IT Academy

by KIBERNUM

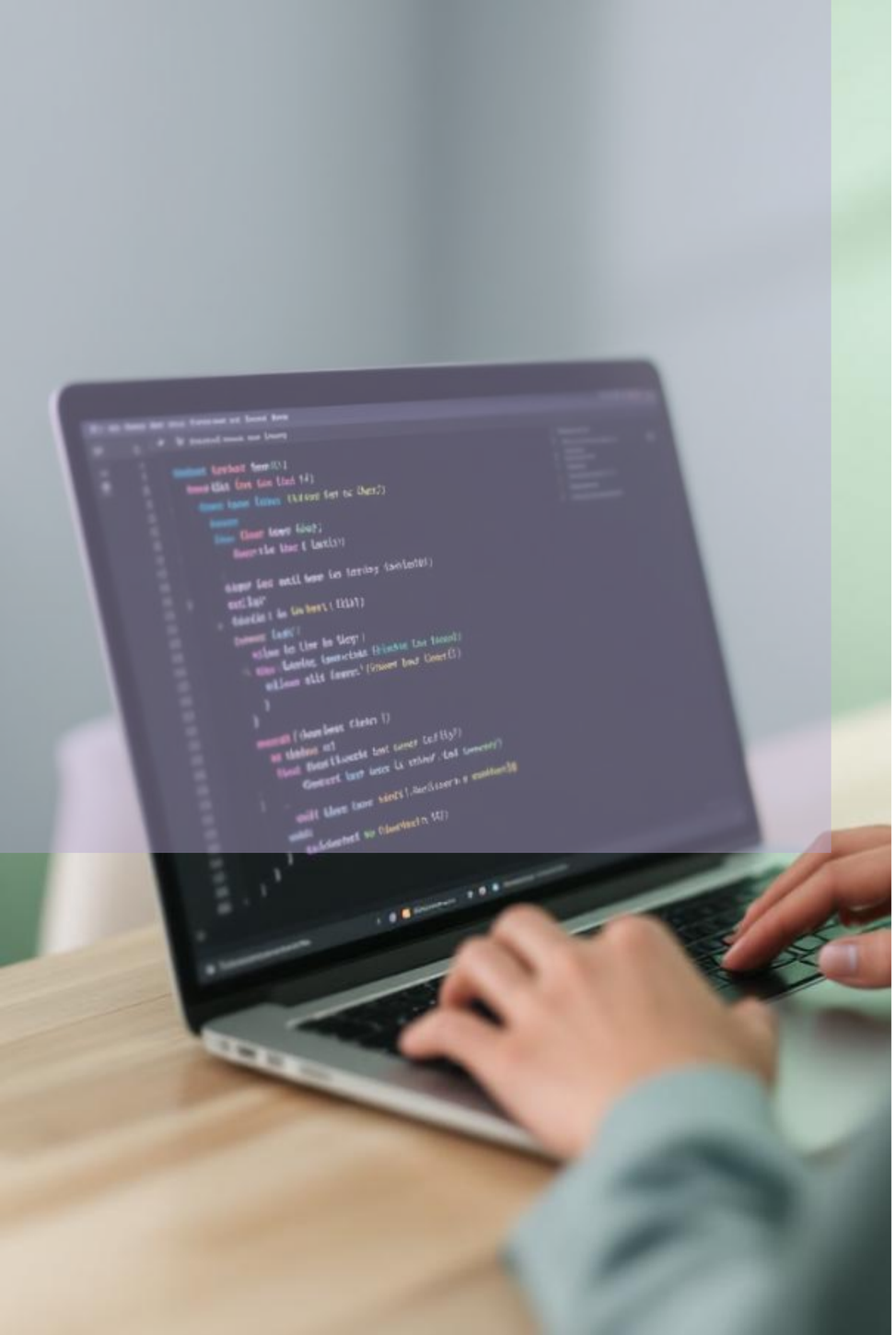
Programa: Desarrollo de Aplicaciones Front-End

-

Módulo 2: Fundamentos de Desarrollo Front-End

Aprendizaje esperado

Gestionar el código fuente de un proyecto utilizando GitHub para mantener un repositorio de código remoto seguro y permitir trabajo concurrente.



