2024

Unidad 3: Tarea individual o grupal - Métricas -

UNIREMINGTON®
CORPORACIÓN UNIVERSITARIA REMINGTON
RES. 2661 MEN JUNIO 21 DE 1996

Brayhan Stiven Orrego Valencia
Jimmis Jhoan Simanca Rojas
Oscar Germanico Motta Riaño
Valentin Ducuara Leguizamo
Osvaldo Andres Peña
[Nombre de la compañía]
6-10-2024

Video de YouTube : https://youtu.be/WhHnXMvHVZ4

1. Investigar sobre 2 herramientas de GCS (Gestión de Configuración de Software) existentes y describa cómo implementan el control de versiones y de cambios.

Git

Git es un sistema de control de versiones distribuido, ampliamente utilizado en proyectos de software de cualquier tamaño. Su diseño está enfocado en la eficiencia y la flexibilidad para manejar ramas y fusiones de manera rápida.

Implementación del Control de Versiones:

- **Versiones Distribuidas**: Git almacena copias completas del repositorio en cada máquina de los desarrolladores. Esto significa que no depende de un servidor central, por lo que cada usuario tiene una "instantánea" completa del proyecto.
- **Commits**: Cada vez que un desarrollador realiza un cambio en el código y lo guarda (hace un commit), Git registra ese estado del proyecto. Cada commit contiene un identificador único (hash SHA-1) que se asocia con el cambio, lo que permite rastrear todas las modificaciones en el tiempo.
- **Ramas (branches)**: Git facilita el uso de ramas, permitiendo a los desarrolladores trabajar en características o correcciones de forma aislada. Las ramas pueden ser fusionadas con otras ramas, como la rama principal (main), una vez que los cambios han sido probados y validados.

Implementación del Control de Cambios:

- **Git Flow**: Un flujo de trabajo común en Git implica el uso de ramas para el desarrollo de nuevas características y la corrección de errores. Las ramas suelen tener nombres como **feature/**, **bugfix/**, y **release/**, lo que permite organizar y gestionar los cambios.
- **Pull Requests**: En plataformas como GitHub o GitLab, los desarrolladores pueden proponer cambios mediante "pull requests". Estas permiten a los equipos de desarrollo revisar y discutir los cambios antes de fusionarlos en la rama principal.
- **Historial y Revertir Cambios**: Git permite ver el historial completo de todos los cambios realizados. Si un cambio es problemático, los desarrolladores pueden revertir un commit específico o restaurar el código a un estado anterior.

2. Apache Subversion (SVN)

SVN es un sistema de control de versiones centralizado, diseñado para rastrear los cambios en archivos y directorios de manera ordenada. Aunque ha sido superado por Git en muchos aspectos, sigue siendo popular en entornos donde se requiere un modelo más centralizado.

Implementación del Control de Versiones:

- **Repositorio Central**: A diferencia de Git, SVN utiliza un único repositorio central donde se almacena todo el código. Los desarrolladores deben hacer "checkouts" de los archivos o directorios del repositorio a sus máquinas locales.
- **Revisiones (revisions)**: SVN asigna un número de revisión secuencial a cada cambio realizado en el repositorio. Cada vez que un desarrollador hace un "commit", el número de revisión del repositorio aumenta, y todos los archivos en el repositorio quedan asociados con ese número de revisión.
- **Ramas y Etiquetas (tags)**: SVN admite la creación de ramas y etiquetas. Las ramas permiten desarrollar características o correcciones por separado, mientras que las etiquetas se utilizan comúnmente para marcar versiones específicas del software.

Implementación del Control de Cambios:

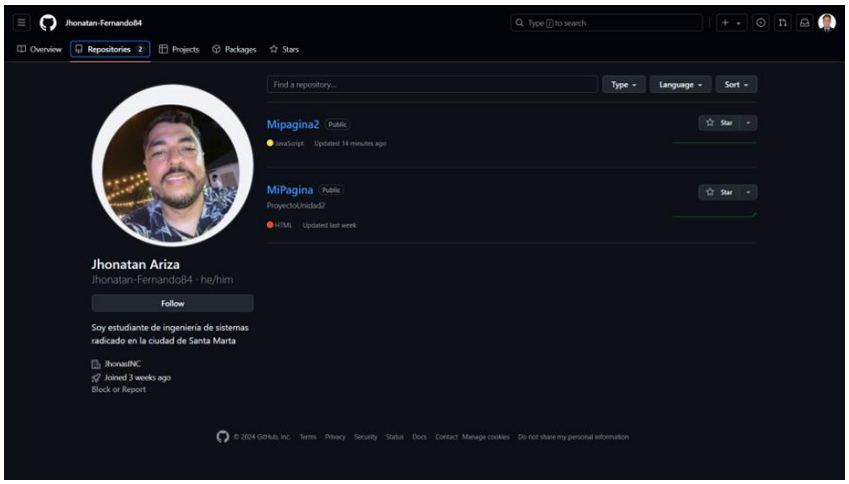
- **Commit Centralizado**: Los desarrolladores deben enviar sus cambios al repositorio central para que queden registrados. Esto permite que todos los cambios sean visibles y accesibles para los demás miembros del equipo.
- **Conflictos y Fusiones**: SVN también tiene capacidades para gestionar conflictos cuando varios desarrolladores trabajan en los mismos archivos. Si dos desarrolladores cambian el mismo archivo, SVN intentará fusionar los cambios automáticamente. Si no puede hacerlo, marca un conflicto que debe ser resuelto manualmente.
- **Historial y Comparación de Cambios**: SVN mantiene un historial completo de todos los cambios realizados en los archivos del repositorio. Los desarrolladores pueden comparar diferentes versiones de un archivo para ver qué ha cambiado.

Comparación Git vs. SVN

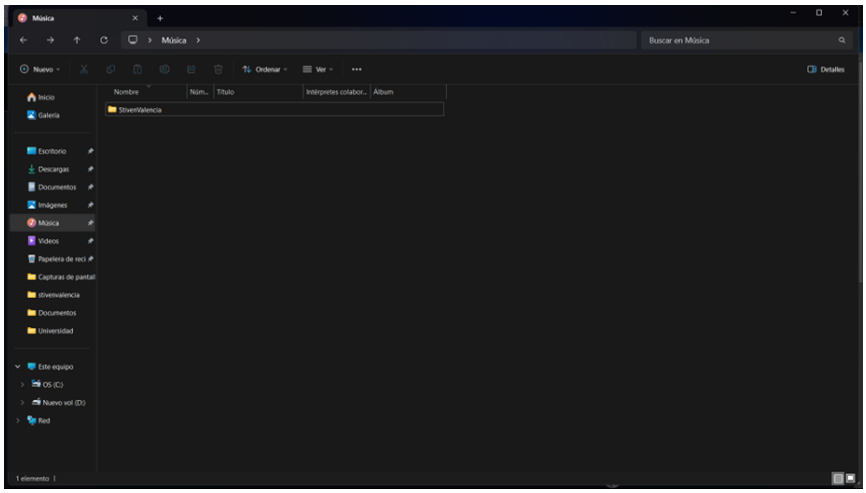
- **Modelo de trabajo**: Git es distribuido, lo que significa que cada desarrollador tiene una copia completa del repositorio, mientras que SVN es centralizado.
- **Rendimiento**: Git suele ser más rápido que SVN para operaciones como commits y branch switching debido a su arquitectura distribuida.
- **Escalabilidad**: Git maneja mejor los proyectos grandes con múltiples desarrolladores y ramas activas.
- **Facilidad de uso**: SVN puede ser más fácil de entender para usuarios que prefieren un control centralizado, mientras que Git puede requerir un aprendizaje inicial más amplio, aunque ofrece mayor flexibilidad.

En general, ambas herramientas ofrecen una sólida capacidad para gestionar versiones y cambios, pero Git se ha convertido en el estándar en la industria del desarrollo de software debido a su capacidad de manejo distribuido y sus flujos de trabajo más avanzados.

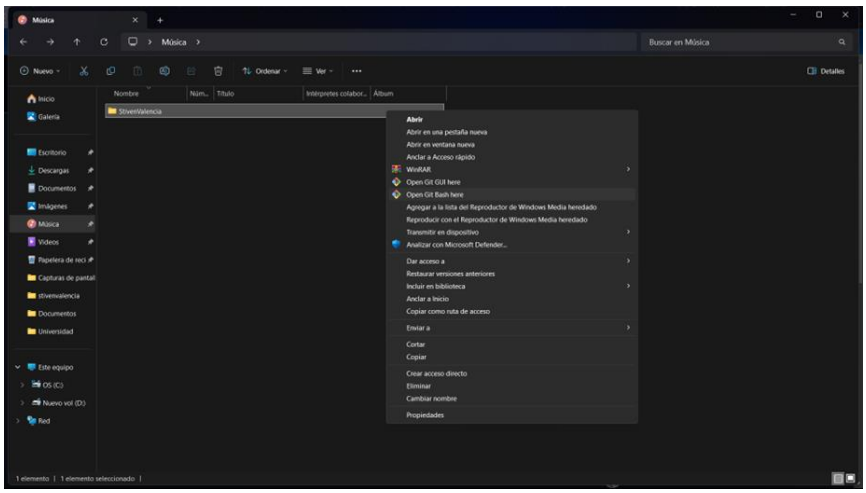
2. Realice un ejemplo práctico de cómo se realiza el control de versiones implementando Git y GitHub.



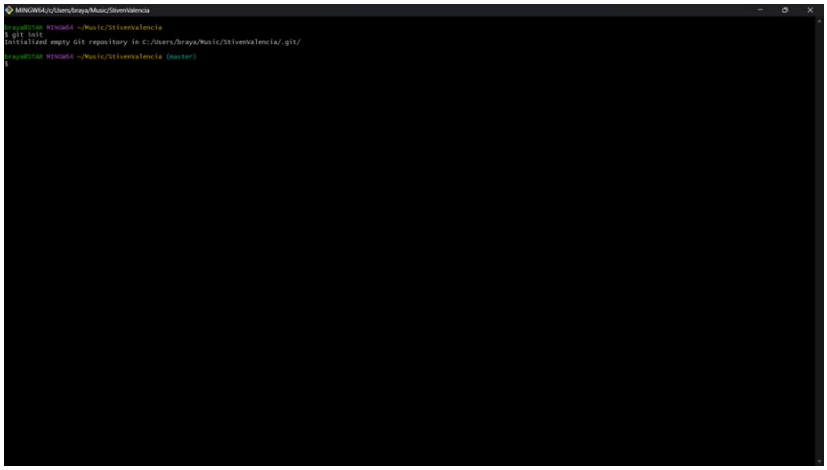
Primero se entra al repositorio que se quiere participar:



