ACTIVIDAD 05

• Diferencias, definiciones y ejemplos de Software y Hardware.

* Hardware: Son las partes físicas de una computadora (ej: teclado, pantalla).

​

* Software: Son los programas e instrucciones que le dicen al hardware qué hacer (ej: Windows, Word).

Diferencias clave:

* El hardware es tangible, el software no.

​

* El hardware se reemplaza, el software se actualiza.

​

* El hardware son los componentes físicos de una computadora, mientras que el software son los programas e instrucciones que se ejecutan en el hardware para realizar tareas específicas. Ambos son esenciales para el funcionamiento de un sistema informático.

• Diferencias, definiciones y ejemplos de Software propietario y libre.

Definiciones

- Software Propietario: Es un software que está sujeto a derechos de autor y tiene restricciones sobre su uso, distribución y modificación impuestas por el propietario . El código fuente no está disponible para el usuario final .

​

- Software Libre: Se refiere al software que otorga a los usuarios la libertad de ejecutar, estudiar, modificar y distribuir el software y adaptarlo a sus necesidades . El código fuente está disponible .

Diferencias Clave

- Código Fuente: En el software propietario, el código fuente generalmente no está disponible para el usuario final, mientras que en el software libre, el código fuente está disponible .

​

- Libertad de Uso: El software libre otorga a los usuarios la libertad de usar, modificar y distribuir el software, mientras que el software propietario impone restricciones sobre estas actividades .

​

- Costo: El software propietario generalmente requiere el pago de una licencia para su uso, mientras que el software libre a menudo está disponible de forma gratuita, aunque algunas distribuciones o servicios asociados pueden tener un costo .

​

- Modificación: Los usuarios pueden modificar el software libre para adaptarlo a sus necesidades, mientras que el software propietario generalmente no permite modificaciones por parte del usuario .

​

- Distribución: El software libre puede ser distribuido libremente, mientras que la distribución del software propietario está restringida y controlada por el propietario .

​

- Soporte: El soporte para software propietario generalmente lo proporciona el proveedor, mientras que el soporte para software libre puede provenir de la comunidad de usuarios o de empresas especializadas .

Ejemplos

- Software Propietario:

​

- Microsoft Windows

​

- Adobe Photoshop

​

- Microsoft Office

​

- Software Libre:

​

- Linux (sistema operativo)

​

- LibreOffice (suite de oficina)

​

- Mozilla Firefox (navegador web)

• Esquema y partes del internet.

Esquema de Internet

* Infraestructura Física

​

* Cables de fibra óptica

​

* Cables de cobre

​

* Redes inalámbricas (Wi-Fi, 4G/5G)

​

* Satélites

​

* Proveedores de Servicios de Internet (ISPs)

​

* Proveen acceso a Internet a usuarios y organizaciones.

​

* Ejemplos: Telmex, AT&T, Movistar, etc.

​

* Redes de Distribución de Contenido (CDNs)

​

* Almacenan contenido cerca de los usuarios para una entrega más rápida.

​

* Ejemplos: Akamai, Cloudflare.

​

* Servidores

​

* Servidores web (almacenan sitios web)

​

* Servidores de correo electrónico

​

* Servidores de bases de datos

​

* Servidores DNS (Domain Name System)

​

* Protocolos de Internet

​

* TCP/IP (Transmission Control Protocol/Internet Protocol)

​

* HTTP/HTTPS (Hypertext Transfer Protocol/Secure)

​

* DNS (Domain Name System)

​

* SMTP, POP3, IMAP (para correo electrónico)

​

* Dispositivos de Usuario Final

​

* Computadoras

​

* Teléfonos inteligentes

​

* Tabletas

​

* Dispositivos IoT (Internet de las Cosas)

​

* Aplicaciones y Servicios

​

* Navegadores web

​

* Correo electrónico

​

* Redes sociales

​

* Servicios de streaming

​

* Aplicaciones móviles

Partes Clave de Internet

1. Infraestructura Física:
2. ​

* Cables: La columna vertebral de Internet, que incluye cables de fibra óptica submarinos y terrestres, así como cables de cobre.

​

* Redes Inalámbricas: Tecnologías como Wi-Fi, 4G y 5G que permiten la conexión inalámbrica a Internet.

​

* Satélites: Usados para proporcionar acceso a Internet en áreas remotas.

​

1. Proveedores de Servicios de Internet (ISPs):
2. ​

* Empresas que ofrecen acceso a Internet a usuarios domésticos y empresariales. Actúan como intermediarios entre los usuarios y la infraestructura de Internet.

