Clase grabada

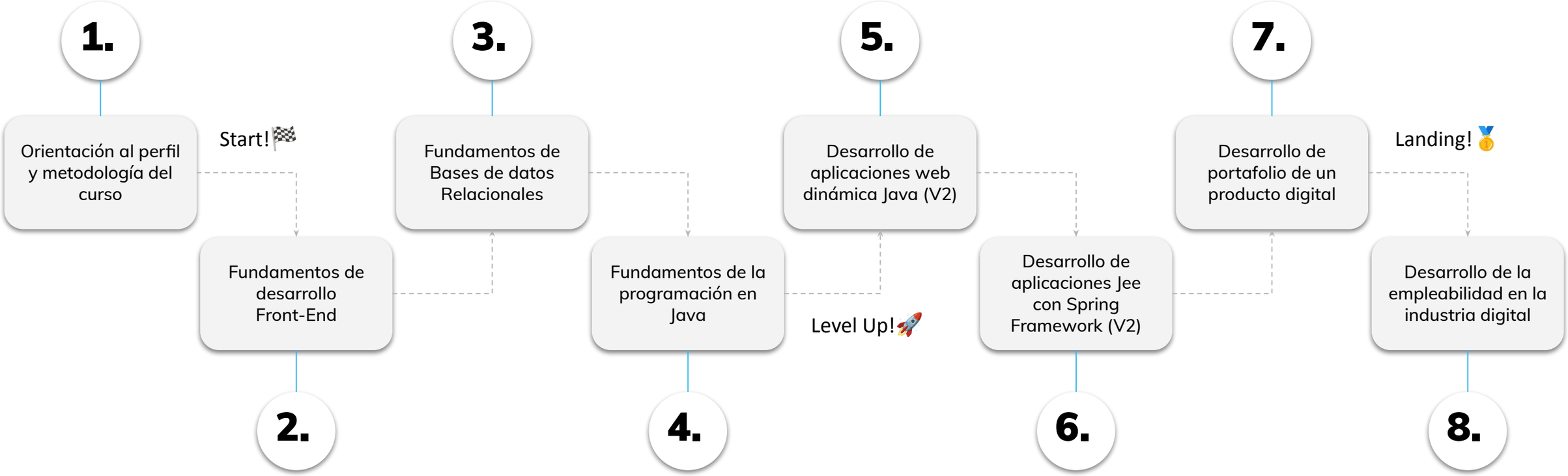
Fundamentos de GIT y GITHUB

Plan formativo: Desarrollo de Aplicaciones Full Stack Java Trainee V2.0

English Always | alkemy

HOJA DE RUTA

¿Cuáles **skill** conforman el programa?



English Always | alkemy

REPASO CLASE ANTERIOR

En la clase anterior trabajamos 📚:

- ✓ *Comprender la importancia del control de versiones y la necesidad de un repositorio de código fuente.*
- ✓ *Instalar y configurar Git en un entorno de desarrollo.*
- ✓ *Dominar los comandos básicos de Git para crear, clonar y gestionar repositorios.*

LEARNING PATHWAY

¿Sobre qué temas trabajaremos?

2. Start! 🏁

FUNDAMENTOS DE DESARROLLO FRONT-END

Fundamentos de GIT y GITHUB

Actividad N° 1: Repositorio GITHUB

Este curso se centra en las habilidades esenciales de GitHub y la gestión de repositorios para el desarrollo de proyectos colaborativos. Aprenderás a trabajar con repositorios remotos, realizar Push y Pull de cambios, y gestionar Pull Requests para colaborar eficazmente en proyectos de código abierto. Explorarás la documentación de proyectos con Markdown y conocerás las mejores prácticas para administrar el flujo de trabajo de desarrollo con GitHub.

OBJETIVOS DE APRENDIZAJE

¿Qué aprenderemos?

- *Comprender los fundamentos de GitHub como una plataforma de gestión de control de versiones.*
- *Dominar la gestión de repositorios remotos y la sincronización de cambios mediante Push y Pull.*
- *Distinguir entre Fetch y Pull y aplicar el concepto apropiado según la situación.*
- *Aprender a clonar un repositorio existente para colaborar en proyectos.*
- *Adquirir habilidades para administrar Pull Requests y revisar y fusionar cambios de colaboradores.*

Repositorios remotos, Push y Pull Fetch v/s Pull

Repositorios remotos

Push y Pull Fetch v/s Pull

Los repositorios remotos, y los comandos push, pull, fetch y pull son conceptos y comandos importantes en Git y GitHub para colaborar con otros desarrolladores y mantener sincronizados los cambios en un proyecto.

