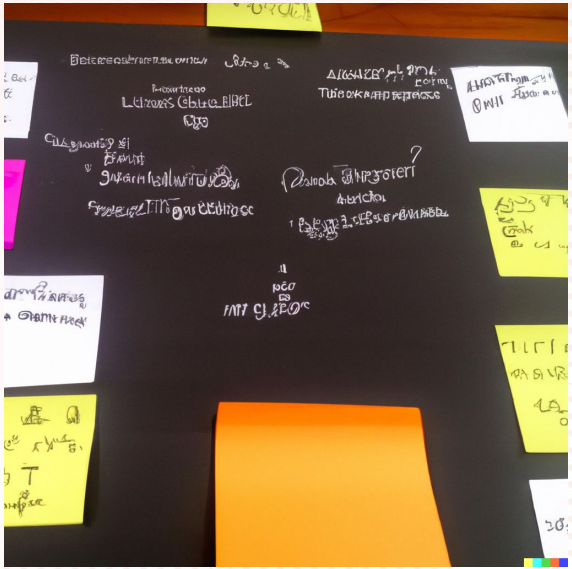
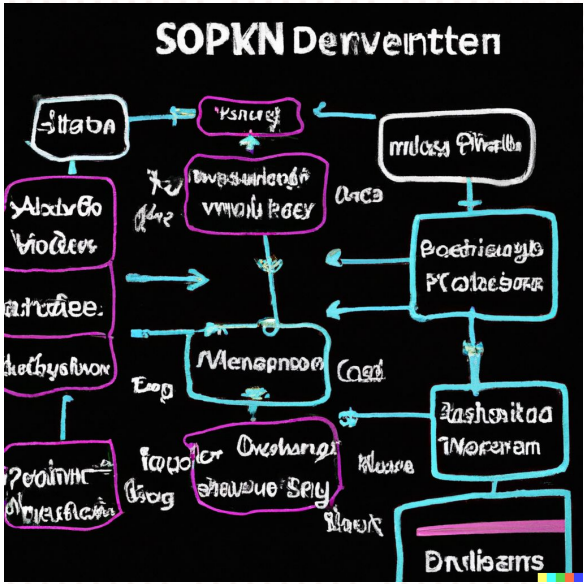
La importancia de la planificación de software

¿Por qué es importante la planeación?





Beneficios de la planificación

La planificación tiene varios beneficios, incluyendo:

- Identificación de riesgos y oportunidades
- Mayor claridad en los objetivos y requisitos.
- Establecimiento de un camino claro hacia el éxito.
- Mayor eficiencia en el uso de los recursos
- Mejor gestión del tiempo y los plazos
- Mayor colaboración y comunicación entre los miembros del equipo.

Herramientas de planificación

TAIGA

CUBE | Software for Controls Companies

Jira Software

Proceso de planificación

El proceso de planificación en proyectos de software puede variar dependiendo del proyecto, pero en general, se divide en varias etapas, que incluyen:

- Definición de los objetivos y requisitos del proyecto
- Estimación de historias de usuarios
- Priorización de historias de usuarios
- Cambiar el estado de las tareas
- Asignación de tareas

Estimación de Historias de usuario

Es el proceso de asignar una cantidad de tiempo o esfuerzo necesario para completar una historia de usuario específica.

En taiga, puedes utilizar puntos de historias para estimar la complejidad de una historia de usuario.

(Demo)

Priorización de historias de usuario

La priorización de historias de usuario es el proceso de terminar el orden en que las historias de usuario se abordarán en el proyecto.

En Taiga, pueden utilizar diferentes métodos para priorizar las historia de usuario, ejemplo:

(demo)

