

GIT Y GITHUB PROFESIONAL

MILUSKA PAJUELO

PRIMER MÓDULO

Introducción
a Git y GitHub

Presentación del docente

Ahora, conoceremos algunos aspectos relevantes de la experta y del contenido a abordar en este curso.

Los temas que abordaremos son:

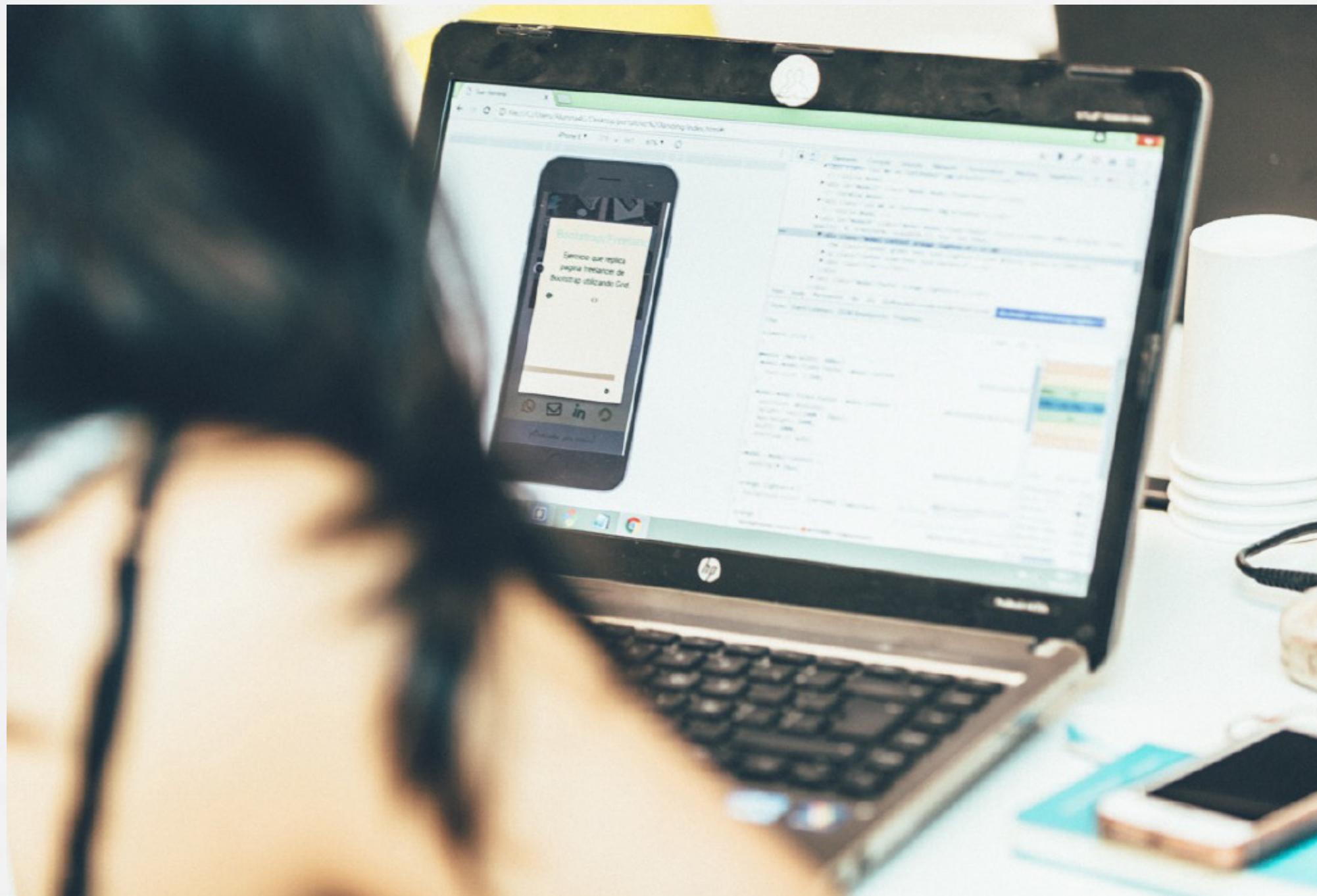
- ¿Quién soy?
- ¿Qué hago? Mi experiencia
- ¿Por qué es importante llevar este curso?
- ¿Qué aprenderán en este curso?



MILUSKA PAJUELO

¿Quién soy?

- Soy Ingeniera Industrial, especializada en Desarrollo de software.
- He desarrollado aplicaciones web responsive, utilizando Python y Javascript.
- He trabajado en equipos de diferentes países como Tailandia, Chile y Canadá. Actualmente, para una empresa ubicada en México de manera remota.



