizar para abordar problemas y desafíos en la vida real.

Objetivos de la asignatura:

Adquirir criterios técnicos que permitan al alumno seleccionar las características de hardware y software para un servidor acorde a las necesidades de las empresas y entidades gubernamentales.

Introducción de la Asignatura

La Teoría de Sistemas, una disciplina multidisciplinaria que nos invita a explorar el funcionamiento interno de las tecnologías que sustentan nuestro mundo digital, es un pilar fundamental en la era de la información. En esta asignatura, embarcaremos juntos en un

viaje que desglosará los complejos entresijos de los computadores, servidores, sistemas operativos y lenguajes de programación que hacen posible la tecnología moderna.

I. DATOS GENERALES DE LA UNIDAD 1

Información General

1.1.	Nivel	Primero					
1.2.	Nombre Unidad	Partes y funciones de computadores y servidores					
1.3.	Tema 1 – Unidad	Funcionalidades de un computador					
1.4.	Tema 2 – Unidad	Definición de servidor					
1.5.	Unidad organizacional	Unidad Profesional					
1.6.	Unidad	1					
1.7.	Total, Horas Unidad	15					
1.8.	Detalle de horas Unidad	ACD	6	AA	4	APE	5

Resultado de Aprendizaje

En la Unidad 1 de Teoría de Sistemas, los estudiantes aprenderán a definir y comprender qué es un servidor, sus características y tipos. Analizarán sus funciones dentro de una red. También se abordarán prácticas de seguridad, mantenimiento y tendencias emergentes en tecnología de servidores, preparando a los estudiantes para manejar y administrar servidores eficazmente en diversos contextos.

Metodología

La clase virtual grabada comienza con una introducción clara sobre lo que los estudiantes aprenderán en esa sesión. Se establecen los objetivos de aprendizaje para que los estudiantes sepan qué esperar y en qué enfocarse.

El contenido de la asignatura se desarrolla dividiéndolo en secciones o módulos. Se utiliza una estructura organizada y lógica para abordar cada tema, combinando diferentes enfoques pedagógicos según el contenido, como explicaciones, ejemplos, demostraciones, gráficos y diagramas.

Aunque la clase es grabada, es importante incorporar elementos interactivos para mantener el compromiso de los estudiantes. Esto se puede lograr a través de preguntas para reflexionar, ejercicios, cuestionarios o pausas para discusión, fomentando su participación activa.

DESARROLLO

Unidad 1: Partes y funciones de computadores y servidores

Tema 1: Funcionalidades de un computador

- Hardware-Software
- Memoria, disco, tarjeta madre, tarjetas gráficas, unidades de almacenamiento
- Dispositivos de entrada y salida
- Procesador y Buses

Objetivo: Proporcionar a los estudiantes una comprensión sólida de los componentes fundamentales que componen un computador y un servidor.

Importancia del Computador:

La importancia del computador en la sociedad moderna radica en su capacidad para procesar información, automatizar tareas, y facilitar la comunicación y el acceso a la información. Desde su invención, los computadores han transformado la forma en que trabajamos, nos comunicamos y vivimos.

Radica en su función como herramienta versátil y poderosa que ha transformado prácticamente todos los aspectos de la sociedad y la vida moderna. Los computadores son fundamentales en la gestión de datos, el procesamiento de información, la automatización de tareas, la comunicación global y la toma de decisiones en una amplia variedad de campos, desde la ciencia y la industria hasta la educación y el entretenimiento.

Sistemas Informáticos:

Los sistemas informáticos son conjuntos de componentes interconectados que incluyen hardware, software, datos y personas, diseñados para adquirir, procesar, almacenar y transmitir información de manera eficiente. Estos sistemas pueden variar en tamaño y complejidad, desde dispositivos personales como teléfonos inteligentes y computadoras portátiles hasta redes empresariales y servidores en la nube. Los sistemas informáticos desempeñan un papel fundamental en la gestión de datos y la toma de decisiones en organizaciones y la vida cotidiana.

Los computadores son sistemas complejos que desempeñan una variedad de funcionalidades esenciales en la sociedad actual. Para comprender mejor estas funcionalidades, es importante desglosar el concepto en elementos clave, incluyendo hardware y software, memoria, dispositivos de entrada y salida, así como el procesador y los buses.

Hardware-Software:

Los computadores operan a través de la interacción entre el hardware y el software.

Hardware: El hardware de un computador comprende todos los componentes físicos, desde la CPU (Unidad Central de Procesamiento) hasta la pantalla. Incluye la memoria, la tarjeta madre, las tarjetas gráficas, las unidades de almacenamiento y los dispositivos de entrada y salida. Estos componentes trabajan en conjunto para ejecutar las instrucciones proporcionadas por el software.

Software: El software abarca todos los programas y sistemas operativos que permiten al usuario interactuar con el hardware. Los sistemas operativos, como Microsoft Windows o macOS, actúan como intermediarios entre el hardware y las aplicaciones de software. Las aplicaciones, como navegadores web o procesadores de texto, realizan tareas específicas y utilizan los recursos de hardware para lograrlo.

Memoria, Disco, Tarjeta Madre, Tarjetas Gráficas, Unidades de Almacenamiento:

Memoria: La memoria de un computador se divide en varias categorías, siendo la RAM (Memoria de Acceso Aleatorio) la más conocida. La RAM almacena temporalmente los datos y programas en uso, lo que permite un acceso rápido a ellos. Cuanta más RAM tenga un computador, más eficiente será en la ejecución de múltiples tareas.

La memoria RAM, o memoria de acceso aleatorio, es un tipo de memoria volátil que se utiliza para almacenar datos e instrucciones que se están utilizando actualmente por el procesador. Es un tipo de memoria de acceso aleatorio, lo que significa que el procesador puede acceder a cualquier dato o instrucción almacenado en ella de forma aleatoria, sin tener que seguir un orden específico.

- **Memoria volátil:** La memoria RAM es una memoria volátil, lo que significa que los datos almacenados en ella se pierden cuando se apaga el ordenador.

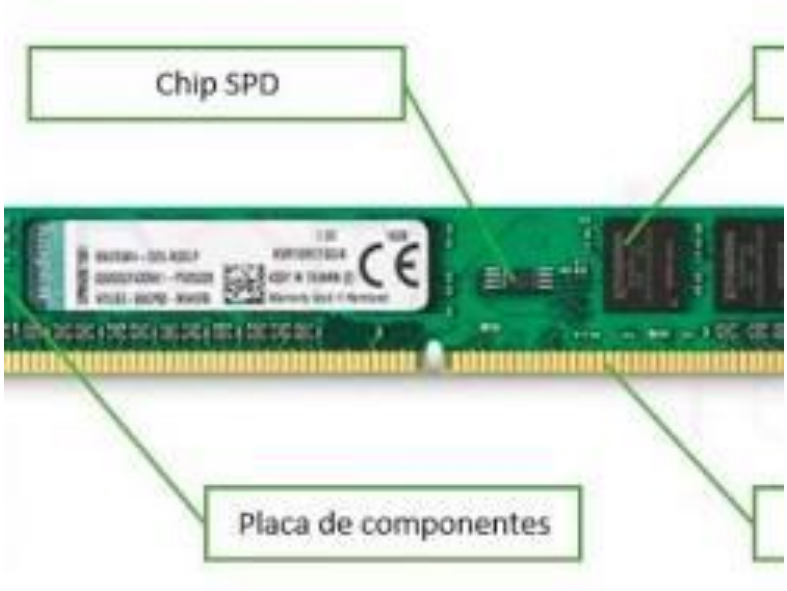
- **Almacena datos e instrucciones:** La memoria RAM almacena datos e instrucciones que se están utilizando actualmente por el procesador.
- **Acceso aleatorio:** El procesador puede acceder a cualquier dato o instrucción almacenado en la memoria RAM de forma aleatoria, sin tener que seguir un orden específico.

Características

- **Tecnología:** La memoria RAM está disponible en diferentes tecnologías, como SDRAM, DDR, DDR2, DDR3, DDR4 y DDR5. Cada tecnología tiene sus propias características, como la velocidad, la capacidad y la latencia.
- **Capacidad:** La capacidad de la memoria RAM se mide en megabytes (MB) o gigabytes (GB). Una mayor capacidad de memoria RAM permite almacenar más datos e instrucciones.
- **Frecuencia o velocidad:** La frecuencia o velocidad de la memoria RAM indica las operaciones (lecturas y escrituras) que puede realizar en un segundo. Se mide en megahercios (MHz), es decir, millones de operaciones por segundo.
- **Transferencia de datos:** La transferencia de datos de la memoria RAM indica la cantidad de datos que puede transferir entre el procesador y la memoria RAM en un segundo. Se mide en bytes por segundo.
- **Timings (Latencias):** Los timings (latencia) de la memoria RAM indican el tiempo que tarda el procesador en acceder a los datos almacenados en la memoria RAM. Se mide en nanosegundos (ns).
- **Latencia efectiva:** La latencia efectiva de la memoria RAM es la suma de la latencia de acceso y la latencia de búsqueda.
- **Voltaje:** La memoria RAM requiere un cierto voltaje para funcionar.
- **Perfiles:** La memoria RAM puede estar disponible en diferentes perfiles, como XMP y DOCP. Estos perfiles permiten al usuario aumentar la velocidad y el rendimiento de la memoria RAM.

Figura 1

Memoria Ram de un computador de escritorio o desktop



Nota. El grafico representa los principales componentes electrónicos de una MRam

Los componentes de la memoria RAM son los siguientes:

Componentes de la memoria RAM

Módulo de memoria: El módulo de memoria es la pieza física de la memoria RAM.

- **Chips de memoria:** Los chips de memoria son los componentes que almacenan los datos y las instrucciones.
- **Conectores:** Los conectores permiten conectar el módulo de memoria a la placa base.

Beneficios de la memoria RAM

La memoria RAM tiene los siguientes beneficios:

- **Mejora el rendimiento del sistema:** Una mayor capacidad de memoria RAM permite almacenar más datos e instrucciones, lo que puede mejorar el rendimiento del sistema.

- **Aumenta la capacidad de respuesta:** Una mayor velocidad de memoria RAM puede reducir el tiempo que tarda el procesador en acceder a los datos, lo que puede mejorar la capacidad de respuesta del sistema.

Consejos para elegir la memoria RAM adecuada

A la hora de elegir la memoria RAM adecuada, hay que tener en cuenta los siguientes factores:

- **La tecnología:** La tecnología de la memoria RAM debe ser compatible con la placa base.
- **La capacidad:** La capacidad de la memoria RAM debe ser suficiente para las necesidades del sistema.
- **La frecuencia:** La frecuencia de la memoria RAM debe ser compatible con la placa base y el procesador.
- **Los timings:** Los timings de la memoria RAM deben ser compatibles con la placa base y el procesador.
- **El voltaje:** El voltaje de la memoria RAM debe ser compatible con la placa base.

Tarjeta Madre: La tarjeta madre (o placa base) es el componente principal que conecta y comunica todos los demás componentes del computador. Contiene el CPU, la RAM, las ranuras de expansión y los puertos de E/S. Es esencial para el funcionamiento del sistema.

Tarjetas Gráficas: Las tarjetas gráficas (GPU) se encargan del procesamiento de imágenes y video. Son fundamentales para la representación visual en una pantalla y son especialmente importantes para juegos y aplicaciones gráficas.

Unidades de Almacenamiento: Además de los discos duros, las unidades de almacenamiento incluyen dispositivos como unidades de estado sólido (SSD) y unidades flash USB. Estos dispositivos almacenan datos a largo plazo y se utilizan para almacenar sistemas operativos, aplicaciones y archivos personales.

Disco Duro: Los discos duros son dispositivos de almacenamiento a largo plazo. Almacenan datos incluso cuando el computador está apagado. Los discos duros tradicionales (HDD) utilizan discos magnéticos giratorios, mientras que las unidades de estado sólido (SSD) son más rápidas y confiables debido a la falta de partes móviles.

Los discos duros son dispositivos de almacenamiento que utilizan discos magnéticos para almacenar datos. Los discos duros son el tipo de almacenamiento más común para los ordenadores personales.