​

3. Redes de Distribución de Contenido (CDNs):

​

* Redes de servidores distribuidos geográficamente que almacenan copias de contenido (sitios web, videos, etc.) para entregar el contenido de manera más rápida y eficiente a los usuarios, reduciendo la latencia.

​

1. Servidores:
2. ​

* Servidores Web: Almacenan y entregan sitios web a los navegadores de los usuarios.

​

* Servidores de Correo Electrónico: Manejan el envío y recepción de correos electrónicos.

​

* Servidores de Bases de Datos: Almacenan y gestionan grandes cantidades de datos.

​

* Servidores DNS: Traducen nombres de dominio (como “google.com”) a direcciones IP, permitiendo a los navegadores encontrar los servidores correctos.

​

5. Protocolos de Internet:

​

* TCP/IP: El conjunto de protocolos fundamental que gobierna cómo los datos se envían y reciben a través de Internet.

​

* HTTP/HTTPS: Protocolos para la transferencia de datos en la web. HTTPS añade una capa de seguridad mediante el cifrado.

​

* DNS: Traduce nombres de dominio a direcciones IP.

​

* SMTP, POP3, IMAP: Protocolos utilizados para enviar y recibir correos electrónicos.

​

1. Dispositivos de Usuario Final:
2. ​

* Dispositivos que los usuarios utilizan para acceder a Internet, incluyendo computadoras, teléfonos inteligentes, tabletas y dispositivos IoT.

​

7. Aplicaciones y Servicios:

​

* Navegadores Web: Permiten a los usuarios acceder y visualizar contenido web.

​

* Correo Electrónico: Permite el envío y recepción de mensajes electrónicos.

​

* Redes Sociales: Plataformas para la interacción social y el intercambio de información.

​

* Servicios de Streaming: Permiten la transmisión de contenido de audio y video en tiempo real.

​

* Aplicaciones Móviles: Aplicaciones diseñadas para dispositivos móviles que utilizan Internet para diversas funciones.

• GitHub y sus utilidades.

¿Qué es GitHub?

GitHub es una plataforma de desarrollo colaborativo basada en la nube que utiliza el sistema de control de versiones Git. Permite a los desarrolladores almacenar, gestionar y colaborar en proyectos de código de manera eficiente .

Utilidades Clave de GitHub

* Control de Versiones: GitHub utiliza Git para rastrear y gestionar los cambios en el código fuente, permitiendo a los desarrolladores revertir a versiones anteriores y trabajar en paralelo sin conflictos .

​

* Colaboración: Facilita la colaboración entre equipos de desarrollo, permitiendo a los miembros revisar el código, proponer cambios y fusionar contribuciones de manera organizada .

​

* Gestión de Proyectos: Ofrece herramientas para la gestión de proyectos, como la creación de “issues” para el seguimiento de errores y nuevas funcionalidades, y la organización de tareas mediante “project boards” .

* Automatización (CI/CD): Permite automatizar los flujos de trabajo de desarrollo de software mediante GitHub Actions, facilitando la integración continua (CI) y la entrega continua (CD) .

* Seguridad: Incluye herramientas de seguridad integradas para identificar y corregir vulnerabilidades en el código, así como para gestionar las dependencias de manera segura .

​

* GitHub Copilot: Asistente de codificación impulsado por IA que sugiere líneas de código y funciones completas directamente en el editor, mejorando la productividad y la experiencia del desarrollador .

* GitHub CLI: Permite interactuar con GitHub desde la línea de comandos, facilitando la gestión de repositorios, la creación de “pull requests” y la revisión de “issues” sin necesidad de utilizar la interfaz web .

​

* GitHub Desktop: Aplicación de escritorio que simplifica el uso de Git y GitHub, ofreciendo una interfaz gráfica intuitiva para realizar operaciones comunes como "commit", “push”, “pull” y “branching” .

• Partes y acciones de GitHub (Repositorios, Commit, Branch, PullRequest, Marge, Organizations, etc).

Partes Clave de GitHub

* Repositorios: Son el elemento más básico de GitHub. Un repositorio contiene todo el código, archivos y el historial de revisiones de cada archivo. Los repositorios pueden tener múltiples colaboradores y ser públicos o privados .

* Commit: Un commit registra los cambios realizados en uno o más archivos en tu “branch” (rama). Cada commit tiene un ID único (SHA o hash) que identifica los cambios específicos, cuándo se hicieron y quién los creó. Al hacer un commit, debes incluir un mensaje que describa brevemente los cambios .