Repositorios remotos

Push y Pull Fetch v/s Pull

Repositorios remotos:

Un repositorio remoto no es más que una copia del repositorio Git en un servidor remoto, como GitHub, GitLab o Bitbucket.

El cual permite colaborar con otros desarrolladores, compartir el código y realizar un seguimiento de los cambios en un proyecto de manera centralizada.

Repositorios remotos

Push y Pull Fetch v/s Pull

Repositorios remotos:

Una gran ventaja es que se puede clonar un repositorio remoto en tu máquina local utilizando el comando git clone.

```
git clone https://github.com/usuario/turepo.git
```

Recorda que debes cambiar https://github.com/usuario/turepo.git por la url de tu proyecto.

Después de clonar, tendrás una copia local del repositorio y podrás realizar cambios y commits en tu máquina.

Repositorios remotos

Push y Pull Fetch v/s Pull

push:

El comando push se utiliza para enviar los commits locales al repositorio remoto. Lo que permite compartir tus cambios con otros desarrolladores y mantener sincronizado el repositorio remoto con tus últimas modificaciones.

Utiliza el comando git push seguido del nombre de la rama y el repositorio remoto al que deseas enviar los cambios.

Repositorios remotos

Push y Pull Fetch v/s Pull

push:

arrancamos clonando nuestro repositorio, utilizaremos push para que los cambios realizados se guarden en el repositorio remoto. Pero ahora, además de clonar, vamos a crearle algunos cambios, para que sean guardados utilizando push

- Ya tenemos esto, del paso anterior:

```
git clone https://github.com/usuario/turepo.git
```

- Podes realizar cambios en el archivo:

Navega al directorio del repositorio clonado y realiza los cambios deseados en un archivo específico utilizando el editor de texto o el IDE de tu elección.

Repositorios remotos

Push y Pull Fetch v/s Pull

push:

- Verificar el estado de los cambios:

Utiliza el comando git status para verificar el estado de los cambios en tu repositorio local. Deberías ver el archivo modificado listado como un archivo sin seguimiento.

- Agregar los cambios al área de preparación:

Utiliza el comando git add seguido del nombre del archivo modificado para agregar los cambios al área de preparación (staging area). Por ejemplo:

```
git add nombre_del_archivo_modificado
```

Repositorios remotos

Push y Pull Fetch v/s Pull

push:

- Realizar el commit:

Realiza un commit para guardar los cambios en el repositorio local.

Utiliza el comando git commit -m seguido de un mensaje descriptivo para el commit.

```
git commit -m "Agregados cambios en el archivo"
```

Repositorios remotos

Push y Pull Fetch v/s Pull

push:

- Realizar el push:

Utiliza el comando git push seguido del nombre del repositorio remoto y la rama a la que deseas enviar tus cambios.

```
git push origin master
```

Reemplaza origin con el nombre del repositorio remoto (generalmente es origin si clonaste el repositorio) y master con el nombre de la rama en la que deseas realizar el push.

Repositorios remotos

Push y Pull Fetch v/s Pull

push:

- Opcional:

Si es la primera vez que haces un push en esta rama, es posible que necesites agregar la opción -u para establecer la rama upstream:

```
git push -u origin master
```

Repositorios remotos

Push y Pull Fetch v/s Pull

push:

- Proporciona tus credenciales:

Si es la primera vez que realizas un push a este repositorio remoto o si se requiere autenticación, se te solicitará ingresar tus credenciales de GitHub (nombre de usuario y contraseña o token de acceso personal).

- Espera a que finalice el push:

Git enviará tus cambios al repositorio remoto en GitHub. Una vez que el push se complete, verás un mensaje indicando que los cambios se han enviado correctamente.

Repositorios remotos

Push y Pull Fetch v/s Pull

pull:

El comando pull se utiliza para recuperar los cambios del repositorio remoto y fusionarlos con tu copia local. Este comando se combina con el comando **fetch** (lo veremos a continuación) que es para recuperar y con el comando **merge** que es para fusionar (este ya lo trabajamos anteriormente) en un solo paso.