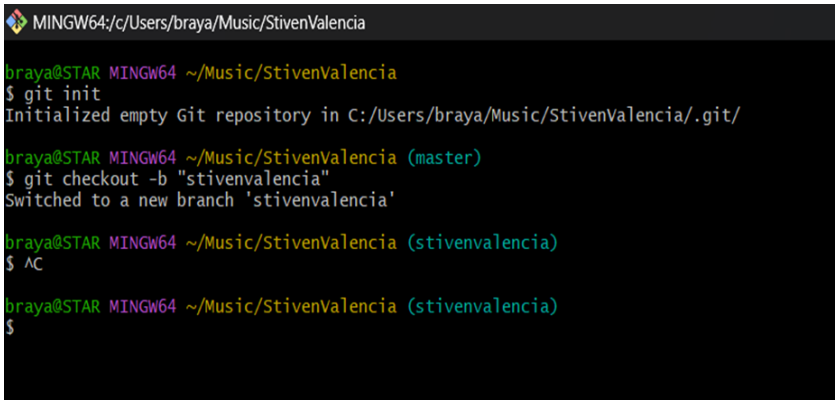
Creamos una carpeta vacía para descargar el repositorio



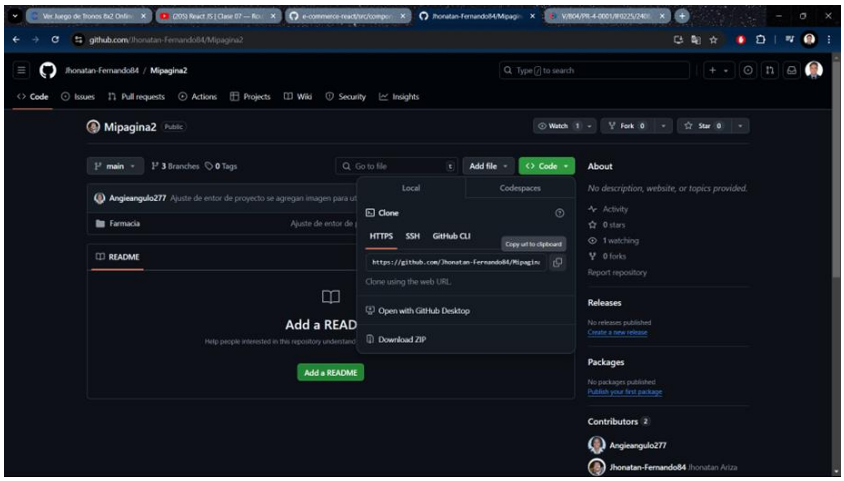
Iniciamos git en la carpeta creada



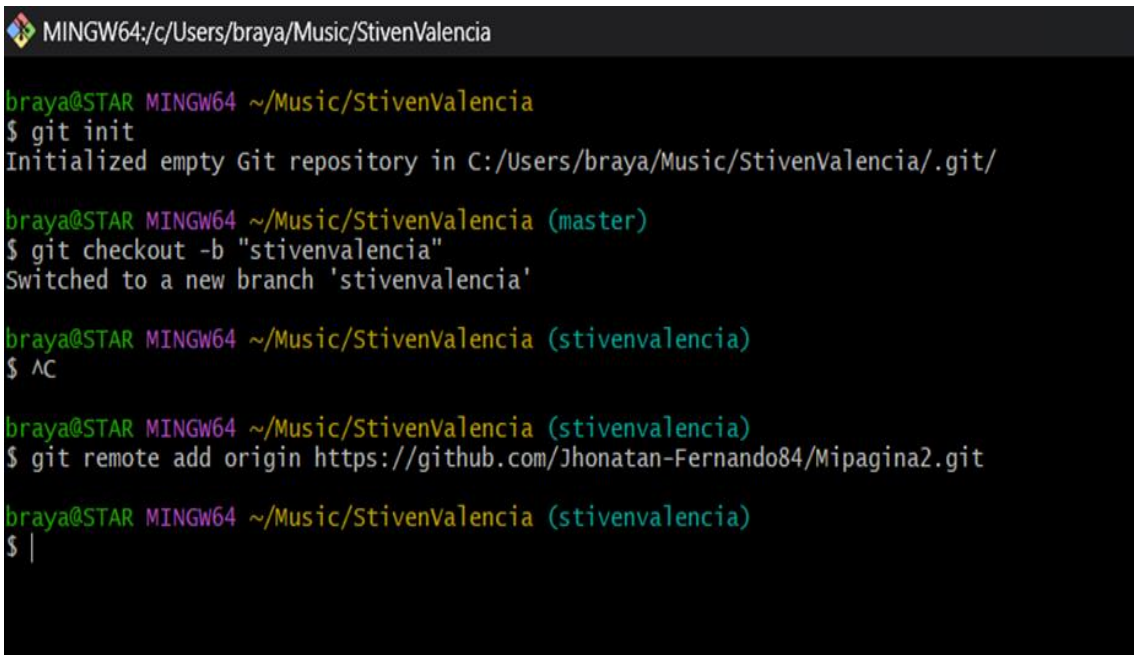
Con el comando git init iniciamos git en esta carpeta



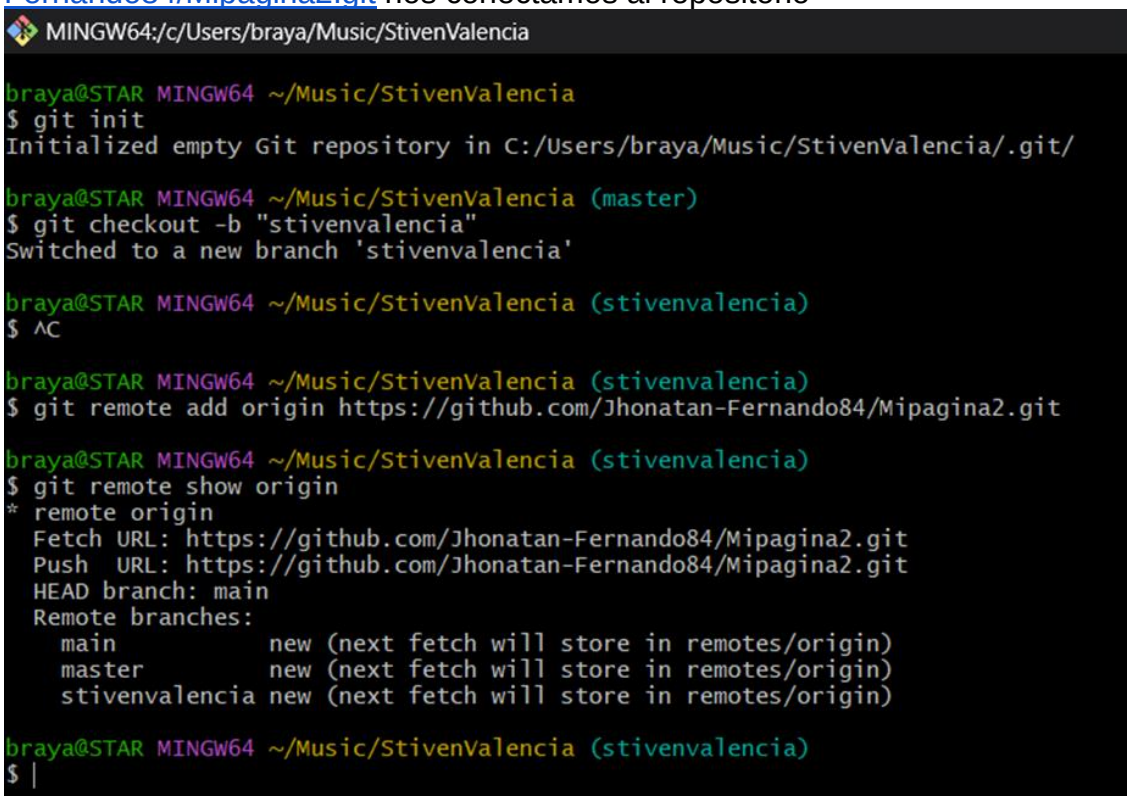
Con el comando git checkout -b "stivenvalencia" nos conectamos a la rama Stiven valencia y en caso de no existir la creamos con este mismo



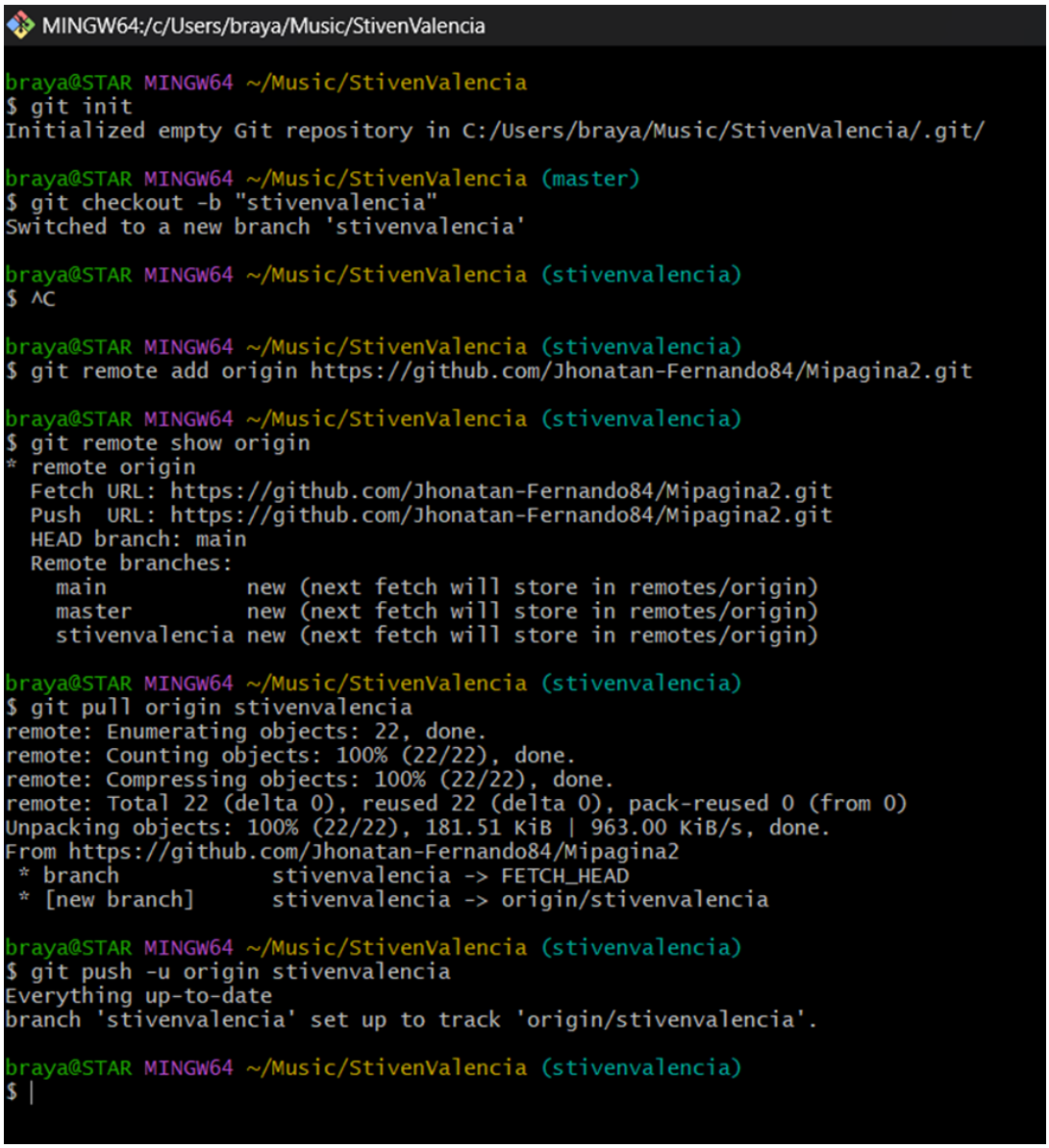
Copiamos el vínculo del repositorio que queremos participar



