UTN
UNIVERSIDAD TECNOLOGICA NACIONAL
FACULTAD REGIONAL TUCUMAN

Tecnicatura Universitaria en Programación (TUP-2024)

Cátedra de Metodología en Sistemas I
Profesora: Petrelli Nadia

Trabajo Práctico Nº4
C9
Grupo 3

Integrantes:

- Díaz Medina Tomás
- Díaz Rossini Juan José
- Gallo Genaro
- Hinojosa Alexis
- Liacoplo Gerónimo Emiliano
- Navarro Victor Leandro

Consignas

1. ¿Que es un Entorno de Desarrollo Integrado?
2. De ejemplos y explique ventajas del punto anterior
3. Qué es y para qué sirve un sistema de control de versiones
4. Definición de Git y GitHub
5. Qué es depuración
6. Qué son las herramientas de análisis
7. Que es un CI/CD (Integración y Despliegue Continuo)
8. Herramientas comunes de los CI/CD

Desarrollo

1. ¿Qué es un Entorno de Desarrollo Integrado?

Un entorno de desarrollo integrado (IDE), es una aplicación de software que agiliza el proceso de desarrollo de código para los programadores, proporcionando una eficiencia notable y elevando la productividad de los desarrolladores al integrar funciones como la edición, creación, prueba y empaquetado de software dentro de una aplicación de fácil manejo.

2. Ventajas de un IDE:

1) **Productividad aumentada:** Al integrar varias herramientas en una sola interfaz (editor, compilador, depurador, etc.), se agiliza el desarrollo.

2) **Autocompletado y sugerencias:** Ayuda a escribir código más rápido y con menos errores.

3) **Depuración eficiente:** Permite encontrar y corregir errores de forma visual y más sencilla.

4) **Gestión de proyectos:** Facilita la organización de archivos, carpetas y dependencias.

5) **Integración con sistemas de control de versiones:** Como Git, para el manejo de versiones del código.

6) **Simulación y pruebas integradas:** Algunos IDEs permiten correr pruebas automáticas o simular aplicaciones sin salir del entorno.

7) **Interfaz gráfica amigable:** Ideal para principiantes y profesionales.

Ejemplos de IDEs:

- **Visual Studio:** Muy usado para desarrollo en C#, .NET y otros lenguajes de Microsoft.
- **Eclipse:** Popular para Java, también soporta otros lenguajes con plugins.
- **IntelliJ IDEA:** IDE muy potente para Java y Kotlin.
- **PyCharm:** Especializado en Python.
- **NetBeans:** También para Java, PHP y C + +.
- **Android Studio:** IDE oficial para desarrollo de apps Android.
- **Xcode:** IDE de Apple para desarrollo en iOS/macOS (Swift y Objective-C).

3. ¿Qué es y para qué sirve un sistema de control de versiones?
   Un sistema de control de versiones es una herramienta la cual se utiliza para gestionar cambios realizados en un archivo/s (en especial el código fuente)
   Este sistema sirve para lo siguiente:
   - Guardar el historial de cambios hechos
   - Recuperar versiones anteriores
   - Colaboración entre varios programadores
   - Fusionar cambios hechos por estos últimos
   - Creación de ramas de desarrollo

4. Antes de explicar que es Git y GitHub, hay que entender que estas aplicaciones son completamente diferentes y no hay que confundirlas, ya que a pesar de sonar muy parecidos, las funciones que ofrecen no son las mismas. Entonces:

   ¿En qué se diferencian?: Git es una herramienta de control de versiones de código abierto creada en 2005 por desarrolladores que trabajan en el sistema operativo Linux. GitHub es una empresa fundada en 2008 que crea herramientas que se integran con Git.
   No necesitas GitHub para usar Git, pero no puedes usar GitHub sin usar Git

   ¿Qué es Git?: es un sistema de control de versiones de código abierto, que facilita este tipo de colaboración de proyectos a través del control de versiones distribuido de los archivos que residen en los repositorios. Git permite integrar los flujos de trabajo realizados por varios colaboradores a lo largo del tiempo en relación con un repositorio determinado.

   ¿Qué es GitHub?: es un servicio de hospedaje basado en web de repositorios de Git. Para un proyecto concreto, GitHub hospeda el repositorio principal, del que los colaboradores pueden realizar copias para llevar a cabo su propio trabajo.

   Características de Git:
   • Git es un sistema distribuido, lo que significa que cada desarrollador tiene una copia completa del repositorio en su máquina local.
   • Puedes rastrear cambios en tu código a lo largo del tiempo, lo que te permite revertir cambios si es necesario.
   • Crear ramas para trabajar en diferentes versiones de tu código de manera simultánea.
   • Fusionar cambios de diferentes ramas en una sola rama.
   • Registrar cambios en tu código con un mensaje descriptivo.