Utiliza el comando **git pull** seguido del nombre de la rama y el repositorio remoto desde donde deseas obtener los cambios.

Repositorios remotos

Push y Pull Fetch v/s Pull

pull:

- Verificar el estado de tus cambios locales:

Antes de realizar un pull, es recomendable verificar el estado de tus cambios locales. Utiliza el comando git status para asegurarte de que no tengas cambios sin confirmar.

- Realizar un pull desde el repositorio remoto:

Utiliza el comando git pull seguido del nombre del repositorio remoto y la rama de la cual deseas obtener los cambios.

```
git pull origin master
```

Reemplaza origin con el nombre del repositorio remoto (generalmente es origin si clonamos el repositorio) y master con el nombre de la rama desde la cual deseas obtener los cambios.

Repositorios remotos

Push y Pull Fetch v/s Pull

pull:

- Proporcionar sus credenciales:

Si se requiere autenticación para acceder al repositorio remoto, es posible que se te solicite ingresar tus credenciales de GitHub (nombre de usuario y contraseña o token de acceso personal).

- Esperar a que finalice el pull:

Git recuperará los últimos cambios del repositorio remoto y los fusionará automáticamente en tu rama local. Si no hay conflictos, el pull se completará sin problemas.

En caso de que haya conflictos, Git te notificará y mostrará los archivos afectados. Deberás resolver los conflictos manualmente editando los archivos en conflicto y luego hacer un commit para finalizar la fusión

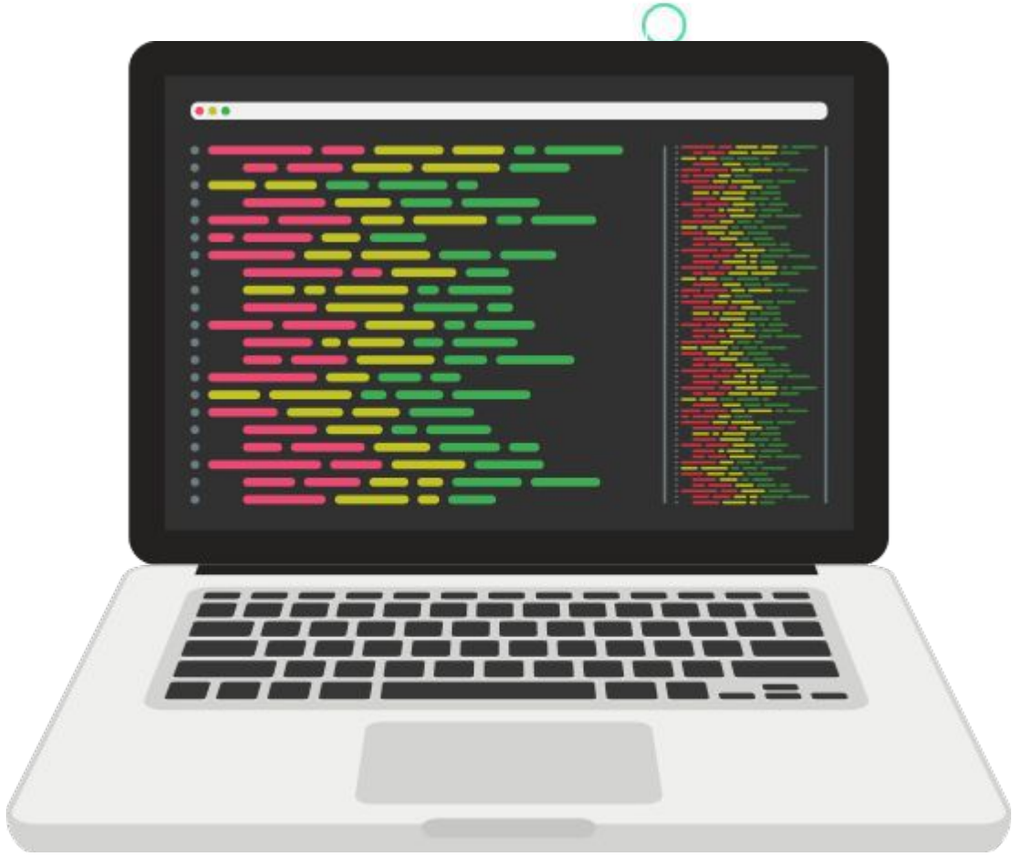
fetch vs pull en GitHub

fetch vs pull en GitHub

En GitHub, fetch se utiliza para obtener los cambios del repositorio remoto sin fusionar automáticamente los cambios en tu rama local.

