TEORIA-CONCEPTOS EN GIT

GitHub avanzado.

Fork, Clone y Pull Requests.

- **Diferencia entre un fork y un clone en Git:** Un fork copia el repositorio en tu cuenta de GitHub, mientras que el clone copia el repositorio localmente

- **Es el comando para vincular un repositorio clonado con el repositorio original (upstream):** git remote add upstream <URL>

- Pull request permite discutir y revisar cambios antes de integrarlos al proyecto principal

- El propósito principal de hacer un fork antes de colaborar en un proyecto de GitHub es la de crear una copia del repositorio que no afecta el proyecto original

Actualización del Fork y flujos de trabajo

- Este comando se usa para obtener los últimos cambios del repositorio original (upstream) sin fusionarlos inmediatamente en tu fork: git fetch upstream

- El propósito de git merge al actualizar un fork es la de combinar los cambios obtenidos mediante git fetch en tu rama actual

- Con Feature Branch, en los flujos de trabajo en GitHub se basa en mantener una rama principal estable y desarrollar en ramas separadas para cada característica.

- Gitflow es la estrategia más adecuada para un equipo pequeño que trabaja en varios cambios simultáneamente, manteniendo control sobre versiones y lanzamientos.

Feature Branch y Revisión del trabajo

- El propósito de usar "feature branch" en Git es la de poder implementar y probar una nueva característica de forma aislada del desarrollo principal

- Este comando se usa para crear y cambiar a una nueva rama en Git: git checkout -b <nombre_rama>

- El pull request permite que otros revisen y comenten el código antes de fusionarlo

- El dar comentarios constructivos que mejoren el código, es importante al revisar el código de un compañero.

Limpieza y recuperación d4-. Limpieza y recuperación de ramas.

- Al mejorar la organización y evitar confusión en el desarrollo es una razón importante para limpiar las ramas de un repositorio.

- git branch -d <nombre_rama> se utiliza para eliminar una rama local en Git:

- git reflog, nos ayuda a recuperar una rama eliminada si todavía existe en el historial de Git.

- Si utilizamos git branch -D <nombre_rama> en lugar de git branch -d <nombre_rama> provocara que la rama se va a eliminar forzadamente, sin importar su estado

GitHub Issues, MileStones y +Colaboradores

- **Características principales de las Github Issues:** Título, una descripción, etiquetas, asignaciones y comentario

- El objetivo de una Issue ayuda a definir un título, por lo tanto su objetivo es la de explicar el problema o la tarea a realizar

- bug, enhancement, documentation, son ejemplos de etiquetas en las Issues

- **El objetivo de los Hitos (Milestone):** Agrupan issues en torno a objetivos específicos del proyecto, permitiendo un seguimiento del progreso.

- Referenciar otros issues o pull requests, creando un historial vinculado que ayuda a entender la relación entre diferentes tareas, es el objetivo de las referencias.

- **Las búsquedas avanzadas permiten:** Buscar issues según palabras clave, estados o asignatarios, facilitando la gestión de grandes cantidades de información

- **Son las mejores prácticas que se recomiendan para crear un Issue:**
  - Antes de crear un issue, verifica que no exista uno similar para evitar duplicados.
  - Usa markdown para estructurar tu descripción, incluyendo listas y bloques de código si es necesario.
  - Mantén una comunicación clara y concisa en los comentarios para facilitar el seguimiento

- **Los pasos para crear una Issue son:**
  - Navega al repositorio
  - Ir a la pestaña Issue
  - Crea un Nuevo Issue
  - Completa el Formulario
  - Envía el Issue

Wikis, Proyectos y GitHub Pages

¿Qué es una wiki en GitHub?

- Un espacio colaborativo para documentar proyectos

¿Cómo se accede a la wiki de un repositorio en GitHub?

- Desde la pestaña "Wiki" en el repositorio

¿Qué formato se utiliza comúnmente para dar formato al contenido en las wikis de GitHub?

- Markdown

¿Qué tipo de contenido puedes incluir en una página de wiki en GitHub? (Selecciona todas las que apliquen)

- Imágenes, Enlaces a otras páginas o repositorios

¿Es posible editar una wiki de GitHub de manera colaborativa?

- Sí, pero se necesita acceso específico para hacerlo

TEORIA-CONCEPTOS EN GIT

¿Cómo puedes organizar múltiples páginas dentro de una wiki de GitHub?