Figura 2

Interior de disco duro, con disco magnéticos



Nota. El grafico representa el interior de un disco dura donde existe un plato magnético y aguja que permite leer la cabecera y pistas del disco.

Características de los discos duros

- Capacidad: La capacidad de un disco duro se mide en gigabytes (GB) o terabytes (TB). Una mayor capacidad permite almacenar más datos.
- Velocidad: La velocidad de un disco duro se mide en revoluciones por minuto (RPM). Las RPM más altas permiten transferir datos más rápido.
- Interfaz: La interfaz de un disco duro es el tipo de conexión que utiliza para conectarse al ordenador. Las interfaces más comunes son SATA y SAS.

Tipos de discos duros

Hay dos tipos principales de discos duros: mecánicos y de estado sólido.

- Discos duros mecánicos: Los discos duros mecánicos utilizan discos magnéticos para almacenar datos. Los discos duros mecánicos son más baratos que los discos duros de estado sólido, pero son más lentos.

Figura 3

Disco duro mecánico



Nota. La imagen representa un disco duro de característica mecánico que tiene la capacidad de 1Tera byte

- Disco de estado sólido (SSD): Los discos de estado sólido utilizan memoria flash para almacenar datos. Los discos de estado sólido son más rápidos que los discos duros mecánicos, pero son más caros.

Figura 4

Disco de estado solido



Nota. La figura muestra un SSD es el acrónimo de Solid State Disk que si lo traducimos a nuestro idioma se define como " disco de estado sólido ".

Beneficios de los discos duros

Los discos duros ofrecen los siguientes beneficios:

- Gran capacidad de almacenamiento: Los discos duros ofrecen una gran capacidad de almacenamiento, lo que los hace ideales para almacenar archivos multimedia, juegos y otros datos de gran tamaño.

- Precio asequible: Los discos duros son relativamente asequibles, lo que los hace una opción popular para los ordenadores personales.
- Facilidad de uso: Los discos duros son fáciles de instalar y usar.

Consejos para elegir un disco duro

Al elegir un disco duro, hay que tener en cuenta los siguientes factores:

- La capacidad: La capacidad del disco duro debe ser suficiente para las necesidades del usuario.
- La velocidad: La velocidad del disco duro es importante para tareas como el arranque del sistema y la carga de aplicaciones.
- El precio: El precio del disco duro es un factor importante por considerar.

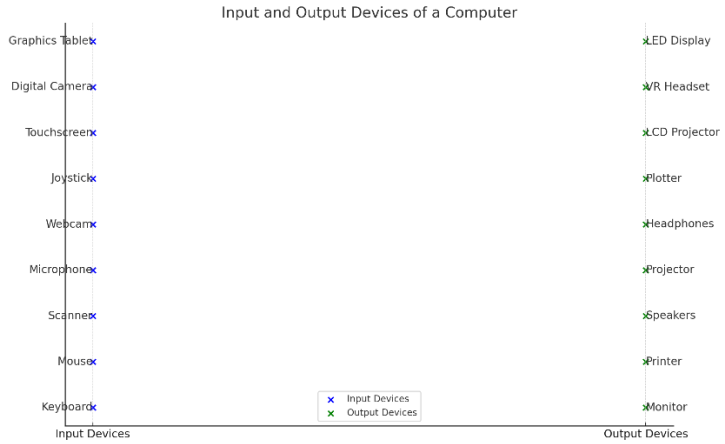
Dispositivos de Entrada y Salida:

Dispositivos de Entrada: Estos incluyen teclados, ratones, pantallas táctiles, cámaras web y micrófonos, entre otros. Permiten al usuario interactuar con el computador al ingresar datos y comandos.

Dispositivos de Salida: Los dispositivos de salida incluyen monitores, impresoras, altavoces y dispositivos hápticos. Proporcionan resultados visuales, impresos o auditivos basados en las acciones del usuario y las operaciones del sistema.

Figura 5

Dispositivos de entrada y salida



Nota. Izquierda dispositivos de entrada, derecha dispositivos de salida

Procesador y Buses:

Procesador (CPU): La CPU es el cerebro del computador y realiza cálculos y ejecuta instrucciones. Cuanto más potente sea la CPU, más rápido podrá ejecutar tareas. Los avances en la miniaturización y la arquitectura de la CPU han permitido un rendimiento excepcional.

Figura 6

Procesador o CPU



Nota: Procesador de la marca Intel, CPU Central Processing Unit, o unidad central de procesamiento, es un componente de hardware y la unidad computacional central

Buses: Los buses son las rutas de comunicación que permiten que los datos se muevan entre los componentes del computador. Los buses de datos, de dirección y de control facilitan la transferencia de información dentro del sistema. Un sistema de buses eficiente es esencial para un rendimiento óptimo.

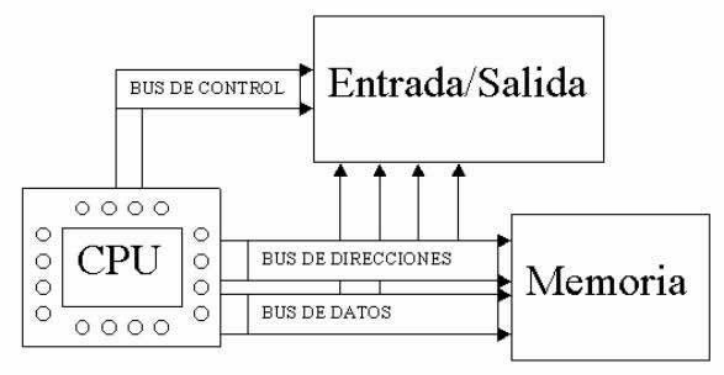
De dirección son las encargadas de indicar la posición de memoria o el dispositivo con el que se desea establecer comunicación. El "ancho de canal" explica así mismo la cantidad de ubicaciones o Direcciones diferentes que el microprocesador puede alcanzar

De control transportan señales de estado de las operaciones efectuadas por la CPU con las demás unidades. El método utilizado por el ordenador para sincronizar las distintas operaciones es por medio de un reloj interno que posee el ordenador y facilita la sincronización y evita las colisiones de operaciones (unidad de control). Estas operaciones se transmiten en un modo bidireccional.

De datos mueve los datos entre los dispositivos del hardware: de Entrada como el teclado, el escáner, el ratón, etc.; de salida como la Impresora, el Monitor o la tarjeta de Sonido; y de Almacenamiento como el Disco Duro, el Disquete o la Memoria-Flash.

Figura 7

Esquema de comunicación de los BUSES



Nota: Representación esquematizada y la forma en que se comunica los buses

ACTIVIDADES PRÁCTICAS

Tabla

Detalle de la Práctica

Practicas	Instrucciones	Duración de la Practica	Mecanismo de Evaluación
Foro de debate	Revisar Actividades Prácticas	0:30 mm	Por rúbricas
Práctica investiga: Taller/Ensayo/Investigación	Revisar Actividades Prácticas	3:30 h	Por rúbricas
Tutorías Prácticas	Revisar Actividades Prácticas	2:00 mm	Por participación en clase
Test evaluativo	Revisar Actividades Prácticas	0:30 mm	Formativa

Nota. Elaboración propia

Consideraciones Generales

1. Lee y revisa la Guía de Estudio.
2. Observa el video de la clase magistral.
3. Participa activamente en el FORO DEBATE.
4. Revisa en el calendario el día y hora en que se realizará las actividades prácticas y la tutoría.
5. Realizar la práctica propuesta por el docente.
6. Revisa las Rúbricas acorde a la actividad prácticas.
7. Toma los principales apuntes que vayas rescatando del texto.
8. Reflexiona sobre los temas tratados en la clase.
9. Sube el documento en formato PDF en la plataforma Moodle.
10. Revisa la calificación de tus actividades prácticas.

FORO DE DEBATE

Tema: ¿Cuál es el componente más crítico de un computador y por qué? Justifica tu respuesta con ejemplos prácticos

Instrucciones para Participar:

- Exponga su punto de vista acerca del tema.
- Comenta las respuestas de otros participantes.
- Promueve un ambiente respetuoso y constructivo.

PRÁCTICA INVESTIGATIVA

Ensayo: Las partes y funciones de computadoras y servidores.

Consigna:

- El trabajo de investigación debe presentarse de manera estructurada y coherente para comunicar efectivamente los hallazgos y conclusiones.
- Asegúrese de que el estilo y formato del trabajo sigan las pautas de citación y presentación establecidas por la institución.
- Utilice un lenguaje claro y evite jergas o términos técnicos innecesarios.
- Revise y corrija la redacción y la gramática antes de la presentación final (ver anexo).

PRÁCTICA TEST

Antes de realizar el test, es importante considerar varios aspectos para aprovechar al máximo la experiencia.

Consideraciones para realizar el test

1. Lee y revisa el documento.
2. Observa el video de la clase magistral.
3. Revisa en el calendario el día y hora en que se debe de realizar el Test.
4. Realiza el Test de la unidad considerando lo Siguiente:

- No cambiarte entre paginas durante la realización del Test.
- Revisa que tu conexión a internet sea estable.
- Recuerda que todo el contenido está inmerso dentro del documento base.

5. Revisa tu puntuación, de inmediato al finalizar de la actividad.

Tema 2: Definición de Servidor

- Aspectos Hardware-Software
- Tipos de servidor

Objetivo: Proporcionar a los estudiantes una comprensión sólida de los componentes fundamentales que componen un computador y un servidor.

Definición de servidor

Un servidor es una computadora o un sistema informático dedicado que proporciona servicios, recursos o datos a otros dispositivos o usuarios, conocidos como clientes, a través de una red. Los servidores desempeñan un papel fundamental en la infraestructura de las tecnologías de la información y son responsables de gestionar y compartir recursos de manera eficiente. Los servicios que un servidor puede ofrecer varían ampliamente y pueden incluir el alojamiento de sitios web, el almacenamiento de datos, el correo electrónico, la gestión de bases de datos, la autenticación de usuarios y muchos otros.

Aspectos Hardware-Software

Un servidor, ya sea para servicios web, almacenamiento de datos, correo electrónico u otras funciones, debe tener ciertos aspectos tanto en hardware como en software para garantizar su rendimiento, confiabilidad y seguridad. A continuación, se detallan los aspectos clave en ambas categorías:

Aspectos Clave en Hardware:

Potencia de Procesamiento: Un servidor debe estar equipado con un procesador (CPU) potente que pueda manejar cargas de trabajo intensivas y ejecutar aplicaciones eficientemente.

Memoria RAM (Random Access Memory): La cantidad de memoria RAM es fundamental para el rendimiento de un servidor. Cuanta más RAM tenga, más aplicaciones y usuarios podrá atender simultáneamente.

Almacenamiento: Los servidores requieren unidades de almacenamiento de alta capacidad y velocidad, como discos duros o unidades de estado sólido (SSD). La elección dependerá de la necesidad de almacenamiento y rendimiento.

Redundancia de Alimentación: Un servidor debe contar con fuentes de alimentación redundantes para evitar interrupciones debido a fallos eléctricos.

Redundancia de Hardware: Para aumentar la fiabilidad, se utilizan componentes redundantes, como discos duros, fuentes de alimentación y ventiladores, de modo que si un componente falla, el servidor pueda seguir funcionando.

Conectividad de Red: Los servidores necesitan múltiples interfaces de red, conexiones Ethernet y, en algunos casos, adaptadores de red avanzados para garantizar la conectividad confiable y la redundancia de red.

Sistema de Enfriamiento: Los servidores generan calor, por lo que es importante contar con sistemas de enfriamiento eficientes, como ventiladores y sistemas de refrigeración por aire o líquido.

Gestión Remota: La capacidad de administrar y supervisar el servidor de forma remota es esencial para el mantenimiento y la resolución de problemas.

Aspectos Clave en Software:

Sistema Operativo: El servidor debe estar equipado con un sistema operativo diseñado para entornos de servidor, como Windows Server, Linux (por ejemplo, CentOS, Ubuntu Server) o FreeBSD.

