# DE LA IDEA A LA MÁQUINA

**ALGORITMO**:
Secuencia ordenada de pasos, sin ambigüedades, que conducen a la solución de un problema dado. Mientras que el *algoritmo* es la receta lógica para resolver un problema, el *programa* es la receta escrita en un lenguaje para ser ejecutada por un ordenador.

**PARADIGMAS PRINCIPALES**:
- *Imperativo*: comandos que la computadora ejecuta en orden (C, Pascal).
- *Orientado a objetos*: Objetos que interactúan entre sí (Java, Python).
- *Declarativo*: Describe el resultado deseado, no el cómo (SQL).
- *Funcional*: Basado en una serie de funciones matemáticas (Haskell, Lisp).

**CLASIFICACIÓN DEL LENGUAJE**:

| Característica | Tipos | Ejemplo |
|---|---|---|
| Nivel de abstracción | Alto (cercano al humano) | Python, Java |
| Nivel de abstracción | Bajo (cercano a la máquina) | Ensamblador |
| Sistema de tipos | Fuerte (estricto) | Java, Python |
| Sistema de tipos | Débil (flexible) | JavaScript |

**EVOLUCIÓN DE LENGUAJES:**
- *1ª Gen*: Lenguaje máquina (binario).
- *2ª Gen*: Primeros ensambladores.
- *3ª Gen*: Primeros lenguajes de alto nivel (C, Pascal).
- *4ª Gen*: Lenguajes RAID y orientado a objetos, capaces de generar código.
- *5ª Gen*: Lenguajes orientados a la IA como LISP.

**FASES DEL TRADUCTOR**:
1) **Análisis léxico**: El código se agrupa en unidades lógicas (tokens).
2) **Análisis sintáctico**: Se comprueba que la estructura gramatical del programa sea correcta.
3) **Análisis semántico**: Verifica que las operaciones tengan sentido y coherencia.
4) **Código intermedio**: Se crea una versión de bajo nivel genérica e independiente.
5) **Código objeto**: Se traduce al código máquina final que ejecutará la CPU.

**LEMA DE JAVA**:
*Compila una vez, ejecuta donde sea*.

**EL PROCESO DE JAVA**:
1) **Código fuente**: El programador escribe el código en un archivo .java.
2) **Compilación**: El compilador lo traduce a un código binario intermedio.
3) **Bytecode**: Se genera el archivo .class, destinado a una máquina virtual.
4) **JVM Interpreta**: La máquina virtual de Java interpreta el Bytecode para ejecutarlo.
5) **Ejecución**: El programa corre en cualquier sistema con una JVM (Windows, Linux, etc.).

La portabilidad garantiza que las aplicaciones se puedan ejecutar en cualquier equipo, sin importar su hardware.

# DESARROLLO DE SOFTWARE

El desarrollo de software es una disciplina que estudia los principios y metodologías para el desarrollo y mantenimiento de sistemas software fiables, eficientes y seguros.

**FASES DEL CICLO DE VIDA**

- **Planificación**: Establecer objetivos y viabilidad del proyecto.
- **Análisis**: Definir el "qué" necesita el cliente. (**Esta es la fase más importante**)
- **Diseño**: Planificar el "cómo" se construirá la solución.
- **Codificación**: Escribir el código fuente del programa.
- **Pruebas**: Descubrir y corregir errores para asegurar la calidad.
- **Mantenimiento**: Evolucionar, corregir y optimizar el software.

**ESTRATEGIAS Y MODELOS**

- **Modelos Clásicos**: Secuenciales y rígidos se planifica todo por adelantado. Ej: Cascada.
- **Metodologías ágiles**: Iterativas y flexibles. Se adaptan al cambio continuo. Ej: Scrum.

**LENGUAJES Y PARADIGMAS**

Un paradigma de programación son las reglas del juego.

| Sistema de tipado | Descripción | Ejemplos de lenguajes |
|---|---|---|
| Estático y fuerte | Verificado en compilación, no hay conversiones implícitas. | C++, Java, C# |
| Dinámico y fuerte | Verificado en ejecución, no hay conversiones implícitas. | Python, Ruby |
| Estático y débil | Verificado en compilación, si hay conversiones implícitas. | C |
| Dinámico y débil | Verificado en ejecución, si hay conversiones implícitas. | JavaScript, PHP |

- **Bajo nivel**: Cercanos a la máquina (ensamblador). rápidos pero complejos y no portables.
- **Alto nivel**: Cercanos al lenguaje humano (Python, Java). Fáciles de usar y portables.