- A través de enlaces dentro del contenido de cada página

¿Qué sucede cuando se edita una página en la wiki de GitHub?

- Se crea una nueva versión y se guarda el historial de cambios

¿Cuál es la principal ventaja de tener una wiki dentro de un repositorio de GitHub?

- Permite gestionar toda la documentación de manera centralizada y versionada

¿Qué puedes hacer si deseas colaborar en una wiki pero no tienes permisos de escritura?

- Proponer cambios a través de un pull request

¿Qué tipo de proyectos es ideal documentar mediante una wiki en GitHub?

- Proyectos colaborativos donde se necesite una documentación actualizada y accesible por todos los colaboradores

Organizaciones y Teams (equipos) en GitHub

- ¿Qué es una organización en GitHub y cuál es su propósito principal?
  - **Un espacio donde los usuarios pueden gestionar y colaborar en repositorios.**

- ¿Cuáles son las principales diferencias entre un repositorio personal y uno dentro de una organización en GitHub?
  - **Los repositorios dentro de una organización permiten asignar roles y permisos a varios miembros.**

- ¿Qué tipos de permisos pueden tener los miembros dentro de una organización en GitHub? Explica al menos tres roles comunes.
  - **Propietario, administrador y miembro.**

- ¿Cómo se crea una organización en GitHub? Explica los pasos principales.
  - **Acceder a la opción "Crear organización" desde la página principal de GitHub y completar los datos necesarios.**

- ¿Qué son los "equipos" dentro de una organización en GitHub y para qué se utilizan?
  - **Grupos de usuarios con permisos específicos para un repositorio determinado.**

- ¿Qué ventajas ofrece crear una organización en lugar de trabajar con repositorios personales cuando se trabaja en equipo?
  - **Permite una colaboración más sencilla, asignando permisos y roles a diferentes miembros.**

- Explica qué son los "repositorios privados" en una organización y cómo se gestionan los permisos para acceder a ellos.
  - **Son repositorios que solo pueden ser accedidos por usuarios que hayan sido invitados específicamente.**

Git con IA – Introducción y exploración de herramientas

- Es importante saber que con Snyk viene siendo una herramienta que se especializa en el análisis de seguridad del código para detectar vulnerabilidades.

TEORIA-CONCEPTOS EN GIT

- La diferencia de Copilot frente a Tabnine es: Que Copilot se entrena con datos de GitHub y proporciona sugerencias completas basadas en IA

- El primer paso para configurar un repositorio en GitHub con integración de una herramienta de IA como SonarQube, es la de instalar SonarQube y vincularlo mediante un webhook o CI/CD pipeline

- Al realizar la configuración de Copilot en GitHub, es necesario que para que la herramienta funcione dentro del repositorio, será la de habilitar la opción de "Copilot" dentro de los ajustes de la cuenta GitHub e instalar la extensión en el IDE

Prácticas de programación con IA

- ¿Qué característica distingue a GitHub Copilot en la implementación asistida de funciones sencillas?
  La que completa funciones completas y genera lógica del código basándose en comentarios y patrones

- Al comparar las sugerencias de Copilot y Tabnine, ¿cuál es una ventaja notable de Tabnine?
  Ofrece compatibilidad con múltiples IDEs y permite mayor personalización de modelos

- ¿Cuál es una funcionalidad destacada de Hugging Face para la generación automatizada de contenido?
  La implementación de modelos preentrenados para clasificación de texto

- ¿Qué paso es fundamental para crear un script básico de clasificación de texto usando Hugging Face?
  Instalar la librería transformers y cargar un modelo preentrenado

Seguridad y calidad del código

- ¿Cuál es el principal objetivo de Snyk en el desarrollo de software?
  Detectar y corregir vulnerabilidades de seguridad en dependencias y configuraciones

- ¿Qué tipo de análisis realiza SonarQube para mejorar el código?
  Análisis estático para evaluar la calidad, seguridad y mantenibilidad del código

- ¿Cuál de las siguientes acciones es necesaria para escanear vulnerabilidades con Snyk en un repositorio de GitHub?
  Activar Snyk mediante línea de comandos y otorgar permisos de escaneo al repositorio

- Después de un análisis estático con SonarQube, el sistema recomienda eliminar código duplicado, pero cuál es el beneficio principal de aplicar esta mejora:
  La mejora vendría siendo la legibilidad y mantenibilidad del código