Software de Seguridad: Se deben implementar soluciones de seguridad, como cortafuegos, software antivirus, y sistemas de detección y prevención de intrusiones.

Software de Gestión de Servidores: Se utilizan herramientas de gestión de servidores, como Microsoft System Center, para administrar y supervisar el servidor de manera eficiente.

Software de Virtualización: En muchos entornos de servidor, se utiliza software de virtualización, como VMware, Hyper-V o KVM, para crear y gestionar máquinas virtuales.

Software de Copia de Seguridad y Recuperación: Se implementan soluciones de copia de seguridad para proteger los datos críticos y garantizar la recuperación en caso de fallos.

Software de Servicios: Dependiendo del propósito del servidor, se instalan aplicaciones y servicios específicos, como servidores web (por ejemplo, Apache, Nginx), bases de datos (por ejemplo, MySQL, PostgreSQL), servicios de correo electrónico, etc.

Actualizaciones y Parches: Mantener el software actualizado con las últimas actualizaciones y parches de seguridad es esencial para proteger el servidor contra vulnerabilidades conocidas.

Políticas de Acceso y Seguridad: Se deben establecer políticas de acceso y seguridad para controlar quién puede acceder al servidor y qué privilegios tienen los usuarios.

Monitorización y Registro: El servidor debe contar con herramientas de monitorización y registro para supervisar el rendimiento, identificar problemas y mantener un registro de eventos.

ACTIVIDADES PRÁCTICAS

Tabla

Detalle de la Práctica

Practicas	Instrucciones	Duración de la Practica	Mecanismo de Evaluación
Foro de debate	Revisar Actividades Prácticas	0:30 mm	Por rúbricas
Práctica investiga: Taller/Ensayo/Investigación	Revisar Actividades Prácticas	3:30 h	Por rúbricas
Tutorías Prácticas	Revisar Actividades Prácticas	2:00 mm	Por participación en clase
Test evaluativo	Revisar Actividades Prácticas	0:30 mm	Formativa

Nota. Elaboración propia

Consideraciones Generales

1. Lee y revisa la Guía de Estudio.
2. Observa el video de la clase magistral.
3. Participa activamente en el FORO DEBATE.
4. Revisa en el calendario el día y hora en que se realizará las actividades prácticas y la tutoría.
5. Realizar la práctica propuesta por el docente.
6. Revisa las Rúbricas acorde a la actividad prácticas.
7. Toma los principales apuntes que vayas rescatando del texto.
8. Reflexiona sobre los temas tratados en la clase.
9. Sube el documento en formato PDF en la plataforma Moodle.
10. Revisa la calificación de tus actividades prácticas.

FORO DE DEBATE

Tema: ¿Considera usted que la redundancia en los servidores es importante?

Justifica tu respuesta con ejemplos prácticos

Instrucciones para Participar:

- Exponga su punto de vista acerca del tema.
- Comenta las respuestas de otros participantes.
- Promueve un ambiente respetuoso y constructivo.

PRÁCTICA INVESTIGATIVA

Tema de invetigación: Los tipos de servidores.

Consigna:

- El trabajo de investigación debe presentarse de manera estructurada y coherente para comunicar efectivamente los hallazgos y conclusiones.
- Asegúrese de que el estilo y formato del trabajo sigan las pautas de citación y presentación establecidas por la institución.
- Utilice un lenguaje claro y evite jergas o términos técnicos innecesarios.
- Revise y corrija la redacción y la gramática antes de la presentación final (ver anexo).

PRÁCTICA TEST

Antes de realizar el test, es importante considerar varios aspectos para aprovechar al máximo la experiencia.

Consideraciones para realizar el test

1. Lee y revisa el documento.
2. Observa el video de la clase magistral.
3. Revisa en el calendario el día y hora en que se debe de realizar el Test.
4. Realiza el Test de la unidad considerando lo Siguiente:

- No cambiarte entre paginas durante la realización del Test.
- Revisa que tu conexión a internet sea estable.
- Recuerda que todo el contenido está inmerso dentro del documento base.

5. Revisa tu puntuación, de inmediato al finalizar de la actividad.

II. DATOS GENERALES DE LA UNIDAD 2

Información General

1.1.	Nivel	Primero					
1.2.	Nombre Unidad	Sistemas operativos					
1.3.	Tema 1 – Unidad	Sistemas operativos					
1.4.	Tema 2 – Unidad	Software libre y propietario					
1.5.	Unidad organizacional	Unidad Profesional					
1.6.	Unidad	2					
1.7.	Total, Horas Unidad	15					
1.8.	Detalle de horas Unidad	ACD	6	AA	4	APE	5

Resultado de Aprendizaje

En la Unidad 2 de Teoría de Sistemas, los estudiantes aprenderán a definir y comprender qué es un servidor, sus características y tipos. Analizarán sus funciones dentro de una red. También se abordarán prácticas de seguridad, mantenimiento y tendencias emergentes en tecnología de servidores, preparando a los estudiantes para manejar y administrar servidores eficazmente en diversos contextos.

Metodología

La clase virtual grabada comienza con una introducción clara sobre lo que los estudiantes aprenderán en esa sesión. Se establecen los objetivos de aprendizaje para que los estudiantes sepan qué esperar y en qué enfocarse.

El contenido de la asignatura se desarrolla dividiéndolo en secciones o módulos. Se utiliza una estructura organizada y lógica para abordar cada tema, combinando diferentes enfoques pedagógicos según el contenido, como explicaciones, ejemplos, demostraciones, gráficos y diagramas.

Aunque la clase es grabada, es importante incorporar elementos interactivos para mantener el compromiso de los estudiantes. Esto se puede lograr a través de preguntas para reflexionar, ejercicios, cuestionarios o pausas para discusión, fomentando su participación activa.

DESARROLLO

Unidad 2: Sistemas operativos

Tema 1: Sistemas operativos

- Definición
- Funciones y objetivos del sistema operativo
- Tipos y características
- Evolución de los sistemas operativo

Objetivo: Proporcionar a los estudiantes una comprensión fundamental de los sistemas operativos, sus funciones esenciales y su importancia en el funcionamiento de las computadoras, permitiéndoles apreciar su papel en el mundo de la informática.

Definición Sistema Operativo

Andrew S. Tanenbaum: "Un sistema operativo es un programa que actúa como intermediario entre el usuario de una computadora y el hardware de la misma. Su objetivo principal es proporcionar un entorno en el cual el usuario pueda ejecutar programas de manera conveniente y eficiente."

Silberschatz, Galvin, y Gagne (Autores del libro "Operating System Concepts"): "Un sistema operativo es un programa que actúa como intermediario entre el usuario y el hardware de una computadora. El propósito de un sistema operativo es proporcionar un entorno en el cual un usuario pueda ejecutar programas de manera eficiente y conveniente."

William Stallings: "Un sistema operativo es un conjunto de programas que gestionan los recursos de hardware de un sistema informático, proporcionando servicios a programas de aplicación. El sistema operativo facilita el uso eficiente del hardware y proporciona servicios básicos para la ejecución de aplicaciones."

Abraham Silberschatz: "Un sistema operativo es el software que, tras ser cargado en la computadora por un arranque o encendido, gestiona el hardware del sistema, proporcionando servicios a otros programas de aplicación.”

Funciones y objetivos del sistema operativo

Las funciones de los sistemas operativos han sido conceptualizadas por diversos autores a lo largo del tiempo.

Gestión de Procesos:

Tanenbaum (2008), señala que una de las funciones más importantes es la gestión de procesos, donde el sistema operativo es responsable de la creación, planificación y finalización de procesos.

Gestión de Memoria:

La gestión eficiente de la memoria es otra función esencial según Tanenbaum (2008).

El sistema operativo asigna y libera memoria para los procesos, evitando conflictos y optimizando el uso de los recursos.

Sistema de Archivos:

Tanenbaum (2008) destaca la importancia de la gestión de archivos. El sistema operativo organiza y controla el almacenamiento de datos en dispositivos de almacenamiento como discos duros.

Gestión de Dispositivos de Entrada/Salida:

Otra función crítica es la gestión de dispositivos de entrada/salida, donde el sistema operativo facilita la comunicación efectiva entre el hardware y los dispositivos periféricos.

Abraham Silberschatz: En "Operating System Concepts" (Conceptos de Sistemas Operativos), un libro coescrito por Silberschatz, Galvin, y Gagne, se presentan funciones similares, resaltando la importancia de:

Gestión de Recursos del Sistema:

Silberschatz y sus coautores subrayan la función esencial de la gestión de recursos, donde el sistema operativo coordina y asigna recursos como la CPU y la memoria.

Interfaz de Usuario:

Destacan la interfaz de usuario como un aspecto crucial, proporcionando una plataforma para que los usuarios interactúen con la computadora de manera conveniente.

Gestión de Procesos y Planificación:

La gestión de procesos y la planificación son fundamentales según Silberschatz, ya que el sistema operativo coordina la ejecución de múltiples procesos.

Seguridad y Protección:

Subrayan la importancia de la seguridad y la protección, donde el sistema operativo implementa mecanismos para resguardar el sistema y los datos contra accesos no autorizados.

Tanenbaum como Silberschatz concuerdan en que los sistemas operativos desempeñan un papel crítico en la coordinación y gestión eficiente de los recursos del sistema, la ejecución de procesos, y la facilitación de la interacción entre el usuario y la computadora. Estas funciones fundamentales garantizan el funcionamiento adecuado y confiable de un sistema informático.

Los objetivos del sistema operativo son metas y propósitos que guían su diseño, implementación y funcionamiento para proporcionar un entorno eficiente y eficaz en el uso de los recursos de una computadora. Estos objetivos varían según los autores y los contextos, pero en general, se pueden identificar los siguientes:

Objetivos del sistema operativo

Eficiencia en el Uso de Recursos: Uno de los objetivos primordiales es maximizar la eficiencia en el uso de los recursos del sistema, incluyendo la CPU, memoria, almacenamiento y dispositivos de entrada/salida.

Proporcionar una Interfaz de Usuario: Facilitar una interfaz que permita a los usuarios interactuar con la computadora de manera efectiva, ya sea mediante una interfaz gráfica de usuario (GUI) o una interfaz de línea de comandos.

Gestión Efectiva de Procesos: Coordinar la ejecución de múltiples procesos, asignando adecuadamente los recursos y optimizando el rendimiento del sistema.

Optimización de la Memoria: Gestionar la memoria de manera eficiente para asegurar la ejecución suave de los programas y evitar la falta de memoria.

Fiabilidad y Tolerancia a Fallos: Proporcionar un entorno estable y confiable, con mecanismos para manejar errores y fallos de manera que se minimice el impacto en la operación del sistema.

Seguridad del Sistema: Implementar medidas de seguridad para proteger los datos y el sistema contra accesos no autorizados, virus y otras amenazas.

Adaptabilidad y Escalabilidad: Ser capaz de adaptarse a diferentes entornos y escalarse para manejar crecimientos en la carga de trabajo, sin perder eficiencia.

Soporte para Redes: Facilitar la conectividad y la comunicación en entornos de red, gestionando la transmisión de datos entre dispositivos.

Proporcionar un Sistema de Archivos Eficiente: Organizar y gestionar la información almacenada en dispositivos de almacenamiento de manera efectiva, permitiendo la creación, lectura, escritura y eliminación de archivos.

Conveniencia para el Usuario: Ofrecer una experiencia de usuario cómoda y eficiente, con herramientas y funcionalidades que faciliten la realización de tareas.

Tipos y Características de los sistemas operativos

Tipos

Sistemas Operativos de Escritorio: Diseñados para computadoras personales.

Ejemplos: Microsoft Windows, macOS, Linux (distribuciones como Ubuntu).

Sistemas Operativos Móviles: Diseñados para dispositivos móviles como teléfonos inteligentes y tabletas.

Ejemplos: Android, iOS.

Sistemas Operativos de Servidores: Optimizados para gestionar servidores y redes.

Ejemplos: Windows Server, Linux Server.

Sistemas Operativos Embebidos: Integrados en dispositivos específicos como electrodomésticos y sistemas embebidos.

