Logotipo

Descripción generada automáticamente

*INSTITUTO   
TECNOLÓGICO DE****LAS AMÉRICAS***

**CENTRO DE EXCELENCIA EN DESARROLLO DE SOFTWARE**

Tecnólogo en Desarrollo de Software

**PROGRAMACIÓN III**

Kelyn Tejada Belliard

**ARLIN YESSABEL PAULINO ESPINOSA**

2015-3187

**TAREA 3 - MÓDULO V**

Herramientas de Administración de Fuentes

**PERÍODO 2024-C2**

29 de julio de 2024

**TAREA 3  
*Módulo V - Herramientas de Administración de Fuentes***

1. ***Desarrolla el siguiente cuestionario.***
2. ***¿Qué es Git?***

Es un sistema de control de versiones que permite a los desarrolladores facilitar la gestión y control de cambios en los códigos fuentes para sus proyectos, permitiendo trabajar sin conexión y en la propia máquina del desarrollador, y facilitando la recuperación ante posibles fallos durante el desarrollo del código.

También permite trabajar con repositorios remotos como GitHub, GitLab, Bitbucket, entre otros, para la colaboración con otros desarrolladores.

1. ***¿Cuál es el propósito del comando* git init *en Git?***

**Git init**sirve para inicializar Git y crear un nuevo repositorio local en el directorio desde el cual se está ejecutando, con el objetivo de crear un nuevo proyecto, y que incluirá todos los archivos necesarios para el proyecto.

De igual manera, sirve para añadir un control de versiones a un proyecto existente o preparar el directorio para la colaboración con otros colaboradores o miembros del equipo de trabajo.

1. ***¿Qué representa una rama en Git y cómo se utiliza?***

Se trata de una línea independiente de desarrollo que permite al desarrollador trabajar en diferentes partes del proyecto de forma aislada. Permiten efectuar cambios en el código, con la posibilidad de experimentar con nuevas características o corregir errores sin afectar la rama principal.

Para trabajar con ramas, primero hay que crear una con el comando **git branch** seguido del nombre de la nueva rama, y se puede cambiar de una rama a otra con los comandos **git checkout** y **git switch**.

También se pueden enlistar todas las ramas de un repositorio local usando **git branch** y **git branch -a** para ver las ramas remotas, y se pueden combinar ramas con **git merge.** Para eliminar ramas, se usa **git branch -d** seguido del nombre de la rama. Para subir ramas a un repositorio remoto se utiliza **git push origin** seguido del nombre de la rama, y para borrarla del repositorio remoto se utiliza **git push origin --delete** seguido del nombre de la rama.

1. ***¿Cómo puedo determinar en qué rama estoy actualmente en Git?***

Para determinar en qué rama se está trabajando actualmente, se pueden usar varios métodos. Uno de ellos, por ejemplo, es usar el comando **git branch** para listar todas las ramas, y la rama actual estará precedida por un asterisco (\*).

Otra forma de hacerlo es usando el comando **git status**, que proporciona información sobre el estado del repositorio, incluyendo la rama en la que se está trabajando.

Sin embargo, la forma más directa es usando el comando **git rev-parse --abbrev-ref HEAD**, que devuelve solo el nombre de la rama actual, sin información adicional.

1. ***¿Quién es la persona responsable de la creación de Git y cuándo fue desarrollado?***

La persona responsable de Git fue Linus Torvalds, y lo desarrolló en el año 2005.

1. ***¿Cuáles son algunos de los comandos esenciales de Git y para qué se utilizan?***

Algunos de los comandos esenciales de Git son los siguientes:

* **Git init:** Se utiliza para crear un nuevo repositorio de Git. Es el primer paso para empezar a utilizar Git en un proyecto.
* **Git add <file>:** Se utiliza para añadir cambios en archivos específicos al área de preparación. Permite seleccionar y revisar los cambios antes de consolidarlos en el historial del repositorio mediante un commit.
* **Git status:** Se utiliza para mostrar el estado del directorio de trabajo y del área de preparación. Es fundamental para entender qué cambios se han realizado, cuáles están preparados para el siguiente commit y cuáles no.
* **Git commit:** Se utiliza para guardar los cambios preparados en el historial del repositorio. Un commit en Git es similar a un punto de control o una instantánea del estado del proyecto en un momento dado.
* **Git push:**Se utiliza para enviar los commits locales a un repositorio remoto.
* **Git pull:**Se utiliza para actualizar el repositorio local con los cambios del repositorio remoto. Combina dos comandos en uno: **git fetch** (que descarga los cambios del repositorio remoto), y **git merge** (que combina esos cambios en la rama actual).
* **Git clone:** Se utiliza para crear una copia de un repositorio remoto en la máquina local. Es el primer paso en el flujo de trabajo de Git cuando se quiere empezar a trabajar en un proyecto existente que ya está alojado en un repositorio remoto.

1. ***¿Puedes mencionar algunos de los repositorios de Git más reconocidos y utilizados en la actualidad?***

Algunos de los repositorios más conocidos son los siguientes:

* ***GitHub Git:*** Es el repositorio del propio sistema de control de versiones Git.
* ***TensorFlow:*** Se trata de una popular biblioteca de código abierto para el aprendizaje automático desarrollada por Google.
* ***React:*** Es una biblioteca de JavaScript para construir interfaces de usuario, fue desarrollada por Facebook.
* ***Bootstrap:*** Es el framework de HTML, CSS y JS más popular para desarrollar proyectos responsivos y móviles primero en la web.
* ***Django:*** Es un framework de alto nivel para el desarrollo web en Python que fomenta el desarrollo rápido y el diseño limpio y pragmático.
* ***Node.Js:*** Se trata de un entorno de ejecución de JavaScript en el lado del servidor que permite ejecutar JavaScript en el servidor.
* ***Linux Kernel:*** Es el código fuente del núcleo del sistema operativo Linux, y es precisamente el proyecto para el cual Linus Torvalds creó Git originalmente.

1. ***Desarrolle un ejercicio práctico en Azure Devops o GitHub con las siguientes características***

* **Link con respuestas:**
* Utiliza OneDrive para la entrega de la teoría o sube el documento al repositorio del proyecto.
* Comparte el documento y asegúrate de configurarlo como público o proporciona acceso al profesor y al monitor. (Si das los accesos no se corregirá)
* **Link del proyecto en GitHub o Azure Devops:**

**Tareas o requerimientos (Seguir los pasos indicados):**

Crea issues para cada tarea.

En el ejemplo explicado verá que la estructura que debe tener el proyecto será la siguientes:

* Branch- **main** -> Es la rama principal o rama productiva.
* **QA** - > Esta es la rama utilizada para el entorno de prueba o entorno de QA donde los equipo de calidad son los responsable de hacer toda la prueba necesaria del codigo en este ambiente.

**Nota:**

En algunos lugares la rama **Main/Master** es **QA**, ya que se utilizan tags para crear la versión productiva del proyecto cuando se terminan las pruebas.

**Dev** ->Rama donde los programadores suben su código para hacer prueba.

**Funcionalidades (Feature branches)**

1. Ramas para desarrollar nuevas características.
2. Fusionadas en la rama principal después de completar.

feature/nueva-funcionalidad

**Correcciones Rápidas (Hotfix Branches):**

1. Ramas para corregir errores críticos.
2. Fusionadas en la rama principal y en las ramas de funcionalidades activas.

hotfix/error-crítico

Pr = Pull request (Este es el nombre que recibe en GITHUB)

Mr = Merge Request (Este es el nombre que recibe en azure devops)

**Crear MR o PR:** Una vez termine de realizar el feature debe de subir el cambio al Dev y luego a QA y luego a Main, se debe de crear un pull request en Github para simular el proceso real que sucede cuando hacemos un feature.

**Importante:**

* Asegúrate de que el repositorio no esté vacío.
* Asegúrate de crear los pull request para ver la evidencia.
* Solo se permite la entrega del código en github o Azure devops.
* Verifica que el repositorio sea público o tenga acceso al profesor y monitor.
* Recuerda configurar el documento como público en Google Docs o dar acceso en Word.