Guía de Actividad 3 – GitHub

ANALISIS Y DESARROLLO DE SOFTWARE

CENTRO DE TECNOLOGIA DE LA MANUFACTURA AVANZADA

MEDELLÍN

PROCESO DIRECCIÓN DE FORMACIÓN PROFESIONAL INTEGRAL

FORMATO GUÍA DE APRENDIZAJE

INTRODUCCIÓN

El objetivo de esta guía es proporcionar al aprendiz los conocimientos necesarios para que pueda utilizar Git y GitHub con confianza, desde el control de versiones hasta la colaboración en equipo. A través de ejemplos prácticos y ejercicios, aprenderá a manejar las herramientas más esenciales de Git y GitHub, lo que le permitirá mejorar su flujo de trabajo y ser más eficiente en sus proyectos.

Palabras claves (key words)

Git, GitHubControl de versiones, Repositorio, Commit, Branch (Rama), Merge (Fusionar), Pull request, Fork (Bifurcación), GitHub Actions, Push, Clone (Clonar).

Preguntas de Introducción:

- ¿Qué es un repositorio en Git y cuál es su función dentro de un proyecto de desarrollo?
- ¿Qué problema resuelve el uso de sistemas de control de versiones como Git en proyectos de desarrollo de software?
- ¿Cómo permite Git la colaboración entre varios desarrolladores sin que sus cambios interfieran entre sí?
- ¿Cuál es la diferencia entre un commit y un push en el flujo de trabajo de Git?
- ¿Qué es un pull request en GitHub y cómo ayuda a revisar y fusionar los cambios de los colaboradores?
- ¿Qué ventajas ofrece el uso de ramas (branches) en Git cuando se está desarrollando una nueva funcionalidad o corrigiendo un error?
- ¿En qué situaciones se recomienda usar un fork en GitHub?

PLANTEAMIENTO DE LAS ACTIVIDADES

En el desarrollo de software moderno, la colaboración y el control de versiones son esenciales para gestionar proyectos de manera eficiente. Git y GitHub son herramientas clave en este contexto, utilizadas por millones de desarrolladores para crear, compartir y mantener código en proyectos de cualquier escala.

Git es un sistema de control de versiones distribuido que permite a los desarrolladores gestionar y realizar un seguimiento de los cambios realizados en su código. A diferencia de los sistemas tradicionales, Git permite trabajar de manera descentralizada, lo que significa que cada desarrollador puede tener una copia completa del repositorio en su máquina local. Esto no solo facilita la colaboración, sino que también proporciona una forma segura de volver a versiones anteriores del código, revisar el historial de cambios y trabajar de manera simultánea en diferentes características del proyecto sin interferir con el trabajo de otros. El aprendiz apropiará conceptos fundamentales de Git, como commit, branching, merge, pull y push, así como las mejores prácticas para gestionar su código y colaborar de forma eficiente en equipo.

GitHub es una plataforma web que utiliza Git para almacenar, gestionar y compartir repositorios de código. Además de ser una herramienta fundamental para el control de versiones, GitHub ofrece una serie de funcionalidades que permiten la colaboración en proyectos de manera más fluida. Entre estas funcionalidades se encuentran pull requests, issues, actions y wikis, que permiten a los equipos de desarrollo coordinarse de forma más efectiva y mantener un flujo de trabajo organizado. No solo es una herramienta valiosa para el desarrollo de software, sino que también se ha convertido en un espacio de colaboración para proyectos de código abierto, donde los desarrolladores pueden contribuir a proyectos globales y compartir su código con la comunidad.

Ilustración 1¿Qué es Git? ¿Qué es GitHub? fuente: https://aprendeia.com/que-es-git-y-github/

Usted está trabajando en el desarrollo de una plataforma web para la empresa “Coquito Amarillo S.A.S.”, cuyo objetivo es evaluar el nivel de satisfacción personal de sus empleados a través de la “Rueda de la vida” (un enfoque utilizado en psicología para medir el bienestar en diferentes áreas). La plataforma permite a los empleados completar un cuestionario interactivo, visualizar los resultados y generar un informe con recomendaciones personalizadas.

