****

**INTEGRANTES**

**Arrieta, Axel**

**Batallón Costa, Juan Pablo**

**Calcagni, Gabriel**

**Delgado, Daniela**

**Jacob, Elías David**

**Morales, Franco**

**Rebolloso, Mariana**

**Salinas, Agustín**

**Sosa, Cintia**

**Tomio Maximiliano**

**Arquitectura y sistemas operativos**

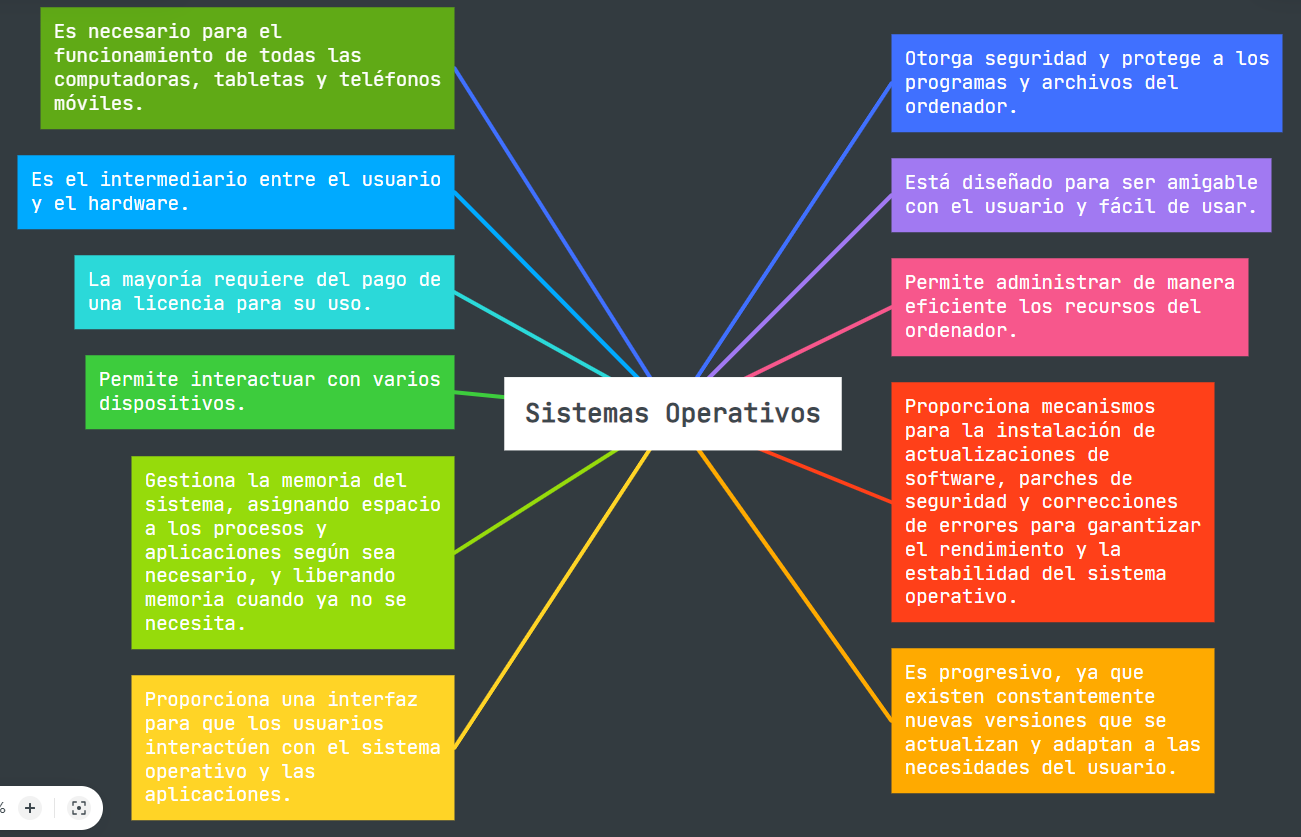
**Trabajo práctico clase N°1**

**Actividad N° 1:**

Realizar un mapa mental con Mindmeister desarrollando los puntos principales sobre sistemas operativos.

<https://www.mindmeister.com/es>

<https://mm.tt/app/map/3233680490?t=FLsldtkBo1>



**Actividad N° 2:**

**Responder las siguientes preguntas**

1. **¿Cuáles son las funcionalidades básicas de un sistema operativo?**

El sistema operativo es un software con características particulares, ya que debe administrar todos los recursos del sistema entre los usuarios y el resto de software. Sus funcionalidades básicas son:

1- Actuar de interfaz entre el usuario y el hardware de manera transparente para el usuario, ofreciendo un soporte a los usuarios para que sus acciones se transmitan con facilidad. No es necesario que sean especialistas en software o hardware para usarlo. ​

2 - Gestionar los recursos de software y hardware del equipo. El uso eficiente de los recursos es primordial puesto que son limitados. Dependiendo del fin y las tareas encomendadas al sistema informático, la eficiencia puede redirigirse a acciones diferentes. ​

1. **¿Cuáles son las características fundamentales de un sistema operativo?**

**Las características fundamentales que debe poseer un sistema operativo son:**

- *Capacidad de reacondicionarse* (mediante actualizaciones usualmente) para hacer uso de características o mejoras, tanto en componentes físicos como software.