Ejemplos: VxWorks, Android Things.

Sistemas Operativos en Tiempo Real: Diseñados para sistemas que deben responder a eventos en tiempo real.

Ejemplos: FreeRTOS, QNX.

Sistemas Operativos de Red: Enfocados en la gestión de redes de computadoras.

Ejemplos: Novell NetWare, BSD UNIX.

Sistemas Operativos de Tiempo Compartido: Permiten que varios usuarios compartan simultáneamente el uso de la computadora.

Ejemplos: UNIX, Linux.

Sistemas Operativos de Tiempo Compartido en la Nube: Se ejecutan en servidores remotos y proporcionan servicios a través de internet.

Ejemplos: Google Cloud, Microsoft Azure.

Características de los Sistemas Operativos:

Interfaz de Usuario: Puede ser una interfaz gráfica de usuario (GUI) o una interfaz de línea de comandos (CLI).

Gestión de Procesos: Coordina la ejecución de programas y procesos en la CPU.

Gestión de Memoria: Administra el acceso y la asignación de memoria para programas y datos.

Sistema de Archivos: Organiza y gestiona la información almacenada en dispositivos de almacenamiento.

Gestión de Dispositivos: Coordina la comunicación entre el hardware y los dispositivos periféricos.

Seguridad y Protección: Implementa medidas de seguridad para proteger el sistema y los datos.

Redes y Comunicación: Facilita la conectividad y la comunicación en redes de computadoras.

Actualizaciones y Mantenimiento: Proporciona mecanismos para actualizar y mantener el sistema operativo.

Multiusuario y Multitarea: Permite que varios usuarios trabajen simultáneamente o que un usuario realice varias tareas al mismo tiempo.

Compatibilidad y Portabilidad: Puede ejecutarse en diferentes tipos de hardware y ser compatible con una variedad de aplicaciones.

Evolución de los sistemas operativos

La evolución de los sistemas operativos ha estado impulsada por avances en hardware, cambios en las necesidades de los usuarios y el desarrollo de nuevas tecnologías.

Sin Sistemas Operativos (1940-1950):

- Las primeras computadoras no tenían sistemas operativos. Los programadores interactuaban directamente con el hardware para realizar tareas específicas.

Lote por Lote (Batch Processing) (1950-1960):

- Introducción de sistemas que procesaban lotes de tareas en secuencia, como el UNIVAC I. Los operadores preparaban las tareas por lotes y las cargaban en la máquina.

Sistemas de Tiempo Compartido (1960-1970):

- Surgieron sistemas que permitían a varios usuarios interactuar simultáneamente con la computadora. El sistema operativo compartía el tiempo de la CPU entre los usuarios.

Sistemas Multitarea (1970-1980):

- Los sistemas operativos evolucionaron para permitir la ejecución simultánea de múltiples tareas. Introducción de sistemas como UNIX y VMS.

Sistemas Operativos de Red (1980-1990):

- Los sistemas operativos comenzaron a enfocarse en la gestión de redes y la comunicación entre computadoras. Surgieron sistemas como Novell NetWare.

Sistemas Operativos para Computadoras Personales (1980-1990):

- Con la popularización de las computadoras personales, surgieron sistemas operativos como MS-DOS y posteriormente Microsoft Windows y macOS.

Sistemas Operativos Móviles (2000 en adelante):

- Con la proliferación de dispositivos móviles, se desarrollaron sistemas operativos específicos para teléfonos inteligentes y tabletas, como Android e iOS.

Computación en la Nube (2010 en adelante):

- Los sistemas operativos evolucionaron para soportar la computación en la nube, donde los recursos se proporcionan a través de internet. Ejemplos incluyen Chrome OS y sistemas operativos para servidores en la nube.

Sistemas Operativos de Código Abierto (2000 en adelante):

- Aumento de sistemas operativos de código abierto como Linux, que se utilizan en una amplia variedad de entornos, desde servidores hasta dispositivos integrados.

Sistemas Operativos para Dispositivos Inteligentes (2020 en adelante):

- Con el auge de la Internet de las cosas (IoT), se están desarrollando sistemas operativos adaptados para dispositivos inteligentes y conectados.

ACTIVIDADES PRÁCTICAS

Tabla

Detalle de la Práctica

Practicas	Instrucciones	Duración de la Practica	Mecanismo de Evaluación
Foro de debate	Revisar Actividades Prácticas	0:30 mm	Por rúbricas
Práctica investiga: Taller/Ensayo/Investigación	Revisar Actividades Prácticas	3:30 h	Por rúbricas
Tutorías Prácticas	Revisar Actividades Prácticas	2:00 mm	Por participación en clase
Test evaluativo	Revisar Actividades Prácticas	0:30 mm	Formativa

Nota. Elaboración propia

Consideraciones Generales

1. Lee y revisa la Guía de Estudio.
2. Observa el video de la clase magistral.
3. Participa activamente en el FORO DEBATE.
4. Revisa en el calendario el día y hora en que se realizará las actividades prácticas y la tutoría.
5. Realizar la práctica propuesta por el docente.
6. Revisa las Rúbricas acorde a la actividad prácticas.
7. Toma los principales apuntes que vayas rescatando del texto.
8. Reflexiona sobre los temas tratados en la clase.
9. Sube el documento en formato PDF en la plataforma Moodle.
10. Revisa la calificación de tus actividades prácticas.

FORO DE DEBATE

Tema: ¿Qué tipo de sistema operativo considera que es más estable y porque S.O. Libre o Con licencia?

Instrucciones para Participar:

- Exponga su punto de vista acerca del tema.
- Comenta las respuestas de otros participantes.
- Promueve un ambiente respetuoso y constructivo.

PRÁCTICA INVESTIGATIVA

Investigación: Ventajas y desventajas de los sistemas operativos de código abierto (como Linux) en comparación con los sistemas operativos propietarios (como Windows o macOS).

Consigna:

- El trabajo de investigación debe presentarse de manera estructurada y coherente para comunicar efectivamente los hallazgos y conclusiones.
- Asegúrese de que el estilo y formato del trabajo sigan las pautas de citación y presentación establecidas por la institución.
- Utilice un lenguaje claro y evite jergas o términos técnicos innecesarios.
- Revise y corrija la redacción y la gramática antes de la presentación final (ver anexo).

PRÁCTICA TEST

Antes de realizar el test, es importante considerar varios aspectos para aprovechar al máximo la experiencia.

Consideraciones para realizar el test

1. Lee y revisa el documento.
2. Observa el video de la clase magistral.
3. Revisa en el calendario el día y hora en que se debe de realizar el Test.
4. Realiza el Test de la unidad considerando lo Siguiente:

- No cambiarte entre paginas durante la realización del Test.
- Revisa que tu conexión a internet sea estable.
- Recuerda que todo el contenido está inmerso dentro del documento base.

5. Revisa tu puntuación, de inmediato al finalizar de la actividad.

Tema 2: Software libre y propietario

- Definición
- Ventajas y desventajas
- Patentes
- Visión histórica del software

Objetivo: Proporcionar a los estudiantes una comprensión fundamental de los sistemas operativos, sus funciones esenciales y su importancia en el funcionamiento de las computadoras, permitiéndoles apreciar su papel en el mundo de la informática.

Definición

Software Libre: El término "software libre" se refiere a un tipo de software que respeta la libertad de los usuarios. Para que un programa sea considerado software libre, debe otorgar a los usuarios cuatro libertades esenciales, conocidas como las "Cuatro Libertades del Software Libre", según la Free Software Foundation (FSF):

Libertad de Ejecución: La libertad de ejecutar el programa como se desee, para cualquier propósito.

Libertad de Estudio: La libertad de estudiar cómo funciona el programa y modificarlo según las necesidades.

Libertad de Distribución: La libertad de redistribuir copias para ayudar a los demás.

Libertad de Mejora: La libertad de mejorar el programa y liberar esas mejoras a la comunidad, para que todos se beneficien.

Un ejemplo de software libre es el sistema operativo Linux y muchas de las aplicaciones asociadas.

Software Propietario:

El software propietario, también conocido como software cerrado o software privativo, es un tipo de software en el cual el código fuente no está disponible públicamente y su uso está sujeto a restricciones impuestas por el titular de los derechos de autor o el titular de la licencia. Las características típicas del software propietario incluyen:

Código Fuente Cerrado: El código fuente del software no está disponible para el público, lo que significa que los usuarios no pueden ver ni modificar el código.

Restricciones de Uso: El uso del software suele estar sujeto a un contrato de licencia que establece las condiciones y restricciones, como la prohibición de modificar o redistribuir el software.

Control por el Desarrollador: El desarrollador o la empresa que posee los derechos de autor tiene un control significativo sobre el desarrollo, distribución y actualización del software.

Costo y Propiedad: A menudo, el software propietario implica el pago de una licencia y no otorga derechos de propiedad o acceso al código fuente.

Un ejemplo de software propietario es Microsoft Windows, cuyo código fuente no está disponible públicamente y su uso está sujeto a licencia.

Ventajas y desventajas entre software libre y Propietario

Ventajas del Software Libre:

- **Libertad de Uso:** Los usuarios tienen libertad para ejecutar, estudiar, modificar y redistribuir el software según sus necesidades.
- **Costo:** El software libre suele ser gratuito, lo que reduce los costos de adquisición y permite a las personas y organizaciones utilizar herramientas potentes sin incurrir en grandes gastos.
- **Colaboración y Comunidad:** Fomenta la colaboración y la creación de comunidades de desarrolladores que comparten conocimientos y trabajan juntos para mejorar el software.
- **Transparencia y Confianza:** El código fuente abierto permite a los usuarios examinar cómo funciona el software, lo que aumenta la transparencia y la confianza en su seguridad y eficiencia.
- **Flexibilidad y Adaptabilidad:** Los usuarios tienen la capacidad de modificar el software para adaptarlo a sus necesidades específicas, lo que brinda una mayor flexibilidad.

- **Soporte Comunitario:** Existe una amplia comunidad de usuarios y desarrolladores que ofrecen soporte y recursos, como foros de discusión y documentación.

Desventajas del Software Libre:

- **Curva de Aprendizaje:** Algunos usuarios pueden enfrentar una curva de aprendizaje al cambiar a software libre, ya que las interfaces y las funciones pueden diferir de las soluciones propietarias más comunes.
- **Compatibilidad:** Puede haber problemas de compatibilidad con ciertos formatos de archivos y aplicaciones que son estándar en el software propietario.
- **Soporte Profesional Limitado:** Aunque hay comunidades de soporte, a veces puede ser difícil encontrar soporte profesional en comparación con soluciones propietarias.
- **Falta de Aplicaciones Específicas:** Para ciertas aplicaciones especializadas o industrias específicas, puede haber una falta de software libre equivalente.
- **Interfaz de Usuario:** Algunos usuarios argumentan que ciertos programas de software libre pueden tener interfaces de usuario menos intuitivas en comparación con sus contrapartes propietarias.

Ventajas del Software Propietario:

- **Soporte Profesional:** Las empresas que desarrollan software propietario suelen ofrecer soporte técnico profesional para resolver problemas y brindar asistencia.
- **Desarrollo Continuo:** Las empresas propietarias tienen recursos para el desarrollo continuo del software, lanzando actualizaciones y nuevas versiones con regularidad.
- **Compatibilidad:** Dado que muchas empresas utilizan software propietario, hay una mayor probabilidad de compatibilidad con estándares de la industria y formatos de archivos.
- **Facilidad de Uso:** Algunos usuarios encuentran que el software propietario ofrece interfaces de usuario más intuitivas y una experiencia de usuario más fácil.
- **Garantía de Calidad:** Las empresas propietarias suelen realizar pruebas extensas y garantizar la calidad del software antes de su lanzamiento.