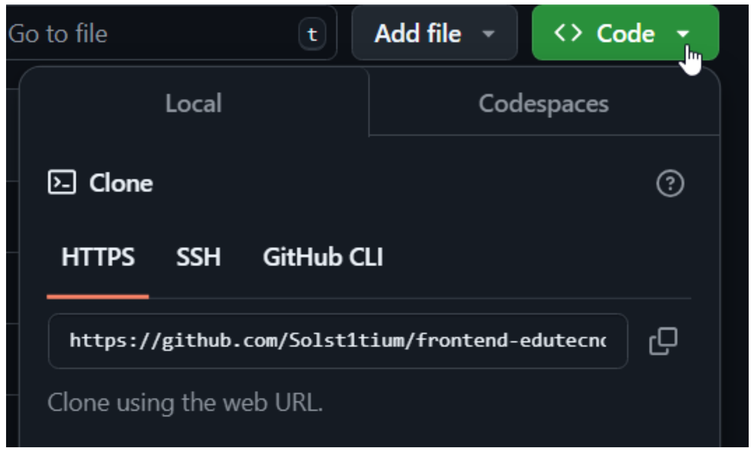
Clonando un Repositorio Remoto

Clonar un repositorio es una tarea fundamental para cualquier usuario de Git. Permite crear una copia exacta de un repositorio remoto en tu máquina local, facilitando la colaboración, el desarrollo y la exploración de proyectos existentes. Al clonar, no solo obtienes los archivos y carpetas del repositorio, sino también todo su historial de cambios.

Pasos para Clonar

- **Copiar la URL**

En GitHub, ve a la página del repositorio que deseas clonar. Haz clic en el botón verde "Code" y selecciona la URL del repositorio (HTTPS o SSH, según tus preferencias). Puedes copiar la URL con un clic derecho o Ctrl+C.



Pasos para Clonar

- **Comando git clone**

Abre tu terminal o consola de comandos. Navega a la carpeta donde deseas guardar el repositorio clonado. Luego, ejecuta el comando `git clone [URL del repositorio]`. Git creará una copia del repositorio remoto en tu máquina, incluyendo todos los archivos, carpetas y el historial completo de commits.

```
    @DESKTOP-Q7LG5E8 MINGW64 ~/Desktop
$ git clone url-del-repositorio
```

Accediendo a la Carpeta Clonada

1

Una vez que el comando `git clone` se completa, se creará una carpeta con el mismo nombre que el repositorio en tu directorio actual. Puedes acceder a esta carpeta con el comando `cd nombre-del-repositorio`.

```
@DESKTOP-Q7LG5E8 MINGW64 ~/Desktop
$ cd nombre-del-repositorio
```

2

Desde aquí, puedes comenzar a trabajar en el proyecto: realizar cambios, crear nuevas ramas y hacer push de tus modificaciones al repositorio remoto (si tienes los permisos necesarios). Git te permitirá trabajar en el proyecto con confianza, sabiendo que tus cambios están seguros y puedes colaborar con otros desarrolladores.

Consideraciones Adicionales

- Autenticación:

❑ Si clonas un repositorio privado o necesitas permisos, asegúrate de tener configuradas tus credenciales de GitHub.