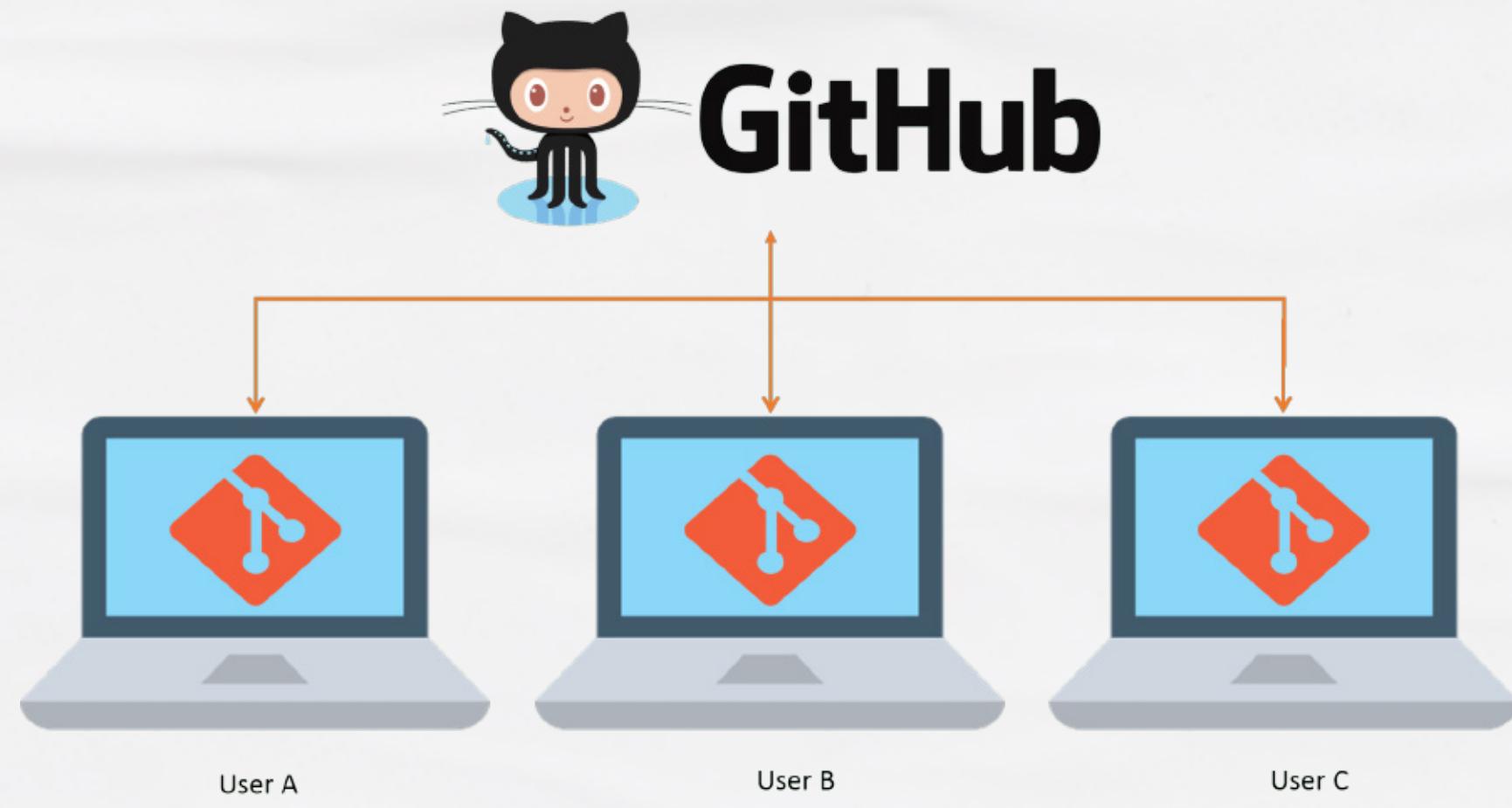
¿Qué hago?

Soy desarrolladora front-end en una compañía mexicana. Formo parte de un equipo multidisciplinario y me encargo del diseño, desarrollo y testeo de aplicaciones web responsive.



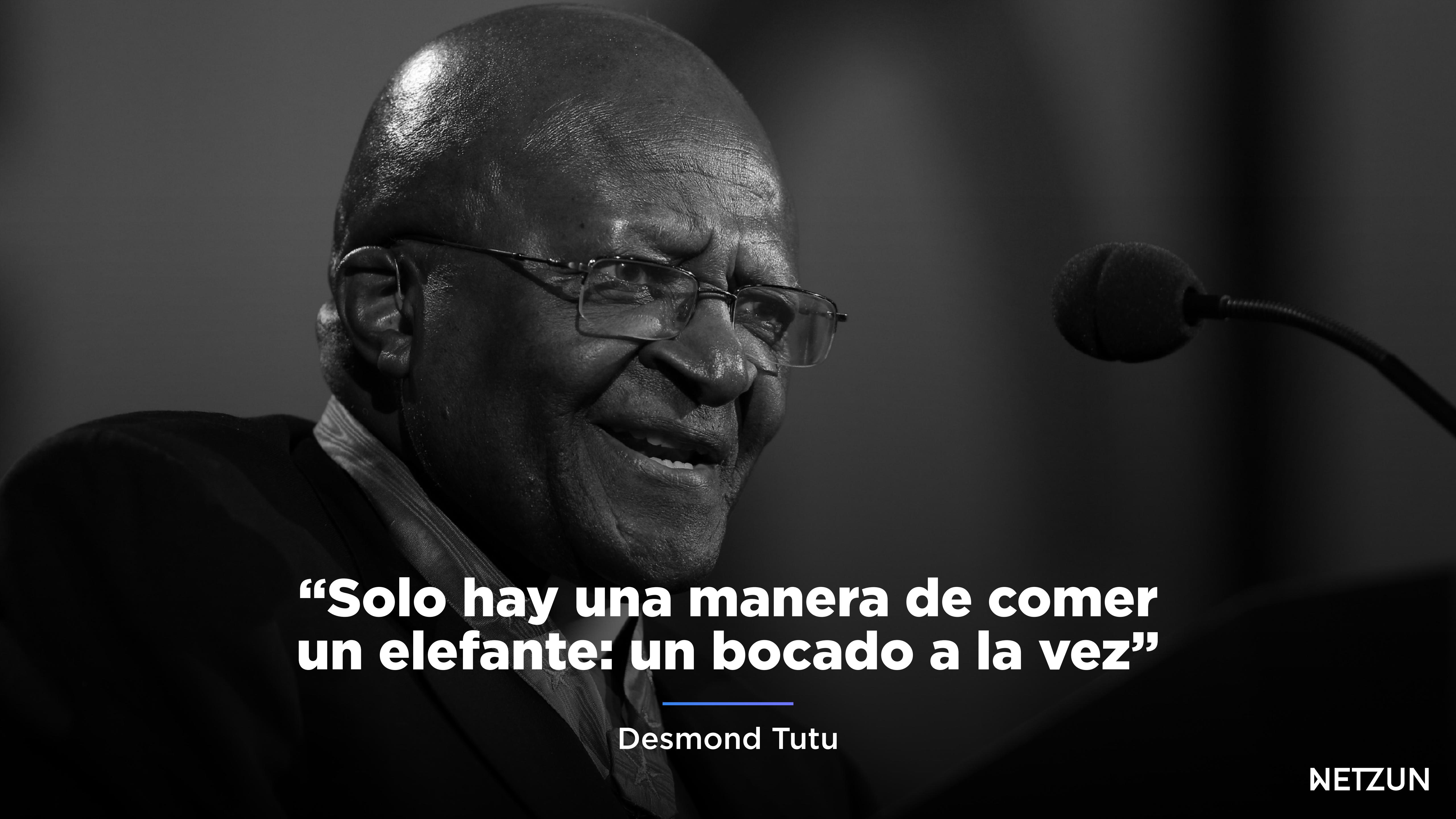
¿Por qué es importante llevar este curso?