Desventajas del Software Propietario:

- **Costo:** El software propietario a menudo implica costos significativos de licencia, actualizaciones y soporte.
- **Falta de Libertad:** Los usuarios están limitados en cuanto a modificar el código fuente y redistribuir el software, ya que estas acciones pueden estar prohibidas por las licencias.
- **Dependencia del Desarrollador:** Los usuarios dependen del desarrollador para actualizaciones, soporte y corrección de errores, lo que puede generar vulnerabilidades en caso de que el desarrollador deje de mantener el software.
- **Falta de Transparencia:** El código fuente cerrado impide a los usuarios ver cómo funciona el software, lo que puede generar preocupaciones sobre la seguridad y la privacidad.
- **Limitaciones de Personalización:** La personalización del software está limitada por las restricciones impuestas por el desarrollador o la empresa propietaria.

Patentes

Una patente es un derecho exclusivo otorgado por un gobierno a un inventor o a la persona que posea los derechos de una invención. Este derecho exclusivo le permite al titular de la patente excluir a otros de hacer, usar, vender, importar o distribuir la invención patentada durante un período de tiempo determinado, generalmente 20 años a partir de la fecha de presentación de la solicitud de patente.

Las patentes son una forma de propiedad intelectual que busca incentivar la innovación y el avance tecnológico al proporcionar al inventor la exclusividad sobre su invención. Para obtener una patente, el inventor debe presentar una solicitud de patente ante la oficina de patentes correspondiente en el país donde busca protección. Esta solicitud debe describir detalladamente la invención y cómo difiere de las tecnologías existentes.

Es importante destacar que las patentes no protegen ideas en abstracto; en cambio, protegen una aplicación específica y práctica de una idea. Además, la información detallada sobre la invención se vuelve pública cuando se otorga la patente, lo que contribuye al conocimiento general y fomenta la innovación al permitir a otros aprender de las ideas patentadas.

Existen varios tipos de patentes, pero las tres categorías principales son:

1. **Patentes de Invención:** Protegen invenciones nuevas y útiles, generalmente procesos, máquinas, composiciones de materiales o mejoras sustanciales en los mismos.
2. **Modelos de Utilidad:** Similar a las patentes de invención, pero a menudo cubren invenciones de menor envergadura y tienen requisitos menos estrictos de novedad y actividad inventiva.
3. **Diseños Industriales:** Protegen la apariencia ornamental de un artículo, como su forma, configuración, ornamentación o combinación de colores.

En Ecuador, la protección de software se contempla en la categoría de "Derechos de Autor" más que en la de "Patentes". El Instituto Ecuatoriano de Propiedad Intelectual (IEPI) es la entidad encargada de administrar los derechos de propiedad intelectual en Ecuador.

Si bien la Ley de Propiedad Intelectual no permite patentar software, ofrece protección a través de derechos de autor para programas de computadora. Esto significa que el titular de un programa de software tiene derechos exclusivos sobre la reproducción, distribución, y ejecución del programa.

Visión histórica del software

La visión histórica del software se puede dividir en varias etapas clave que han marcado su evolución desde los primeros días de la informática hasta la actualidad.

Década de 1940 - Primeros Pasos:

Durante esta época, los primeros computadores electrónicos se construyeron para realizar cálculos específicos. El software consistía en programas cableados físicamente para ejecutar tareas específicas.

Década de 1950 - Lenguajes de Ensamblaje:

Se desarrollaron lenguajes de programación de bajo nivel, como el lenguaje ensamblador, que permitían una programación más eficiente y menos dependencia de la codificación de hardware específico.

Década de 1960 - Lenguajes de Alto Nivel:

Surgieron lenguajes de programación de alto nivel como Fortran y COBOL, facilitando la programación y aumentando la portabilidad del software entre diferentes computadoras.

Década de 1970 - Revolución de los Microprocesadores:

La introducción de microprocesadores y el desarrollo de sistemas operativos como UNIX allanaron el camino para la proliferación de computadoras personales y la diversificación del software.

Década de 1980 - Computadoras Personales y Software Comercial:

La popularización de las computadoras personales, junto con el surgimiento de sistemas operativos como MS-DOS y el lanzamiento de la suite de productividad Microsoft Office, marcó una era de software comercial.

Década de 1990 - Internet y Software Distribuido:

El auge de Internet impulsó el desarrollo de software distribuido y aplicaciones web. Surgieron sistemas operativos como Windows 95 y se produjo un aumento significativo en la conectividad.

Década de 2000 - Software de Código Abierto y Móvil:

El software de código abierto, representado por proyectos como Linux, ganó relevancia. Además, el auge de los teléfonos inteligentes y las aplicaciones móviles marcó una nueva era en la informática personal.

Década de 2010 - Computación en la Nube y Software como Servicio (SaaS):

El surgimiento de la computación en la nube permitió acceder y utilizar aplicaciones directamente desde la web. Modelos como Software como Servicio (SaaS) se volvieron comunes.

Década de 2020 - Inteligencia Artificial y Desarrollo Ágil:

El desarrollo de software se ha centrado en la integración de inteligencia artificial, aprendizaje automático y enfoques ágiles para mejorar la eficiencia y la capacidad de respuesta a cambios rápidos en los requisitos.

ACTIVIDADES PRÁCTICAS

Tabla

Detalle de la Práctica

Practicas	Instrucciones	Duración de la Practica	Mecanismo de Evaluación
Foro de debate	Revisar Actividades Prácticas	0:30 mm	Por rúbricas
Práctica investiga: Taller/Ensayo/Investigación	Revisar Actividades Prácticas	3:30 h	Por rúbricas
Tutorías Prácticas	Revisar Actividades Prácticas	2:00 mm	Por participación en clase
Test evaluativo	Revisar Actividades Prácticas	0:30 mm	Formativa

Nota. Elaboración propia

Consideraciones Generales

1. Lee y revisa la Guía de Estudio.
2. Observa el video de la clase magistral.
3. Participa activamente en el FORO DEBATE.
4. Revisa en el calendario el día y hora en que se realizará las actividades prácticas y la tutoría.
5. Realizar la práctica propuesta por el docente.
6. Revisa las Rúbricas acorde a la actividad prácticas.
7. Toma los principales apuntes que vayas rescatando del texto.
8. Reflexiona sobre los temas tratados en la clase.
9. Sube el documento en formato PDF en la plataforma Moodle.
10. Revisa la calificación de tus actividades prácticas.

FORO DE DEBATE

Tema: ¿Considera usted que un software libre tiene más ventajas que el software propietario? Fundamente su respuesta.

Instrucciones para Participar:

- Exponga su punto de vista acerca del tema.
- Comenta las respuestas de otros participantes.
- Promueve un ambiente respetuoso y constructivo.

PRÁCTICA INVESTIGATIVA

Ensayo: La importancia de las Patentes en la Industria del software

Consigna:

- El trabajo de investigación debe presentarse de manera estructurada y coherente para comunicar efectivamente los hallazgos y conclusiones.
- Asegúrese de que el estilo y formato del trabajo sigan las pautas de citación y presentación establecidas por la institución.
- Utilice un lenguaje claro y evite jergas o términos técnicos innecesarios.
- Revise y corrija la redacción y la gramática antes de la presentación final (ver anexo).

PRÁCTICA TEST

Antes de realizar el test, es importante considerar varios aspectos para aprovechar al máximo la experiencia.

Consideraciones para realizar el test

1. Lee y revisa el documento.
2. Observa el video de la clase magistral.
3. Revisa en el calendario el día y hora en que se debe de realizar el Test.
4. Realiza el Test de la unidad considerando lo Siguiente:

- No cambiarte entre paginas durante la realización del Test.
- Revisa que tu conexión a internet sea estable.
- Recuerda que todo el contenido está inmerso dentro del documento base.

5. Revisa tu puntuación, de inmediato al finalizar de la actividad.

III. DATOS GENERALES DE LA UNIDAD 3

Información General

1.1.	Nivel	Primero					
1.2.	Nombre Unidad	Paradigmas y lenguajes de programación					
1.3.	Tema 1 – Unidad	Paradigmas					
1.4.	Tema 2 – Unidad	Lenguaje de Programación					
1.5.	Unidad organizacional	Unidad Profesional					
1.6.	Unidad	3					
1.7.	Total, Horas Unidad	15					
1.8.	Detalle de horas Unidad	ACD	6	AA	4	APE	5

Resultado de Aprendizaje

En la Unidad 3 de Teoría de Sistemas, los estudiantes podrán comprender y analizar los diferentes paradigmas de programación y lenguajes de programación, incluyendo sus definiciones, tipos y tendencias actuales, y aplicar este conocimiento en la selección y uso adecuado de lenguajes de programación en diversos contextos.

Metodología

La clase virtual grabada comienza con una introducción clara sobre lo que los estudiantes aprenderán en esa sesión. Se establecen los objetivos de aprendizaje para que los estudiantes sepan qué esperar y en qué enfocarse.

El contenido de la asignatura se desarrolla dividiéndolo en secciones o módulos. Se utiliza una estructura organizada y lógica para abordar cada tema, combinando diferentes enfoques pedagógicos según el contenido, como explicaciones, ejemplos, demostraciones, gráficos y diagramas.

Aunque la clase es grabada, es importante incorporar elementos interactivos para mantener el compromiso de los estudiantes. Esto se puede lograr a través de preguntas para reflexionar, ejercicios, cuestionarios o pausas para discusión, fomentando su participación activa.

DESARROLLO

Unidad 3: Paradigmas y lenguajes de programación

Tema 1: Paradigmas

- Definición
- Tipos de paradigmas
- Tendencias

Los paradigmas en teoría de sistemas son modelos o patrones fundamentales que proporcionan una estructura y guía para la construcción y organización de sistemas. Estos paradigmas no solo influyen en el diseño de sistemas, sino que también afectan la forma en que los programadores piensan y resuelven problemas. En esencia, son enfoques conceptuales que dictan cómo se debe abordar la construcción de sistemas.

En el contexto de la informática y la programación, un paradigma se refiere a un modelo o enfoque conceptual que proporciona pautas y técnicas para resolver problemas y diseñar sistemas. Estos paradigmas influyen en la forma en que los programadores estructuran y abordan la creación de software, dictando cómo se deben escribir y organizar los programas. Cada paradigma tiene sus propias reglas y estilos distintivos, lo que impacta en la comprensión y la solución de problemas en el ámbito de la programación.

Ejemplos de Paradigmas:

1. **Paradigma Imperativo:**

Definición: El paradigma imperativo se centra en describir paso a paso cómo realizar una tarea. Utiliza variables para almacenar datos y cambios de estado para expresar la lógica del programa. Las instrucciones imperativas especifican cómo se deben realizar las operaciones.

Ejemplo: El lenguaje de programación C es un ejemplo clásico de un paradigma imperativo. En C, los programadores escriben instrucciones detalladas para ejecutar tareas específicas, manipulando variables y utilizando estructuras de control de flujo.

2. **Paradigma Orientado a Objetos (OO):**

    - *Definición:* El paradigma orientado a objetos organiza el código alrededor de "objetos", que son instancias de clases que encapsulan datos y

comportamientos. Este paradigma promueve la reutilización y el modularidad del código al modelar el mundo real a través de objetos y sus interacciones.

- *Ejemplo:* Java es un lenguaje de programación que sigue el paradigma orientado a objetos. En Java, los programas se estructuran en torno a clases y objetos, permitiendo la encapsulación, la herencia y el polimorfismo para facilitar el diseño y la implementación de sistemas complejos.

Tipos de Paradigmas en Teoría de Sistemas:

1. **Paradigma Imperativo:**
    - Se centra en describir paso a paso cómo realizar una tarea. Utiliza variables y cambios de estado para expresar la lógica del programa. Ejemplos incluyen lenguajes como C y Pascal.
2. **Paradigma Declarativo:**
    - Se enfoca en describir qué se quiere lograr en lugar de como hacerlo. Incluye el paradigma funcional y lógico.
        - **Funcional:** Se basa en funciones matemáticas y evita el cambio de estado. Ejemplos son Haskell y Lisp.
        - **Lógico:** Se basa en reglas lógicas y relaciones. Prolog es un ejemplo.
3. **Paradigma Orientado a Objetos (OO):**
    - Organiza el código alrededor de "objetos" que encapsulan datos y comportamientos. Promueve la reutilización y el modularidad. Lenguajes como Java, Python y C++ siguen este paradigma.
4. **Paradigma de Programación Estructurada:**
    - Se centra en el uso de estructuras de control de flujo (secuencia, selección y repetición) para organizar el código. C es un lenguaje emblemático de este paradigma.

