# Introducción a la informática

## Módulo 3. Herramientas de trabajo

## Clase 14. Lenguajes, paradigmas de programación y máquinas virtuales

Los lenguajes de programación son los medios a través de los cuales damos instrucciones a la computadora para indicarle lo que nuestro programa debe hacer. Estas instrucciones se compnene de una serie de pasos claros y precisos (algoritmos)

**Lenguajes específicos.** Resuelven problemas puntuales.

**Lenguajes generales:** Permiten desarrollar aplicaciones independientes del contexto.

**Lenguajes de alto nivel:** Se encuentran más cercanos al lenguaje natural. Ej: JavaScript.

**Lenguajes de bajo nivel:** Son utilizados para instrucciones específicas que permiten utilizar al máximo los recursos disponibles.

### Paradigmas de Programación

Simula 67: Nacimiento de la Programación Orientada a Objetos.

Paradigma: Forma de pensar bajo un modelo preestablecido.

**- Paradigma estructurado**
Sigue una línea de pensamiento donde se suele ejecutar una instrucción a la vez y uno se rige en un acotado set de instrucciones.

**- Paradigma de programación orientado a objetos**
El código puede agruparse de tal forma que llegue a representar una entidad y que interprete mensajes. Su fortaleza yace en utilizar abstracciones y crear entidades.

**- Paradigma funcional**
Se basa en las funciones matemáticas. Se caracteriza por su inmutabilidad.

**- Paradigma lógico**
Utiliza reglas lógicas para consultar al sistema y él mismo infiere qué hacer con base en las reglas lógicas establecidas.

**- Paradigma de programación con lenguaje específico de dominio**
Tratan de resolver problemáticas específicas.

**- Multiparadigma**
Ej: JavaScript: Permite programación estructurada, orientada a objetos y funcional.

## Ecosistema de los lenguajes

**Compilador:** Herramienta que convierte un código de alto nivel en instrucciones que la computadora puede entender y ejecutar: el código máquina.

Para que un programa ejecutado pueda funcionar en una máquina diferente a aquella en la que compiló se requiere:
- Una arquitectura de CPU similar
- Un sistema operativo similar

¿Cómo lograr que el código sea independiente de la plataforma en que se ejecuta? Dos opciones: máquinas virtuales e Intérpretes.

**Máquinas virtuales.** El código es compilado a código que entienda la VM. Las VM son propias del lenguaje de programación y deben ser actualizadas según los cambios en arquitectura.

**Intérprete:** Traduce línea por línea a lenguaje de máquina. El código fuente no es compilado previamente a código máquina para crear un ejecutable.

Existen lenguajes de programación:
**- Compilados.** El rendimiento es mayor.
**- Ejecutados en VM.** El código es portable.
**- Interpretados.** La traducción se hace línea por línea, haciendo más lenta la ejecución.

## Frameworks, librerías, IDE y editores de texto

**Framework:** Es un patrón o esquema que ayuda a la programación a estructurar el código, ahorrando tiempo y esfuerzos a los programadores. No están ligados necesariamente a un lenguaje concreto. Un framework es un esqueleto que predetermina la estructura del proyecto.
Ej: Laravel, Vue.js, Flask, ExpressJS, Spring MVC, Django, Angular, Rails.

**Librerías:** Una librería no es más que un conjunto de código que alguien ha realizado para que podamos reutilizar dentro de nuestro sproyectos. Normalmente están enfocadas a resolver problemas concretos; no brindan una estructura para el proyecto.
Grunt, SQLite, axios, Redux, JUint5.

**Editores de texto:** Se crearon para mostrar el código de una forma agradable y realizaban algunas acciones simples.

Atom, Sublime Text, Brackets.

**IDE (Entorno de desarrollo integrado):** A diferencia de los editores de texto, no trabajan con archivos y carpetas, sino con proyectos. Un proyecto es una carpeta en el disco duro, pero el IDE crea archivos adicionales al código para optimizar la experiencia del usuario.
Permiten realizar debbug en tiempo real, visualizar gráficamente diferentes elementos y ofrecen ayudas en tiempo real.

## Escritorios remotos

Son programas que permiten acceder e interactuar con una computadora a distancia a través de una conexión a Internet. La computadora a la que se accede de forma remota es llamada Host, mientras que la computadora desde la que se trabaja físicamente es el Cliente.

Para que se pueda realizar esta operación se requiere:
- Acceso a Internet
- Que ambas computadoras tengan la misma aplicación de escritorio remoto.
- Que ambas computadoras estén encendidas de manera simultánea.

Apps: TeamViewer, AnyDesk, Assist, Chrome Remote Desktop, Windows Remote Desktop.

Desventajas: Puede ser objeto de Ciberataques; el rendimiento depende completamente de la conexión a Internet.

## Máquinas virtuales

Formato .iso: Fotografía configuraciones y datos de un sistema de archivos, como un programa o sistema.

Una máquina virtual es un software que puede contener en su interior en sistema operativo, simulando una máquina real.

Existen dos clases:
VM de Sistemas: Emula una computadora completa. Software que permite ejecutar otro SO en su interior. La MV es creada en el Hipervisor, es una capa de software que se instala sobre la parte física de la memoria y le asigna recursos específicos.
VM de Procesos: Emula solamente un proceso, por ej. Una aplicación.

Dos tipos de Hipervisor:
Tipo 1: Corre directamente sobre el hardware.
Tipo 2: Corre sobre el SO.

Las VM permiten:
- Probar otros sistemas operativos sin cambiar el hardware.
- Ejecutar programas antiguos.
- Ejecutar aplicaciones de otros SO.
- Ofrece un entoerno de seguridad.
- Mejora el aprovechamiento del hardware

Actualmente: Virtualización del almacenamiento y las redes.

Desventajas:
- Son menos eficientes.

Componentes de Virtualización:

- Máquinas virtuales
- Administrador de máquinas virtuales
- Sistemas operativos base
- Hardware (Servidores físicos)

Beneficios de la virtualización:
- Tiempo de actividad
- Despliegue
- Ahorro de energía
- Snapshots
- Backups
- Alta disponibilidad
- Costo
- Eficiencia