Hoy en día, aprendemos, compartimos, creamos, construimos y colaboramos en proyectos tecnológicos con personas que se encuentran en diferentes partes del mundo. Por lo tanto, debemos ser cuidadosos y gestionar una adecuada comunicación entre los equipos.



¿Qué aprenderán en este curso?

Aprenderás a instalar Git, así como el uso de comandos básicos, avanzados y de emergencia, además de trabajar colaborativamente en GitHub y aplicar buenas prácticas para manejar el control de tus proyectos basado en Git-Flow, Conventional Commits y GitHub projects.



**“Solo hay una manera de comer
un elefante: un bocado a la vez”**

Desmond Tutu

Fundamentos de Git

Ahora, conoceremos conceptos fundamentales de Git.

Los temas que abordaremos son:

- ¿Qué es Git?
- ¿Por qué usar un sistema de versiones para tu proyecto?
- ¿Por qué usar Git para el control de versiones?

¿Qué es Git?

Es un sistema de control de versiones distribuido, gratuito y de código abierto (open source).

Un sistema de control de versiones

Se refiere a un sistema que graba los cambios ocurridos en archivos y/o proyectos, y podemos acceder a cualquier versión de este utilizando comandos.

Un código abierto

Se refiere a un código que está diseñado para ser de acceso público: cualquiera puede ver, modificar y distribuir el código como mejor le parezca.

Distribuido

El proyecto se ubica localmente en nuestras computadoras y no necesitas Internet para guardar información del mismo.

¿Por qué usar un sistema de versiones para tu proyecto?



En proyectos de tecnología, estamos frecuentemente cambiando, mejorando e iterando los documentos, archivos, tecnologías, adaptándolos a los requerimientos del usuario/cliente.



Por lo tanto, necesitamos contar con un gestor de versiones para así guardarlas todas y mantener la trazabilidad del proyecto.

¿Por qué usar Git para el control de versiones?

Es el sistema de versión más moderno y popular.

Git es un gestor de versiones que nos permite guardar las versiones de nuestro código fuente y a la vez mantener un respaldo de la información de forma local y remota.



v 1.0/v 2.0/v 3.0



GG

**Intenta fallar rápido, aprende e
intenta fallar nuevamente y así**

JJ

Miluska Pajuelo

Fundamentos de GitHub

Ahora, conoceremos conceptos fundamentales de GitHub.

Los temas que abordaremos son:

- ¿Qué es GitHub?
- ¿Por qué usar GitHub para el control de versiones?
- Conociendo la plataforma GitHub
- ¿Cómo crear mi primer repositorio desde GitHub?

¿Qué es GitHub?

Es una plataforma de alojamiento que Microsoft compró en el 2018 por US\$ 7 500 millones de dólares.

Permite la creación de repositorios de código y el guardado en la nube de forma segura (remota), usando un sistema de control de versiones de Git.

A través de su plataforma amigable, nos permite centralizar el repositorio en un solo lugar y compartirlo con personas en diferentes partes del mundo.



¿Por qué usar GitHub para el control de versiones?

GitHub permite lo siguiente:

- 1 Trabajar de forma colaborativa en un solo proyecto al mismo tiempo de forma remota
- 2 Reducir el riesgo de trabajo duplicado o conflictivo
- 3 Alojar de forma privada o pública tus repositorios y proyectos
- 4 Reducir el tiempo de producción
- 5 Desplegar tu web estática de forma gratuita



Conociendo la plataforma GitHub

Este tema será desarrollado a lo largo
de la explicación del videotutorial.



¿Cómo crear mi primer repositorio desde GitHub?

Este tema será desarrollado a lo largo de la explicación del videotutorial.



“Se necesitan dos piedras para hacer fuego”

Louisa May Alcott

Ideas resumen del módulo 1

01

El comando git clone nos permite realizar modificaciones, eliminar o agregar archivos al proyecto original.

02

Para crear un repositorio de Git de forma local, es necesario usar el comando git init. Notaremos un archivo .git en la carpeta del proyecto que indica que el repositorio fue creado.

03

El comando git tag nos permite generar la versión de nuestro proyecto, también podemos generar la versión desde GitHub.

SEGUNDO MÓDULO

**Primeros
pasos en Git**

Instalando Git

Ahora, entenderemos cómo se instala Git.

Los temas que abordaremos son:

- Instalando Git/Git Bash en Windows



Instalando Git

Este tema será desarrollado a lo largo de la explicación del videotutorial.