Con el comando git remote add origin https://github.com/Jhonatan-Fernando84/Mipagina2.git nos conectamos al repositorio



El comando git remote show origin permite ver más información sobre el remoto origen en Git



Con el comando git pull origin stivenvalencia descargamos en nuestra carpeta local el repositorio remoto

```
braya@STAR MINGW64 ~/Music/StivenValencia (stivenvalencia)
$ git pull origin stivenvalencia
remote: Enumerating objects: 22, done.
remote: Counting objects: 100% (22/22), done.
remote: Compressing objects: 100% (22/22), done.
remote: Total 22 (delta 0), reused 22 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (22/22), 181.51 KiB | 963.00 KiB/s, done.
From https://github.com/Jhonatan-Fernando84/Mipagina2
 * branch            stivenvalencia -> FETCH_HEAD
 * [new branch]      stivenvalencia -> origin/stivenvalencia

braya@STAR MINGW64 ~/Music/StivenValencia (stivenvalencia)
$ git push -u origin stivenvalencia
Everything up-to-date
branch 'stivenvalencia' set up to track 'origin/stivenvalencia'.

braya@STAR MINGW64 ~/Music/StivenValencia (stivenvalencia)
$ git commit -m "Cambios realizados prueba"
On branch stivenvalencia
Your branch is up to date with 'origin/stivenvalencia'.

nothing to commit, working tree clean
```

Con el comando git push -u origin stivenvalencia subimos los cambios realizados

```
MINGW64:/c/Users/braya/Music/StivenValencia

braya@STAR MINGW64 ~/Music/StivenValencia
$ git init
Initialized empty Git repository in C:/Users/braya/Music/StivenValencia/.git/

braya@STAR MINGW64 ~/Music/StivenValencia (master)
$ git checkout -b "stivenvalencia"
Switched to a new branch 'stivenvalencia'

braya@STAR MINGW64 ~/Music/StivenValencia (stivenvalencia)
$ ^C

braya@STAR MINGW64 ~/Music/StivenValencia (stivenvalencia)
$ git remote add origin https://github.com/Jhonatan-Fernando84/Mipagina2.git

braya@STAR MINGW64 ~/Music/StivenValencia (stivenvalencia)
$ git remote show origin
* remote origin
  Fetch URL: https://github.com/Jhonatan-Fernando84/Mipagina2.git
  Push  URL: https://github.com/Jhonatan-Fernando84/Mipagina2.git
  HEAD branch: main
  Remote branches:
    main           new (next fetch will store in remotes/origin)
    master         new (next fetch will store in remotes/origin)
    stivenvalencia new (next fetch will store in remotes/origin)

braya@STAR MINGW64 ~/Music/StivenValencia (stivenvalencia)
$ git pull origin stivenvalencia
remote: Enumerating objects: 22, done.
remote: Counting objects: 100% (22/22), done.
remote: Compressing objects: 100% (22/22), done.
remote: Total 22 (delta 0), reused 22 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (22/22), 181.51 KiB | 963.00 KiB/s, done.
From https://github.com/Jhonatan-Fernando84/Mipagina2
 * branch            stivenvalencia -> FETCH_HEAD
 * [new branch]      stivenvalencia -> origin/stivenvalencia

braya@STAR MINGW64 ~/Music/StivenValencia (stivenvalencia)
$ git push -u origin stivenvalencia
Everything up-to-date
branch 'stivenvalencia' set up to track 'origin/stivenvalencia'.

braya@STAR MINGW64 ~/Music/StivenValencia (stivenvalencia)
$ git commit -m "Cambios realizados prueba"
On branch stivenvalencia
Your branch is up to date with 'origin/stivenvalencia'.

nothing to commit, working tree clean

braya@STAR MINGW64 ~/Music/StivenValencia (stivenvalencia)
$ ^C

braya@STAR MINGW64 ~/Music/StivenValencia (stivenvalencia)
$ git push -u origin stivenvalencia
Everything up-to-date
branch 'stivenvalencia' set up to track 'origin/stivenvalencia'.

braya@STAR MINGW64 ~/Music/StivenValencia (stivenvalencia)
$
```

Con el comando git commit -m "Cambios realizados prueba" hacemos un comentario corto pero detallado de que cambios se realizaron

Con el comando git push -u origin stivenvalencia subimos nuestros cambios y commit al repositorio remoto para ser aprobado o rechazado por el dueño del repositorio

Bonus:

Es común cometer errores al subir cambios a la rama main en GitHub. Git ofrece varias herramientas para revertir estos cambios y mantener tu historial limpio.

Para estos casos tenemos los siguientes comandos

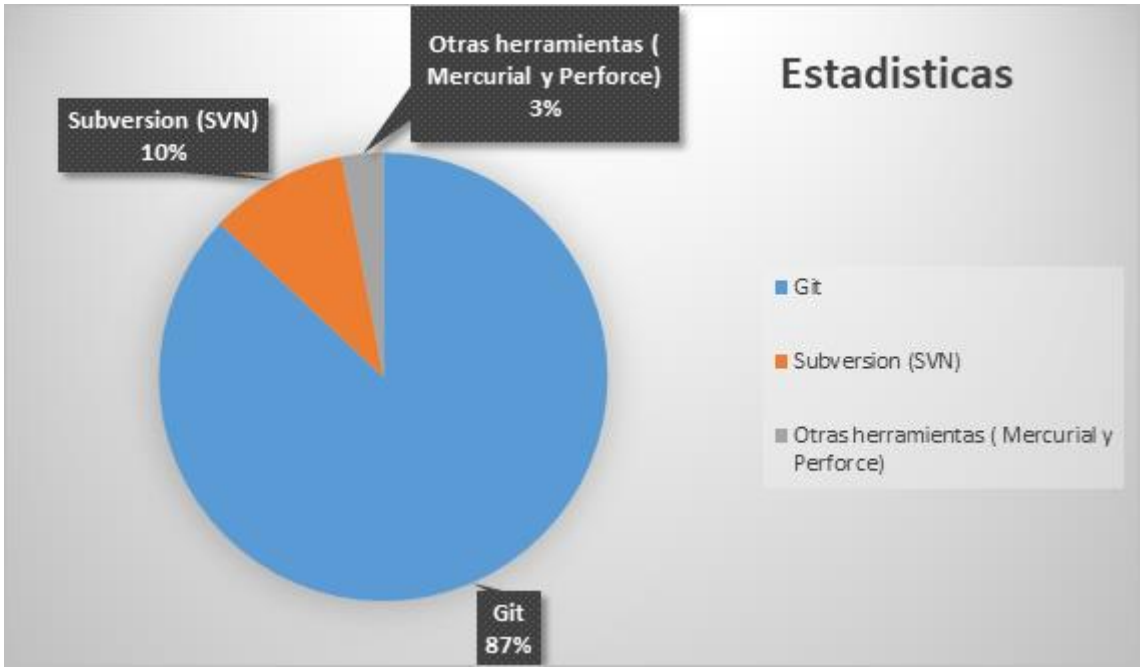
git revert: Ideal para deshacer cambios específicos y mantener un historial limpio.

git reset: Útil para retroceder varias confirmaciones o eliminar cambios no deseados.

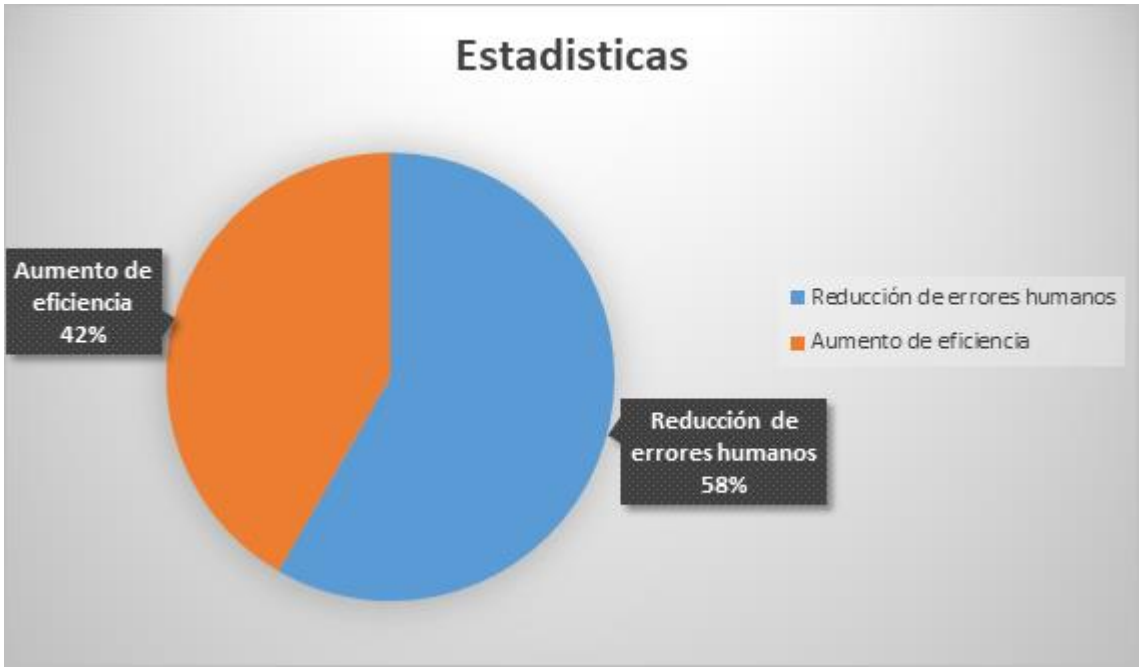
git checkout: Sirve para cambiar de rama o restaurar archivos individuales.

3. Presentar estadísticas relevantes sobre las herramientas de Gestión de Configuración de Software. Utilizar imágenes que representen las estadísticas.