5. **Paradigma de Programación Concurrente:**

    - Permite la ejecución simultánea de varias partes de un programa. Lenguajes como Erlang y Java (con su soporte para hilos) se ajustan a este paradigma.

Tendencias en Paradigmas de Programación:

1. **Paradigmas Híbridos:**

    - La combinación de características de diferentes paradigmas para abordar de manera más eficiente problemas específicos.

2. **Paradigmas Cuánticos:**

    - Con el avance de la computación cuántica, se exploran nuevos paradigmas que aprovechan los principios de la mecánica cuántica para realizar cálculos de manera revolucionaria.

3. **Paradigmas para la Inteligencia Artificial:**

    - Con el auge de la inteligencia artificial, se están desarrollando paradigmas específicos que permiten la creación y gestión eficiente de algoritmos de aprendizaje automático y sistemas inteligentes.

4. **Programación Cuántica:**

    - En el contexto de la computación cuántica, este paradigma emergente busca aprovechar los principios cuánticos para la resolución eficiente de problemas complejos.

ACTIVIDADES PRÁCTICAS

Tabla

Detalle de la Práctica

Practicas	Instrucciones	Duración de la Practica	Mecanismo de Evaluación
Foro de debate	Revisar Actividades Prácticas	0:30 mm	Por rúbricas
Práctica investiga: Taller/Ensayo/Investigación	Revisar Actividades Prácticas	3:30 h	Por rúbricas
Tutorías Prácticas	Revisar Actividades Prácticas	2:00 mm	Por participación en clase
Test evaluativo	Revisar Actividades Prácticas	0:30 mm	Formativa

Nota. Elaboración propia

Consideraciones Generales

1. Lee y revisa la Guía de Estudio.
2. Observa el video de la clase magistral.
3. Participa activamente en el FORO DEBATE.
4. Revisa en el calendario el día y hora en que se realizará las actividades prácticas y la tutoría.
5. Realizar la práctica propuesta por el docente.
6. Revisa las Rúbricas acorde a la actividad prácticas.
7. Toma los principales apuntes que vayas rescatando del texto.
8. Reflexiona sobre los temas tratados en la clase.
9. Sube el documento en formato PDF en la plataforma Moodle.
10. Revisa la calificación de tus actividades prácticas.

FORO DE DEBATE

Tema: Impacto Ético de la Inteligencia Artificial en la Sociedad.

Instrucciones para Participar:

- Exponga su punto de vista acerca del tema.
- Comenta las respuestas de otros participantes.
- Promueve un ambiente respetuoso y constructivo.

PRÁCTICA INVESTIGATIVA

Taller: Los paradigmas y tendencias en la programación.

Consigna:

- El trabajo de investigación debe presentarse de manera estructurada y coherente para comunicar efectivamente los hallazgos y conclusiones.
- Asegúrese de que el estilo y formato del trabajo sigan las pautas de citación y presentación establecidas por la institución.
- Utilice un lenguaje claro y evite jergas o términos técnicos innecesarios.
- Revise y corrija la redacción y la gramática antes de la presentación final (ver anexo).

PRÁCTICA TEST

Antes de realizar el test, es importante considerar varios aspectos para aprovechar al máximo la experiencia.

Consideraciones para realizar el test

1. Lee y revisa el documento.
2. Observa el video de la clase magistral.
3. Revisa en el calendario el día y hora en que se debe de realizar el Test.
4. Realiza el Test de la unidad considerando lo Siguiente:

- No cambiarte entre paginas durante la realización del Test.
- Revisa que tu conexión a internet sea estable.
- Recuerda que todo el contenido está inmerso dentro del documento base.

5. Revisa tu puntuación, de inmediato al finalizar de la actividad.

Tema 2: Lenguaje de programación

- Definición
- Tipos de lenguajes
- Tendencias de los lenguajes de programación

Objetivo: Proporcionar a los estudiantes a Desarrollar una comprensión integral de los lenguajes de programación, incluyendo su definición, clasificación en diversos tipos, y las tendencias emergentes en su desarrollo, para aplicar este conocimiento en el análisis y diseño de sistemas informáticos.

Lenguaje de programación en Teoría de Sistemas:

Definición

Un lenguaje de programación es un lenguaje informático especialmente diseñado para describir el conjunto de acciones consecutivas o instrucciones que un equipo informático debe ejecutar.

En el contexto de la Teoría de Sistemas se refiere a un conjunto estructurado de instrucciones y reglas sintácticas que permiten a los programadores comunicarse con un sistema informático para desarrollar, implementar y controlar sus funciones y operaciones. el lenguaje de programación facilita la representación y expresión de la lógica y la estructura inherentes a un sistema, permitiendo la especificación detallada de su comportamiento y las interacciones entre sus componentes. Este enfoque contribuye a la conceptualización, diseño y gestión de sistemas complejos, al tiempo que favorece el modularidad, la reutilización de código y la eficiencia en el desarrollo de software.

la sinergia entre el lenguaje de programación y la Teoría de Sistemas es esencial para el desarrollo efectivo de software, ya que proporciona las bases teóricas y prácticas necesarias para abordar la complejidad inherente a la construcción y gestión de sistemas informáticos.

Ejemplos de lenguaje de programación:

1. **Simulación de Sistemas Dinámicos:**
    - *Definición:* La simulación de sistemas dinámicos es una técnica que utiliza modelos matemáticos y algoritmos numéricos para imitar el comportamiento

de sistemas en el tiempo. Estos sistemas pueden ser físicos, biológicos, sociales o tecnológicos, y la simulación ayuda a comprender su evolución a lo largo de un periodo temporal.

- *Ejemplo:* El lenguaje de programación C es un ejemplo clásico de un paradigma imperativo. En C, los programadores escriben instrucciones detalladas para ejecutar tareas específicas, manipulando variables y utilizando estructuras de control de flujo.

2. **Paradigma Orientado a Objetos (OO):**

    - *Definición:* El paradigma orientado a objetos organiza el código alrededor de "objetos", que son instancias de clases que encapsulan datos y comportamientos. Este paradigma promueve la reutilización y el modularidad del código al modelar el mundo real a través de objetos y sus interacciones.

Ejemplo: Ejemplo en Python: Simulación de un Péndulo Simple se simula el comportamiento de un péndulo simple utilizando la ecuación diferencial que describe su movimiento. La simulación se realiza mediante el método solve_ivp de la biblioteca SciPy en Python, y los resultados se grafican para visualizar cómo el ángulo del péndulo cambia con el tiempo. Este tipo de simulación es útil para comprender fenómenos dinámicos y evaluar el impacto de diferentes parámetros en el sistema a lo largo del tiempo

Tipos de Paradigmas en Teoría de Sistemas:

Lenguajes de Bajo Nivel:

Ensamblador: Utilizado para programar directamente en el nivel de la arquitectura de la computadora, utilizando mnemotécnicos representando instrucciones de máquina.

Lenguajes de Máquina: Los más básicos, representando directamente instrucciones binarias ejecutables por la CPU.

Lenguajes de Alto Nivel:

Imperativos: Se centran en describir cómo realizar una tarea y utilizan instrucciones detalladas.

C: Un lenguaje de programación de propósito general que sigue un paradigma imperativo.

Declarativos: Se centran en describir qué resultado se desea obtener sin detallar los pasos específicos.

SQL: Lenguaje utilizado para gestionar bases de datos relacionales.

Lenguajes Orientados a Objetos:

Java: Un lenguaje orientado a objetos que organiza la lógica del programa en clases y objetos.

Python: Un lenguaje de programación versátil que admite tanto programación imperativa como orientada a objetos.

Lenguajes Funcionales:

Haskell: Un lenguaje funcional que se centra en funciones puras y evita los efectos secundarios.

Lisp: Conocido por su poderosa capacidad de manipulación de listas y su enfoque funcional.

Lenguajes de Scripting:

JavaScript: Ampliamente utilizado para desarrollo web, es interpretado y ejecutado en el navegador del usuario.

Bash: Un lenguaje de scripting de shell utilizado en sistemas operativos tipo Unix.

Lenguajes de Dominio Específico (DSL):

HTML y CSS: Utilizados para describir la estructura y estilo de las páginas web, respectivamente.

MATLAB: Diseñado para realizar cálculos numéricos y visualización de datos.

Tendencias de los lenguajes de programación:

1. Python en Ascenso:

Python ha experimentado un crecimiento significativo y se ha convertido en uno de los lenguajes de programación más populares. Su sintaxis clara y su versatilidad lo hacen atractivo para una variedad de aplicaciones, desde desarrollo web hasta inteligencia artificial.

2. JavaScript Ubicuo:

JavaScript sigue siendo esencial para el desarrollo web, y su uso se ha ampliado con el auge de las tecnologías de desarrollo frontend como React, Angular y Vue.js. Además, Node.js permite a los desarrolladores utilizar JavaScript en el lado del servidor.

3. TypeScript en Crecimiento:

TypeScript, una extensión de JavaScript que agrega tipado **estático** ha ganado popularidad. Muchos desarrolladores lo eligen para proyectos más grandes debido a su capacidad para detectar errores en tiempo de compilación.

4. Rust por Seguridad:

Rust ha ganado atención debido a su enfoque en la seguridad y el rendimiento. Es utilizado en situaciones donde se requiere un control preciso sobre los recursos de la máquina y la prevención de errores de memoria.

5. Desarrollo Móvil con Kotlin y Swift:

Kotlin se ha convertido en el lenguaje preferido para el desarrollo de aplicaciones Android, mientras que Swift sigue siendo esencial para el desarrollo de aplicaciones iOS. Ambos lenguajes ofrecen mejoras sobre sus predecesores (Java y Objective-C, respectivamente).

6. Lenguajes Funcionales:

Lenguajes funcionales como Haskell, Clojure y Scala han ganado interés, especialmente en aplicaciones donde la programación funcional puede mejorar la concurrencia y la escalabilidad.

ACTIVIDADES PRÁCTICAS

Tabla

Detalle de la Práctica

Practicas	Instrucciones	Duración de la Practica	Mecanismo de Evaluación
Foro de debate	Revisar Actividades Prácticas	0:30 mm	Por rúbricas
Práctica investiga: Taller/Ensayo/Investigación	Revisar Actividades Prácticas	3:30 h	Por rúbricas
Tutorías Prácticas	Revisar Actividades Prácticas	2:00 mm	Por participación en clase
Test evaluativo	Revisar Actividades Prácticas	0:30 mm	Formativa

Nota. Elaboración propia

Consideraciones Generales

1. Lee y revisa la Guía de Estudio.
2. Observa el video de la clase magistral.
3. Participa activamente en el FORO DEBATE.
4. Revisa en el calendario el día y hora en que se realizará las actividades prácticas y la tutoría.
5. Realizar la práctica propuesta por el docente.
6. Revisa las Rúbricas acorde a la actividad prácticas.
7. Toma los principales apuntes que vayas rescatando del texto.
8. Reflexiona sobre los temas tratados en la clase.
9. Sube el documento en formato PDF en la plataforma Moodle.
10. Revisa la calificación de tus actividades prácticas.

FORO DE DEBATE

Tema: En que consiste la Evolución de los lenguajes de programación.

Instrucciones para Participar:

- Exponga su punto de vista acerca del tema.
- Comenta las respuestas de otros participantes.
- Promueve un ambiente respetuoso y constructivo.

PRÁCTICA INVESTIGATIVA

Investigación: Evolución de los lenguajes de programación desde los años 80hasta la actualidad

Consigna:

- El trabajo de investigación debe presentarse de manera estructurada y coherente para comunicar efectivamente los hallazgos y conclusiones.
- Asegúrese de que el estilo y formato del trabajo sigan las pautas de citación y presentación establecidas por la institución.
- Utilice un lenguaje claro y evite jergas o términos técnicos innecesarios.
- Revise y corrija la redacción y la gramática antes de la presentación final (ver anexo).

