Actividad de investigación Andreu Orenga 1ºDAW

1. Requisitos indispensables de un editor de código. Explicar que debe tener al menos un editor para considerarse una herramienta adecuada para crear código.

Los requisitos necesarios que debe tener un editor deben ser los siguientes:

- Sintaxis resaltada: el editor debe resaltar la sintaxis y diferenciarla con colores para crear un código mas comprensible.
- Autocompletado: debe ofrecer sugerencias mientras escribes que te ayuden a completar partes del código de forma mas rápida.
- Gestor de archivos y carpetas: organiza tus archivos y carpetas de manera intuitiva para facilitar la búsqueda y el acceso a los archivos y documentos necesarios para tus proyectos
- Control de versiones integrado: debe llevar un control de versiones integrado como puede ser Git.
- Extensible: tienen la posibilidad de añadir extensiones y elementos que puedas necesitar o que personalicen la experiencia del usuario.
- Soporte para múltiples lenguajes: es capaz de manejar varios lenguajes de programación.
- Interfaz amigable: debe tener una interfaz intuitiva y que facilite la navegación y sea fácil de entender para los principiantes.
- Previsualización y depuración: tiene capacidad de poder ver los errores del código y facilitar su corrección.

2. Diferencias entre un EC y un IDE. Poner en una tabla al menos 3 ejemplos de cada.

Un editor de código se centra en la edición y la escritura de código, es mas ligero y simple en comparación con los IDE, suele proporcionar solo las funciones esenciales que puedes ampliar con extensiones que deberás saber implementar.

Los IDE por otra parte ofrecen un conjunto mas amplio de herramientas aparte de la edición de codigo, lleva ya las funciones integradas de depuracion, compilación etc, no necesita la instalación de ningún tipo de complemento para hacer según que tipo de funciones.

Editores de Codigo	IDEs
Visual Estudio Code	Visual Estudio
Sublime text	Eclipse
	IntelliJ IDEA

3. **Apache Maven:**
    a. ¿Cuál es el propósito principal de Apache Maven en el desarrollo de software?

        Maven se utiliza principalmente para la gestión y construcción de proyectos de software. Su propósito es simplificar y automatizar el proceso de construcción, permitiendo a los desarrolladores gestionar fácilmente el ciclo de vida del software, desde la compilación hasta la distribución.

    b. ¿Cómo ayuda Maven en la gestión de dependencias en un proyecto?

        Maven simplifica la gestión de dependencias ya que descarga automáticamente las bibliotecas necesarias para un proyecto desde un repositorio centralizado. Esto facilita mucho la utilización de estas bibliotecas ya que no es necesario buscarlas, descargarlas y gestionarlas manualmente cuando las necesitas para algún proyecto

c. ¿En qué situaciones sería preferible utilizar Maven en lugar de otras herramientas de construcción?

Es preferible utilizar Maven en proyectos donde se valora más la convención que la configuración. Es muy útil en proyectos Java o también en proyectos vasados en JVM.

También es muy útil en proyectos donde se trabaja en entornos empresariales, donde la estandarización y la consistencia son esenciales.

4. **Apache Ant:**
   a. ¿Qué función cumple Apache Ant en el proceso de construcción de software?

   Apache ant es una herramienta de construcción que se utiliza para automatizar tareas en los procesos de desarrollo de software. Su función principal es ejecutar tareas de construcción , como compilar código, empaquetar dependencias y ejecutar pruebas, según las necesidades del proyecto.

   b. ¿Cómo se diferencia Apache Ant de otras herramientas de construcción, como Maven o Gradle?

   A diferencia de Maven o Gradle, Apache ant no se basa en convenciones. En lugar de utilizar una estructura predeterminada, Ant permite a los desarrolladores especificar las tareas de construcción de forma más personalizada en archivos XML. Esto proporciona flexibilidad, pero necesita una configuración más manual.

   c. ¿En qué tipo de proyectos o escenarios sería más adecuado utilizar Apache Ant?

   Es mas adecuado para proyectos que requieren un nivel más alto de personalización en el proceso de construcción. Es especialmente útil cuando la estructura del proyecto no sigue las convenciones estándar o cuando se integra con procesos o sistemas existentes que no admiten fácilmente otras herramientas de construcción.

5. **npm (Node Package Manager):**
    a. ¿Cuál es el propósito principal de npm en el ecosistema de desarrollo de JavaScript?

    El propósito principal de npm en el ecosistema de desarrollo de JavaScript es facilitar la gestión de paquetes (bibliotecas etc.) que se utilizan en proyectos basados en Node.js. Permite a los desarrolladores instalar, compartir y gestionar dependencias de manera eficiente.

    b. ¿Cómo utiliza npm la gestión de dependencias en proyectos basados en Node.js?

    Npm simplifica la gestión de dependencias al permitir a los desarrolladores especificar las dependencias de un proyecto en un archivo (package.json). Cuando se ejecuta el comando “npm install”, npm lee este archivo y descarga todas las dependencias automáticamente desde el repositorio npm. Asegura que todos los miembros del equipo de desarrollo utilizaran las mismas dependencias y versiones.

    c. ¿En qué medida npm facilita la automatización de tareas en el desarrollo web?

    Npm facilita la automatización de tareas en el desarrollo a través de scrips que se definen en el archivo “packcage.json”. Los desarrolladores pueden ejecutar comandos personalizados para la compilación de código, la ejecución de pruebas y la optimización de recursos, utilizando para ello la interfaz de línea de comandos de npm. Todo esto contribuye a la automatización de trabajos y mejora la eficiencia en el desarrollo.

6. **Bazel:**
    a. ¿Cuál es el objetivo principal de Bazel en el desarrollo de proyectos a gran escala?