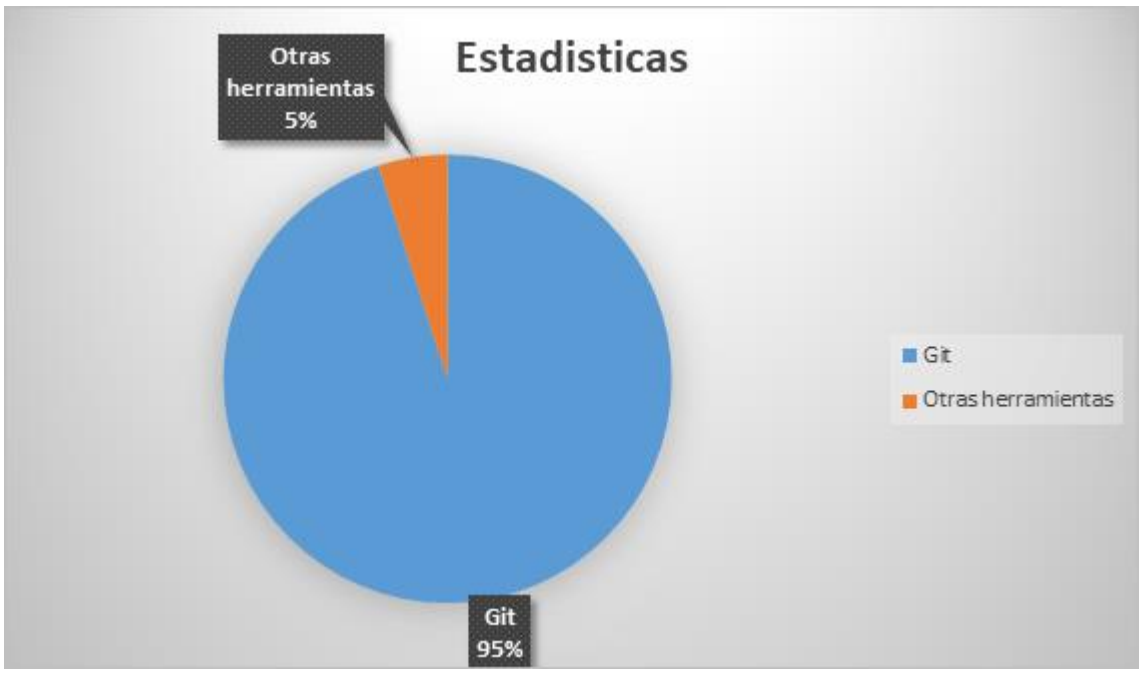
Popularidad de las Herramientas de SCM (2023):



Ahorro de Tiempo:



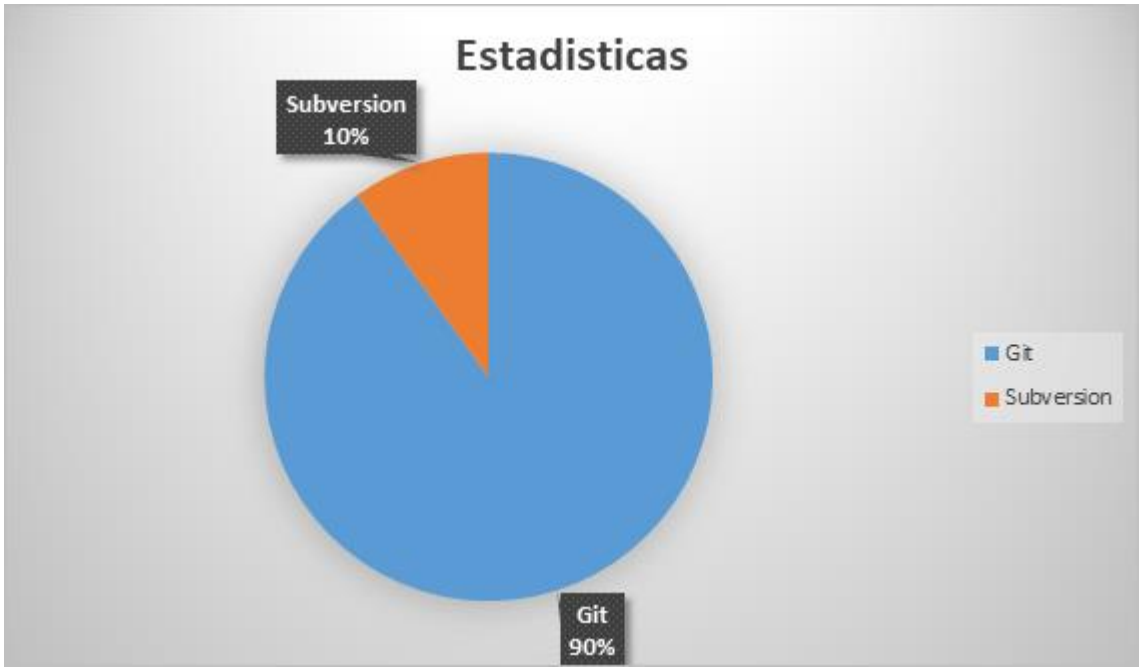
Preferencias por Sectores:



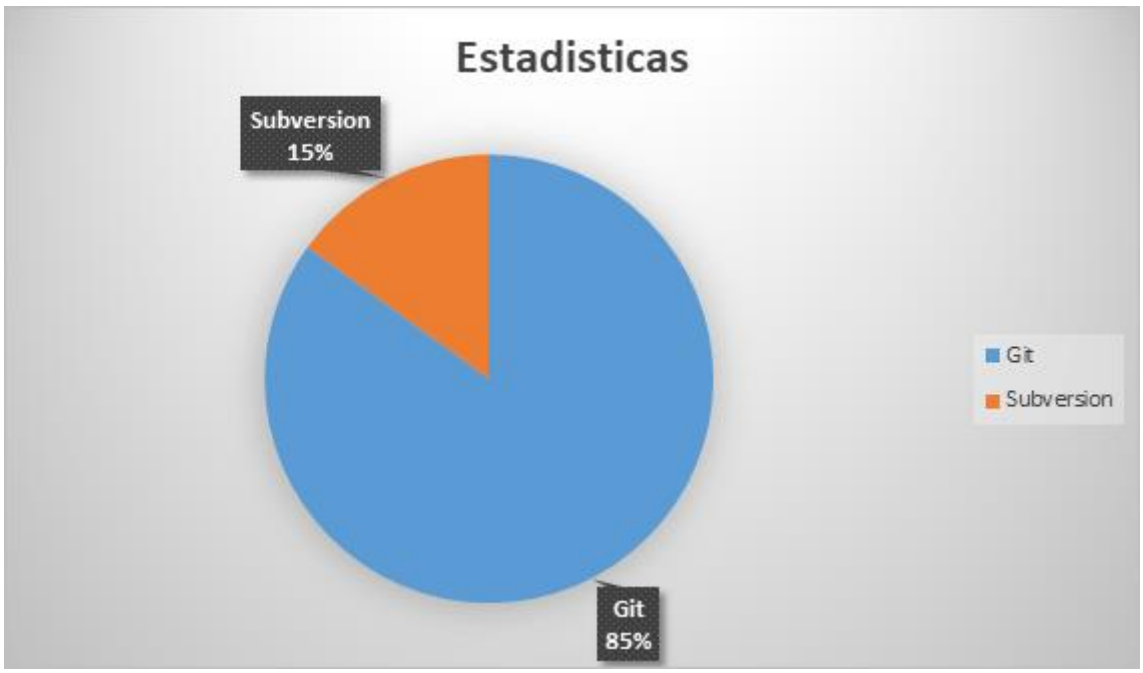
Este grafico aplica para empresas que estas empezando y empresas grandes.

Distribución Geográfica del Uso de SCM:

América del Norte y Europa

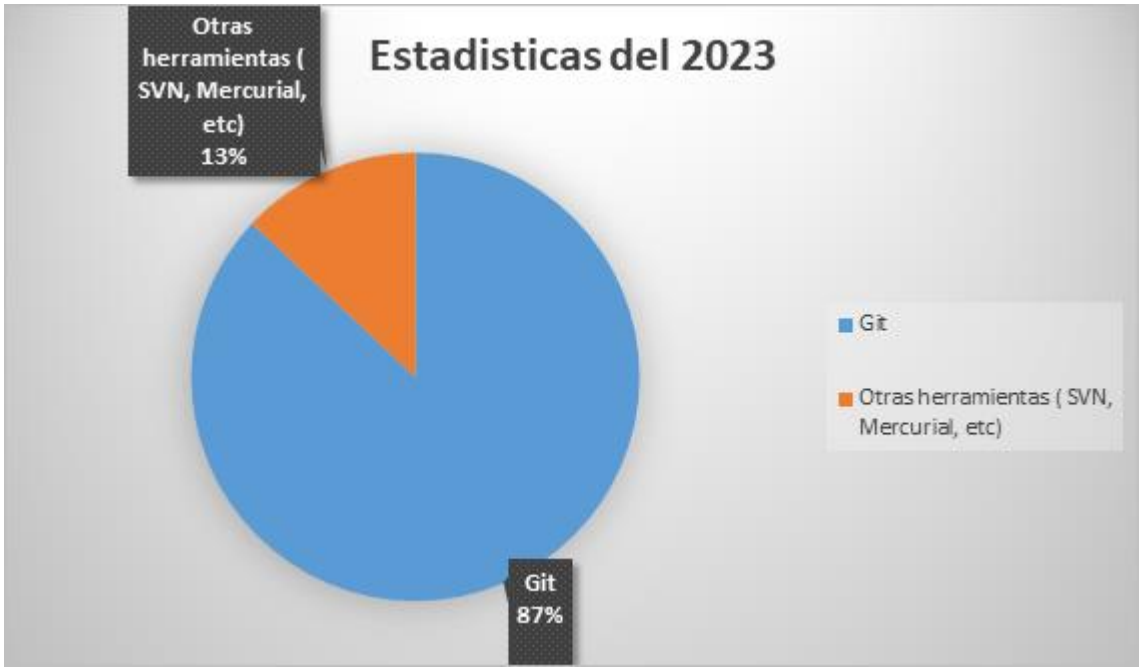


Asia y Latinoamérica



Crecimiento del Uso de Git (2010-2023)


Estadisticas del 2010
Git
30%
Otras herramientas (SVN, Mercurial, etc)
70%
Git
Otras herramientas (SVN, Mercurial, etc)


Otras herramientas (SVN, Mercurial, etc)
13%
Estadisticas del 2023
Git
87%
Git
Otras herramientas (SVN, Mercurial, etc)

4. ¿Cuáles son los beneficios de realizar control de versiones a través de estas herramientas?

El control de versiones, ya sea con herramientas como Git o Subversion (SVN), proporciona numerosos beneficios en el desarrollo de software, especialmente en entornos colaborativos y proyectos complejos. Como, por ejemplo:

1. Historial Completo de Cambios

- Registro detallado: Cada cambio realizado en el código, ya sea una corrección menor o una nueva funcionalidad, queda registrado en el sistema. Esto permite a los desarrolladores rastrear quién realizó el cambio, cuándo, y por qué, a través de mensajes de commit que documentan cada modificación.
- Auditoría y rastreo: En caso de errores o bugs, es fácil identificar cuándo y cómo se introdujo el problema y revertir los cambios si es necesario. Esto facilita auditorías internas y la revisión del código.

2. Recuperación de Versiones Anteriores

- Reversión de cambios: Si alguna versión del código introduce errores, las herramientas de control de versiones permiten revertir el software a un estado anterior en cualquier momento. Esto es útil para solucionar rápidamente problemas críticos que pueden surgir tras actualizaciones o fusiones.
- Acceso a versiones históricas: El sistema permite trabajar con diferentes versiones del código, facilitando el acceso a versiones anteriores para revisiones, pruebas o comparación de rendimiento.