**EL EQUIPO PROFESIONAL**

- **Arquitecto**: Decide la estructura y la tecnología del proyecto. se involucra activamente en la fase de diseño para definir el "cómo" a nivel global.
- **Analista de sistemas**: Estudia el problema, realiza reuniones con el cliente y define los requisitos del software. Es la figura clave en la fase de análisis.
- **Programador**: Se encarga de codificar las tareas y probar los módulos transformando el diseño y los requisitos en un programa funcional.
- **Testeadores**: Su misión es conseguir que el programa funcione incorrectamente para descubrir y corregir defectos, asegurando la calidad del software.

# ENTORNOS DE DESARROLLO: TU TALLER DE PROGRAMACIÓN

Un **IDE** (***Integrated Development Environment***) es una aplicación que reúne en un solo lugar todas las herramientas necesarias para programar: escribir, probar, depurar y gestionar código.
Su objetivo es hacer la programación más rápida, eficiente y de mayor calidad.

**COMPONENTES ESENCIALES DE UN IDE**

1) **Editor de código**
   - Resaltado de sintaxis: mejora la legibilidad y detecta errores visuales.
   - Autocompletado / IntelliSense: sugiere código, nombres de variables, funciones, etc.
   - Análisis léxico y sintáctico: subraya errores en tiempo real (como faltas de punto y coma o paréntesis).

2) **Compilador o intérprete**
   - Convierte el código fuente en un formato que la máquina entiende.
   - Compilador: traduce todo el programa de una vez (crea un ejecutable).
   - Intérprete: ejecuta el código línea por línea.

3) **Depurador**
   - Permite detener el programa en puntos concretos (breakpoints) y examinar el estado de las variables.
   - Facilita encontrar errores y entender cómo funciona el código paso a paso.
   - Convierte la “caza de errores” en un proceso sistemático y seguro.

4) **Control de versiones**
   - Con herramientas como Git se pueden guardar versiones del código, volver atrás y trabajar en equipo sin conflictos.
   - GitKraken es un cliente gráfico que ayuda a visualizar los cambios.
   - Es esencial para el trabajo colaborativo y para experimentar sin miedo.

5) **Plugins o extensiones**
   - Permiten añadir funcionalidades al IDE: soporte de lenguajes, frameworks, correctores, etc.
   - Se instalan fácilmente desde “marketplaces”.
   - Adaptan el entorno a las necesidades del programador y mejoran la productividad.

# ¿QUÉ ES UML Y POR QUÉ ES ESENCIAL PARA EL DESARROLLO DE SOFTWARE MODERNO?

***UML*** es un estándar gráfico que permite especificar, visualizar y documentar sistemas de software y sus estructuras, comportamientos y relaciones. Se utiliza para modelar aplicaciones antes de programarlas, ayudando a planificar, diseñar y comunicar cómo funcionará el sistema.

**Características de UML**:

- Permite diseñar la arquitectura del software antes de escribir código, reduciendo errores y costos.
- Ayuda a gestionar la complejidad, especialmente en proyectos grandes.
- Facilita la reutilización de módulos y componentes.
- Garantiza que el sistema sea escalable, seguro, robusto y mantenible.
- Actúa como los planos de un edificio, pero para el software.

**Beneficios del modelado**:

- Visualiza el sistema completo antes de implementarlo.
- Permite verificar requisitos y funcionalidades.
- Aumenta las probabilidades de éxito de los proyectos.
- Facilita el trabajo colaborativo y la comunicación entre equipos.
- Eleva el nivel de abstracción, mostrando solo lo esencial y ocultando detalles innecesarios.

**UML 2.0 y sus mejoras principales**:

- Clasificadores anidados: permite crear estructuras jerárquicas (clases dentro de componentes o comportamientos dentro de clases).
- Modelado de comportamientos mejorado: los distintos tipos de diagramas de comportamiento ahora se integran mejor entre sí.
- Relación más fuerte entre modelos estructurales y de comportamiento.
- Permite representar no sólo clases y objetos, sino también procesos, regla de negocio y arquitecturas completas.
- Incluye nuevas herramientas como el Lenguaje de Restricciones de Objetos (OCL) y la Semántica de Acciones.