​

* Branch (Rama): Un branch permite desarrollar funciones, corregir errores o experimentar con nuevas ideas de forma segura en un área contenida del repositorio. Siempre se crea un branch a partir de uno existente, generalmente el branch principal del repositorio .

* Pull Request: Es una propuesta para fusionar un conjunto de cambios de un branch en otro. Los colaboradores pueden revisar y discutir los cambios propuestos antes de fusionarlos .

​

* Merge: Es la acción de integrar los cambios de un pull request en el branch de destino. Cualquier persona con acceso de “push” al repositorio puede completar el “merge” .

​

* Organizations: Son cuentas compartidas donde las empresas y los proyectos de código abierto pueden colaborar en muchos proyectos a la vez, con funciones administrativas y de seguridad sofisticadas .

Acciones Comunes en GitHub

* Crear un Repositorio: Para iniciar un nuevo proyecto o almacenar código existente .

​

* Clonar un Repositorio: Para crear una copia local de un repositorio remoto en tu computadora .

​

* Crear un Branch: Para aislar el trabajo de desarrollo sin afectar otros branches en el repositorio .

​

* Hacer un Commit: Para guardar los cambios realizados en los archivos del repositorio con un mensaje descriptivo .

​

* Crear un Pull Request: Para proponer cambios y colaborar en ellos antes de fusionarlos en el branch principal .

​

* Fusionar un Pull Request (Merge): Para integrar los cambios propuestos en el branch de destino después de la revisión y aprobación .

​

* Eliminar un Branch: Para limpiar los branches que ya no son necesarios después de que sus cambios se han fusionado .

Estas partes y acciones permiten a los desarrolladores colaborar de manera eficiente, gestionar el código y mantener un historial de cambios organizado en GitHub.

• Definición e interacción de Sistemas operativos con el Hardware.

Definición de Sistema Operativo (SO)

Un Sistema Operativo (SO) es un software esencial que actúa como intermediario entre el hardware de una computadora y las aplicaciones de usuario. Gestiona los recursos del hardware y proporciona servicios comunes para las aplicaciones, permitiendo que los usuarios interactúen con la computadora de manera eficiente .

Interacción del Sistema Operativo con el Hardware

1. Gestión de Recursos:
2. ​

* CPU: El SO asigna tiempo de procesamiento de la CPU a diferentes procesos y aplicaciones, optimizando el rendimiento del sistema.

​

* Memoria: Gestiona la asignación y liberación de memoria RAM para las aplicaciones, evitando conflictos y asegurando un uso eficiente de la memoria.

​

* Almacenamiento: Controla el acceso a dispositivos de almacenamiento (discos duros, SSDs) y gestiona el sistema de archivos.

​

* Dispositivos de Entrada/Salida (E/S): Facilita la comunicación entre el hardware (teclado, ratón, impresora) y las aplicaciones a través de controladores de dispositivos.

​

1. Abstracción del Hardware:
2. ​

* El SO proporciona una capa de abstracción que oculta la complejidad del hardware a las aplicaciones. Esto permite a los desarrolladores escribir software que funcione en diferentes tipos de hardware sin necesidad de conocer los detalles específicos de cada dispositivo.

​

3. Servicios del Sistema:

​

* Gestión de Procesos: Crea, programa y finaliza procesos, gestionando su ejecución y asignación de recursos.

​

* Sistema de Archivos: Organiza y gestiona los archivos y directorios en los dispositivos de almacenamiento.

​

* Interfaz de Usuario (UI): Proporciona una interfaz gráfica (GUI) o de línea de comandos (CLI) para que los usuarios interactúen con el sistema.

​

* Seguridad: Implementa mecanismos de seguridad para proteger el sistema contra accesos no autorizados y amenazas.

​

1. Comunicación con el Hardware:
2. ​

* El SO utiliza controladores de dispositivos (drivers) para comunicarse con el hardware. Los controladores son software específico que permite al SO entender y utilizar las funciones de cada dispositivo.

​

* El SO envía comandos al hardware a través de interrupciones y llamadas al sistema, permitiendo que las aplicaciones soliciten servicios del hardware.

​

5. Ejemplos de Interacción:

​

* Cuando abres un archivo, la aplicación solicita al SO que acceda al disco duro. El SO utiliza el controlador del disco duro para leer los datos y los entrega a la aplicación.

​

* Cuando escribes en el teclado, el hardware envía una señal al SO, que la interpreta y la muestra en la pantalla.

<https://github.com/dianacortess-eng/Cultura-digital/blob/main/Tarea04.pdf>