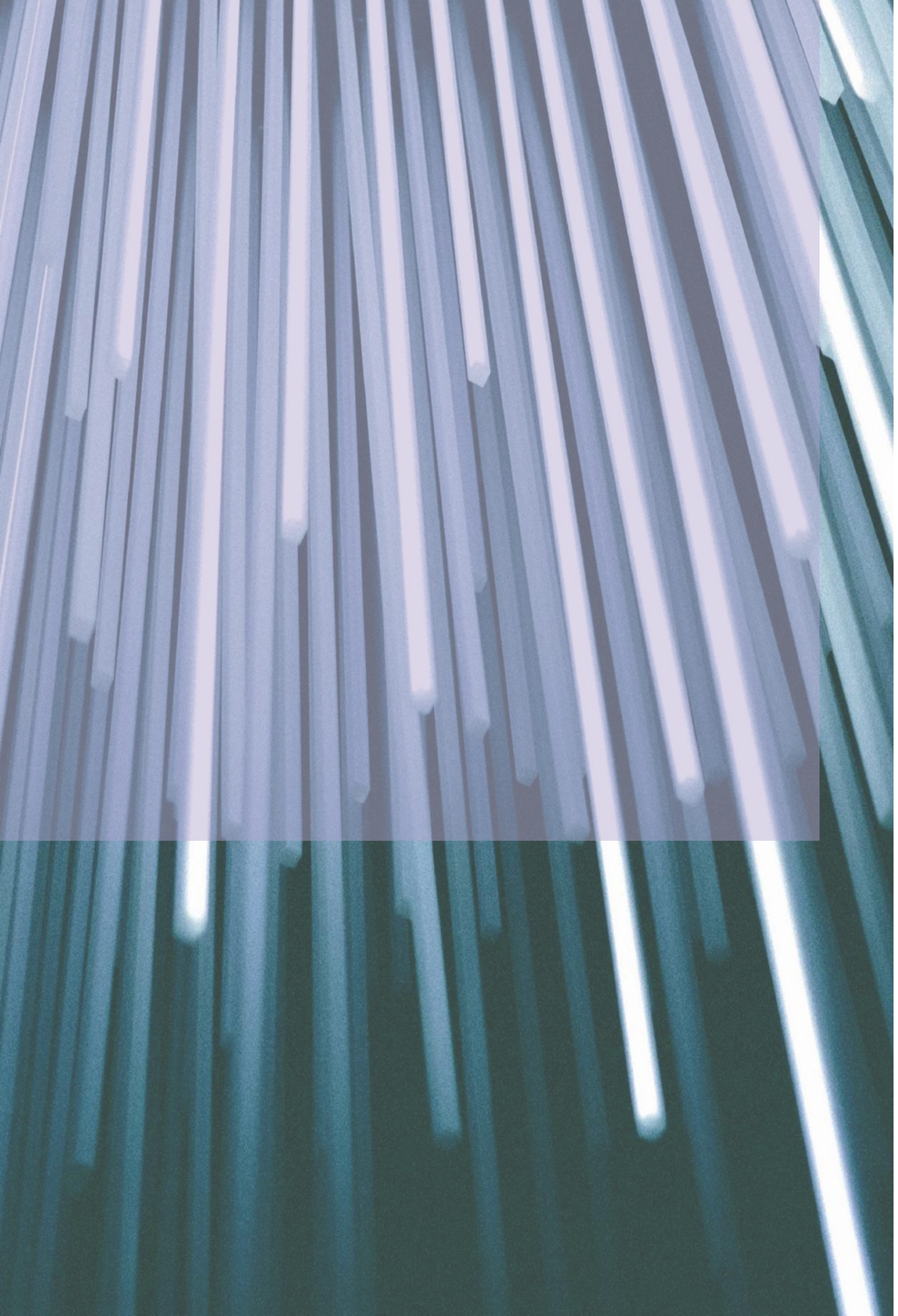
❑ Puedes hacerlo mediante un token de acceso personal o configurando claves SSH.

- Actualizaciones:

❑ Después de clonar, mantén tu copia local sincronizada con el repositorio remoto usando git pull regularmente.

- Ramas:

❑ Git trae todas las ramas del repositorio remoto, lo que te permite trabajar en diferentes características o correcciones de errores.



Documentando un Proyecto con Markdown

Markdown es un lenguaje de marcado ligero, utilizado principalmente para la documentación en proyectos de GitHub. El archivo README.md es fundamental en la mayoría de los repositorios, proporcionando una descripción clara del proyecto y facilitando la colaboración.

Ventajas Clave de Markdown

1 **Simplicidad**

Fácil de usar, no requiere editores complejos ni conocimientos avanzados de HTML o CSS para crear documentos formateados.

2 **Compatibilidad**

Soportado por GitHub y muchas otras plataformas, ideal para documentación que se verá en múltiples entornos.

