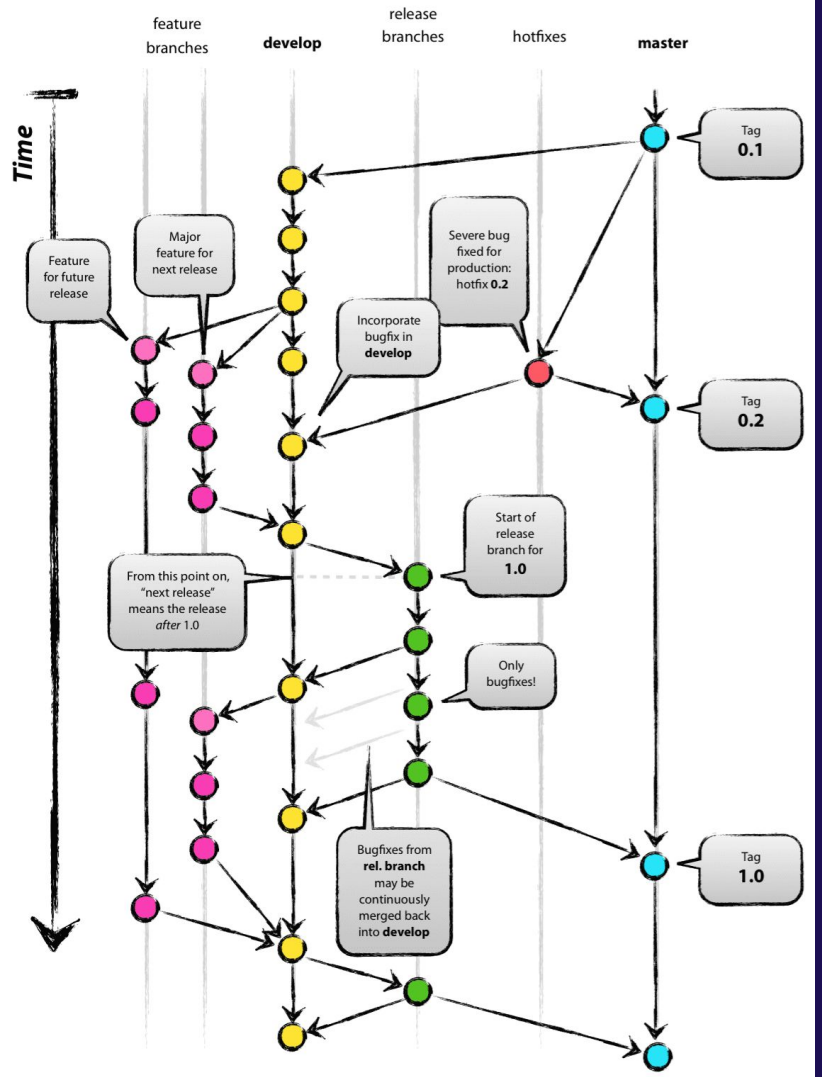


Control de versiones

El control de versiones es una práctica esencial en el desarrollo de software que permite gestionar y rastrear los cambios en el código fuente a lo largo del tiempo. Facilita la colaboración entre equipos, asegurando la integridad y la historia del proyecto al mantener un registro detallado de todas las modificaciones realizadas.

¿Qué es el control de versiones?

Definición

Un sistema de control de versiones (VCS) es un sistema informático que organiza, gestiona y almacena cambios en un sistema de archivos. Generalmente se utiliza para gestionar el desarrollo de programas informáticos.

Características

Un VCS almacena cada cambio declarado asignándole un identificador hash y ubicándolo en una secuencia. Esta secuencia puede dividirse en ramas que divergen de la secuencia principal y que posteriormente pueden converger de nuevo.

Objetivos

El objetivo de un VCS es mantener una secuencia de cambios reproducible, navegable y segura. De esta forma se puede ver qué cambios se han hecho, quién los ha realizado, revertir dichos cambios si es necesario y abrir nuevas ramas de desarrollo.

Tipos de VCS

Centralizados

Un sistema de control de versiones centralizado (CVCS) almacena los cambios en un servidor central. Los usuarios deben conectarse a este servidor para poder enviar los cambios y actualizar su código con los cambios de otros programadores.

Distribuidos

Un sistema de control de versiones distribuido (DVCS) permite a cada usuario tener una copia completa del repositorio, incluyendo todo su historial, en su máquina local. Esto permite trabajar sin conexión y realizar cambios y fusiones de manera eficiente antes de sincronizar con otros repositorios, mejorando la flexibilidad y la resiliencia del desarrollo colaborativo.

Semantic versioning

Este número representa la versión principal del software.
Cuando se incrementa, significa que la nueva versión sea incompatible con versiones anteriores del sistema.
Es lo que se conoce como un “breaking change”.