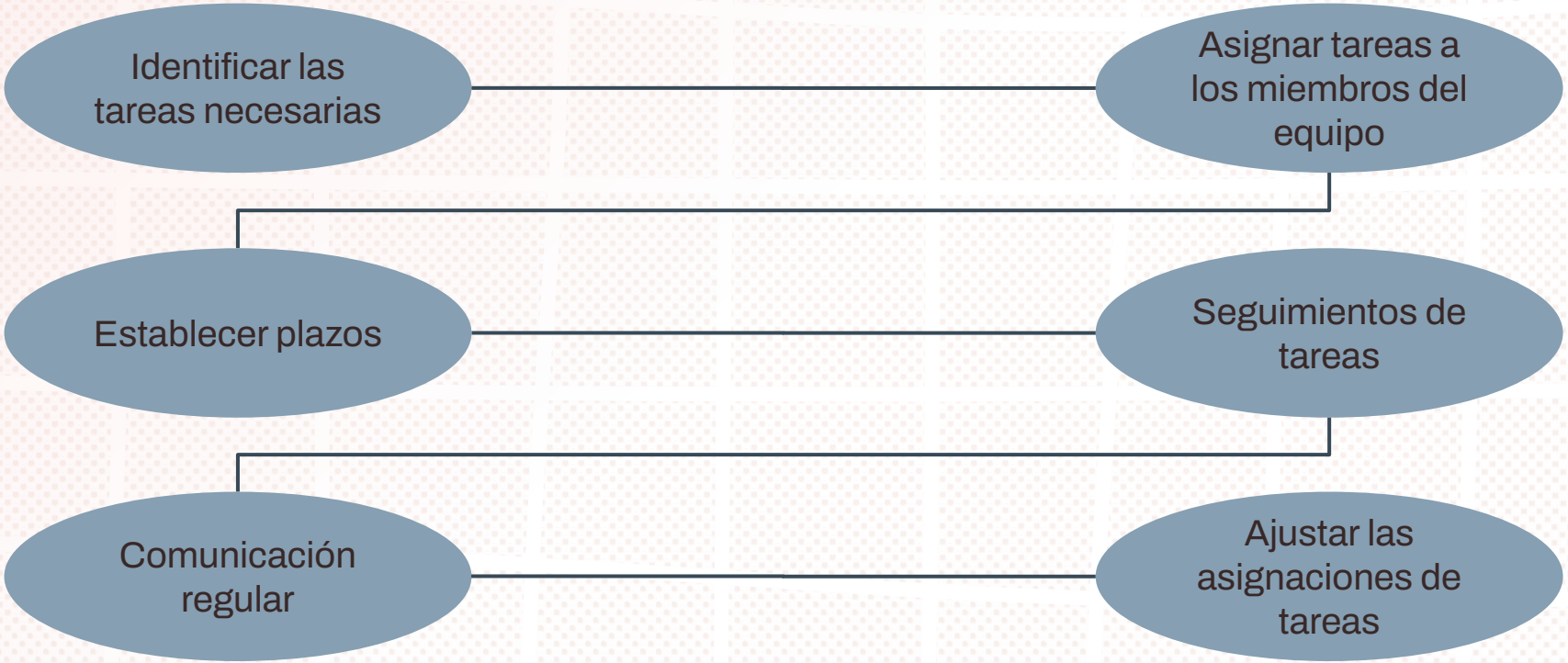
Cambiar el estado de las tareas

Las herramientas de gestión de proyectos permiten a los usuarios cambiar el estado de las tareas en función de su progreso. Los cambios de estado de las tareas en Taiga pueden incluir los siguientes:

- Nueva tarea
- En progreso
- En revisión
- Completado
- Cancelado
- Bloqueado

Estos son algunos de los cambios de estado de tarea comunes en Taiga, pero es importante destacar que los usuarios pueden personalizar los estados de las tareas para adaptarse a las necesidades de su proyecto específico.

La asignación de tareas



Proyectos de software que fallaron debido a la falta de planeación.

Healthcare.gov

El lanzamiento fallido de Healthcare.gov en 2013

tráfico aéreo de Londres

El lanzamiento del sistema de control de tráfico aéreo de Londres en 2002

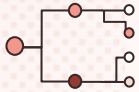
No Man's Sky

El lanzamiento del videojuego "No Man's Sky" en 2016

La planificación es clave para el éxito en proyectos de software

En conclusión, la planificación es clave para el éxito en proyectos de software. Nos ayuda a establecer objetivos claros, identificar riesgos y oportunidades, y mantenernos en el camino correcto a medida que avanzamos en el proyecto. La falta de planificación puede ser fatal para un proyecto, y es importante que dediquemos tiempo y recursos para planificar adecuadamente.

Estrategias de trabajo en equipo con git



Git Flow

Establecer un flujo de trabajo claro

Branch

Utilizar ramas

Pull requests

Realizar pull requests

Checkout

Realizar revisiones de código

Tag

Utilizar etiquetas (tags)

Mensajes de Commit

Documentar los cambios

Estrategia Git Flow

Gitflow es un modelo alternativo de creación de ramas en Git en el que se utilizan ramas de función y varias ramas principales.

- Es un modelo de desarrollo más estricto en el que solo determinadas personas pueden aprobar los cambios en el código principal.
- Se basa en tener dos ramas principales: master (principal) y develop (desarrollo), y varias ramas auxiliares: feature (funciones), release (publicación) y hotfix (corrección).
- Este flujo de trabajo ayuda a mantener la calidad del código y a minimizar el número de errores, pero también puede ser más complejo y lento.

Estrategia Git Trunk

Git Trunk más abierto, en el que todos los desarrolladores tienen acceso al código principal, llamado trunk o main.