3. Facilita el Trabajo Colaborativo

- Desarrollo simultáneo: Varias personas pueden trabajar en el mismo proyecto, incluso en los mismos archivos, sin sobrescribir el trabajo de otros. Git y SVN gestionan automáticamente las fusiones de cambios y resaltan conflictos, permitiendo la resolución colaborativa.
- Ramas y fusiones: Las ramas permiten a los desarrolladores trabajar en características o correcciones de errores por separado sin afectar el código principal (o rama principal). Las fusiones permiten integrar esos cambios al proyecto cuando están listos.

4. Mejora la Organización y Estructura del Proyecto

- Manejo de versiones y lanzamientos: Las herramientas como Git y SVN permiten etiquetar versiones específicas del software, como "v1.0", "v2.0", lo cual es esencial para el lanzamiento de versiones estables y mantenimiento de actualizaciones o parches.
- Flujos de trabajo estandarizados: Herramientas como Git fomentan el uso de flujos de trabajo estandarizados, como GitFlow, donde las ramas se

organizan para el desarrollo, pruebas y lanzamientos. Esto mejora la organización del ciclo de vida del software.

5. Mejora la Calidad del Código

- Revisión de código: Con la ayuda de pull requests (en Git) o revisiones de cambios (en SVN), es posible realizar revisiones de código antes de que los cambios sean integrados en el proyecto principal. Esto mejora la calidad del código, ya que otros desarrolladores pueden identificar errores o sugerir mejoras.
- Facilita las pruebas: La creación de ramas permite probar nuevas características o cambios en un entorno aislado sin interrumpir la funcionalidad del proyecto principal. Esto asegura que los cambios son estables y seguros antes de ser desplegados.

6. Aumenta la Seguridad y Redundancia

- Copia de seguridad automática: El control de versiones actúa como una copia de seguridad del código. Si un desarrollador pierde su máquina o se daña, el código puede recuperarse desde el repositorio central (en SVN) o desde cualquier otra copia local del equipo (en Git).
- Acceso controlado: Estas herramientas permiten establecer permisos para que solo ciertos usuarios puedan realizar ciertas acciones, como crear ramas, fusionar cambios o hacer commits, mejorando la seguridad del proyecto.

7. Optimización de Flujo de Trabajo y Automatización

- Automatización con CI/CD: En entornos modernos de DevOps, el control de versiones se integra con herramientas de integración continua (CI) y entrega continua (CD), automatizando pruebas, compilaciones y despliegues. Esto agiliza los procesos de desarrollo y despliegue.
- Colaboración en la nube: Herramientas como Git se integran con plataformas en la nube (GitHub, GitLab, Bitbucket), lo que facilita la colaboración entre equipos distribuidos geográficamente.

8. Facilidad en la Resolución de Conflictos

- Manejo de conflictos en fusión: Cuando dos desarrolladores modifican el mismo archivo simultáneamente, el control de versiones detecta los conflictos y permite resolverlos de forma colaborativa. Git y SVN proporcionan herramientas que ayudan a combinar cambios o resolver conflictos de forma manual si es necesario.

9. Soporte Multiplataforma

- Adaptabilidad a diferentes lenguajes y entornos: Tanto Git como SVN son agnósticos al lenguaje de programación o entorno de desarrollo utilizado, por lo que pueden usarse en proyectos de cualquier tipo, desde pequeños scripts hasta grandes aplicaciones empresariales.

Conclusión

El control de versiones, a través de herramientas como **Git** y **SVN**, es fundamental para garantizar la integridad, colaboración y eficiencia en el desarrollo de software. Proporciona un historial completo de cambios, permite trabajar de manera colaborativa y segura, mejora la calidad del código mediante revisiones y pruebas, y facilita la recuperación ante errores. Estas herramientas son indispensables para gestionar proyectos de software de cualquier tamaño de manera organizada y eficiente.

5. Investigar 5 técnicas o métodos que permitan identificar riesgos en los proyectos de software.

En el mundo del desarrollo de software, la gestión de riesgos ha tomado mucha popularidad en los últimos años, según (Torres, 2022). Esto se debe a que algunas grandes compañías han presentado pérdidas económicas y reputacionales significativas por no haber realizado una buena gestión de riesgos en sus proyectos, lo cual, a largo plazo, provoca este tipo de incidentes. De aquí la importancia de llevar a cabo una adecuada gestión de riesgos para evitar estos problemas. Existen varias técnicas o métodos que ayudan a identificar riesgos futuros en el proyecto o incluso a encontrar riesgos ya presentes de manera más eficiente. A continuación, se presentan algunas de ellas:

1.Análisis de Causa-Raíz: Esta técnica se utiliza habitualmente para analizar la raíz del problema, riesgo o fallo en los proyectos. Se emplean diagramas como el de Ishikawa o espina de pescado, el diagrama de Pareto o la técnica de los 5 porqués. Se caracteriza por no solo identificar los riesgos, sino también las causas del problema. Sin embargo, requiere tiempo y esfuerzo para analizar cada situación en detalle (Borsalli, 2022).

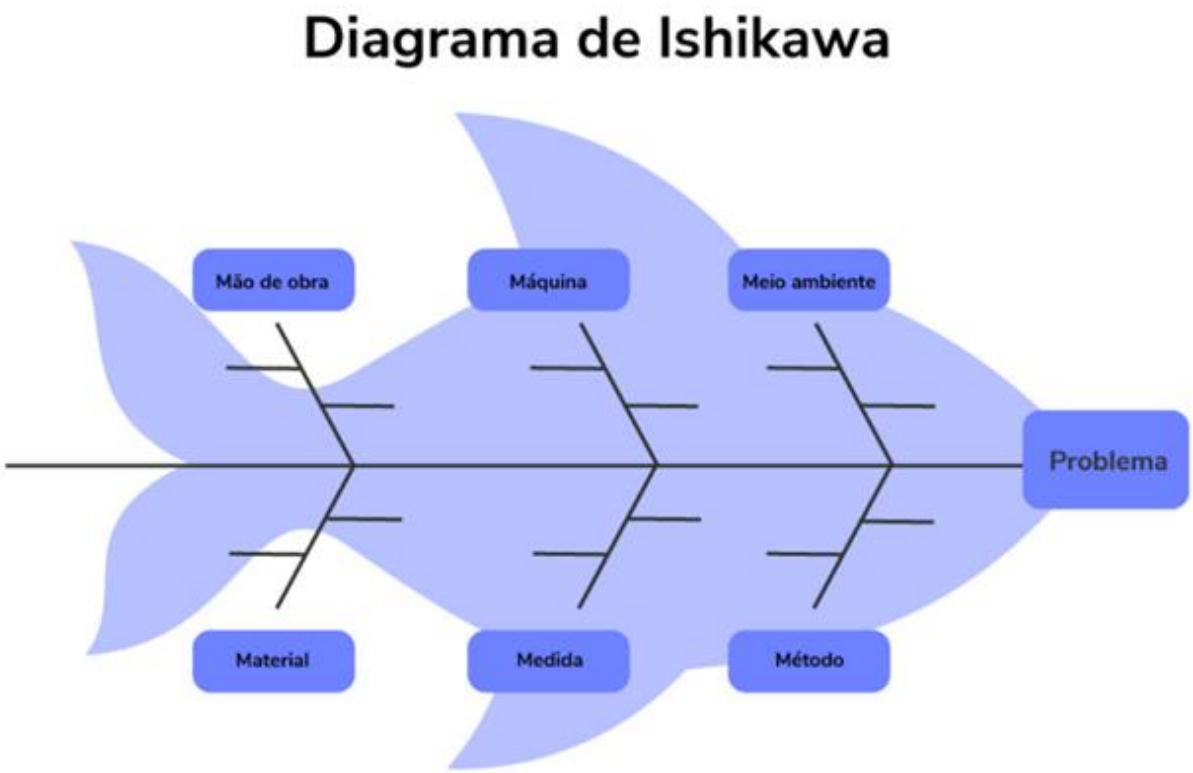


Imagen tomada de: [Diagrama-de-Ishikawa.png (1097×705) (automacaoindustrial.info)]

2. Técnica Delphi: Este método se centra en recolectar información u opiniones de especialistas para identificar posibles riesgos mediante rondas de preguntas, buscando así detectar los riesgos potenciales. La información se recolecta de manera anónima, lo que permite a los especialistas opinar libremente. No obstante, puede ser costoso si se involucran muchos expertos en el tema y requiere una

buena coordinación para llevar a cabo las rondas de consultas de manera efectiva (EALDE, 2020).

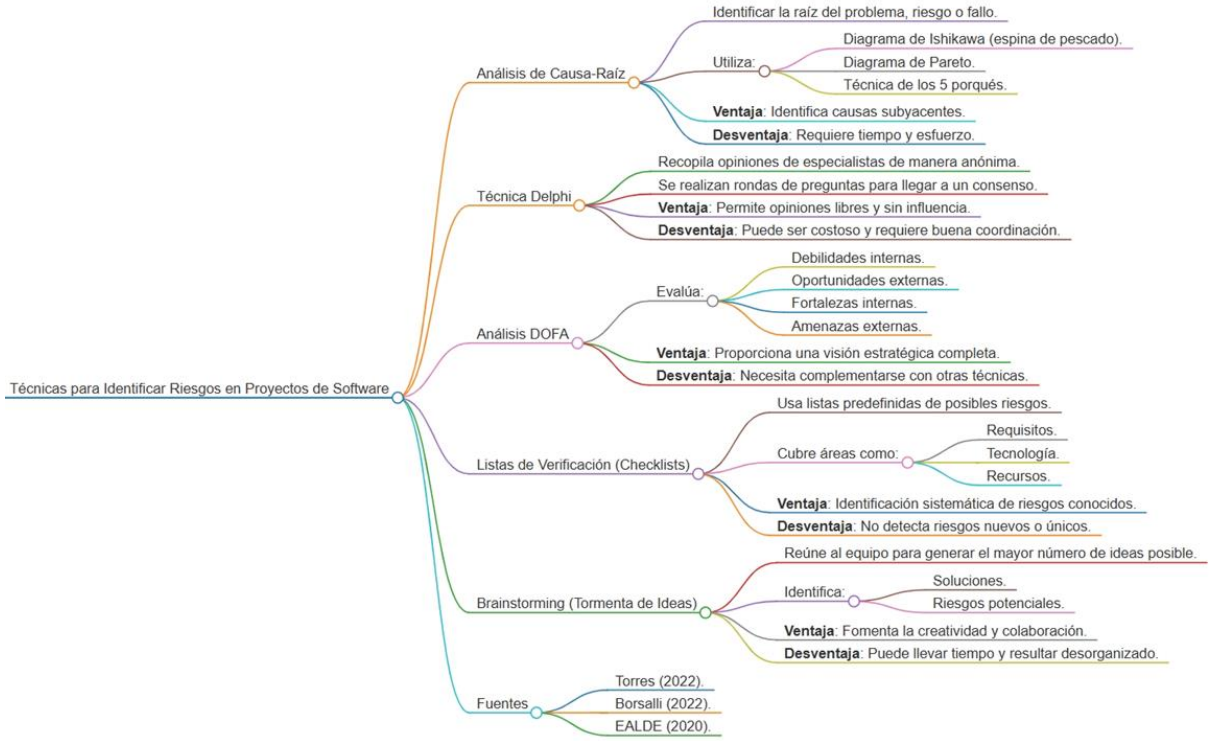
3. Análisis DOFA: Es el acrónimo de Debilidades, Oportunidades, Fortalezas y Amenazas. Se utiliza principalmente para evaluar, de manera interna, las debilidades y fortalezas, y de manera externa, las amenazas y oportunidades que afectan al proyecto de software en su fase inicial. Suele combinarse con otras técnicas para aumentar su efectividad (Borsalli, 2022).