3 **Legibilidad**

El texto en Markdown es fácil de leer incluso sin procesamiento, útil para revisar archivos de texto plano.

Estructura Básica de Markdown

🔹 Títulos

❑ Se crean con el símbolo #

```
# Título principal
## Subtítulo
### Subtítulo de tercer nivel
```

🔹 Listas

Pueden ser:

❑ ordenadas (1. 2. 3.)

❑ desordenadas (- * +).

```
- Item 1
- Item 2
- Item 3
```

Estructura Básica de Markdown

🔹 Enlaces

❑ Se crean con el formato [Texto del enlace](URL). Permiten conectar a recursos externos.

```
[Texto del enlace](https://ejemplo.com)
```

🔹 Imágenes

Similar a los enlaces, pero precedido por un signo de exclamación: ![Texto alternativo](URL de la imagen).

```
![Texto alternativo](https://ejemplo.com/imagen.jpg)
```

Uso de Markdown en GitHub

- **Archivos README.md:**

Documenta el proyecto, incluye descripción, instalación, dependencias e instrucciones de contribución. GitHub lo renderiza en la página principal.

- **Otras Documentaciones**

Archivos como CONTRIBUTING.md (guía para colaboradores) y LICENSE.md (términos de licencia) también utilizan Markdown.

- **Issues y Pull Requests**

Markdown facilita discusiones formateadas y claras con títulos, listas y enlaces para describir problemas y soluciones.

Ejemplo:

Un archivo README.md típico incluiría:

- Descripción del proyecto.
- Instrucciones de instalación.
- Cómo ejecutar el proyecto.
- Contribuciones y licencias.

````
# Nombre del Proyecto
Descripción breve de lo que hace el proyecto.

## Instalación
Instrucciones para instalar el proyecto localmente:
```bash
git clone https://github.com/usuario/proyecto.git
cd proyecto
npm install
````
````

¿Cómo usarlo?

- **Clonar el Proyecto**

Primero debes clonar el proyecto en tu computadora desde GitHub utilizando el comando: git clone [URL del repositorio].

```
@DESKTOP-Q7LG5E8 MINGW64 ~/Desktop
$ git clone https://github.com/usuario/proyecto.git
```

- **Ingresar a la Carpeta**

Luego debes ingresar en la carpeta del proyecto con: cd [nombre del proyecto].

```
@DESKTOP-Q7LG5E8 MINGW64 ~/Desktop
$ cd proyecto
```

¿Cómo usarlo?