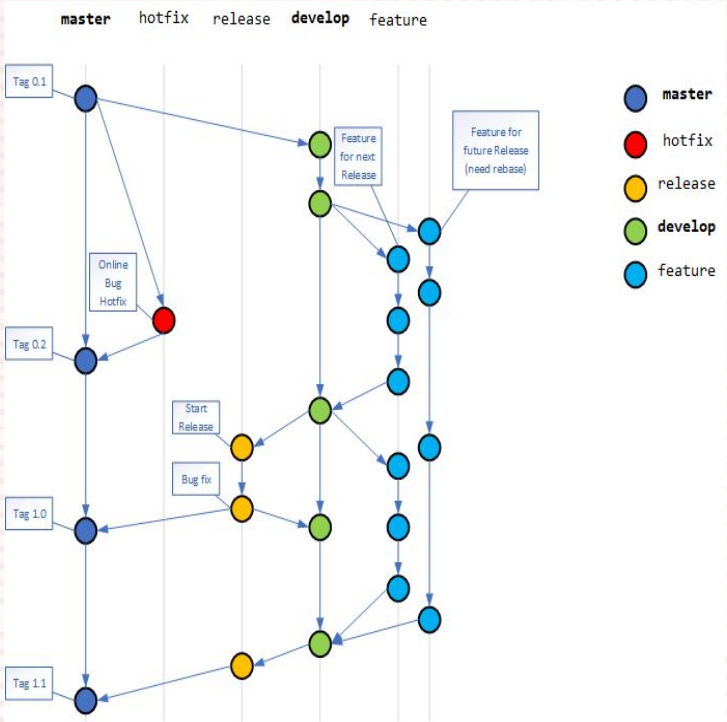
- Se basa en fusionar pequeñas actualizaciones de forma frecuente en el tronco, usando ramas cortas o efímeras para aislar los cambios.
- Este flujo de trabajo ayuda a lograr la integración continua y el despliegue continuo.
- Este flujo de trabajo ayuda a aumentar la entrega de software y el rendimiento de la organización.
- Este flujo de trabajo puede requerir más disciplina y comunicación entre los desarrolladores.

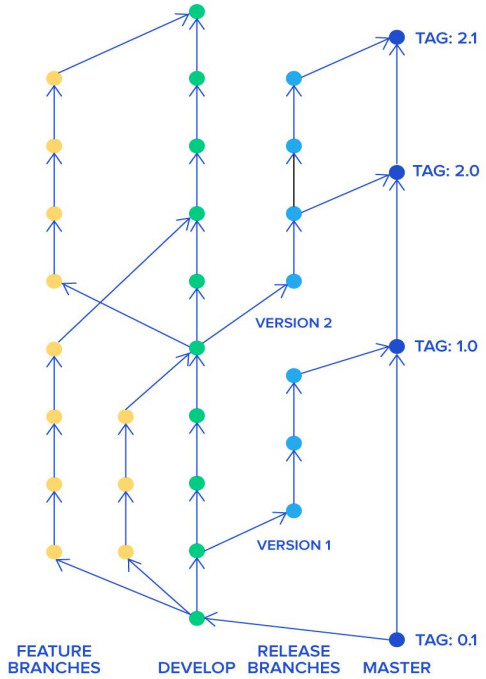
Estrategia Git Trunk

Prácticas recomendadas en el desarrollo basado en troncos (Git Trunk)

- Desarrollar en lotes pequeños.
- Marcas de función.
- Implementar pruebas automatizadas exhaustivas.
- Realizar revisiones de código asíncronas.
- Tener tres o menos ramas activas en el repositorio de código de la aplicación.
- Fusionar las ramas con el tronco al menos una vez al día.
- Reducir el número de bloqueos de código y fases de integración.
- Compilar rápido y ejecutar de inmediato.

Comparación Git Flow y Git Trunk





Conclusion Estrategia Git

El desarrollo basado en troncos (Trunk) es actualmente el estándar para los equipos de ingeniería de alto rendimiento, ya que establece y mantiene un ritmo de publicación de software mediante el uso de una estrategia de creación de ramas Git simplificada. Además, el desarrollo basado en troncos o Git Trunk, ofrece a los equipos de ingeniería más flexibilidad y control sobre la forma de lanzar el software para el usuario final. [1]

Referencias

- PRATIK SHINDE (2022). 8 Essential Steps for Planning a Software Development [Blog post] https://simpleprogrammer.com/software-development-project-planning/
- Anastasia Kushnir (2022). Software Development Planning: Why It Is Important [Blog post] https://bambooagile.eu/insights/software-development-planning/
- Alex Circei (2021). 9-Step Guide: How to Plan Effective Software Development Projects [Blog post] https://waydev.co/software-projects-planning/
- KALEIDOS VENTURES, S.L. Taiga: la herramienta de gestión de proyectos de software libre y gratuita. https://www.taiga.io/es
- Atlassian. (s/f) (2019). Flujo de trabajo de Gitflow. Atlassian. https://www.atlassian.com/es/git/tutorials/comparing-workflows/gitflow-workflow
- KEV ZETTLER (2021). Desarrollo basado en troncos. Atlassian. https://www.atlassian.com/es/continuous-delivery/continuous-integration/trunk-based-development