PRÁCTICA TEST

Antes de realizar el test, es importante considerar varios aspectos para aprovechar al máximo la experiencia.

Consideraciones para realizar el test

1. Lee y revisa el documento.
2. Observa el video de la clase magistral.
3. Revisa en el calendario el día y hora en que se debe de realizar el Test.
4. Realiza el Test de la unidad considerando lo Siguiente:

- No cambiarte entre paginas durante la realización del Test.
- Revisa que tu conexión a internet sea estable.
- Recuerda que todo el contenido está inmerso dentro del documento base.

5. Revisa tu puntuación, de inmediato al finalizar de la actividad.

IV. DATOS GENERALES DE LA UNIDAD 4

Información General

1.1.	Nivel	Primero					
1.2.	Nombre Unidad	Enfoques de Programación					
1.3.	Tema 1 – Unidad	La programación					
1.4.	Tema 2 – Unidad	Enfoques					
1.5.	Unidad organizacional	Unidad Profesional					
1.6.	Unidad	4					
1.7.	Total, Horas Unidad	15					
1.8.	Detalle de horas Unidad	ACD	6	AA	4	APE	5

Resultado de Aprendizaje

Al finalizar la Unidad 4 sobre Enfoques de Programación, los estudiantes serán capaces de comprender y aplicar los diferentes paradigmas de programación, incluyendo la programación imperativa, estructurada y orientada a objetos, así como identificar y analizar las tendencias emergentes en el desarrollo de software, incluyendo Cloud Computing y sistemas ERP, para adaptarse eficazmente a las demandas cambiantes de la industria tecnológica.

Metodología

La clase virtual grabada comienza con una introducción clara sobre lo que los estudiantes aprenderán en esa sesión. Se establecen los objetivos de aprendizaje para que los estudiantes sepan qué esperar y en qué enfocarse.

El contenido de la asignatura se desarrolla dividiéndolo en secciones o módulos. Se utiliza una estructura organizada y lógica para abordar cada tema, combinando diferentes enfoques pedagógicos según el contenido, como explicaciones, ejemplos, demostraciones, gráficos y diagramas.

Aunque la clase es grabada, es importante incorporar elementos interactivos para mantener el compromiso de los estudiantes. Esto se puede lograr a través de preguntas para reflexionar, ejercicios, cuestionarios o pausas para discusión, fomentando su participación activa.

DESARROLLO

Unidad 4: Enfoques de Programación

Tema 1: La programación

- Clasificación
- Programación imperativa
- Programación estructurada
- Programación Orientada a Objetos

Objetivo: Proporcionar a los estudiantes a desarrollar una comprensión profunda de los distintos enfoques de programación, incluyendo su clasificación y las características específicas de la programación imperativa, estructurada y orientada a objetos, para aplicar estos paradigmas en el diseño y desarrollo de sistemas informáticos eficientes y mantenibles.

La Programación

La programación es un proceso fundamental en el desarrollo de software que implica la creación de algoritmos y la codificación de instrucciones para que una computadora pueda realizar tareas específicas. En este tema, exploraremos diferentes enfoques de programación que han evolucionado a lo largo del tiempo para abordar diversos desafíos en el desarrollo de software.

Clasificación de la Programación

La programación puede clasificarse en diferentes paradigmas, cada uno con sus propias características y técnicas de implementación. Los principales paradigmas de programación incluyen la programación imperativa, la programación estructurada y la programación orientada a objetos.

Programación Imperativa

Es uno de los paradigmas más antiguos y fundamentales en la programación de computadoras. Se centra en describir cómo se deben realizar las operaciones mediante la especificación de los pasos o instrucciones que la computadora debe seguir para alcanzar un resultado deseado. Los lenguajes de programación imperativos, como C y Pascal, se basan en este enfoque y utilizan estructuras como bucles y condicionales para controlar el flujo de ejecución del programa.

1. **Programación Imperativa:**
   - Ejemplo en C:

```c
#include <stdio.h>
int main() {
    int numero = 5;
    if(numero > 0) {
        printf("El número es positivo.\n");
    } else {
        printf("El número es negativo o cero.\n");
    }
    return 0;
}
```

En este ejemplo, el programa determina si un número es positivo o negativo utilizando una estructura de control `if` para evaluar una condición y tomar decisiones basadas en ella.

Programación Estructurada

La programación estructurada es un enfoque más organizado y fácil de entender para escribir programas. Se basa en la idea de dividir un programa en bloques de código lógicos y cohesivos, como funciones o procedimientos, que realizan tareas específicas. Este enfoque promueve el modularidad, la legibilidad y la reutilización del código. Los lenguajes de programación como Python y Java son ejemplos de lenguajes que fomentan la programación estructurada.

- Ejemplo en Python:

```python
def calcular_area_rectangulo(base, altura):
    area = base * altura
    return area

base = 10
altura = 5
area_rectangulo = calcular_area_rectangulo(base, altura)
print("El área del rectángulo es:", area_rectangulo)
```

Este ejemplo muestra una función en Python para calcular el área de un rectángulo. La función `**calcular_area_rectangulo**` toma la base y la altura como argumentos, realiza el cálculo y devuelve el resultado. El programa principal luego llama a esta función y muestra el área calculada.

Programación Orientada a Objetos

La programación orientada a objetos (POO) es un paradigma de programación que se basa en el concepto de "objetos", que son entidades que contienen datos y funciones relacionadas. En la POO, los programas se estructuran en torno a clases y objetos, lo que permite modelar entidades del mundo real de manera más natural y modular. Este enfoque facilita la reutilización del código, la encapsulación de datos y el mantenimiento del software. Lenguajes como Java, C++ y Python son ampliamente utilizados en el desarrollo de software orientado a objetos.

- Ejemplo en Java:

java — Copy code

```java
class Circulo {
    double radio;

    // Método para calcular el área del círculo
    double calcularArea() {
        return Math.PI * radio * radio;
    }
}

public class Main {
    public static void main(String[] args) {
        Circulo circulo1 = new Circulo();
        circulo1.radio = 5;
        System.out.println("El área del círculo es: " + circulo1.calcularArea
    }
}
```

En este ejemplo, se define una clase `Circulo` que tiene una variable de instancia `radio` y un método `calcularArea()` que calcula el área del círculo. Luego, en la clase `Main`, se crea una instancia de `Circulo`, se establece el radio y se llama al método `calcularArea()` para obtener el área del círculo.

ACTIVIDADES PRÁCTICAS

Tabla

Detalle de la Práctica

Practicas	Instrucciones	Duración de la Practica	Mecanismo de Evaluación
Foro de debate	Revisar Actividades Prácticas	0:30 mm	Por rúbricas
Práctica investiga: Taller/Ensayo/Investigación	Revisar Actividades Prácticas	3:30 h	Por rúbricas
Tutorías Prácticas	Revisar Actividades Prácticas	2:00 mm	Por participación en clase
Test evaluativo	Revisar Actividades Prácticas	0:30 mm	Formativa

Nota. Elaboración propia

Consideraciones Generales

1. Lee y revisa la Guía de Estudio.
2. Observa el video de la clase magistral.
3. Participa activamente en el FORO DEBATE.
4. Revisa en el calendario el día y hora en que se realizará las actividades prácticas y la tutoría.
5. Realizar la práctica propuesta por el docente.
6. Revisa las Rúbricas acorde a la actividad prácticas.
7. Toma los principales apuntes que vayas rescatando del texto.
8. Reflexiona sobre los temas tratados en la clase.
9. Sube el documento en formato PDF en la plataforma Moodle.
10. Revisa la calificación de tus actividades prácticas.

FORO DE DEBATE

Tema: Cual es el mejor enfoque de programación según su punto de vista, argumente la respuesta.

Instrucciones para Participar:

- Exponga su punto de vista acerca del tema.
- Comenta las respuestas de otros participantes.
- Promueve un ambiente respetuoso y constructivo.

PRÁCTICA INVESTIGATIVA

Ensayo: Argumentar sobre los desafíos y dificultades que enfrentan los programadores al cambiar de un paradigma de programación a otro.

Consigna:

- El trabajo de investigación debe presentarse de manera estructurada y coherente para comunicar efectivamente los hallazgos y conclusiones.
- Asegúrese de que el estilo y formato del trabajo sigan las pautas de citación y presentación establecidas por la institución.
- Utilice un lenguaje claro y evite jergas o términos técnicos innecesarios.
- Revise y corrija la redacción y la gramática antes de la presentación final (ver anexo).

PRÁCTICA TEST

Antes de realizar el test, es importante considerar varios aspectos para aprovechar al máximo la experiencia.

Consideraciones para realizar el test

1. Lee y revisa el documento.
2. Observa el video de la clase magistral.
3. Revisa en el calendario el día y hora en que se debe de realizar el Test.
4. Realiza el Test de la unidad considerando lo Siguiente:

- No cambiarte entre paginas durante la realización del Test.
- Revisa que tu conexión a internet sea estable.
- Recuerda que todo el contenido está inmerso dentro del documento base.

5. Revisa tu puntuación, de inmediato al finalizar de la actividad.

Tema 2: Enfoques

- Tendencia de Desarrollo
- Clouds (SAAS, PAAS, IAS)
- Tendencia de Desarrollo ERP

Objetivo: Proporcionar a los estudiantes a desarrollar una comprensión integral de las tendencias actuales en el desarrollo de software, incluyendo la adopción de soluciones en la nube (SaaS, PaaS, IaaS) y las tendencias en el desarrollo de sistemas de planificación de recursos empresariales (ERP), para aplicar estos conocimientos en la implementación y gestión de sistemas informáticos avanzados y adaptables.

Tendencia de Desarrollo

Es un concepto amplio que abarca las prácticas, metodologías y tecnologías emergentes que están dando forma al panorama del desarrollo de software. Desde la adopción de metodologías ágiles hasta la integración de inteligencia artificial y aprendizaje automático, las tendencias de desarrollo reflejan la rápida evolución del campo tecnológico. Un ejemplo de tendencia de desarrollo es la programación sin código (no-code), que permite a los usuarios crear aplicaciones complejas sin necesidad de conocimientos profundos de programación. Esta tendencia está democratizando el desarrollo de software y permitiendo a una amplia gama de personas crear soluciones tecnológicas.

Caso de Estudio: Programación sin código en una empresa de comercio electrónico

Una empresa de comercio electrónico busca mejorar su plataforma de ventas en línea para ofrecer una experiencia de usuario más personalizada. Utilizando una herramienta de programación sin código, el equipo de desarrollo implementa una serie de mejoras en el sitio web, como la creación de páginas de producto dinámicas, la optimización del proceso de pago y la integración de recomendaciones de productos basadas en el historial de compras. El resultado es una plataforma más ágil y adaptable, que permite a la empresa responder rápidamente a las necesidades y preferencias cambiantes de sus clientes.

Clouds (SAAS, PAAS, IAS): El Cloud Computing, o computación en la nube, ha revolucionado la forma en que las empresas y los individuos acceden, almacenan y procesan datos. Este enfoque proporciona servicios de infraestructura de TI, plataformas de desarrollo y software a través de internet, permitiendo una escalabilidad, flexibilidad y eficiencia sin precedentes. Dentro del Cloud Computing, se distinguen tres modelos de servicio: Software as a Service (SaaS), Platform as a Service (PaaS) e Infrastructure as a

Service (IaaS). Por ejemplo, un servicio de correo electrónico como Gmail es un ejemplo de SaaS, mientras que una plataforma de desarrollo como Google App Engine representa un ejemplo de PaaS, y proveedores como Amazon Web Services ofrecen IaaS, proporcionando infraestructura de servidores virtualizados a través de la nube.

Clouds (SAAS, PAAS, IAS):

Caso de Estudio: Implementación de un sistema de gestión empresarial en la nube

Una empresa de servicios financieros decide migrar su sistema de gestión empresarial a la nube para mejorar la accesibilidad, escalabilidad y seguridad de sus datos. Utilizando un servicio de SaaS, el equipo implementa un ERP que incluye módulos para contabilidad, gestión de recursos humanos y gestión de clientes.