Durante el proceso de desarrollo, el equipo se enfrenta a varios desafíos relacionados con el control de versiones y la colaboración efectiva. El proyecto está siendo desarrollado por un equipo de varios miembros, y la coordinación de los cambios, la integración de nuevas funcionalidades y el despliegue del sitio web son aspectos críticos.

- ¿Cómo podría estructurar el flujo de trabajo del equipo utilizando Git y GitHub para asegurar que todos los miembros colaboren de manera eficiente y no se presenten conflictos de código?

- Imagine que un miembro del equipo ha realizado cambios importantes en una funcionalidad de la evaluación de la rueda de la vida, pero al hacer un merge de la rama en la que estaba trabajando, se presentan conflictos con la rama principal del proyecto. ¿Cómo resolvería este conflicto utilizando Git y GitHub?

- Como parte de su rol en el proyecto, decide automatizar las pruebas unitarias de las funcionalidades de la web utilizando GitHub Actions. ¿Qué pasos seguiría para configurar un flujo de trabajo en GitHub Actions que ejecute las pruebas automáticamente con cada push a la rama principal y despliegue el sitio web de manera continua?

Las actividades siguientes permiten al aprendiz implementar la gestión de proyecto y versiones de código mediante el uso de GitHub.

Actividad 1: Configuración Inicial

Para aprender debemos hacer, empecemos por realizar la configuración del repositorio y la estructura del proyecto en GitHub; para ello vamos a crear nuestro perfil de desarrollador y un repositorio en GitHub y configuraremos su estructura básica:

1. Crear su perfil de usuario en la plataforma GitHub (https://www.github.com).

2. Crear en su perfil una organización (Ej: Coquito Amarillo S.A.S)
3. Crear en su organización un proyecto (Ej: Web Corporativa)
4. Crear en su proyecto un repositorio con un nombre apropiado, como “coquito-amarillo-web”.
5. Clonar el repositorio en su máquina local utilizando `***git clone <URL del repositorio>***`.
6. Crear la estructura inicial de directorios dentro del repositorio (por ejemplo, una carpeta `documentación` para la gestión administrativa del proyecto, `*src*` para los archivos del código y una carpeta `*assets*` para imágenes, estilos y otros recursos).
7. Realizar el primer *commit* para guardar la estructura general y los archivos base del proyecto (*Readme*, Licenciamiento, Requerimientos, *product backlog*, HTML, CSS, imágenes).
8. Subir los cambios al repositorio remoto con el comando `**git push**`.

Herramientas de Git a utilizar:

- ✓ `**git init**`
- ✓ `**git clone**`
- ✓ `**git add**`
- ✓ `**git commit**`
- ✓ `**git push**`

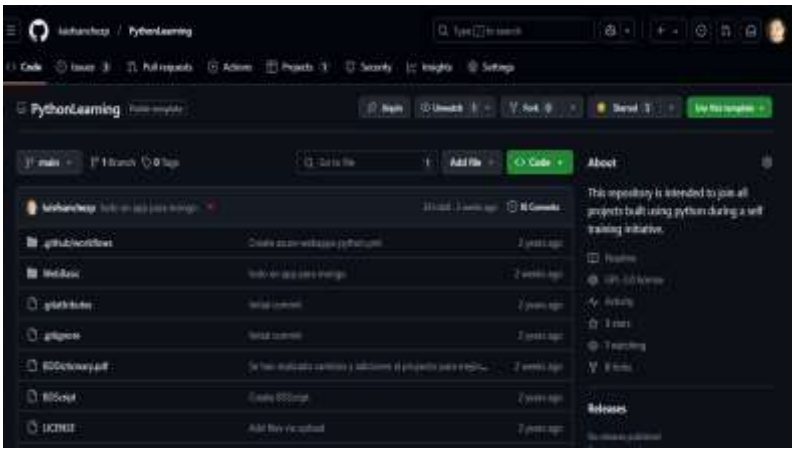

Ilustración 2Repositorio de GitHub - Construcción propia

Evidencias:

- ✓ Sustentación de su proyecto, presentando su perfil de GitHub y la estructura general de la organización, proyecto y repositorio.