“Piensa constantemente cómo podrías hacer mejor las cosas”

Elon Musk

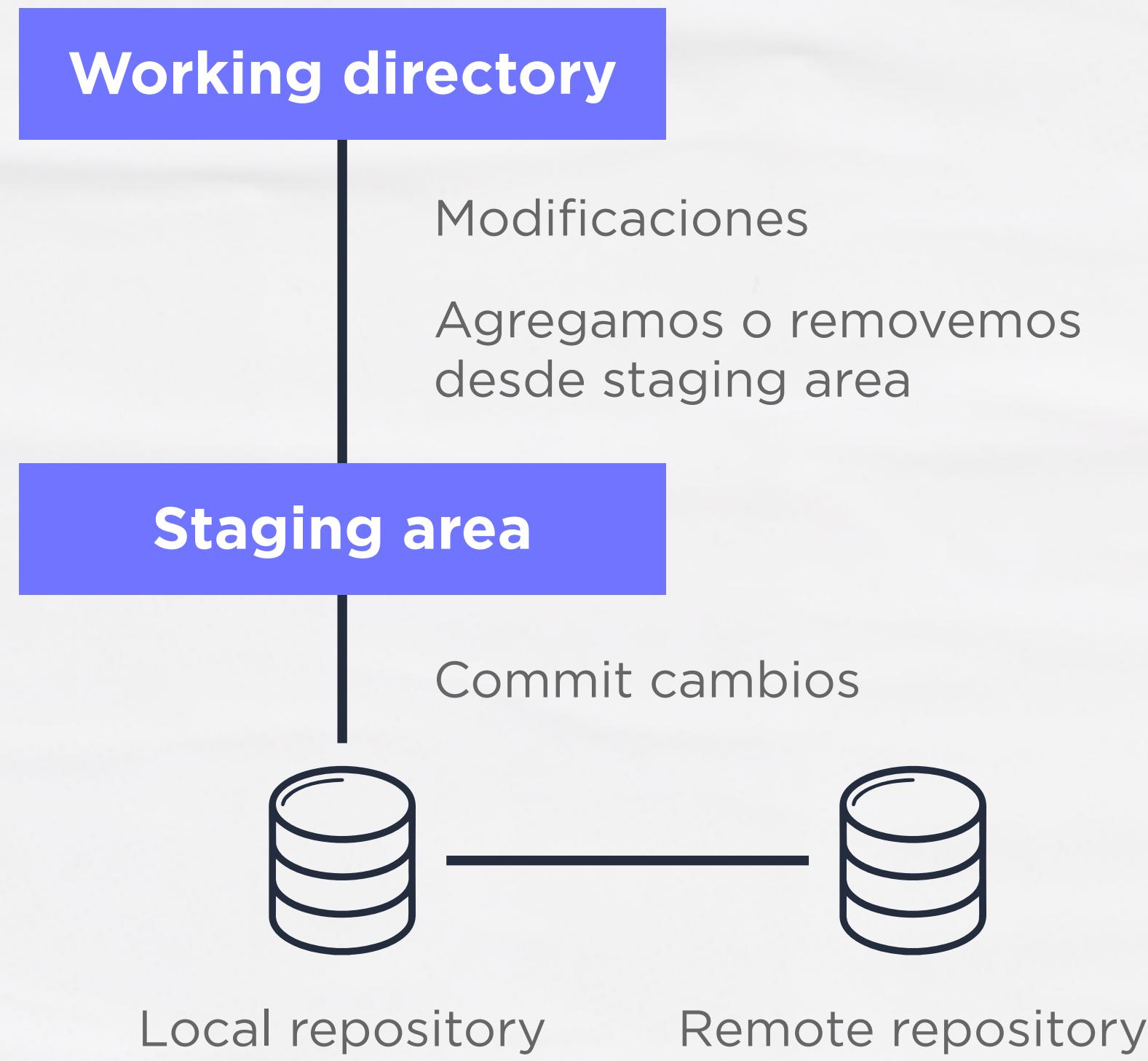
Comandos básicos de Git

Ahora, conoceremos los comandos básicos de Git.

Los temas que abordaremos son:

- Diagrama de flujo de trabajo
- Crea tu primer repositorio desde el CLI
- Clona tu primer repositorio desde Github
- Primeros comandos para actualizar un proyecto

Diagrama de flujo de trabajo



Working directory

Es el directorio donde trabajamos, considerado como un repositorio.

Staging area o zona de montaje

Allí están los cambios antes de hacer la confirmación o commit.

Local repository o repositorio local

Es el que se encuentra en nuestra computadora.

Remote repository o repositorio remoto

Es el que se encuentra en un servidor como GitHub.



Crea tu primer repositorio desde el CLI

Este tema será desarrollado a lo largo
de la explicación del videotutorial.



Clona tu primer repositorio desde Github

Este tema será desarrollado a lo largo de la explicación del videotutorial.



Primeros comandos para actualizar un proyecto

Este tema será desarrollado a lo largo de la explicación del videotutorial.



“Ten en cuenta que la información es poder”

Bill Gates

Comandos para un trabajo colaborativo

Ahora, conoceremos los comandos para realizar un trabajo colaborativo en Git.

Los temas que abordaremos son:

- Comandos para el trabajo colaborativo Git merge
- Trabajo con ramas o branches
- Crear mi primera versión desde el CLI



Comandos para un trabajo colaborativo

Este tema será desarrollado a lo largo de la explicación del videotutorial.



“Desde el principio, no pensaba en otra cosa que no fuera tener éxito”

Bill Gates

Ideas resumen del módulo 2

01

El comando git clone nos permite realizar modificaciones, eliminar o agregar archivos al proyecto original.

02

Para crear un repositorio de Git de forma local, es necesario usar el comando git init. Notaremos un archivo .git en la carpeta del proyecto que indica que el repositorio fue creado.

03

El comando git tag nos permite generar la versión de nuestro proyecto, también podemos generar la versión desde GitHub.

TERCER MÓDULO

**Comandos avanzados
de Git y GitHub**

Comandos avanzados para múltiples entornos

Ahora, conoceremos los comandos avanzados para múltiples entornos y cómo funcionan.