Se incrementa cuando se añade nueva funcionalidad al software sin reducir su compatibilidad con versiones anteriores.

Se incrementa cuando se reparan errores en el sistema que no rompen la compatibilidad con versiones anteriores.

https://semver.org/

Instalando git

https://git-scm.com/downloads

Estrategias de branching

Existen distintas estrategias para manejar el desarrollo de una aplicación de forma organizada, asegurando que el código sea desplegable, actualizable, y reversible.

Ahora veremos las estrategias más comunes:

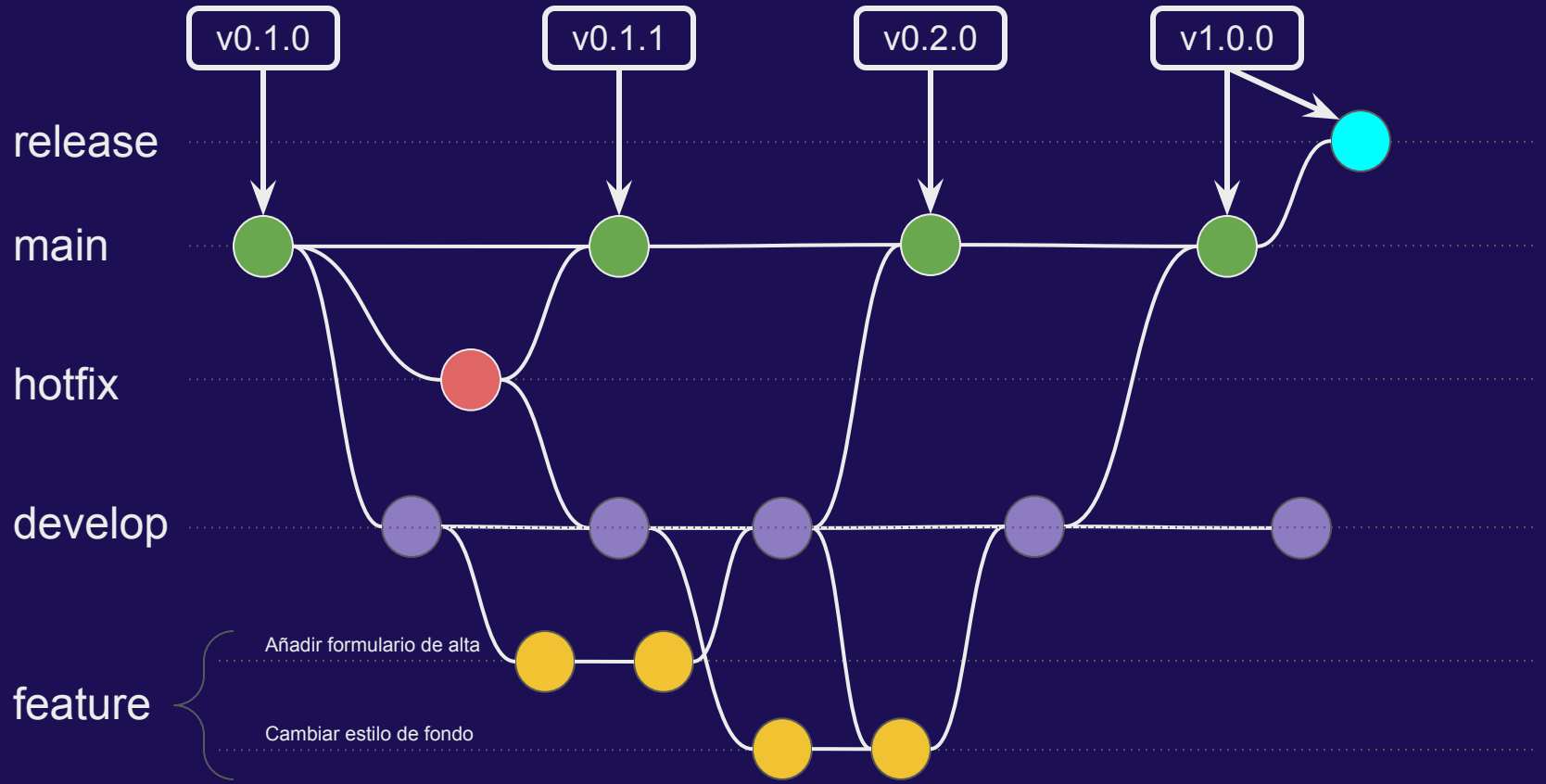
- Git flow
- GitHub flow

Git flow

Esta estrategia se centra en una mejor organización del desarrollo. En ella se distinguen cinco tipos de ramas:

- La rama principal (***main***): Se mantendrá durante todo el desarrollo. Contiene código que ya ha sido probado y está listo para desplegar en producción.
- La rama de desarrollo (***develop***): Se mantendrá durante todo el desarrollo. Cualquier nueva funcionalidad se mezclará en esta rama una vez haya terminado el desarrollo y esté libre para ser testada.
- Ramas de funcionalidad (***feature branches***): Se crean a partir de la rama ***develop***, una para cada funcionalidad que se está desarrollando. Cuando se termina el desarrollo, la rama se mezcla en develop y desaparece.
- Rama de publicación (***release***): Contiene el código que se ha publicado en producción.
- Ramas de reparación (***hotfix branches***): Son ramas que se abren de la rama ***main*** cuando es necesario hacer una reparación de urgencia. Cuando se termina el desarrollo, esta rama se debe mezclar tanto en ***main*** como en ***develop*** para asegurarnos de que el arreglo está en ambas ramas.

Git flow


v0.1.0
v0.1.1
v0.2.0
v1.0.0
release
main
hotfix
develop
feature
Añadir formulario de alta
Cambiar estilo de fondo

GitHub flow

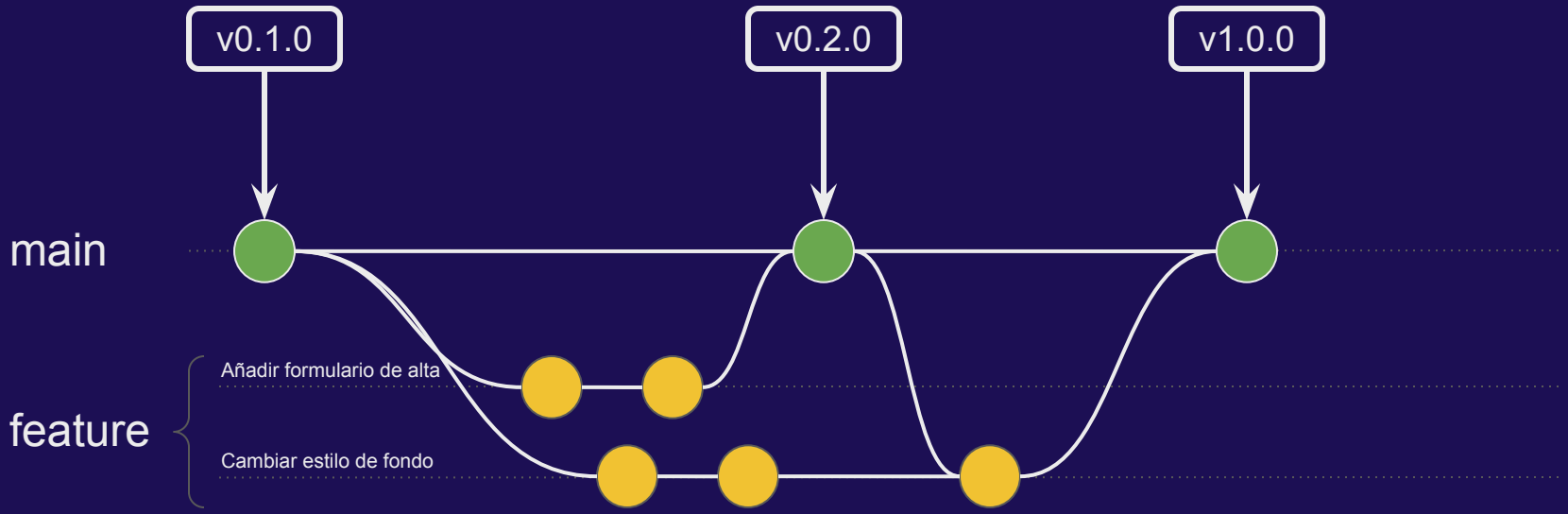
En esta estrategia es más sencilla y se centra en despliegue ágil a producción (también llamado integración contínua):

- La rama principal (***main***): Se mantendrá durante todo el desarrollo. Contiene código que ya ha sido probado y está listo para desplegar en producción.
- Ramas de funcionalidad (***feature branches***): Se crean a partir de la rama ***develop***, una para cada funcionalidad que se está desarrollando. Cuando se termina el desarrollo, la rama se mezcla en develop y desaparece.

Hay ciertos principios a seguir si se va a utilizar GitHub flow:

- El código en ***main*** debe ser siempre desplegable (debe estar testado e integrado).
- El código en las ***feature branches*** debe actualizarse regularmente.
- Una vez se mezcla en ***main***, se debe desplegar en producción lo antes posible.

GitHub flow


v0.1.0
v0.2.0
v1.0.0
main
feature
Añadir formulario de alta
Cambiar estilo de fondo