Programacion 3

Tarea GIT

Wesly estevez
Matricula:2022-1065
Kellyn tejada beliar

Calificación: la que indica la plataforma.

Punto1:

Desarrolla el siguiente Cuestionario

1. **¿Qué es Git?**

Git es un sistema de control de versiones distribuido que permite a múltiples desarrolladores trabajar en el mismo proyecto de software de manera simultánea sin sobrescribir los cambios de los demás. Fue creado por Linus Torvalds en 2005 para el desarrollo del kernel de Linux. Algunas características clave de Git incluyen:

Distribuido: Cada desarrollador tiene una copia completa del repositorio, incluyendo toda la historia del proyecto, lo que permite trabajar sin conexión y realizar cambios de manera local antes de compartirlos con otros.

Ramas (Branches): Git facilita el uso de ramas, que son versiones independientes del proyecto que se pueden trabajar en paralelo. Esto es útil para desarrollar nuevas características, corregir errores o probar nuevas ideas sin afectar la versión principal del código.

Commits: Los cambios en el código se guardan como "commits", que son puntos de referencia en la historia del proyecto. Cada commit incluye un mensaje que describe los cambios realizados, facilitando el seguimiento y la revisión de la evolución del proyecto.

Fusión (Merge): Git permite combinar cambios de diferentes ramas mediante un proceso de fusión, asegurando que los trabajos paralelos se integren de manera coherente.

Historia: Git guarda un registro completo de todos los cambios realizados en el proyecto, lo que permite revertir a versiones anteriores si es necesario y entender cómo y por qué se realizaron ciertos cambios.

Colaboración: Git facilita la colaboración en proyectos de software, ya que los desarrolladores pueden trabajar en sus propias copias del repositorio y luego compartir sus cambios con el resto del equipo mediante pull requests y revisiones de código.

2- Cuál es el propósito del comando git init en Git?

El comando git init en Git se utiliza para crear un nuevo repositorio de Git en un directorio. Este comando inicializa un nuevo repositorio vacío o re-inicializa un repositorio existente. Su propósito principal es configurar el directorio actual como un repositorio de Git, permitiendo el seguimiento de versiones y la gestión de código fuente. A continuación, se detallan los pasos y efectos del uso de git init:

Crear un Repositorio Vacío: Si se ejecuta en un directorio que no contiene un repositorio de Git, git init creará uno nuevo. Esto incluye la creación de un subdirectorio oculto llamado. git que contiene toda la información necesaria para el repositorio (como objetos, referencias, configuraciones, etc.).

Re-inicializar un Repositorio Existente: Si se ejecuta en un directorio que ya contiene un repositorio de Git, git init puede re-inicializarlo. Esto no perderá la historia ni los archivos del repositorio, sino que puede actualizar la configuración o corregir problemas con la estructura del repositorio.

Configurar el Control de Versiones: Una vez inicializado el repositorio, Git comenzará a rastrear los archivos y los cambios en ese directorio. Los comandos como git add, git commit, git status, y otros estarán disponibles para gestionar los archivos y la historia del proyecto.

3-¿Qué representa una rama en Git y cómo se utiliza?

En Git, una rama (o branch) representa una línea de desarrollo independiente dentro de un repositorio. Es una versión del proyecto en la que puedes trabajar de manera aislada del resto del código base. Las ramas permiten a los desarrolladores trabajar en diferentes características, correcciones de errores o experimentos sin afectar la versión principal del código (generalmente llamada main o master). Una vez que los cambios en una rama están listos y probados, se pueden fusionar de nuevo en la rama principal.

4-¿Cómo puedo determinar en qué rama estoy actualmente en Git?

Para determinar en qué rama estás actualmente en Git, puedes usar los siguientes métodos:

Comando git branch:

git branch
Este comando muestra una lista de todas las ramas en el repositorio y marca la rama actual con un asterisco *.

Comando git status:

git status
Este comando proporciona el estado del repositorio e incluye la información de la rama actual al inicio de la salida.