- **Instalar Dependencias**

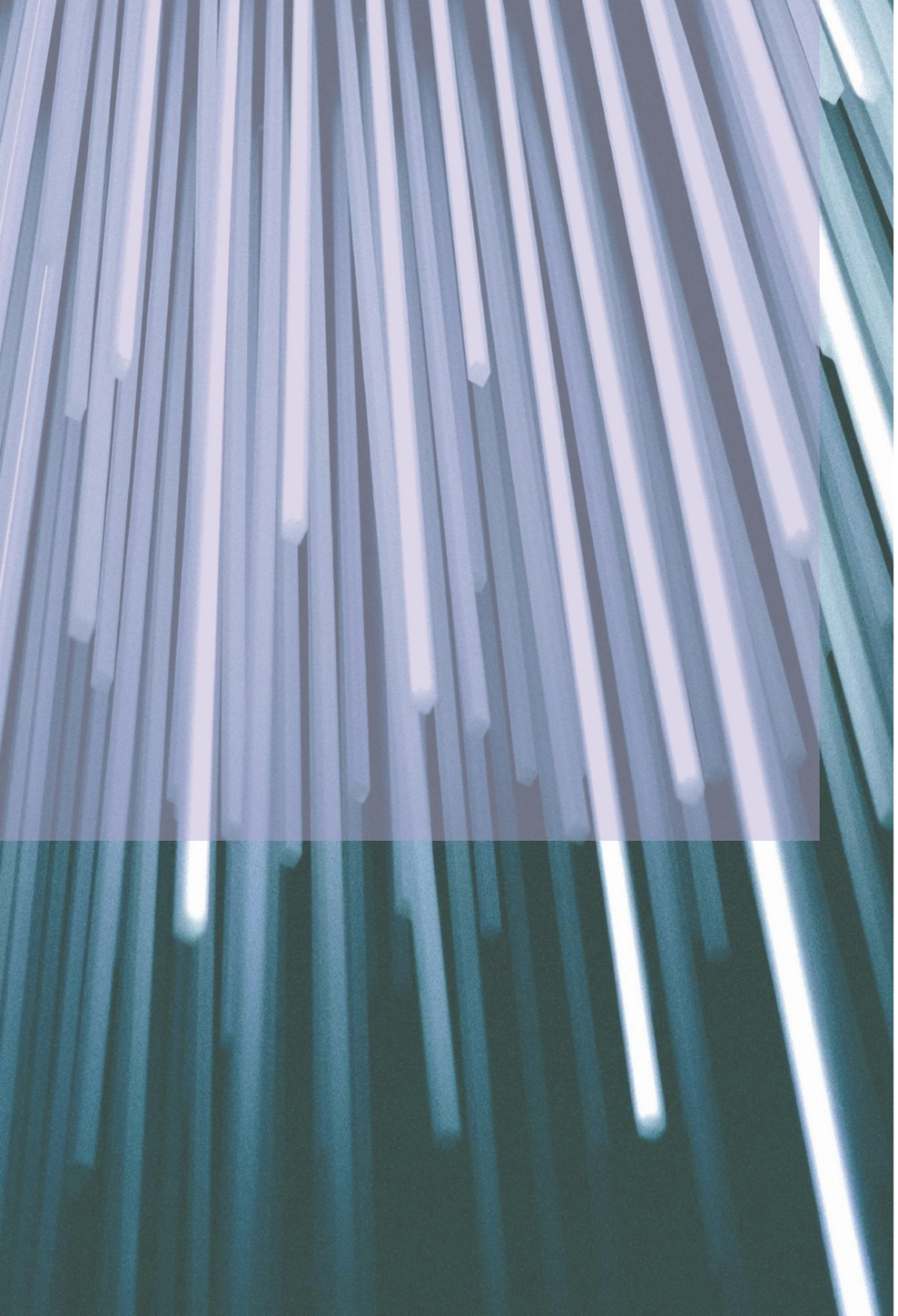
Después, es necesario instalar las dependencias del proyecto. En este caso, se está utilizando npm (Node Package Manager) para instalar todas las dependencias del proyecto.

```
@DESKTOP-Q7LG5E8 MINGW64 ~/Desktop
$ npm install
```

- **Ejecutar el proyecto**

Una vez que el proyecto está instalado, esta sección te muestra cómo ejecutarlo. Usando el siguiente comando, iniciarás el proyecto, por ejemplo, si es una aplicación web:

```
@DESKTOP-Q7LG5E8 MINGW64 ~/Desktop
$ npm start
```



Administrando Pull Requests en GitHub

Los Pull Requests (PR) son una funcionalidad clave en GitHub para gestionar la colaboración en proyectos. Permiten a los desarrolladores solicitar la integración de los cambios realizados en una rama hacia otra, facilitando un flujo de trabajo estructurado y seguro.

Creación de un Pull Request

- **Subir Cambios al Repositorio**

Antes de crear un PR, sube tus cambios locales al repositorio remoto en GitHub. Asegúrate de haber realizado git add y git commit, luego usa git push origin nombre-rama. Esto envía tus cambios a la rama especificada en GitHub, dejándolos listos para el PR..

```
@DESKTOP-Q7LG5E8 MINGW64 ~/Desktop
$ git push origin nombre-de-la-rama
```