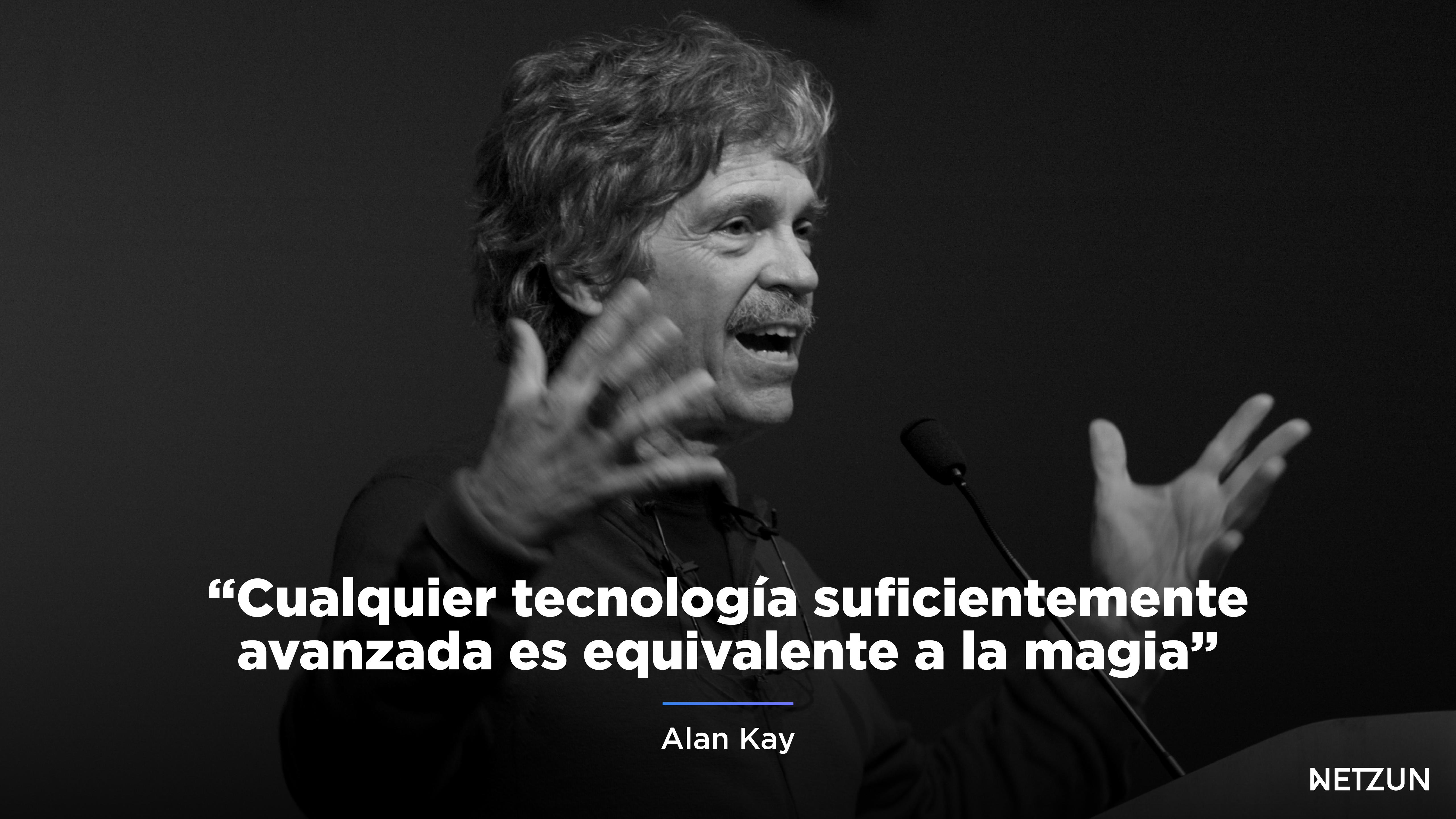
Los temas que abordaremos son:

- Git Stash
- Git commit --amend



Comandos avanzados para múltiples entornos

Este tema será desarrollado a lo largo
de la explicación del videotutorial.



“Cualquier tecnología suficientemente avanzada es equivalente a la magia”

Alan Kay

Comandos avanzados para emergencia

Ahora, conoceremos los comandos avanzados para emergencia y cómo funcionan.

Los temas que abordaremos son:

- Git revert
- Git reset



Comandos avanzados para emergencia

Este tema será desarrollado a lo largo
de la explicación del videotutorial.

GG

**Que algo no haya salido
como hayas querido no
significa que sea inútil**

JJ

Mark Kennedy

¿Cómo publicar tu página web con GitHub Pages?

Ahora, entenderemos cómo se hace un despliegue del proyecto con Github pages.

Los temas que abordaremos son:

- Comandos para hacer un deploy a GitHub pages
- Despliega tu primer proyecto en un hosting gratuito



¿Cómo publicar tu página web con **GitHub Pages?**

Este tema será desarrollado a lo largo
de la explicación del videotutorial.

“

**Existen muchas plataformas para
publicar tus proyectos de forma
gratuita, mantén la curiosidad**

”

Miluska Pajuelo

Ideas resumen del módulo 3

01

La diferencia entre git revert y git reset es que el primero agrega una nueva historia del proyecto sin modificar la historia existente y el segundo opera en el historial de commits, git índice y el trabajo directorio.

02

Usando git revert, se puede crear un nuevo commit que revierte los cambios realizados en el commit incorrecto.

03

La diferencia entre git log y git log -oneline es que con git log podemos ver el detalle del historial de commits.

CUARTO MÓDULO

**Buenas prácticas
en Git y GitHub**

Reglas para crear un historial de commits

Ahora, conoceremos las reglas para crear un historial de commits.

Los temas que abordaremos son:

- ¿Qué es conventional commits?
- ¿Por qué usar conventional commits?
- ¿Qué estructura deben tener mis commits?