Actividad 2: Ramas

Una de las principales características de GitHub es la posibilidad de usar ramas para mantener aislados los aspectos del proyecto y asegurar una mayor agilidad en su desarrollo. Utilizar ramas para desarrollar funcionalidades específicas sin afectar la versión principal del proyecto permitirá tener mayor control de los flujos de trabajo y hacer un seguimiento más adecuado a los colaboradores y sus asignaciones.

Ejecute en su proyecto las siguientes actividades:

1. Crear una nueva rama para una funcionalidad específica, como el diseño de la página principal o la implementación de la evaluación de la rueda de la vida.
    a. Ejemplo: `***git checkout -b diseño-página-inicial***`
2. Trabajar en los archivos correspondientes (HTML, CSS, python) dentro de esa rama.
3. Realizar *commits* frecuentemente para registrar los avances realizados en la rama.
4. Una vez completada la funcionalidad, realizar un *merge* de la rama con la rama principal (`*main*`) mediante un *pull request* en GitHub.
5. Resolver cualquier conflicto que pueda surgir al intentar fusionar las ramas.

Herramientas de Git a utilizar:

- ✓ `***git branch***`
- ✓ `***git checkout***`
- ✓ `***git merge***`
- ✓ ***Pull Request en GitHub***

Evidencias:

- ✓ Sustentación de su proyecto, presentando su proyecto en GitHub y la estructura general del repositorio.

Actividad 3: Pull Request e Issues

Nadie trabaja solo, por eso es esencial en el desarrollo de software la capacidad de colaborar, pero la colaboración tiene sus costos. Fomentar la colaboración utilizando ***pull requests*** e ***issues*** para gestionar tareas y problemas facilita la construcción de proyectos de mayor tamaño y complejidad.

Realice las siguientes actividades para comprender mejor estos aspectos:

1. Crear un ***issue*** en GitHub para cualquier tarea específica, como "Implementar la evaluación de la rueda de la vida para empleados".
2. Asignar el ***issue*** a un miembro del equipo o a usted mismo.
3. Desarrollar la funcionalidad necesaria en una nueva rama (por ejemplo, `evaluación-rueda-vida`).
4. Una vez completado, abrir un ***pull request*** para fusionar los cambios en la rama principal y mencionar el ***issue*** relacionado en el ***Pull Request***.
5. Revisar y discutir el ***pull request*** en el equipo, realizar pruebas y hacer ajustes si es necesario.

6. Una vez aprobado, fusionar el ***pull request*** y cerrar el ***issue*** .

Herramientas de Git y GitHub a utilizar:

- ✓ `***git branch***`
- ✓ ***Pull Request en GitHub***
- ✓ ***Issues en GitHub***

Evidencias:

- ✓ Sustentación de su proyecto, presentando su repositorio en GitHub y el historial de actividades relacionado.

Actividad 4: Actions

Automatización de Flujos de Trabajo con GitHub Actions permite por ejemplo automatizar tareas como la ejecución de pruebas unitarias o el despliegue de la web al entorno de producción.

A continuación se propone un reto para que el aprendiz intente desarrollar este tipo de actividad en el proyecto:

1. Configurar un archivo de flujo de trabajo en ***GitHub Actions*** para automatizar el proceso de CI/CD (Integración continua y despliegue continuo).
2. Crear un archivo `***.yml***` en el directorio `***.github/workflows***` para definir el flujo de trabajo, por ejemplo, para ejecutar pruebas unitarias sobre el código cada vez que se haga un `***push***` a la rama principal.
3. Implementar pasos en el flujo de trabajo que automáticamente desplieguen la web en un entorno de producción o *staging* (como *GitHub Pages, Netlify o Heroku*) cuando se realice un ***push*** .
4. Configurar el flujo de trabajo para ejecutar **pruebas de integración** y revisar el código en cada ***pull request***, para asegurarse de que no haya errores.
5. Verificar el funcionamiento de ***GitHub Actions*** observando los resultados en la pestaña de ***Actions*** en GitHub.

Herramientas de GitHub a utilizar:

- ✓ GitHub Actions
- ✓ Archivos de configuración `.yml`

Evidencias:

- ✓ Sustentación de su proyecto, presentando su repositorio en GitHub y el historial de actividades relacionado.