# UML: DIAGRAMAS DE CASOS DE USO

Describe la funcionalidad del sistema desde el punto de vista del usuario. Para ello:

- Define las interacciones entre actores (los que interactúan con la app) y sistema para realizar funciones concretas (caso de uso).
- Define las funcionalidades que tiene el sistema (que hace la app).
- Proporciona vista de alto nivel. No entra en detalles.
- Facilita la comunicación entre desarrolladores.
- Facilita escenarios de prueba.

# PREGUNTAS Y RESPUESTAS

1) **Explica cada una de las fases del modelo de cascada de desarrollo de software.**
   *Análisis de requisitos:*
   - Se recopila toda la información sobre lo que el cliente necesita.
   - Se definen qué funciones debe tener el sistema, sus entradas, salidas y restricciones.

   *Diseño del sistema:*
   - Se define cómo se implementarán los requisitos.
   - Se diseña la arquitectura general, la estructura de datos, los módulos, las interfaces y los diagramas UML.

   *Implementación (codificación):*
   - Los programadores desarrollan el código siguiendo el diseño previo.

   *Pruebas (testing):*
   - Se verifica que el sistema funcione correctamente y cumpla los requisitos.
   - Incluye pruebas unitarias, de integración y de sistema.

   *Implantación o despliegue:*
   - El sistema se instala en el entorno real del usuario.
   - Se capacita a los usuarios y se inicia su uso en producción.

   *Mantenimiento:*
   - Se corrigen errores detectados tras la entrega.
   - Se realizan mejoras o actualizaciones según nuevas necesidades.

2) **¿En qué parte del diseño serían útiles los gráficos UML de casos de uso, actividad, etc?**
   En la fase de diseño, porque ayudan a representar de forma gráfica la estructura y comportamiento del sistema.

3) **¿Qué es la metodología ágil (por ejemplo Scrum) y qué ventajas o diferencias implica respecto al modelo tradicional de diseño en cascada?**
   Metodología ágil (Scrum, Kanban, etc.): Es un modelo iterativo e incremental donde el desarrollo se divide en pequeños ciclos llamados sprints (normalmente de 2-4 semanas).
   - En cada sprint se planifica, desarrolla, prueba y entrega una parte funcional del producto.
   - El cliente participa continuamente, dando retroalimentación y ajustando prioridades.
   - Los equipos son autónomos y colaborativos.
   - Las metodologías ágiles son mejores para entornos cambiantes y colaborativos, donde se busca rapidez y flexibilidad.

4) **¿En qué se diferencia una página web de una aplicación web?**
   Una página web se centra principalmente en mostrar información al usuario.
   Por otro lado, una aplicación web permite al usuario realizar acciones, interactuar con los datos y ejecutar procesos, como ocurre en Gmail, Google Docs o Trello.

5) **¿En qué se diferencia una aplicación web de una aplicación instalable o de escritorio?**
   Una aplicación web no necesita instalación: se accede directamente desde el navegador y puede utilizarse desde cualquier dispositivo con conexión a internet. Las actualizaciones son automáticas, ya que el usuario siempre entra en la versión más reciente.

   En cambio, una aplicación de escritorio sí necesita instalarse en el ordenador y solo puede usarse desde ese dispositivo. Generalmente ofrece un rendimiento más rápido, pero requiere actualizaciones manuales y depende del sistema operativo.

6) **Nombra tres ejemplos de aplicaciones instalables y su correspondiente aplicación web.**
   - Microsoft Word tiene su versión web en Word Online.
   - Spotify, que se puede instalar como programa, también dispone de una versión web.
   - Adobe Photoshop tiene una alternativa online muy popular llamada Photopea.

7) **Como usuario de internet y desarrollador informático ¿qué ventajas le encuentras al uso y desarrollo de aplicaciones web?**
   Como usuario, las aplicaciones web ofrecen muchas ventajas: no requieren instalación, pueden utilizarse desde cualquier dispositivo, se actualizan de forma automática y guardan la información en la nube, por lo que no se pierde al cambiar de ordenador o móvil.
   Como desarrollador, las aplicaciones web son más prácticas porque un único desarrollo puede funcionar en distintos sistemas operativos. Además, es más fácil mantenerlas y actualizarlas, ya que los cambios se realizan en el servidor y se aplican a todos los usuarios al instante. También facilitan el trabajo en equipo y la colaboración en tiempo real.