¿Qué es conventional commits?

Conventional Commits es una especificación inspirada en las Guías Angular.



Es una convención ligera sobre los mensajes del commit.

```
git commit -m "EL MENSAJE"
```

Proporciona un conjunto sencillo de reglas para crear un historial de commits explícito, lo que facilita la escritura de herramientas automatizadas encima.

Esta convención encaja con SemVer al describir las funciones, las correcciones y los cambios importantes realizados en los mensajes del commit.

¿Por qué usar conventional commits?

Básicamente:



Comunicar la naturaleza de los cambios a los compañeros de equipo, el público y otras partes interesadas.



Comunicar la naturaleza de los cambios a los compañeros de equipo, el público y otras partes interesadas.

¿Qué estructura deben tener mis commits?

<type>(<scope>): <short summary>



[@commitlint/config-conventional](#)
(based on the [the Angular convention](#))

Links:

<https://bit.ly/3a2ZSNT>
<https://bit.ly/3sIcm45>

- **feat** (nueva funcionalidad para el usuario, no una nueva función para el script de compilación)
- **fix** (corrección de errores para el usuario, no una corrección para un script de compilación)
- **docs** (cambios en la documentación)
- **style** (formato, puntos y coma faltantes, etc. sin cambio de código de producción)
- **refactor** (refactorización de código de producción, por ejemplo, cambiar el nombre de una variable)
- **test** (agregar pruebas faltantes, refactorizar pruebas sin cambio de código de producción)
- **chore** (actualización de tareas sin cambio de código de producción)

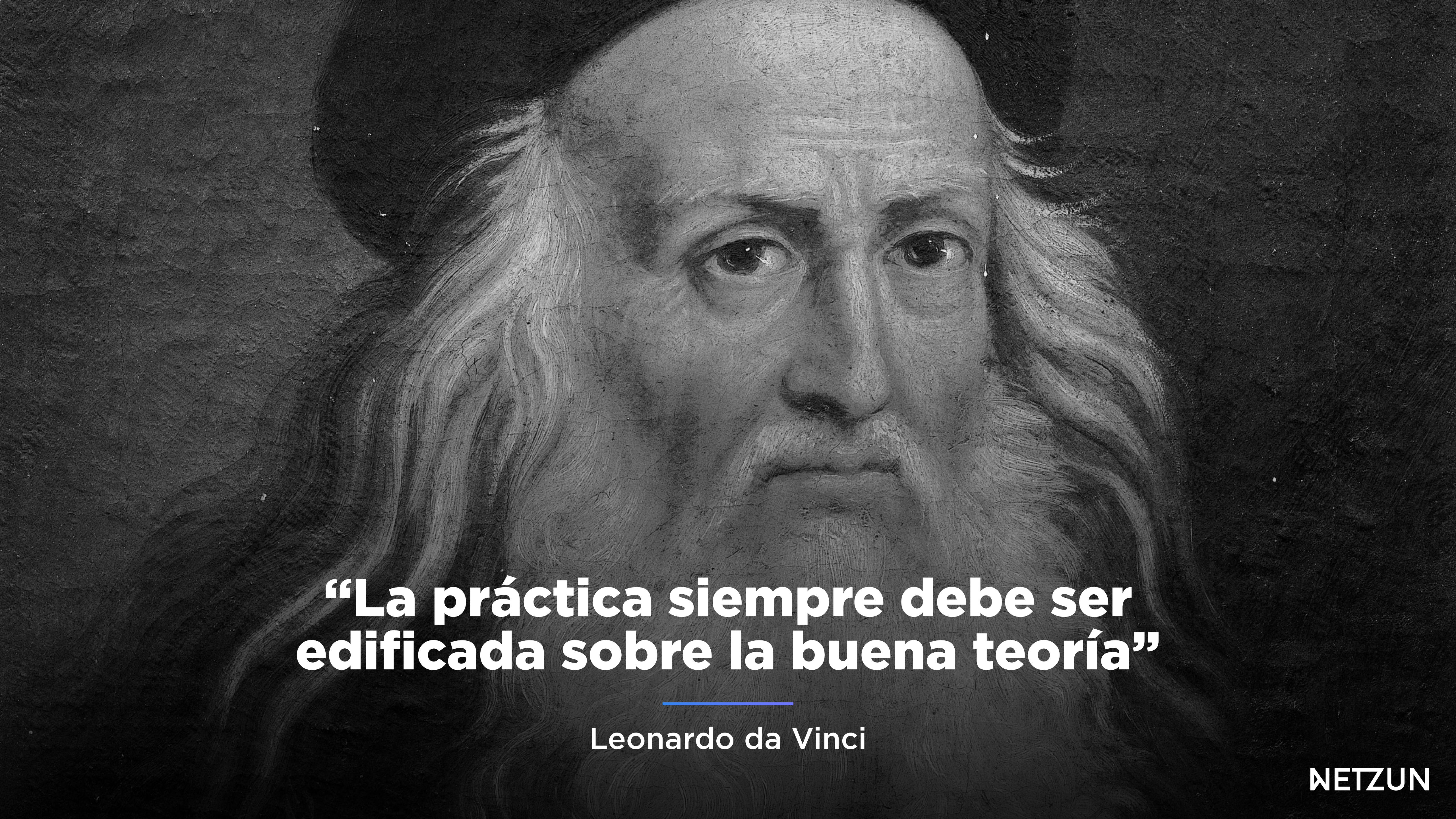
¿Qué estructura deben tener mis commits?

Ejemplos

A Feat(shopping cart): add the amazing button

B Build(release): add version to 1.0.0

C Feat(login): add component button



“La práctica siempre debe ser edificada sobre la buena teoría”

Leonardo da Vinci

Manejo de ramas con Git Flow

Ahora, conoceremos sobre el manejo de ramas con Git Flow.