   Características de GitHub:
   • Permite crear repositorios para almacenar y compartir tu código con otros.
   • Colaborar con otros desarrolladores en proyectos abiertos o privados.
   • Crear issues para rastrear errores o tareas pendientes en tu proyecto.
   • Crear pull requests para revisar y fusionar cambios en tu proyecto.
   • Tener una gran comunidad de desarrolladores que comparten y colaboran en proyectos de código abierto.

5. ¿Qué es depuración?
   La depuración (o debugging) es el proceso de detectar, analizar y corregir errores o fallos (bugs) en un sistema, programa o aplicación. La depuración se realiza cuando un sistema/programa no funciona correctamente, produce resultados inesperados o se bloquea.

El objetivo es identificar la causa del problema y modificar el código para solucionarlo.

6. ¿Qué son las herramientas de análisis?

Las herramientas de análisis son aplicaciones, software o metodologías diseñadas para examinar, interpretar y extraer información significativa de datos. Su propósito principal es facilitar la comprensión de patrones, tendencias, relaciones y anomalías dentro de conjuntos de datos, lo que a su vez ayuda en la toma de decisiones informadas y la resolución de problemas. En esencia, estas herramientas transforman datos brutos en insights accionables.

Se pueden clasificar en varios tipos según su función y el tipo de datos que manejan. Algunos ejemplos comunes incluyen:

- Herramientas de análisis de datos: Se centran en la limpieza, transformación, modelado y análisis de datos para descubrir información útil. **Ejemplos incluyen Excel, Python (con librerías como Pandas y NumPy), R, SQL, Tableau y Power BI.**
- Herramientas de análisis web: Permiten rastrear y analizar el comportamiento de los usuarios en sitios web, proporcionando métricas sobre el tráfico, las fuentes de visitantes, las páginas más visitadas, las tasas de conversión, etc. **Google Analytics y Yandex Metrica son ejemplos populares.**
- Herramientas de análisis de procesos de negocio: Ayudan a comprender, modelar, analizar y mejorar los procesos dentro de una organización. **Ejemplos incluyen ProcessMaker, Celonis y IBM Process Mining.**
- Herramientas de análisis estadístico: Diseñadas para realizar cálculos estadísticos complejos, pruebas de hipótesis y modelado predictivo. **R y SPSS son ejemplos.**
- Herramientas de visualización de datos: Facilitan la creación de gráficos, diagramas y otros elementos visuales para comunicar los hallazgos del análisis de manera efectiva. **Tableau, Power BI y librerías de Python como Matplotlib y Seaborn son ejemplos.**

7. ¿Qué es CI/CD (Integración y Despliegue Continuo)?

CI/CD es básicamente una práctica que se usa en desarrollo de software para trabajar de forma más ordenada y automática.

- **La parte de Integración Continua (CI)** trata de que, cada vez que un desarrollador hace cambios en el código, esos cambios se integran rápidamente al proyecto principal y se prueban de forma automática para detectar errores lo antes posible. Esto ayuda a que no se acumulen problemas y que todo se mantenga funcionando.

- **Después está la Entrega o Despliegue Continuo (CD),** que se enfoca en automatizar el paso final: llevar esos cambios al entorno donde se prueba o incluso donde ya lo usa la gente (producción). Si todo sale bien en los pasos anteriores, el código nuevo se publica sin que nadie tenga que hacerlo manualmente.

8. ¿Herramientas comunes de los CI/CD?

Herramientas comunes de CI/CD:

1. Integración Continua (CI)

Estas herramientas permiten compilar, probar y validar el código automáticamente cada vez que se realiza un cambio.

- **Jenkins**: Muy popular y extensible mediante plugins.
- **GitHub Actions**: Integrada en GitHub, permite definir flujos de trabajo automáticos.
- **GitLab CI/CD**: Integración directa con GitLab, muy usada en entornos DevOps.
- **CircleCI**: Plataforma rápida y flexible para CI basada en la nube.

- **Travis CI**: Fácil de usar, especialmente con proyectos en GitHub.

2. Entrega / Despliegue Continuo (CD)
Estas herramientas se enfocan en automatizar el despliegue del software en diferentes entornos (pruebas, producción, etc.).

- **Argo CD**: Muy usado para despliegue continuo con Kubernetes.
- **Spinnaker**: Desarrollado por Netflix, permite desplegar a múltiples entornos (AWS, Kubernetes, etc.).
- **Octopus Deploy**: Ideal para despliegues complejos en múltiples etapas.
- **Flux**: Automatiza despliegues en Kubernetes usando GitOps.

3. Automatización y gestión de pipelines

- **Azure DevOps**: Plataforma completa con CI/CD, gestión de código y tareas.
- **Bitbucket Pipelines**: Integrado con Bitbucket, ideal para equipos Atlassian.
- **Bamboo**: De Atlassian, permite CI/CD con integración a JIRA y Bitbucket.