5-¿Quién es la persona responsable de la creación de Git y cuándo fue desarrollado?

La persona responsable de la creación de Git es Linus Torvalds, el creador del kernel de Linux. Git fue desarrollado en abril de 2005. Torvalds creó Git para gestionar el desarrollo del kernel de Linux, buscando una herramienta de control de versiones distribuida que fuera rápida, eficiente y capaz de manejar grandes proyectos con facilidad.

6-¿Cuáles son algunos de los comandos esenciales de Git y para qué se utilizan?

Claro, aquí tienes una versión más corta y directa de los comandos esenciales de Git:

git init: Inicializa un nuevo repositorio.
git clone <url>: Clona un repositorio.
git add <archivo>: Añade archivos al área de preparación.
git commit -m "mensaje": Crea un commit con un mensaje.
git status: Muestra el estado del repositorio.
git log: Muestra el historial de commits.
git branch: Lista las ramas y muestra la rama actual.
git checkout <rama> / git switch <rama>: Cambia de rama.
git merge <rama>: Fusiona una rama en la actual.
git pull: Descarga y fusiona cambios de un remoto.
git push: Sube cambios a un remoto.
git remote: Gestiona repositorios remotos.
git diff: Muestra diferencias entre commits.
git reset <archivo> / git reset --hard <commit>: Deshace cambios.
git stash: Guarda y restaura cambios temporales.

7-¿Puedes mencionar algunos de los repositorios de Git más reconocidos y utilizados enla actualidad?

Sí, algunos de los repositorios de Git más reconocidos y utilizados en la actualidad incluyen:

Linux Kernel: El repositorio del kernel de Linux, desarrollado por Linus Torvalds y otros colaboradores.

URL: https://github.com/torvalds/linux
TensorFlow: Un marco de trabajo de aprendizaje automático de código abierto desarrollado por Google.

URL: https://github.com/tensorflow/tensorflow
React: Una biblioteca de JavaScript para construir interfaces de usuario, mantenida por Facebook.

URL: https://github.com/facebook/react
Vue.js: Un marco de trabajo progresivo para construir interfaces de usuario.

URL: https://github.com/vuejs/vue
Kubernetes: Un sistema de orquestación de contenedores de código abierto, inicialmente desarrollado por Google.

URL: https://github.com/kubernetes/kubernetes
Django: Un marco de trabajo de alto nivel para el desarrollo de aplicaciones web en Python.

URL: https://github.com/django/django
Node.js: Un entorno de ejecución para JavaScript construido con el motor V8 de Chrome.

URL: https://github.com/nodejs/node
Bootstrap: Un popular marco de trabajo de front-end para el desarrollo de sitios web y aplicaciones web.

URL: https://github.com/twbs/bootstrap
Angular: Un marco de trabajo de aplicaciones web de código abierto desarrollado por Google.

URL: https://github.com/angular/angular
Electron: Un marco de trabajo para construir aplicaciones de escritorio con tecnologías web.

URL: https://github.com/electron/electron

Punto 2:
Desarrolle un ejercicio práctico en Azure Devops o GitHub con las siguientes características

Link con respuestas:

- Utiliza OneDrive paa 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.

Email de los calificadores:

Azure Devops**/**Github/JIRA**: 20186927@itla.edu.do**

Azure Devops**/**Github/JIRA**: ktejada@itla.edu.do**

Link de ejemplo hecho por el monitor:

Plataforma Youtube:{Aqui colocare un video de explicacion}

Recordatorios

- 1- Acceso a los link al profesor y monitor con los link indicado.
- 2- No se puede enviar tarea en ningún momento por email solo por la plataforma.
- 3- Cualquier dudas se hará únicamente por correo al profesor y monitor.
- 4- Respetar las fechas acordadas.
- 5- La nota será 0 si no cumplen con el requerimiento del paso 1, 2, 3 y 4.

Criterio de evaluación

Permiso al repositorio	10%
Cumplir con la estructura indicada	60%
Crear los PR o MR	30%
En caso de no dar el permiso completo	-100%
Plagio	-100%