Además, aprovechan un servicio de PaaS para personalizar y desarrollar funcionalidades específicas para sus necesidades comerciales. Con la infraestructura proporcionada por un proveedor de IaaS, la empresa garantiza la disponibilidad y la fiabilidad del sistema en todo momento.

Tendencia de Desarrollo ERP

Los sistemas de planificación de recursos empresariales (ERP) son aplicaciones de software integradas que gestionan y automatizan procesos empresariales clave, como contabilidad, recursos humanos, inventario y producción. La tendencia de desarrollo ERP se centra en la evolución de estos sistemas para adaptarse a las necesidades cambiantes de las organizaciones en un entorno digitalizado.

Por ejemplo, la integración de análisis avanzados y la inteligencia artificial en los sistemas ERP permite a las empresas obtener información valiosa sobre sus operaciones y tomar decisiones más informadas.

Asimismo, la tendencia hacia los ERP basados en la nube está ganando impulso, ya que proporciona mayor flexibilidad y escalabilidad a las empresas, permitiéndoles acceder a sus sistemas desde cualquier lugar y en cualquier momento.

Caso de Estudio: Implementación de un sistema ERP con análisis avanzado en una empresa manufacturera

Una empresa manufacturera busca optimizar sus operaciones implementando un sistema ERP que integre análisis avanzados para mejorar la eficiencia y la productividad. El nuevo sistema recopila datos en tiempo real de las líneas de producción y utiliza algoritmos de

aprendizaje automático para predecir la demanda de productos y optimizar la gestión de inventario. Además, el sistema proporciona paneles de control personalizados que permiten a los gerentes supervisar el rendimiento de la producción y tomar decisiones basadas en datos. Como resultado, la empresa logra reducir los costos de producción, mejorar la calidad del producto y aumentar la satisfacción del cliente.

ACTIVIDADES PRÁCTICAS

Tabla

Detalle de la Práctica

Practicas	Instrucciones	Duración de la Practica	Mecanismo de Evaluación
Foro de debate	Revisar Actividades Prácticas	0:30 mm	Por rúbricas
Práctica investiga: Taller/Ensayo/Investigación	Revisar Actividades Prácticas	3:30 h	Por rúbricas
Tutorías Prácticas	Revisar Actividades Prácticas	2:00 mm	Por participación en clase
Test evaluativo	Revisar Actividades Prácticas	0:30 mm	Formativa

Nota. Elaboración propia

Consideraciones Generales

1. Lee y revisa la Guía de Estudio.
2. Observa el video de la clase magistral.
3. Participa activamente en el FORO DEBATE.
4. Revisa en el calendario el día y hora en que se realizará las actividades prácticas y la tutoría.
5. Realizar la práctica propuesta por el docente.
6. Revisa las Rúbricas acorde a la actividad prácticas.
7. Toma los principales apuntes que vayas rescatando del texto.
8. Reflexiona sobre los temas tratados en la clase.
9. Sube el documento en formato PDF en la plataforma Moodle.
10. Revisa la calificación de tus actividades prácticas.

FORO DE DEBATE

Tema: En la programación, existen distintos enfoques de desarrollo que permiten solucionar problemas de manera estructurada y eficiente.

- ✓ ¿Cuál crees que es el enfoque de programación más útil para resolver problemas complejos?

Instrucciones para Participar:

- Exponga su punto de vista acerca del tema.
- Comenta las respuestas de otros participantes.
- Promueve un ambiente respetuoso y constructivo.

PRÁCTICA INVESTIGATIVA

Tema: Uso de Software y Herramientas en Diferentes Enfoques de Programación.

- ✓ Investiga herramientas o entornos de desarrollo que faciliten la implementación de cada enfoque (ejemplo: Python para funcional, Java para POO, C para estructurada).
- ✓ Realiza una breve demostración en un software y explica los resultados obtenidos.
- ✓ Entrega un informe en Word o PDF, con capturas de pantalla del proceso.

Consigna:

- El trabajo de investigación debe presentarse de manera estructurada y coherente para comunicar efectivamente los hallazgos y conclusiones.
- Asegúrese de que el estilo y formato del trabajo sigan las pautas de citación y presentación establecidas por la institución.
- Utilice un lenguaje claro y evite jergas o términos técnicos innecesarios.
- Revise y corrija la redacción y la gramática antes de la presentación final (ver anexo).

PRÁCTICA TEST

Antes de realizar el test, es importante considerar varios aspectos para aprovechar al máximo la experiencia.

Consideraciones para realizar el test

1. Lee y revisa el documento.
2. Observa el video de la clase magistral.
3. Revisa en el calendario el día y hora en que se debe de realizar el Test.
4. Realiza el Test de la unidad considerando lo Siguiente:

- No cambiarte entre paginas durante la realización del Test.
- Revisa que tu conexión a internet sea estable.
- Recuerda que todo el contenido está inmerso dentro del documento base.

5. Revisa tu puntuación, de inmediato al finalizar de la actividad.

V. Bibliografía

Bibliografía básica

G.M.R., W. (S.A.). Informatica Para No Iniciados. Ediciones Deusto, S.A.

Hernández Bejarano, M., & Baquero Rey, L. E. (2023). Programación Orientada A Objetos En Java: Buenas Prácticas. EDICIONES DE LA U.

Winters, T., Manshreck, T., & Wright, H. (2022). Ingeniería de Software en Google: Lecciones sobre Programación Aprendidas a lo largo del tiempo. MARCOMBO.

Trejos Buritica, O. I. (2021). Lógica de Programación: Solucionario en pseudocódigo/ Ejercicios resueltos, EDICIONES DE LA U

Trejos Buritica, O. I. (2023). Lógica de Programación. 2 edición. EDICIONES DE LA U

Bibliografía de consulta

Luhmann, N. (1996). Introducciuón a la Teoiria de Sistemas. Mexico: Reis. Obtenido de https://books.google.es/books?hl=es&lr=&id=U-_sNyAtvCQC&oi=fnd&pg=PA315&dq=Teor%C3%ADa+de+Sistemas&ots=tSshuLBTpG&sig=h42IpoKCwWPvs2i76wnN2GRrXnU#v=onepage&q=Teor%C3%ADa%20de%20Sistemas&f=false

Anexo 1: Modelo de Foro

Título del Foro:

[Nombre del Tema o Pregunta]

Desarrollo:

[Breve descripción del tema o pregunta]

Reglas del Foro:

Regla 1: Antes de participar en el debate, realiza una investigación exhaustiva sobre el tema en cuestión. Comprender los hechos, datos y argumentos relacionados con el tema fortalecerá tu posición y mejorarán la calidad de la discusión.

Regla 2: Muestra respeto hacia tus compañeros de debate, incluso si no estás de acuerdo con sus opiniones. Escucha activamente sus argumentos y evita interrupciones. Expresa tus puntos de vista de manera respetuosa y constructiva.

Regla 3: Contribuye de manera equitativa al debate, evitando la monopolización de la conversación. Respeta el tiempo asignado y asegúrate de que otros también tengan la oportunidad de expresar sus opiniones.

Anexo 2: Esquema de Ensayo

Caratula: (Escribe aquí los datos referentes a la materia, nombres completos, tema, periodo, Institución Educativa y logos)

Título:

(Escribe aquí el título de tu ensayo)

Introducción:

Desarrollo:

Conclusión:

Bibliografía

Anexo 3: Esquema de Taller/Investigación

Caratula:

(Escribe aquí los datos referentes a la materia, tema, periodo, Institución Educativa y logos)

Título:

(Escribe aquí el título principal de tu investigación)

Temas a investigar:

- **Tema 1:**
- **Tema 2:**
- **Tema 3:**

Conclusión:

Bibliografía

Anexos

Anexo 4: Rúbrica de Foro

INDICADORES	MS 5 PUNTOS	S 3 PUNTOS	PS 2 PUNTOS	PUNTAJE
Comprensión del tema	La respuesta argumentada demuestra conocimiento concreto del tema tratado.	La respuesta argumentada demuestra conocimiento general de la mayoría del tema.	La respuesta argumentada demuestra un conocimiento limitado del tema.	20 %
Cuestionamientos y criterios	Presentó las dudas e inquietudes que se le presentaron durante el desarrollo del tema, o al menos sacó diferentes conclusiones, que las expuso en el debate.	Presenta al menos 2 preguntas o argumentos, aunque demuestra que no ha comprende ni revisado en profundidad el material.	No presenta dudas ni argumentos, aunque conozca del tema, no aporta al debate.	20 %
Redacción y ortografía	No presenta errores ortográficos ni de redacción.	Presenta algunos errores ortográficos y/o de redacción.	Presenta muchos errores de redacción y de ortografía.	20 %
Interacción: relación/ confrontación con intervenciones de otros/as	Comparte activamente con de sus compañeros los temas planteados y resuelve dudas de sus compañeros con argumentos valederos en más de 2 intervenciones.	Comparte con de sus compañeros los temas planteados, argumentando en al menos de 2 intervenciones.	Plantea dudas a sus compañeros con el tema, sin embargo, no participa ni responde con argumentos lógicos. Solo interviene una vez.	20 %
Participación	Participa con más de 2 intervenciones, además de su contestación.	Participa al menos con 5 Intervenciones, además de su contestación.	No interviene en otra participación a más de la suya.	20 %
			TOTAL=	100 %

Puntaje: 5 puntos

Anexo 5: Rubrica de Ensayo

INDICACIONES	MS 10 PUNTOS	S 6 PUNTOS	PS 2 PUNTOS	PUNTAJE
Enfoque o Idea principal	La idea principal nombra el tema del ensayo y esquematiza los puntos principales a discutir.	La idea principal nombra el tema del ensayo.	La idea principal esquematiza algunos o todos los puntos a discutir, pero no menciona el tema.	20 %
Transiciones o Conectores	Una variedad bien pensada de transiciones fue usada. Estas muestran claramente como están las ideas conectadas.	Las transiciones muestran cÃ³mo están las ideas conectadas, pero hay muy poca variedad.	Algunas transiciones funcionan bien, pero la conexiÃ³n entre las ideas no es clara.	20 %
Conclusión	La conclusión es fuerte y dejó al lector con una idea absolutamente clara de la posición del autor. Un parafraseo efectivo de la idea principal empieza la conclusión.	La conclusión es evidente. La posición del autor es parafraseada en las primeras dos oraciones de la conclusión.	La posición del autor es parafraseada en la conclusión, pero no al principio de la misma.	20 %
Gramática y ortografía	El autor no comete errores de gramática ni de ortografía que distraen al lector del contenido del ensayo.	El autor comete 1 a 2 errores gramaticales u ortográficos que distraen al lector del contenido del ensayo.	El autor comete 3 a 4 errores gramaticales u ortográficos que distraen al lector del contenido del ensayo.	20 %
Fuentes	Todas las fuentes usadas para las citas, las estadísticas y los hechos son creíbles y están citadas correctamente.	Todas las fuentes usadas para las citas, las estadísticas y los hechos son creíbles y la mayoría está citada correctamente.	La mayoría de las fuentes usadas para las citas, las estadísticas y los hechos es creíble y está citada correctamente.	20 %
			TOTAL=	100 %

Puntaje: 10 puntos

Anexo 6: Rúbrica de Taller/Investigación

INDICADORES	MS 10 PUNTOS	S 6 PUNTOS	PS 2 PUNTOS	PUNTAJE
Organización de las ideas	El texto es coherente y tiene sentido toda la lectura.	Presenta ideas aisladas, no enlazadas entre sí.	El texto no tiene coherencia y las ideas no están claras.	30 %
Ortografía	Tiene 4 o menos faltas ortográficas	Tiene entre 5 y 9 faltas ortográficas.	Tiene más de 10 faltas ortográficas.	30 %
Calidad de la información presentada	La información está claramente relacionada con el tema y proporciona varios niveles de relación con sus diferentes entornos.	La información da respuesta a la consigna principal del artículo, mas no aporta niveles de relaciones de la comunicación con el entorno próximo.	La información proporcionada guarda relación con las categorías, aunque su calidad argumentativa es deficiente.	40 %
			TOTAL=	100 %

Puntaje: 10 puntos