Creación de un Pull Request

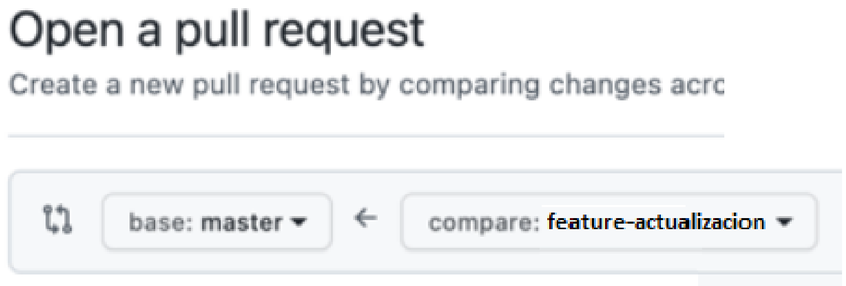
- **Navegar y Seleccionar "Compare & pull request"**

Dirígete a GitHub.com e ingresa a tu repositorio. Una vez que hayas subido la nueva rama, GitHub mostrará un mensaje en la página del repositorio indicando la opción "Compare & pull request". Haz clic para comenzar el proceso de PR.

Detalles del Pull Request

- **Revisar Ramas de Origen y Destino**

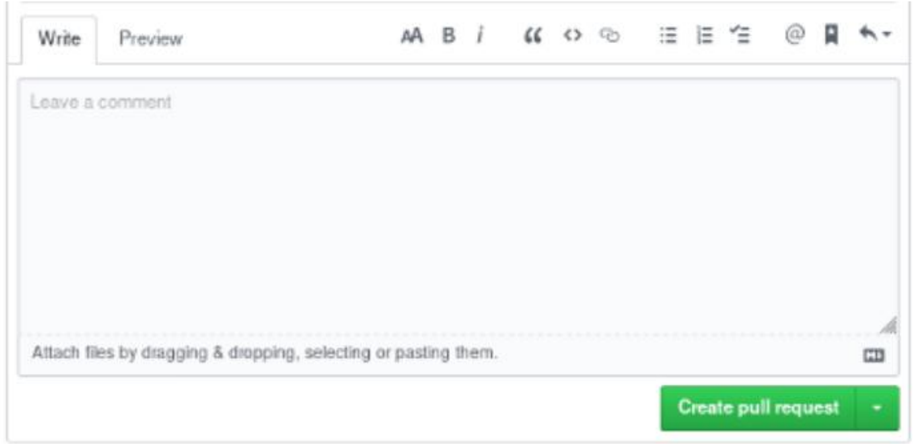
En la página de creación del PR, verás menús desplegables para elegir la rama de origen y destino. La rama base (destino) suele ser la principal (main o master), y la rama compare (origen) es la que contiene los cambios a fusionar.



Detalles del Pull Request

- **Proporcionar Descripción Clara**

Incluye un título descriptivo y agrega una descripción en el campo correspondiente. Describe el problema o mejora que resuelve el PR, así como detalles específicos de las modificaciones realizadas. Esto ayudará a los revisores a entender el contexto.



Detalles del Pull Request

- **Enviar el Pull Request**

Haz clic en "Create pull request" para enviar el PR. Tu PR ahora estará en la lista de Pull Requests de tu repositorio, y otros colaboradores podrán revisarlo o solicitar cambios antes de aprobarlo para su fusión.

Revisión de Código

- **Comentar Líneas de Código**

Señalar posibles errores, sugerir mejoras o hacer preguntas directamente en líneas específicas del código.

- **Aprobar o Rechazar Cambios**

Los revisores pueden aprobar el PR si los cambios son correctos, o solicitar modificaciones si consideran que algo debe ajustarse.

- **Discusiones Dentro del PR**

Iniciar conversaciones dentro del propio PR para aclarar dudas o discutir soluciones antes de proceder.

Fusión (Merge)