- *Facilidad de uso:* considerando el fin con el que se empleará el sistema informático, la facilidad de manejo ha de ser primordial. Generalmente, una mayor comodidad implica mayor gasto de recursos (como por ejemplo un sistema gráfico de ventanas). Por eso, algunos sistemas operativos ganan en eficiencia a costa de restringir su manejabilidad.

- *Eficiencia: los recursos* (procesadores y núcleos, RAM, acceso a discos, red o cola de impresión) son limitados. El sistema operativo debe atender todas las peticiones de usuarios, programas y el propio sistema operativo para facilitar el acceso a los recursos. Ello debe hacerse barajando la importancia de cada solicitud y de quién desee hacer uso de los recursos. Esta tarea es muy compleja y crítica, ya que repercutirá en todo el sistema​​. El sistema operativo debe administrar de forma eficiente los recursos, atendiendo a sus objetivos.

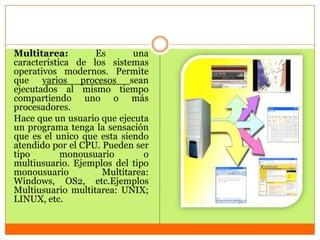
1. **Busca ejemplos (2 ó 3 de cada uno) de sistemas multiusuario y monotarea, y de multitarea y monousuario.**

**Ejemplos:**

**Multiusuario:** sistemas operativos con característica de multiusuario son VMS y Unix, así como sus múltiples derivaciones (e.g. IRIX, Solaris, etc.)

**Monotarea:** Lavarropas, impresoras, microondas.

**Multitarea:**



**Monousuario:** Al ser capaz de ejecutar diferentes procesos en un mismo momento, se pueden ejecutar diferentes programas y aplicaciones. Esto, por supuesto, aumenta mucho *la productividad* del sistema y suple esta necesidad creada por los sistemas de monotarea.

**Ejemplos de sistemas operativos monousuario son** (soportan un solo usuario por vez)**:**

* **MS-DOS.**
* **AmigaOS.**
* **Classic MacOS.**
* **Windows 1.0.**
* **Windows 2.0.**
* **Windows 3.0.**
* **Windows 3.1.x.**
* **Windows 95.**
* **Windows 98.**
* **Windows ME.**

1. **Busca en internet dos versiones de sistemas operativos únicamente textuales y explica por qué no presentan interfaz gráfica. Busca dos versiones gráficas de sistemas operativos.**

**Unix**

**El sistema operativo Unix** no tiene una interfaz gráfica predeterminada porque fue diseñado originalmente como un sistema operativo de línea de comandos. Aunque existen versiones de Unix que tienen interfaces gráficas, como Linux, Unix en sí mismo no está basado en una interfaz gráfica.

El servicio individual más importante en un sistema UNIX es provisto por init. init es el primer proceso que se inicia en todo sistema UNIX, siendo la última acción que el núcleo realiza al arrancar. Cuando init comienza su ejecución, continúa con el proceso de arranque del sistema, realizando varias tareas de inicio (chequear y montar sistemas de archivos, iniciar dominios, etc.).

La lista exacta de cosas que init realiza depende del sistema tipo UNIX con el que estemos trabajando; existen varios para elegir. init normalmente proporciona el concepto de modo de usuario individual (single user mode), en el cual nadie puede iniciar una sesión y root utiliza un intérprete de comandos en la consola; el modo usual es llamado modo multiusuario (multiuser mode). Algunos sistemas UNIX generalizan esto como niveles de ejecución (run levels). Así, los modos individual y multiusuario son considerados dos niveles de ejecución, y pueden existir otros niveles adicionales para, por ejemplo, ejecutar X-Windows en la consola.

**GNU**

**GNU/Linux** permite tener hasta 10 niveles de ejecución (runlevels) distintos, 0-9, pero normalmente solo algunos de estos niveles están definidos por defecto. El nivel de ejecución 0 se define como “sistema detenido (system halt)”. El nivel de ejecución 1 se define como “modo de usuario individual (single user mode)”. El nivel de ejecución 6 se define como “reinicio del sistema (system reboot)”. Los niveles de ejecución restantes dependen de como la distribución particular de GNU/Linux los haya definido, y varían significativamente entre distribuciones. Observando el contenido del archivo /etc/inittab podemos hacernos una idea de los niveles de ejecución preestablecidos en nuestro sistema y de como se encuentran definidos.

En el funcionamiento normal, init se asegura de que getty se encuentre trabajando para permitir que los usuarios puedan iniciar una sesión, y también se encarga de adoptar procesos huérfanos (aquellos cuyo proceso padre murió; en UNIX todos los procesos deben estar en un árbol individual, y por esta razón los procesos huérfanos deben ser adoptados).

Al cerrar el sistema, es init quien se encarga de matar todos los procesos restantes, desmontar todos los sistemas de archivos, y por último detener el procesador, además de cualquier otra cosa que haya sido configurado para hacer.