Los temas que abordaremos son:

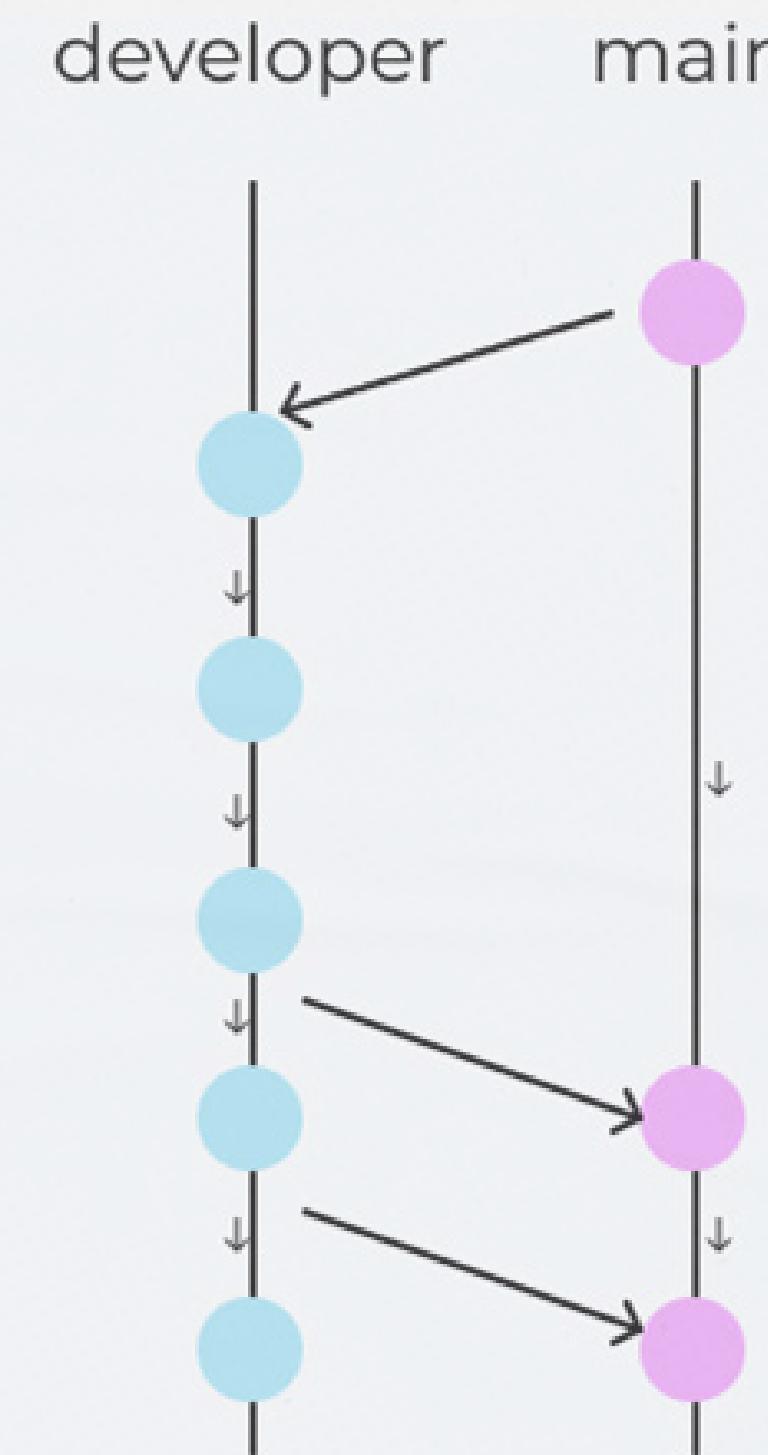
- ¿Qué es Git Flow?
- Tipo de ramas
- Interacción de ramas

Interacción de ramas

Es un modelo de ramificación para Git, creado por Vincent Driessen. Ha llamado mucho la atención, porque se adapta muy bien a la colaboración y al escalado del equipo de desarrollo.

Ramas principales

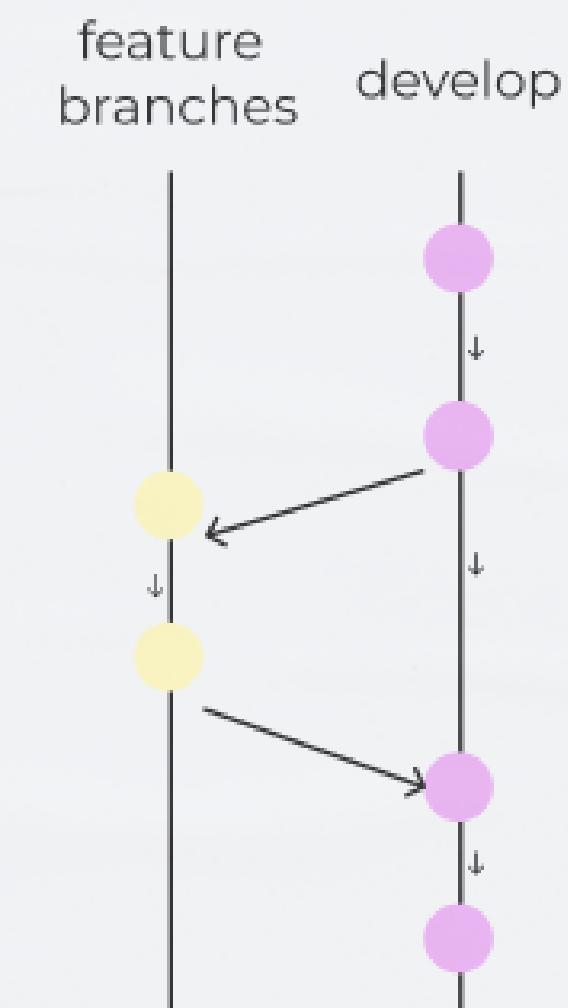
En esencia, el modelo de desarrollo está muy inspirado en los modelos existentes. El repositorio central contiene dos ramas principales con una vida útil infinita.