    El objetivo principal de Bazel en el desarrollo de proyectos a gran escala es proporcionar un sistema de construccion eficiente y escalable. Bazel se enfoca en optimizar la velocidad de compilación y la gestión de dependencias para proyectos extensos, permitiendo un desarrollo más rápido.

b. ¿Cómo aborda Bazel los desafíos específicos asociados con la construcción de proyectos de gran envergadura?

Aborda este tipo de proyectos mediante la implementación de un modelo de construcción basado en reglas. Este enfoque permite construir solo las partes del proyecto que han cambiado, reduciendo en gran parte los tiempos de compilación. Además es capaz de gestionar eficientemente grandes conjuntos de código fuente.

c. ¿En qué situaciones sería recomendable elegir Bazel sobre otras herramientas de construcción?

Se recomienda elegir Bazel en situaciones donde se desarrollan proyectos a gran escala, especialmente cuando se trabaja en entornos con grandes equipos de ingenieros y múltiples repositorios. Bazel brilla en proyectos complejos donde la velocidad de compilación, la gestión eficiente de dependencias y la reproducibilidad del entorno de construcción son críticas.

7. **CMake:**
   a. ¿Cuál es el propósito de CMake en el desarrollo de proyectos basados en C y C++?

   CMake tiene como propósito facilitar la construcción de proyectos en C y C++. Su objetivo principal es proporcionar una forma, independiente de la plataforma, para describir el proceso de construcción, permitiendo crear archivos de construcción para diferentes sistemas y entornos.

   b. ¿Cómo aborda CMake la generación de archivos de construcción para diferentes plataformas?

   CMake utiliza la generación de scripts, creados con un archivo llamado CMakeLists.txt, para describir la estructura del proyecto y las dependencias. CMake luego genera archivos de construcción específicos para las plataformas , como makefiles en Unix o Visual Studio en Windows, en las que se van a utilizar.

c. ¿En qué escenarios o proyectos CMake podría ser la opción preferida en comparación con otras herramientas de construcción?

CMake es la opción preferida en proyectos donde la portabilidad es esencial, ya que permite a los desarrolladores generar archivos de construcción para diferentes sistemas operativos y entornos de compilación. También es útil en proyectos que utilizan bibliotecas externas ya que simplifica la integración de estas. Aporta flexibilidad y personalización en los procesos de construcción de proyectos.

8. **Gradle:**
   a. ¿Qué es Gradle y para qué se utiliza en el desarrollo de software?

   Gradle es una herramienta de automatización de tareas y construcción de proyectos. Se utiliza para compilar, empaquetar y desplegar aplicaciones. Utiliza un DLS basado en Groovy o Kotlin para describir la estructura y las tareas del proyecto.

   b. ¿Cuál es la principal ventaja de utilizar Gradle en comparación con otras herramientas de construcción de proyectos?

   La principal ventaja de Gradle es su flexibilidad y adaptación en diferentes escenarios. Ofrece un rendimiento sólido y soporte para la gestión de dependencias, lo que facilita la construcción de proyectos grandes y complejos.

   c. ¿En qué tipo de proyectos y lenguajes de programación es comúnmente utilizado Gradle?

   Es comúnmente utiliza en los proyectos que utilizan Java como lenguaje principal, pero también lo vemos en proyectos de Kotlin y Android. También se adapta fácilmente a proyectos tanto grandes como pequeños ya que es muy versátil y tiene capacidad para integrar diversas tecnologías.

d. ¿Cómo facilita Gradle la automatización de tareas en el ciclo de vida del desarrollo de software?

   Gradle utiliza un DLS para la automatización de tareas durante el ciclo de vida del desarrollo. Estas automatizaciones incluyen tareas como compilación, ejecución de pruebas, generación de informes, despliegue y más.

9. **JUnit:**
   a. ¿Qué es JUnit y cuál es su propósito en el desarrollo de software?

      El propósito principal de Junit es facilitar la escritura y la ejecución de pruebas unitarias en el desarrollo de software en Java, permite a os desarrolladores validar el comportamiento correcto de pequeñas unidades de código, como métodos y funciones.

   b. ¿Cómo ayuda JUnit a los desarrolladores a realizar pruebas unitarias de manera efectiva?

      JUnit proporciona anotaciones y clases que permiten definir y ejecutar pruebas unitarias de manera estructurada. Facilita la creación de casos de prueba, la organización de las pruebas y la verificación de resultados.

   c. ¿En qué entornos de desarrollo y proyectos es comúnmente utilizado JUnit?

      JUnit es comúnmente utilizado en entornos de desarrollo Java, tanto para proyectos pequeños como para grandes aplicaciones empresariales. Es una opción estándar para pruebas en el ecosistema Java y se integra fácilmente con varias herramientas de desarrollo y entornos de integración.

d. ¿Cuáles son los beneficios clave de implementar pruebas unitarias con JUnit en un proyecto de software?

-Mejora la calidad del código al detectar errores.

-Facilita la detección y corrección de problemas durante el desarrollo.

-Proporciona documentación con ejemplos de uso de código.

- Facilita la refactorización ya que garantiza que las unidades de código mantengan su funcionalidad.

e. ¿Puede proporcionar un ejemplo práctico de cómo se utiliza JUnit para realizar una prueba unitaria simple?

Supongamos que tienes una clase con un método que suma dos números. Una prueba simple de este código podría verse así:

```
import static org.junit.jupiter.api.Assertions.assertEquals;
import org.junit.jupiter.api.Test;

public class CalculatorTest {

    @Test
    public void testAdd() {
        Calculator calculator = new Calculator();
        int result = calculator.add(3, 4);
        assertEquals(7, result, "La suma de 3 y 4 debería ser 7");
    }
}
```

Esta prueba se utiliza para verificar que el resultado que esperamos del método es correcto.