METODOLOGÍA DE SISTEMAS 1

Ing. Nadia Petrelli

¿QUÉ ES UN IDE? (ENTORNO DE DESARROLLO INTEGRADO)

Un **IDE** es una aplicación que ofrece herramientas esenciales para escribir, compilar, depurar y ejecutar código en un solo lugar. Está pensado para facilitar el trabajo del programador y mejorar su productividad.
Componentes comunes de un IDE:

- **Editor de código fuente** : permite escribir el código con resaltado de sintaxis y autocompletado.
- **Compilador o intérprete** : convierte el código en un formato que la máquina puede ejecutar.
- **Depurador (debugger)** : herramienta para detectar errores, seguir la ejecución paso a paso, revisar valores de variables, etc.
- **Consola de salida** : muestra resultados de ejecución, errores de compilación o mensajes personalizados.
- Algunos IDEs también permiten la gestión de proyectos, integración con bases de datos, soporte para múltiples lenguajes, y conexiones a sistemas de control de versiones.

EJEMPLOS DE IDE

- **Visual Studio Code (VSCode)** : Muy liviano, extensible y gratuito. Soporta decenas de lenguajes gracias a sus extensiones.
- **IntelliJ IDEA** : Ideal para Java, aunque tiene soporte para otros lenguajes.
- **Eclipse** : Muy usado en el entorno académico y profesional, también orientado a Java.
- **NetBeans** : Similar a Eclipse, enfocado en Java pero con soporte para otros lenguajes.
- **PyCharm** : IDE especializado en Python.
- Cada uno tiene características específicas, pero todos comparten la estructura básica mencionada antes.

VENTAJAS

- **Productividad** : El autocompletado y las plantillas reducen el tiempo de escritura de código.
- **Reducción de errores** : La detección en tiempo real de errores evita problemas posteriores.
- **Mejor organización** : Se pueden gestionar archivos, carpetas, librerías y configuraciones desde un solo lugar.
- **Depuración fácil** : El entorno permite encontrar y corregir errores paso a paso.

¿QUÉ ES Y PARA QUÉ SIRVE UN SISTEMA DE CONTROL DE VERSIONES?

Un **Sistema de Control de Versiones (VCS)** es una herramienta que permite llevar un registro detallado de todos los cambios realizados en los archivos de un proyecto a lo largo del tiempo.

Ventajas clave:

- Permite volver atrás a versiones anteriores del código si algo sale mal.
- Facilita el trabajo en equipo: varios desarrolladores pueden trabajar al mismo tiempo en diferentes partes del proyecto.
- Deja un historial de cambios que permite entender quién hizo qué y cuándo.

Git y GitHub

- **Git** es el sistema de control de versiones más usado del mundo.
  Es distribuido, lo que significa que cada desarrollador tiene una copia completa del proyecto.
- **GitHub** es una plataforma que permite almacenar y compartir proyectos Git en la nube.

Flujo básico de trabajo con Git:

1. git init: inicializa un repositorio.
2. git add: prepara archivos para ser versionados.
3. git commit: guarda los cambios con un mensaje.
4. git push: envía los cambios al repositorio remoto (por ejemplo, en GitHub).
5. git pull: descarga cambios realizados por otros colaboradores.

Estos comandos permiten trabajar colaborativamente, mantener el orden del proyecto y evitar la pérdida de código.

DEPURACIÓN: ¿QUÉ ES?

La **depuración (debugging)** es el proceso de identificar, analizar y corregir errores (bugs) en el código. Una herramienta de depuración permite:

- **Establecer puntos de interrupción (breakpoints)** : para pausar la ejecución en una línea específica.
- **Ejecutar el programa paso a paso** : para observar el flujo y detectar errores lógicos.
- **Inspeccionar variables y estructuras de datos** : para comprobar si tienen los valores esperados.
- **Evaluar expresiones durante la ejecución** .
- Esto ayuda a entender exactamente qué está ocurriendo dentro del programa en tiempo real.

HERRAMIENTAS DE ANÁLISIS DE CÓDIGO

Estas herramientas analizan automáticamente el código para encontrar:

- Errores comunes.
- Problemas de estilo o formato.
- Fragmentos de código duplicado o innecesariamente complejos.

Ejemplos:

- **ESLint (JavaScript)** : analiza el código JS en busca de errores y ayuda a mantener un estilo uniforme.
- **PyLint (Python)** : chequea errores y también sugiere buenas prácticas.
- **SonarQube** : permite un análisis más profundo del código (calidad, seguridad, cobertura de tests, etc.).
- Estas herramientas son muy útiles para equipos grandes y para mantener proyectos limpios y eficientes a largo plazo.

¿QUÉ ES CI/CD? (INTEGRACIÓN Y DESPLIEGUE CONTINUO)

CI/CD es un conjunto de prácticas que automatizan las etapas del desarrollo de software, desde que un programador sube un cambio hasta que ese cambio llega al usuario final.

🔁 CI – Integración Continua

- Los desarrolladores integran frecuentemente su código (varias veces al día).
- Cada integración se verifica automáticamente con pruebas automatizadas.
- Se detectan errores de forma temprana, cuando son más fáciles de corregir.

🚀 CD – Despliegue Continuo

- Los cambios verificados se despliegan automáticamente en un entorno de pruebas o producción.
- Esto permite entregar software más rápidamente y con mayor confiabilidad.

Herramientas comunes de CI/CD

- **GitHub Actions**: permite crear flujos de trabajo (workflows) que se ejecutan automáticamente al hacer push o pull request.
- **GitLab CI/CD**: viene integrado con GitLab y permite definir pipelines directamente desde el repositorio.
- **Jenkins**: muy configurable, se usa en entornos más avanzados.
- **CircleCI, Travis CI**: otras opciones populares que ofrecen integración con múltiples plataformas.

Ejemplo de pipeline básico:

1. Se hace un push al repositorio.
2. Se ejecutan automáticamente pruebas unitarias.
3. Si pasan las pruebas, se genera una nueva versión del software.
4. Se despliega en un servidor de prueba o producción