GLOSARIO

- Git: Sistema de control de versiones distribuido utilizado para gestionar y realizar un seguimiento de los cambios en el código fuente durante el desarrollo de software. Permite a los desarrolladores trabajar de forma independiente en diferentes partes de un proyecto.
- Repositorio (Repository): Es un espacio donde se almacenan todos los archivos y el historial de cambios de un proyecto. Puede ser local (en tu máquina) o remoto (en un servidor como GitHub).
- Commit: Acción de guardar los cambios en el repositorio. Un "commit" registra un conjunto de modificaciones, y cada uno tiene un mensaje que describe lo que se ha cambiado. Cada commit tiene un identificador único (hash).
- Branch (Rama): Una rama es una versión paralela del proyecto. Permite trabajar en nuevas funcionalidades sin alterar el código principal. La rama principal por lo general se llama `main` o `master`.
- Merge (Fusionar): Acción de combinar los cambios de una rama con otra. Generalmente, se fusionan cambios de una rama secundaria (por ejemplo, una rama de características) a la rama principal.
- Clone (Clonar): Acción de crear una copia local de un repositorio remoto. Se utiliza para obtener una versión del proyecto desde un repositorio en línea.
- Pull: Es el comando para obtener los cambios más recientes de un repositorio remoto y fusionarlos con el repositorio local.
- Push: Es el comando utilizado para enviar los cambios realizados en el repositorio local al repositorio remoto.
- Staging Area (Área de preparación): Es el área donde se colocan los cambios que serán parte del siguiente commit. Permite a los desarrolladores seleccionar qué cambios se van a guardar.
- Diff (Diferencias): Muestra las diferencias entre dos versiones de un archivo o entre dos commits. Esto ayuda a revisar los cambios antes de hacer un commit.
- HEAD: Es una referencia a la rama o commit actual. HEAD apunta al último commit realizado en la rama en la que se está trabajando.
- Fork (Bifurcación): Crear una copia de un repositorio para realizar cambios. Es común en proyectos de código abierto, donde los desarrolladores crean un fork del repositorio para trabajar de manera independiente.
- Rebase: Permite incorporar cambios de una rama en otra de manera más lineal, reescribiendo el historial de commits.
- Tag (Etiqueta): Es una referencia fija a un punto específico en la historia de un repositorio, generalmente usada para marcar versiones o lanzamientos importantes.

- Conflict (Conflicto): Ocurre cuando dos ramas modifican el mismo fragmento de un archivo de manera diferente. Git no puede fusionar estos cambios automáticamente y requiere intervención manual.
- GitHub: Plataforma web que utiliza Git para ofrecer almacenamiento de repositorios de código, gestión de versiones, y colaboración en proyectos. GitHub permite a los desarrolladores almacenar, gestionar y compartir su código de manera eficiente.
- Pull Request (PR): Es una solicitud para fusionar los cambios realizados en una rama de un repositorio a otra rama (generalmente a la rama principal). Es una herramienta clave para la colaboración en proyectos de desarrollo.
- Issue: Herramienta de GitHub para gestionar y rastrear errores, mejoras y tareas relacionadas con un proyecto. Los desarrolladores pueden usar Issues para discutir cambios y mejoras.
- Actions: Herramienta de GitHub que permite automatizar flujos de trabajo, como compilaciones, pruebas y despliegues, directamente desde el repositorio de GitHub.
- Wiki: Espacio en GitHub donde los proyectos pueden almacenar documentación, guías, manuales y otros recursos relacionados con el proyecto.
- GitHub Pages: Funcionalidad de GitHub que permite alojar sitios web estáticos directamente desde un repositorio de GitHub. Es útil para proyectos de documentación o sitios web personales.
- Gist: Servicio de GitHub que permite compartir fragmentos de código o notas de manera sencilla. Un gist es un archivo individual o un pequeño conjunto de archivos.
- Collaborators (Colaboradores): Personas que tienen acceso a un repositorio privado para realizar cambios, enviar pull requests o simplemente leer el código.
- Stars: Son "me gusta" o favoritos que los usuarios de GitHub pueden dar a un repositorio para indicar su interés o apoyo al proyecto.
- Contributors (Contribuyentes): Son los usuarios que han realizado contribuciones a un repositorio, ya sea a través de commits, pull requests, o reportes de problemas.
- Actions Workflow (Flujo de trabajo de GitHub Actions): Definición de los procesos automáticos (como CI/CD) que GitHub ejecuta en respuesta a ciertos eventos, como cuando se hace un commit o un pull request.
- Commit History (Historial de commits): Registro de todos los commits realizados en un repositorio. Permite ver qué cambios se hicieron, cuándo y por quién.
- Webhooks: Son mecanismos que permiten a GitHub notificar a un sistema externo sobre eventos específicos de un repositorio, como un push o pull request.