- **Merge Directo**

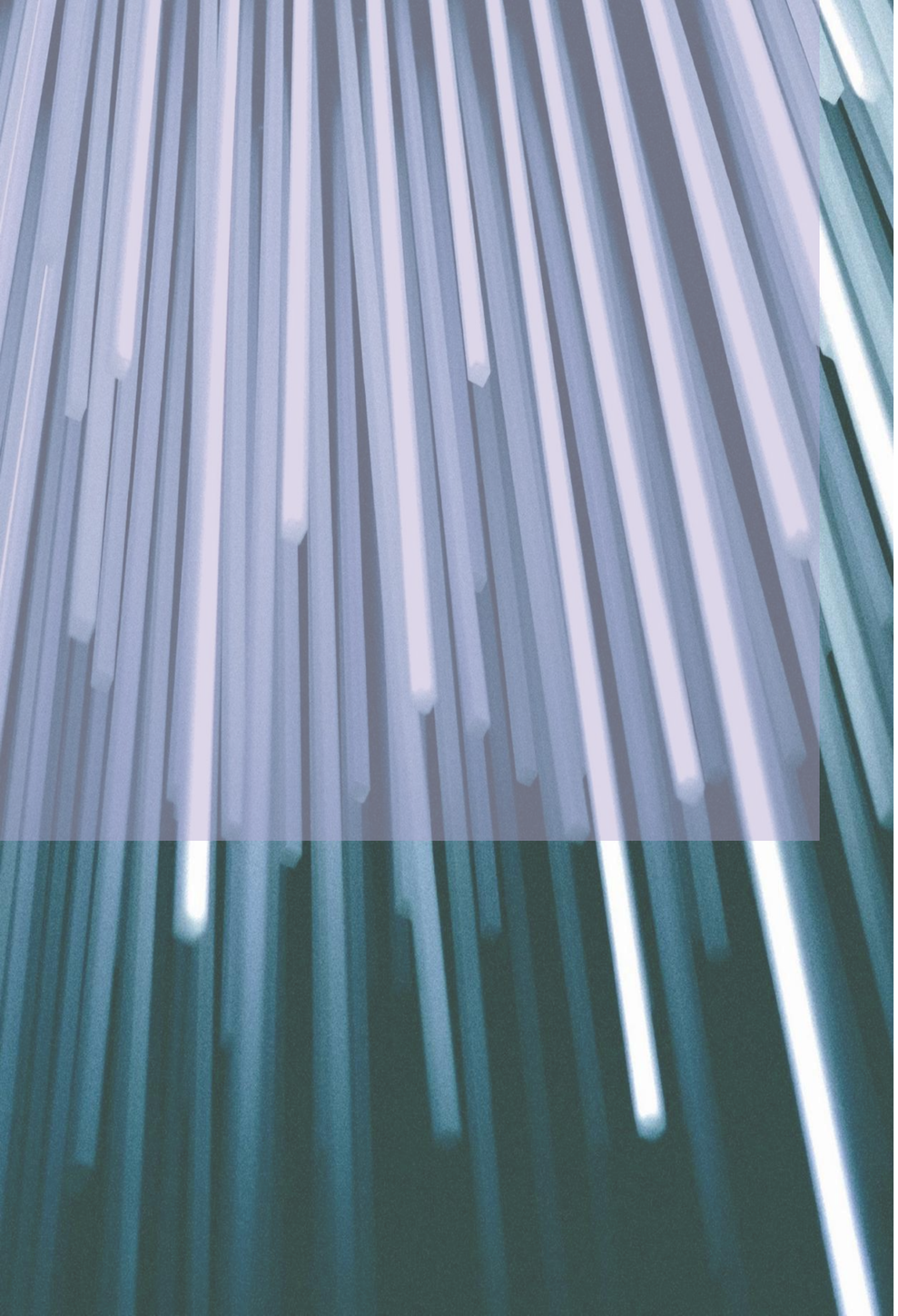
Los cambios se fusionan directamente en la rama de destino, manteniendo el historial completo de commits.

- **Squash and Merge**

Todos los commits se combinan en un solo commit antes de ser fusionados, manteniendo un historial más limpio.

- **Rebase and Merge**

Los commits del PR se aplican sobre la rama de destino uno por uno, integrando los cambios sin crear un merge commit adicional.



Automatización e Integración Continua

- GitHub permite la integración de acciones automáticas dentro de los PR, como la ejecución de pruebas automáticas o la verificación del estilo de código a través de herramientas de CI/CD.

- Esto permite que los cambios sean validados automáticamente antes de ser revisados por los colaboradores, asegurando que el código cumple con ciertos estándares de calidad y no rompe funcionalidades existentes.