Tipo de ramas

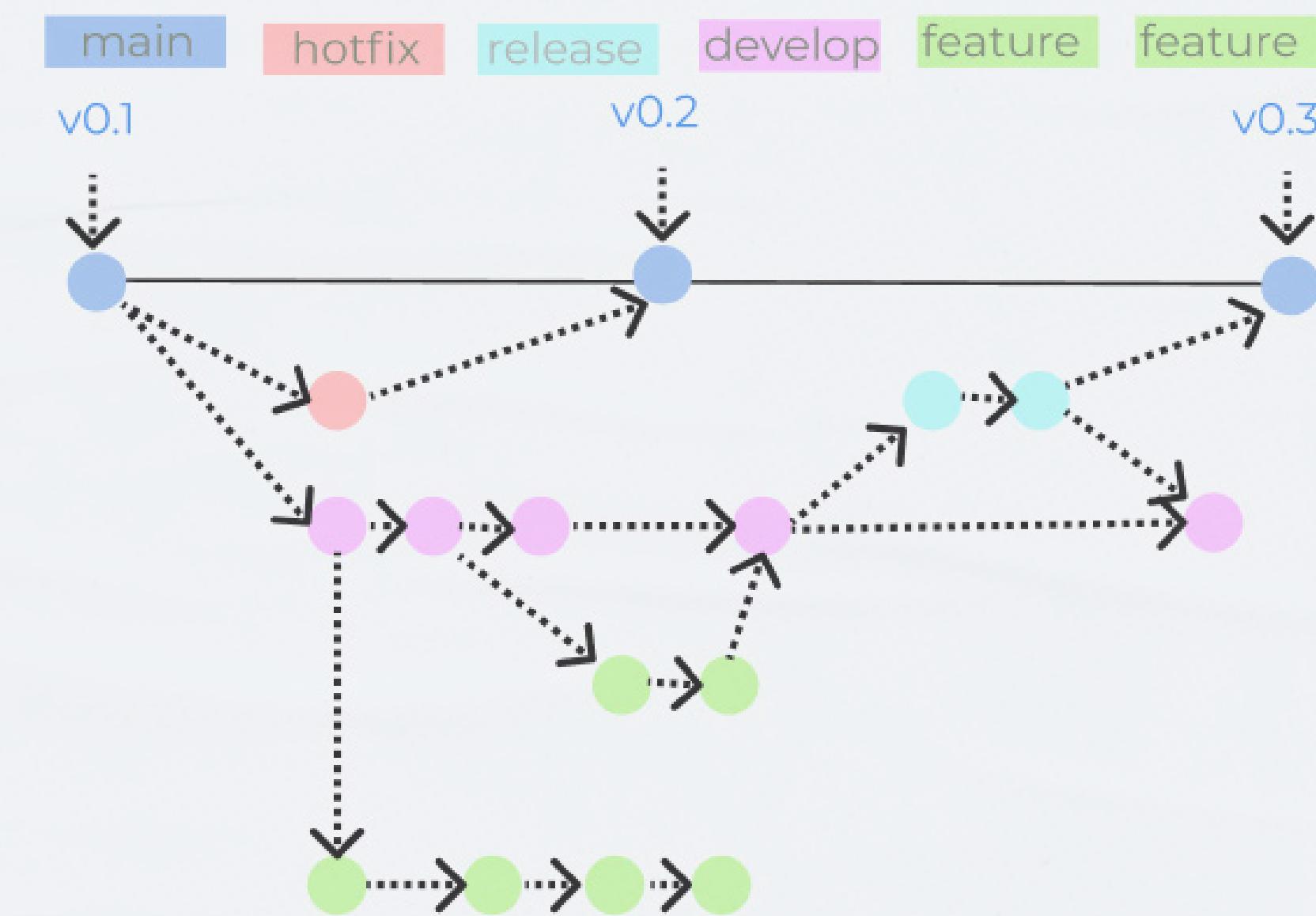
Rama de soporte

- Feature Branch
- Hotfix Branch
- Release Branch



Interacción de ramas

En el siguiente ejemplo, podemos ver cómo interactúan las ramas, generando versiones v0.1, v0.2, v0.3.





**“La práctica siempre debe ser
edificada sobre la buena teoría”**

Leonardo da Vinci

Gestión de proyectos con GitHub project

Ahora, entenderemos cómo gestionar
un proyecto con GitHub project.

Los temas que abordaremos son:

- ¿Cómo crear Issues?
- ¿Para qué sirven los Milestones?
- ¿Cómo utilizar el tablero?
- Cierre y despedida



Gestión de proyectos con GitHub project

Este tema será desarrollado a lo largo
de la explicación del videotutorial.

GG

**Las pequeñas oportunidades
son a menudo el inicio de
grandes proyectos**

JJ

Demóstenes

Ideas resumen del módulo 4

01

Usar un conventional commit nos permite trabajar de manera más coordinada con nuestro equipo y leer de una forma más detallada el historial de commits.

02

El uso de git flow nos ofrece una solución mucho más robusta para proyectos que van creciendo y necesitan de actualizaciones constantes. Funciona bien cuando contamos con muchas personas en el proyecto.

03

Utilizar GitHub projects nos permite gestionar nuestro proyecto, asignando roles a cada issue, agrupándolos en milestones y realizando el monitoreo de los issues con un tablero Kanban.

Bibliografía

- GitHub (2015). Git Cheat Sheet.
<https://training.github.com/downloads/github-git-cheat-sheet.pdf>

Enlaces de interés

- <https://github.com/>
- <https://bit.ly/3wyI0mN>
- <https://git-scm.com/>
- <https://pages.github.com>

NETZUN