ACTIVIDADES DE EVALUACIÓN

La evaluación de los aprendizajes será realizada en base a los siguientes criterios:

- ✓ Calidad en el uso de GitHub: ¿Se exploraron nuevos elementos adicionales a los mostrados en el documento? ¿Está bien estructurado el esquema organización-proyecto-repositorio?
- ✓ ¿Se identificaron y apropiaron herramientas adicionales que faciliten la creación del código?
- ✓ Calidad de la documentación: ¿La documentación de los resultados es clara, precisa y contiene todos los detalles necesarios?

Evidencias de Aprendizaje	Criterios de Evaluación	Técnicas e Instrumentos de Evaluación
Evidencias de Conocimiento: Prueba de conocimiento sobre principios y conceptos de Git y GitHub (Cuestionario). **Evidencias de Desempeño:** Verificación de los conceptos aplicados al diseño de la organización, proyecto, repositorio. (Listas de Chequeo, sustentación). **Evidencias de Producto:**	✓ Diseña la estructura del repositorio de acuerdo con los requerimientos especificados. ✓ Define el proyecto y sus características de acuerdo con las condiciones del entorno de producción. ✓ Realiza los entrenamientos propuestos por el instructor. ✓ Documenta las actividades para mantener la trazabilidad del proyecto en GitHub.	Observación directa, entrevistas, pruebas escritas, proyecto practica de aula, informe final.

Entrega del repositorio en línea (Sustentación, Listas de Chequeo).		

EVIDENCIAS A PRESENTAR

Evidencias por desempeño

- ✓ *Desarrollo completo de las actividades planteadas en el documento.*

Evidencias de conocimiento

- ✓ *Informe en formato PDF con las respuestas a cada una de las preguntas y retos planteados en el documento.*

Evidencias de producto

- ✓ *Perfil de GitHub, con su respectiva organización, proyecto y repositorio creados.*
- ✓ *Informe en formato PDF con las evidencias gráficas de los procesos ejecutados para cumplir con las actividades y sus correspondientes observaciones, descripciones y conclusiones.*

REFERENTES BILBIOGRÁFICOS

- ✓ Chacon, S., & Straub, B. (2014). Pro Git (2ª ed.). Apress.
- ✓ GitHub, Inc. (2020). Git Pocket Guide. O'Reilly Media.
- ✓ Bell, J. (2021). Learning GitHub Actions. Packt Publishing.
- ✓ Loeliger, J., & McCullough, M. (2021). Version Control with Git (3ª ed.). O'Reilly Media.

Cibergrafía

- ✓ itHub. (s.f.). GitHub Docs: Introducción a GitHub. Recuperado el 6 de marzo de 2025, de https://docs.github.com/es/github
- ✓ Microsoft. (s.f.). Introducción a Git y GitHub en Visual Studio Code. Recuperado el 6 de marzo de 2025, de https://learn.microsoft.com/es-es/github

CONTROL DEL DOCUMENTO

	Nombre	Cargo	Dependencia	Fecha
Autor (es)	Luis Fernando Sánchez Pérez.	Instructor	CTMA/TICS-Electrónica.	20/4/2025

CONTROL DE CAMBIOS (diligenciar únicamente si realiza ajustes a la guía)

	Nombre	Cargo	Dependencia	Fecha	Razón del Cambio
Autor (es)	Luis Fernando Sánchez Pérez	Instructor	CTMA/TICS-Electrónica.	9/5/2025	Modificación de Actividades de Apropiación del conocimiento