Imagen recuperada de: [7 métodos y herramientas para la identificación de riesgos (softexpert.com)] (labitstudio, s.f.)

4. Listas de Verificación (Checklists): Esta técnica consiste en utilizar listas con los posibles riesgos y verificarlos en el proyecto de software, cubriendo áreas como requisitos, tecnología, etc. Facilita la identificación de riesgos conocidos, pero al basarse en riesgos predefinidos, puede no ser útil para identificar riesgos nuevos o únicos.

5. Brainstorming (Tormenta de Ideas): Se basa en reunir a los miembros del equipo para que aporten una lluvia de ideas con el mayor número posible de propuestas. De esta manera, se busca encontrar posibles soluciones o riesgos en el proyecto, aprovechando la experiencia y conocimientos de cada miembro del equipo. Esta técnica fomenta la creatividad y la colaboración (Borsalli, 2022).

Mapa Mental



6. Investigar 2 estándares que contengan métricas para evaluar la calidad de un producto de software, mencione las métricas, cómo se implementan y qué evalúan.

En el ámbito de TI y desarrollo de software, es fundamental seguir estándares de medición para asegurar la calidad del producto final. La calidad del software es medible y requiere de un marco de referencia, apoyándose en estándares de calidad reconocidos internacionalmente.

Según (Domínguez, 2017), “los estándares son normas y protocolos internacionales que deben cumplir los productos de cualquier índole para su distribución”, lo que implica que deben satisfacer las necesidades del cliente.

Los estándares se pueden clasificar en tres grupos principales:

Estándares que miden el producto (ej. ISO 25000 o SQUARE)

Estándares que miden el proceso (ej. CMMI)

Estándares específicos para TI (ej. ISO/IEC 9126)

Hablemos de dos de estos estándares:

ISO 25000(SQUARE): Este estándar esta más enfocado a la evaluación y mejora de calidad del software incluyendo mantenibilidad y usabilidad (Medina, 2022).

De acuerdo con (Medina, 2022)esta se compone de las siguientes divisiones

- Gestión de calidad
- Modelo de calidad
- Mediciones de calidad
- Requisitos de calidad
- Evaluación de la calidad
- Estándares de extensión

Este evalúa las siguientes características del proyecto de software

- Adecuación funcional
- Fiabilidad
- Usabilidad
- Eficiencia
- Compatibilidad
- Seguridad
- Mantenibilidad
- Portabilidad

Implementación: Se aplica en todo el ciclo de vida del software desde que se planifica hasta que se entrega.

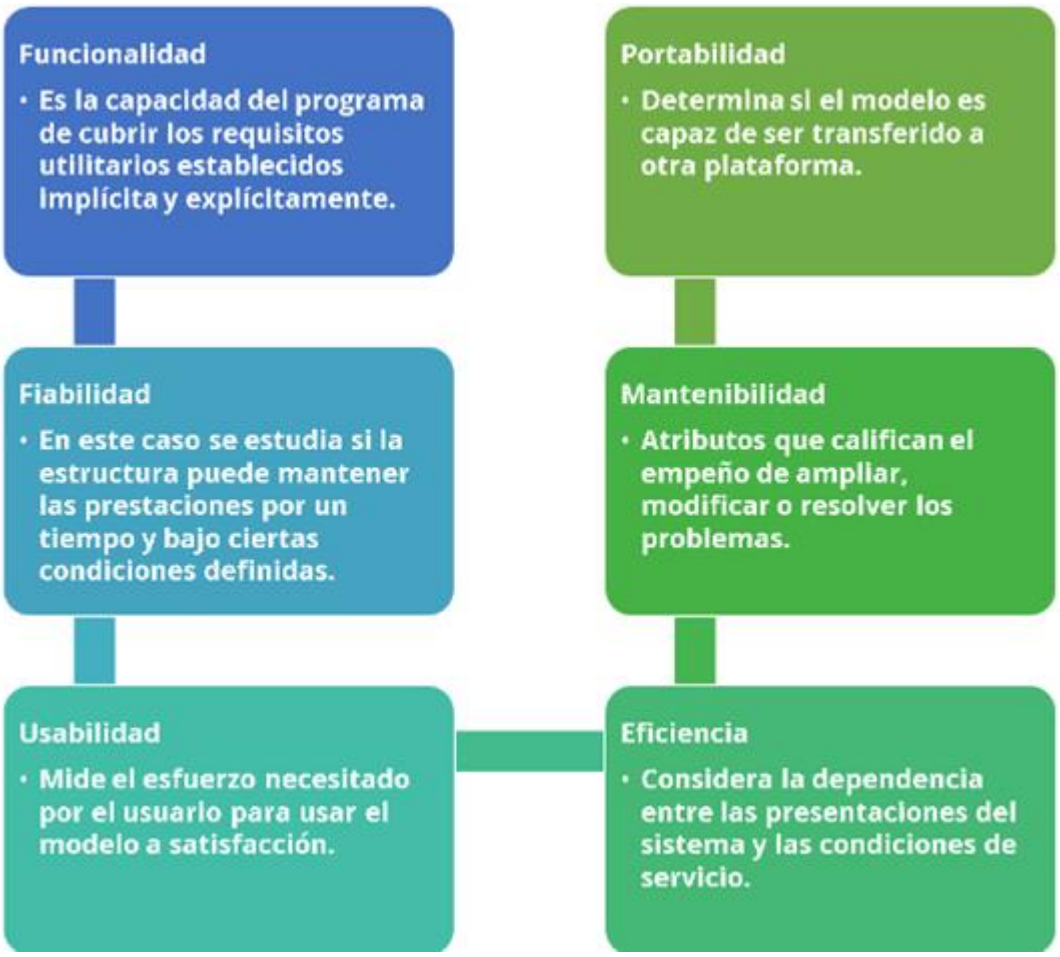
Métricas:

- Tiempo medio de detección: Indica cuánto tiempo se necesita para detectar errores.
- Tiempo medio de preparación: Mide el tiempo necesario para corregir defectos
- Prueba de confiabilidad: Evalúan la robustez y estabilidad del software.
- Cobertura de ejecución de pruebas: Cuantifica qué porcentaje del código se ha ejecutado durante las pruebas.
    - Cobertura de código: Analiza la cantidad de código fuente cubierto por pruebas.

ISO 9126: Es uno de los estándares que nos permite realizar la medición de la calidad del software, implementando siete características claves: funcionalidad, confiabilidad, usabilidad, eficiencia, mantenibilidad, portabilidad y satisfacción cada

una facilitan la medición de los atributos internos y externos del software. Lo que hace que esta norma sea crucial ya que nos proporciona métricas que permiten cuantificar y calificar la calidad del software, permitiendo que las empresas puedan estar a la vanguardia de la transformación digital asegurando que el software sea muy excelente y así solventar la satisfacción del cliente.

Pero según (GRUPO DIGITAL 360°, 2018)estas métricas están siendo reemplazadas por el proyecto ISO 25000, el cual tiene características y conceptos similares



Métricas

1. **Valoración de requerimientos**: permite asegurarse de que en cada fase se ha cumplido con las exigencias correspondientes. (GRUPO DIGITAL 360°, 2018)

2. **Integridad del sistema**: analiza las amenazas potenciales, el número de ocasiones que se presentan y cuántas veces la página se logra recuperar.

3. **Madurez del software**: indica el nivel de mantenimiento considerando la cantidad de módulos actuales, modificados, añadidos y eliminados. El número se aproxima a 1 cuando aumenta la estabilidad. En cuyo caso se necesitará menos tiempo en la corrección de errores o implementación de mejoras.

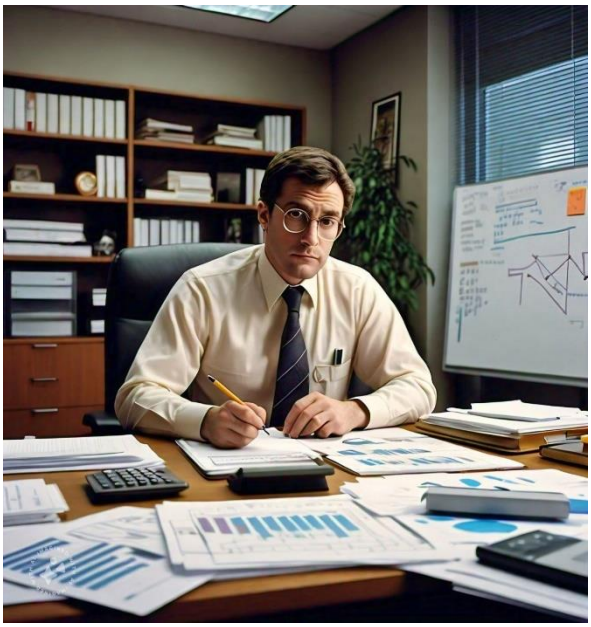
4. **Eficacia de detección y eliminación de errores**: las fallas se analizan por separado

Herramientas útiles para la medición o detección de errores

SonarQube: SonarQube es una herramienta de análisis on-premise diseñada para detectar problemas de codificación en +30 idiomas, frameworks y plataformas IaC. Al integrarse directamente con su canalización de CI (SonarQube, s.f.)

Documentación de SonarQube 10.6: SonarQube 10.6 (sonarsource.com)

7. ¿Qué es la estimación de proyectos?



La estimación de proyectos se refiere al proceso de estimar el tiempo, los costos y los recursos necesarios para completar un proyecto. Un proceso simplificado de estimación de proyectos es parte de cualquier estrategia exitosa de gestión de recursos y, por lo general, se realiza durante la fase inicial de planificación del proyecto.

