***TRABAJO PRACTICO N\*3***

***HERRAMIENTAS DE DESARROLLO***

1. Las herramientas de desarrollo permiten agilizar el proceso de desarrollo de software.

Una aplicación o programa puede contar con múltiples herramientas que se encarguen de funciones especificas y afines para realizar una tarea.

Existen estos tipos de herramientas:

* IDE.
* Sistema de control de versiones.
* Bug tracker.
* Pruebas de desempeño.
* Documentación.
* Sistema de integración continuo.

1. IDE significa sistema de desarrollo integrado y es un sistema de software para el diseño de aplicaciones que combina herramientas comunes para desarrolladoras en una sola interfaz de usuario grafica

Ejemplos de IDE’s son:

* jetBrains web storm
* Microsoft visual studio code
* Sublime text
* Eclipse
* Net beans
* Aptana
* Dreamweaber
* Notepad++

1. Un entorno de programación suele estar integrado pos las siguientes herramientas:

* Editor de código fuente: es un procesador de textos orientado para escribir código fuente de aplicaciones en general en lenguajes de programación.
* Depurador o debugger: permite probar y eliminar los errores del programa.
* Compilador: traduce el lenguaje de programación al lenguaje máquina, código intermedio o texto.
* Interprete: realiza la traducción a medida que sea necesaria instrucción por instrucción y no guardan el resultado de la traducción.

1. Un sistema de control de versiones o VCS permiten gestionar los archivos de un proyecto (y sus versiones) y que sus integrantes puedan acceder remotamente a ellos.

Sus ventajas son que:

* Permite llevar cuenta de los cambios de un conjunto de archivos digitales en el tiempo.
* Cada versión registrada en el tiempo es una revisión
* Permite tanto a un desarrollador como a un grupo de desarrolladores, gestionar el código del proyecto.

1. Git es un sistema de control de versiones pensado en la eficiencia, la confiabilidad y compatibilidad de mantenimiento de versiones de aplicaciones cuando estas tienen un gran numero de archivos de código fuente

Los servidores de git son:

* GitHub
* Bitbucker
* GitLab

1. La documentación en código es importante porque te permite entender que se esta haciendo y porque y también mantener el código.
2. JSDoc es una sintaxis para agregar documentación de la api al código fuente de javascrips.
3. Estas son las líneas de comando GIT para:
4. Clonar un repositorio: GIT CLONE
5. Actualizar tu repositorio local al commit mas nuevo: GIT PULL
6. Registrar cambios (añadirlos al índex): GIT ADD
7. Hacer un commit: GIT COMMIT
8. Enviar cambios al repositorio remoto: GIT PUSH

Gráfico, Diagrama

Descripción generada automáticamente con confianza media

* 1: git add
* 2: git commit -m
* 3: git commit -a -m
* 4:
* 5:git push
* 6:
* 7:
* 8:git diff
* 9:
* 10:git pull

1. Para asegurar una buena calidad en los proyectos de dimensión considerable es importante utilizar un bug tracker para realizar el seguimiento de los defectos o bugs que surgen y son detectados.

Los bug tracker mas empleados son:

* Jira
* Bugzilla
* Flyspray
* Trac
* The Bug Genie
* MantisBT

1. Las pruebas de desempeño son test que se le hacen a un sistema para ver su desempeño en diferentes condiciones.

Por ejemplo:

* Loan Test: simular la realidad a la que esta expuesto un sistema cuando esta en producción.
* Stress test: simular mas carga de la esperada.
* Endurance o resistencia: desempeño del sistema después de una carga duradera por un periodo de tiempo largo.

1. Es una buena practica generar tests para verificae el correcto funcionamiento de las partes del software.

Estos test pueden ser manuales (los realiza una persona según un documento) o atomaticos (código que prueba otro código). En el caso de los test automatizados, se recurre muchas veces a servidores que corran los test, para poder correrlos en distintos ambientes. A estos servidores se les llama de integración continua.

Para esto se puede emplear las siguientes herramientas:

**PROPIO SERVIDOR**

* GitLab CL
* JenKins
* Drone.io

**CLOUD**

* Travis.ci
* Codeship
* CircleCI