Automatización en los Pull Requests (PR)

- **Automatización en PRs:**
  - Ejecución de pruebas automáticas.
  - Verificación de estilo de código.

- **Herramientas CI/CD:**
  - Se integran para validar automáticamente los cambios antes de ser revisados por colaboradores.
  - Asegura que el código cumpla con los estándares y no rompa funcionalidades existentes.

Cierre de un Pull Request (PR)

- **Fusión exitosa:** GitHub cierra automáticamente el PR.

- **Fusión no realizada:** El PR puede cerrarse sin realizar la fusión, dejando el código intacto en la rama principal.

- **Flujo de trabajo eficiente:** Los PRs gestionan el ciclo de vida del código, garantizando que solo los cambios aprobados se integren.

Flujos de Trabajo con GitHub

- Git Flow

Ideal para proyectos de desarrollo a largo plazo. Se basa en la creación de ramas específicas para la versión estable, la versión en desarrollo, las funcionalidades, las correcciones y los lanzamientos.

- Ramas principales:

- Master: Contiene la versión estable en producción.
- Develop: Contiene la versión de desarrollo con características en construcción.

- Ramas adicionales:

- Feature: Para nuevas funcionalidades.
- Hotfix: Para correcciones urgentes.
- Release: Para preparar la versión de lanzamiento.

Flujos de Trabajo con GitHub

- Feature Branch Workflow

Se centra en la creación de ramas para cada nueva funcionalidad o corrección. Una vez finalizados los cambios, se abre un PR para que el equipo revise y apruebe los cambios antes de integrarlos en la rama principal.

- Creación de ramas por funcionalidad o corrección:

- Cada tarea (nueva funcionalidad o corrección de errores) se realiza en su propia rama.
- Una vez terminados los cambios, se crea un Pull Request (PR) para revisión.

- Beneficios:

- Colaboración fluida.
- Código organizado y modular.

Flujos de Trabajo con GitHub

- Forking Workflow

Muy utilizado en proyectos de código abierto. Los colaboradores externos hacen un fork (copia) del repositorio original en su cuenta de GitHub, implementan mejoras o correcciones, y luego crean un PR hacia el repositorio original.

- Flujo de trabajo:

- Fork: El colaborador hace un fork del repositorio.
- Desarrollo en una rama: Implementa mejoras o correcciones.
- Pull Request: Crea un PR hacia el repositorio original para revisión.

- Ventajas:

Protege el repositorio principal.

Facilita contribuciones externas.

Resumen

✅ GitHub es una herramienta esencial para el desarrollo colaborativo:

- Permite a varios desarrolladores trabajar en un mismo proyecto sin sobrescribir cambios.
- Facilita la comunicación entre equipos a través de Pull Requests y revisiones de código.

✅ Facilita el control de versiones y la organización del código:

- Registra cada modificación con commits, permitiendo revertir cambios si es necesario.
- Usa ramas (branches) para desarrollar nuevas características sin afectar el código principal.

✅ Es clave para proyectos de cualquier tamaño, desde personales hasta empresariales:

- En proyectos pequeños, ayuda a mantener un historial ordenado del código.
- En empresas, permite una gestión eficiente con integraciones de CI/CD, automatización y permisos de acceso.

¿Cómo mejorarías tu flujo de trabajo con Git y GitHub en el futuro?

Conceptos Fundamentales

Commits Claros y Frecuentes

✅Realiza commits frecuentes y con mensajes claros que describan los cambios realizados. Esto facilita la comprensión del historial de cambios y mejora la colaboración.

Usa ramas para nuevas características

✅ Trabaja siempre en una rama separada para nuevas características o correcciones. Esto mantiene el código limpio y organizado.

Mantén tu repositorio actualizado

✅ Usa git pull regularmente para mantener tu repositorio local actualizado con los cambios de otros colaboradores.

•Evita nombres genéricos como x o dato.
•Usa nombres claros como precioProducto o nombreUsuario.

Conceptos Fundamentales

Documenta bien con README

✅Asegúrate de tener un archivo README.md claro y detallado que explique el propósito del proyecto, cómo configurarlo y cómo contribuir..

IT Academy

by KIBERNUM