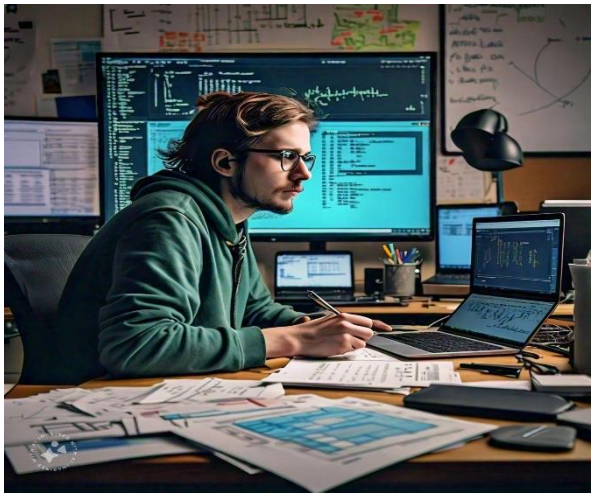
Estimar un proyecto con precisión te ayuda a ti y a las partes interesadas, a comprender los requerimientos de un proyecto antes de su inicio. De esa manera, puedes estar seguro de que tienes todo lo que necesitas para ejecutar y completar el proyecto con éxito, desde los recursos hasta el presupuesto correcto. Además, realizar estimaciones precisas ayuda a establecer expectativas para todos los involucrados, lo que ayuda a reducir la falta de comunicación y los malentendidos.

8. ¿Por qué es importante estimar en un proyecto?

¿QUÉ ES LA ESTIMACIÓN DE PROYECTOS?

La programación es una parte crucial de la gestión de un proyecto y está directamente influida por estimaciones precisas de recursos y tiempo. Cuando un equipo de proyecto proporciona estimaciones detalladas, puede crear un calendario claro que respalde la ejecución de cada fase del proyecto. Es probable que surjan imprevistos, como plazos irrazonables y cargas de trabajo subestimadas, que perturben la productividad si no se dispone de estimaciones adecuadas. Una preparación satisfactoria también implica definir los logros esenciales, los resultados y la secuencia de actividades necesarias para cumplir los objetivos del trabajo.

IMPORTANCIA DE ESTIMAR EN UN PROYECTO Y SU PLANIFICACIÓN



Las previsiones especifican el plazo dentro del cual deben cumplirse las tareas, lo que ayuda a los gestores de proyectos a planificar las operaciones y optimizar la asignación de recursos. Además, una preparación adecuada ayuda a identificar la interdependencia de las tareas, lo que facilita la gestión de los cambios y la adaptación a imprevistos. Una preparación insuficiente puede dar lugar a excesos de costes e interrupciones, comprometiendo la satisfacción del cliente y la reputación de la organización. En consecuencia, la elaboración de presupuestos es fundamental para que los grupos desarrollen una estrategia razonable y ejecutable que se ajuste a las expectativas de las partes interesadas y a los objetivos de la organización.

CONTROL DE RECURSOS

El control de recursos es un proceso crítico en la gestión de proyectos que asegura que los recursos necesarios estén disponibles y se utilicen de manera eficiente. La estimación precisa de recursos es vital para determinar cuántos empleados, materiales y financiamiento serán necesarios en cada fase del proyecto. Si un proyecto se basa en suposiciones imprecisas sobre los recursos, puede haber escasez de personal, lo que retrasa el avance, o un exceso de recursos, lo que incrementa los costos operativos sin aportar valor.

Una gestión competente de los recursos permite a los gestores de proyectos planear cómo y cuándo se desplegarán los recursos, asegurándose de que haya suficientes para cumplir los plazos acordados. Esto incluye la creación de un calendario de recursos, la distribución de la carga de trabajo entre los miembros del equipo y la prevención del agotamiento. Asimismo, la supervisión periódica del consumo de recursos proporciona información vital sobre su rendimiento y eficacia, lo que permite introducir modificaciones en tiempo real. Predecir y controlar

eficazmente los recursos ayuda a mantener el proyecto en marcha y aumenta la probabilidad de alcanzar los objetivos previstos.

EXPECTATIVAS DE STAKEHOLDERS

Gestionar las expectativas de las partes interesadas es fundamental para el éxito del proyecto y se basa en la apertura y la colaboración. Clientes, ejecutivos y miembros del equipo son partes interesadas con distintas prioridades y expectativas en cuanto a los plazos y presupuestos del proyecto. Hacer previsiones precisas permite a los gestores de proyectos mantener una conversación clara y educada sobre lo que cabe esperar, reduciendo así la posibilidad de que surjan interpretaciones erróneas.

Las partes interesadas pueden decidir cómo avanzar cuando las estimaciones son precisas y están bien fundamentadas. Esto es especialmente importante en situaciones en las que todos deben estar de acuerdo y las prioridades pueden cambiar. Además, fomenta la confianza porque la frustración y la decepción pueden evitarse a lo largo de la fase de estimación mediante una buena comunicación y gestión de las expectativas. Por otra parte, la gestión de la participación y la satisfacción de las partes interesadas exige mantenerlas informadas sobre el estado del proyecto y el cumplimiento de los plazos. Por todo ello, la estimación se convierte en un instrumento táctico para coordinar objetivos y establecer vínculos de confianza con las partes interesadas.

IDENTIFICACIÓN DE RIESGOS

La Identificación de riesgos es un proceso dinámico cuyo objetivo es predecir posibles dificultades durante la vida de un proyecto y que se beneficia enormemente de unas estimaciones correctas. Los gestores pueden identificar las áreas de incertidumbre y vulnerabilidad que pueden obstaculizar el avance del proyecto examinando el tiempo, el dinero y los recursos necesarios. Hacer estimaciones detalladas da a los jefes de proyecto una base sobre la que evaluar los posibles peligros, lo que les permite crear métodos de mitigación adecuados antes de que surjan esos riesgos.

Por poner un caso, en caso de que un proyecto sea demasiado lento, se puede descubrir que es probable que haya errores mortales de ese proyecto, como las instrucciones inadecuadas del personal necesario o la posibilidad de que los suministros no se entreguen a tiempo. Detectar estos peligros a tiempo le permiten realizar correcciones, capacitación del personal o buscar nuevas opciones. Por otro parte, la fijación de peligros y sus estimaciones también pueden ser beneficiosos en la fase posterior, que finaliza con éxito posteriormente el proyecto, como la reactividad y la recuperación de desafíos. En el caso de que los se encuentren y se sepan los peligros, también se garantiza que el proyecto se complete a tiempo y dentro del presupuesto.

TOMA DE DECISIONES

La toma de decisiones es importante en el procedimiento de la dirección de proyectos y depende de que la información sea exacta y puntual, sobre todo la facilitada a través de los cálculos estipulados. Cuando los directores del proyecto disponen de datos precisos, los medios necesarios y el tiempo requerido, están en excelentes condiciones de tomar decisiones fundamentadas que pueden influir en el curso del proyecto. Esto es fundamental cuando hay que explorar múltiples alternativas o modificar los planes iniciales en respuesta a cambios inesperados.

Las estimaciones sirven a los administradores de los proyectos evaluar las implicaciones de las distintas opciones. La dirección puede emplear proyecciones, por ejemplo, para evaluar si la introducción de nuevas funciones afectará al presupuesto y al calendario. Este tipo de evaluación facilita la priorización de tareas y el equilibrio entre objetivos de tiempo, gastos y plazos. Además, las proyecciones exactas disminuyen el peligro de sobrecarga de trabajo al proporcionar una evaluación honesta de lo que puede lograrse dentro de una cantidad de tiempo y dinero. Por último, una toma de decisiones eficaz basada en estimaciones exactas es fundamental para guiar la empresa hacia la consecución de sus objetivos.

Evaluación de Viabilidad

La evaluación de la viabilidad es un proceso de diagnóstico fundamental que permite comprobar si un proyecto es viable desde el punto de vista económico, técnico y de funcionamiento. En este proceso influyen directamente las estimaciones minuciosas que establecen una base para evaluar si los beneficios previstos compensan los costes. Las estimaciones permiten a los gestores de proyectos elaborar un presupuesto y un calendario detallados, factores críticos para la evaluación de la viabilidad.

cuando vamos arrancar hacer un proyecto, los directores deben establecer si tienen todo lo necesario como es el capital y la mano de obra, se puedan realizar de manera correcta y sin inconvenientes. Si las proyecciones exponen que el gasto superará el presupuesto o que el proyecto tardará demasiado en terminarse, es necesario dar esos cambios notables o replantearse la elaboración del proyecto. Una excelente evaluación de la viabilidad no sólo ayuda a evitar riesgos económicos, sino que también indica si la estrategia del proyecto es viable a largo plazo. Es decir, la viabilidad se transforma en un importante criterio que ayuda a las empresas a tomar decisiones con conocimiento de causa y a garantizar que sus inversiones en proyectos estén justificadas y vinculadas a objetivos estratégicos de mayor envergadura.

Mejora Continua

La mejora continua es un procedimiento encaminado a perfeccionar técnicas y prácticas a través del tiempo, lo que permite a las empresas evolucionar y transformarse. La estimación en la gestión de proyectos es fundamental en este contexto porque ofrece un marco para comparar el rendimiento real con los objetivos especificados. Al comparar estimaciones anteriores con resultados reales, los equipos pueden descubrir áreas de desarrollo y aprender de los errores.

Expongo el siguiente caso, si un proyecto se sale del presupuesto o no se completa a tiempo, el equipo puede examinar las estimaciones iniciales para entender qué variables contribuyeron a las diferencias. Este estudio no sólo perfecciona los procesos de estimación, sino que también identifica las lecciones aprendidas que pueden aplicarse a futuros proyectos. La incorporación de información sobre estimaciones anteriores desarrolla una cultura de aprendizaje en el equipo, haciendo hincapié en la adaptabilidad y la creatividad. Como resultado, el proceso de mejora continua, ayudado por estimaciones precisas y cuidadosas, se transforma en un instrumento estratégico para impulsar la eficiencia de los proyectos y el rendimiento de la organización.

9. Mencione técnicas que se implementen en la actualidad para realizar estimación de proyectos. Realice un ejemplo con el equipo de trabajo y documéntese.

1. Claridad en los Objetivos

La claridad en los objetivos es fundamental para el éxito de cualquier proyecto. Cuando todos los miembros del equipo comprenden los objetivos de manera precisa, se fomenta la alineación y el compromiso hacia un fin común. Una comunicación efectiva ayuda a definir estos objetivos de forma clara y comprensible, asegurando que cada miembro del equipo sepa lo que se espera de ellos. Esto implica no solo especificar qué se espera alcanzar, sino también por qué esos objetivos son importantes para la organización.

Una falta de claridad puede dar lugar a confusión, errores y esfuerzos desperdigados que alejan al equipo de sus metas. La comunicación clara y continua permite que se realicen ajustes rápido si es necesario, asegurando que todos se mantengan en la misma página. Además, fomenta un ambiente de transparencia donde se pueden plantear dudas, y se crea un espacio donde se valida el feedback. En última instancia, la claridad en los objetivos potencia la motivación y la colaboración, lo que facilita el progreso hacia los resultados deseados. Una buena práctica es revisar periódicamente los objetivos para asegurarse de que sigan siendo relevantes y que todos estén actualizados sobre cualquier cambio.

2. Facilitación del Trabajo en Equipo

La facilitación del trabajo en equipo es una de las dimensiones más críticas en la gestión de proyectos, y la comunicación efectiva juega un papel esencial en su éxito. Un equipo cohesionado debe tener la capacidad de compartir información, coordinar esfuerzos y resolver conflictos de manera constructiva. Las herramientas de comunicación, ya sean reuniones, correos electrónicos o plataformas colaborativas, permiten que los integrantes del equipo compartan ideas y progresos, lo cual fomenta la confianza mutua y la colaboración.

Cuando la comunicación fluye libremente, se minimizan los malentendidos, y los miembros pueden abordar los problemas y obstáculos de inmediato. Esto no solo mejora la moral del equipo, sino que también incrementa la eficiencia, ya que cada persona puede contribuir con su expertise en la solución de problemas. Además, una buena comunicación promueve la inclusión, asegurando que todos los miembros, independientemente de su rol, se sientan valorados y escuchados. Establecer normas claras sobre cómo y cuándo comunicarse puede facilitar la colaboración, ayudando a definir roles y responsabilidades desde el comienzo. Aunque la comunicación efectiva requiere esfuerzo, los beneficios de un equipo comprometido y colaborativo son invaluables.

3. Gestión de Conflictos

La gestión de conflictos es un aspecto esencial en la dinámica de cualquier equipo de trabajo, y una comunicación adecuada es clave para su resolución. En un

entorno de proyecto, es inevitable que surjan desacuerdos; sin embargo, la forma en que se manejan estos conflictos puede determinar el éxito del equipo. Una comunicación abierta y honesta permite a los miembros del equipo expresar sus diferencias de manera constructiva, facilitando la búsqueda de soluciones.

La identificación temprana de conflictos a través de un diálogo efectivo puede evitar que pequeñas disputas crezcan y afecten la moral del equipo o retrasen el proyecto. Además, es importante que los líderes de proyecto fomenten un entorno donde todos se sientan seguros para expresar sus opiniones. Mediante el uso de técnicas de resolución de conflictos, como la mediación y el compromiso, se pueden transformar situaciones tensas en oportunidades de aprendizaje y mejora. La gestión de conflictos a través de una comunicación efectiva no solo resuelve problemas inmediatos, sino que también fortalece las relaciones interpersonales en el equipo, creando un ambiente de trabajo más cohesionado y centrado en las metas compartidas.

4. Aumento de la Productividad

El aumento de la productividad en un equipo de proyecto es un objetivo crucial que se ve influenciado directamente por la calidad de la comunicación. Cuando los miembros de un equipo tienen una comunicación efectiva, pueden compartir información, coordinar tareas y seguir el avance de manera más eficiente. Esto se traduce en una reducción de malentendidos y errores, lo que ahorra tiempo y recursos, y permite centrarse en las tareas que realmente importan.

La comunicación también establece expectativas claras sobre plazos y entregables, lo que permite que cada miembro del equipo conozca sus responsabilidades y se comprometa a cumplir con ellas. Este sentido de responsabilidad compartida impulsa a todos a trabajar con mayor rapidez y eficiencia. Además, con un flujo de información constante, se pueden identificar y resolver problemas o cuellos de botella antes de que se conviertan en obstáculos significativos. Para alcanzar este objetivo, es vital implementar canales de comunicación que faciliten el intercambio regular de información, como reuniones diarias cortas (scrums) o el uso de herramientas de seguimiento de proyectos. En última instancia, una comunicación eficaz no solo promueve la mejora en la productividad individual, sino que también impulsa el rendimiento general del equipo.

5. Adaptación al Cambio

La adaptación al cambio es un componente inevitable en la gestión de proyectos, especialmente en entornos dinámicos. Las circunstancias, requerimientos y prioridades pueden cambiar, y la capacidad para adaptarse rápidamente a estas alteraciones es crucial. La comunicación efectiva es un facilitador clave en este proceso, permitiendo que el equipo reciba información sobre cambios importantes y comprenda su impacto en el proyecto.

Un flujo constante de información ayuda al equipo a anticipar cambios y a desarrollar planes de acción adecuados, minimizando así la resistencia y el desconcierto que suelen acompañar a nuevas directrices. La apertura en la comunicación no solo permite que las personas expresen sus preocupaciones o dudas sobre los cambios propuestos, sino que también les da un sentido de

propiedad sobre el proceso. Cuando los miembros del equipo son informados y participan en la toma de decisiones, son más propensos a comprometerse con nuevas direcciones. Además, implementar retroalimentación en tiempo real permite ajustar enfoques y estrategias a lo largo del proyecto. Así, la adaptabilidad, potenciada por la comunicación efectiva, se convierte en una ventaja competitiva que facilita la evolución y el éxito en proyectos en constante cambio.

6. Fortalecimiento de Relaciones

El fortalecimiento de relaciones es un factor crucial que impacta directamente en la cohesión y efectividad de un equipo de proyecto. La comunicación juega un papel central al construir y mantener relaciones saludables entre los miembros del equipo. Un ambiente donde se fomentan interacciones abiertas y respetuosas crea un sentido de comunidad y pertenencia, lo que contribuye a un mejor ambiente laboral.

Una buena comunicación no solo implica informar y compartir actualizaciones, sino también estar dispuesto a escuchar y considerar las perspectivas de los demás. Fomentar una cultura de respeto y empatía puede reducir las tensiones y promover la resolución constructiva de conflictos. Al invertir en la comunicación y el desarrollo de relaciones interpersonales sólidas, los equipos pueden crear un espacio donde cada miembro se sienta valorado y motivado para colaborar. Además, las relaciones fuertes facilitan una colaboración más efectiva, ya que los miembros se sienten cómodos intercambiando ideas y ofreciendo feedback. Una notable conexión entre los miembros del equipo se traduce en una mayor lealtad, satisfacción laboral y, en última instancia, un mejor rendimiento en el proyecto.

7. Retroalimentación Efectiva

La retroalimentación efectiva es fundamental en el proceso de comunicación de un proyecto, ya que permite la evaluación continua del rendimiento y el avance. Proporcionar y recibir retroalimentación en tiempo real ayuda a los miembros del equipo a entender su rendimiento individual y colectivo, así como a identificar áreas de mejora. La comunicación efectiva en este contexto asegura que la retroalimentación sea constructiva, específica y orientada a soluciones, lo que a su vez mejora el aprendizaje y el crecimiento.

Establecer un entorno de confianza donde se pueda ofrecer feedback sin temor a represalias es crucial para la eficacia de este proceso. La retroalimentación regular también permite ajustarse rápidamente ante cualquier desviación de los objetivos del proyecto, facilitando correcciones antes de que se conviertan en problemas mayores. Además, al ofrecer oportunidades para discutir no solo el desempeño, sino también los desafíos que se enfrentan, se fomenta un ambiente colaborativo. Esto no solo mejora la moral del equipo, sino que también alienta a la responsabilidad compartida. La inclusión de sesiones de retroalimentación estructuradas, como revisiones periódicas, puede ser un mecanismo efectivo para maximizar la colaboración y el rendimiento.

Referencias

- Vesperman, J. (2006). Essential CVS (2nd ed.). O'Reilly Media.
- Sink, E. (2011). Versión Control by Example. Pyrenean Gold Press.
- Swicegood, T. (2008). Pragmatic Version Control Using Git. Pragmatic Bookshelf.
- Somasundaram, R. (2013). Git: Version Control for Everyone. Packt Publishing.
- Silverman, M. (2013). Git Pocket Guide: A Working Introduction. O'Reilly Media.
- Pipinellis, A. (2015). GitHub Essentials. Packt Publishing
- *Apache Subversion*. (n.d.). Apache.org. Retrieved September 27, 2024, from https://subversion.apache.org/
- *gestion de la configuracion del software*. (n.d.). Angelfire.com. Retrieved September 27, 2024, from https://planificacionymodelado.angelfire.com/gestion-de-la-configuracion-del-software.html
- *Subversion Documentation*. (n.d.). Apache.org. Retrieved September 27, 2024, from https://subversion.apache.org/docs/
- Trabajo de Diploma Para Optar Por el Título, de I. en C. I. (n.d.). *Procesos de Gestión de la Configuración de Software en el desarrollo de distribuciones GNU/Linux"*. Uci.Cu. Retrieved September 27, 2024, from https://repositorio.uci.cu/jspui/bitstream/ident/TD_05035_11/1/TD_05035_11.pdf
- Ospino, L. A. C. (2020, May 26). *ESTIMACIÓN EN GESTIÓN DE PROYECTOS*. Linkedin.com. https://www.linkedin.com/pulse/estimaci%C3%B3n-en-gesti%C3%B3n-de-proyectos-luis-cagua-ospino/?originalSubdomain=es
- *Técnicas de Estimación de Proyectos*. (n.d.). Timecamp.com. Retrieved September 27, 2024, from https://www.timecamp.com/es/planner/glossary/tecnicas-de-estimacion-de-proyectos/
- Borsalli, B. (28 de 02 de 2022). *SoftExpert Blog*. Obtenido de SoftExpert Blog: https://blog.softexpert.com/es/identificacion-riesgos/
- Domínguez, I. R. (08 de 03 de 2017). *Calidad del software | MODELOS Y ESTÁNDARES DE UN PRODUCTOI*. Obtenido de Youtube: https://www.youtube.com/watch?v=2lycHFiG-co
- EALDE. (10 de 11 de 2020). *EALDE*. Obtenido de EALDE: https://cursos.ealde.es/blogs/gestion-de-riesgos-direccion-de-proyectos-online/tecnicas-y-herramientas-para-la-identificacion-de-riesgos-en-proyectos#:~:text=Algunos%20ejemplos%20de%20t%C3%A9cnicas%20de%20recopilaci%C3%B3n%20de%20informaci%C3%B3n,de%20deb

- GRUPO DIGITAL 360°. (13 de 12 de 2018). *ISO 9126*. Obtenido de GRUPO DIGITAL 360°: https://grupodigital360.com/iso-9126-metricas-para-calidad-web/
- labitstudio. (s.f.). *labitstudio*. Obtenido de labitstudio: https://labitstudio.com/diferencias-sistemas-control-de-versiones/
- Medina, I. F. (05 de 12 de 2022). *Los estándares de calidad del software más importantes*. Obtenido de hiberus blog: https://www.hiberus.com/crecemos-contigo/los-estandares-de-calidad-del-software-mas-importantes/
- SonarQube. (s.f.). *Documentación de SonarQube 10.6*. Obtenido de SonarQube: https://docs.sonarsource.com/sonarqube/latest/
- Torres, O. P. (14 de 10 de 2022). *pirani*. Obtenido de pirani: https://www.piranirisk.com/es/blog/gestion-de-riesgos-proyectos-de-software
- Bonilla, G. A. (06 de Octubre de 2016). *Universidad Mariana*. Obtenido de file:///C:/Users/Osvaldo/Downloads/adm-ojs2014,+Herramientas+para+la+gesti%C3%83%C2%B3n.pdf
- Cruz-Alonso, J. A. (17 de Diciembre de 2021). *Revista ecosistemas*. Obtenido de https://www.revistaecosistemas.net/index.php/ecosistemas/article/view/2332
- Holcombe, J. (17 de Abril de 2023). *Kinsta*. Obtenido de https://kinsta.com/es/blog/github-estadisticas/