Con fetch, los cambios se descargan a tu repositorio local, pero no se fusionan automáticamente, lo que te permite revisar los cambios antes de incorporarlos.

Para fusionar los cambios después de realizar fetch, podes utilizar el comando git merge o git rebase para combinar los cambios descargados con tu rama local.

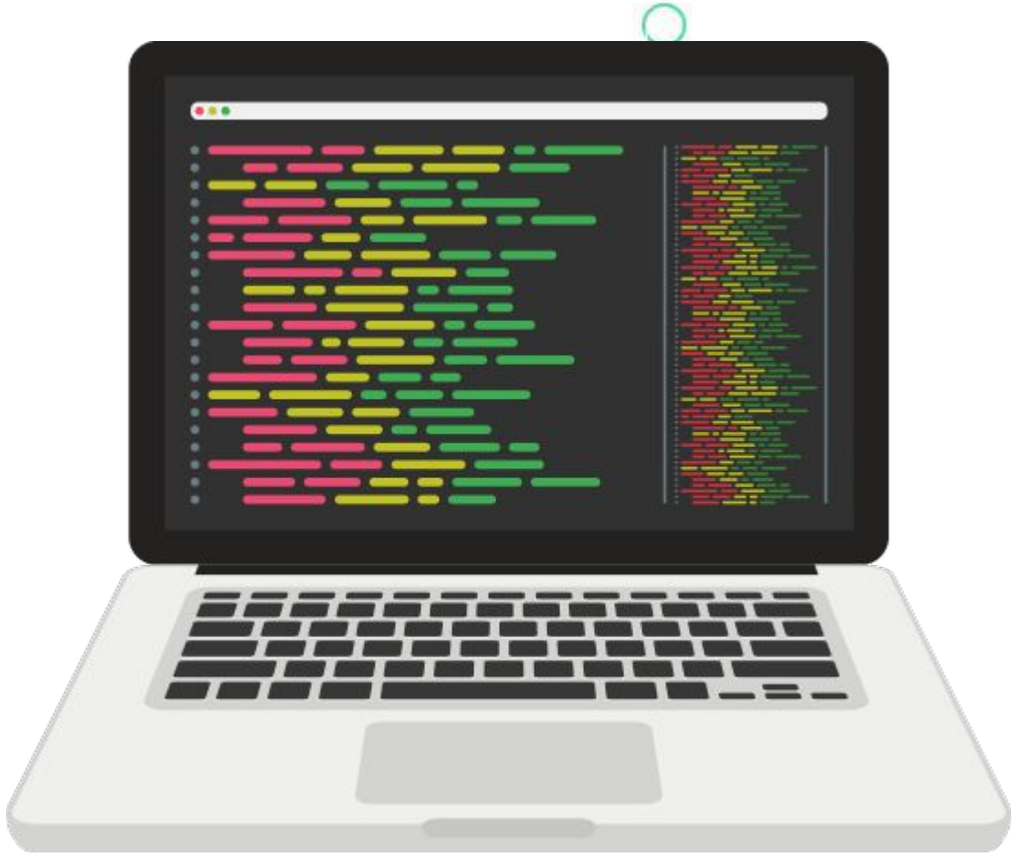


fetch vs pull en GitHub

Por otro lado, pull en GitHub es similar a fetch pero automáticamente fusiona los cambios descargados en tu rama local sin necesidad de ejecutar un comando de fusión adicional.

Si estás trabajando en un proyecto en solitario o en una rama dedicada, pull puede ser conveniente.

Sin embargo, si estás colaborando con otros desarrolladores y deseas revisar los cambios antes de fusionarlos, podes optar por fetch y luego realizar la fusión manualmente.



fetch vs pull en GitHub

Fetch:

- Cuando realizas un "Fetch", obtienes una copia de los cambios del repositorio remoto en tu repositorio local.
- Los cambios se descargan y se almacenan en tu repositorio local, pero no se aplican automáticamente a tu rama de trabajo actual.
- Puedes inspeccionar los cambios descargados antes de decidir fusionarlos en tu rama actual.

Pull:

- *Un "Pull" es una operación que combina dos pasos en uno: primero, obtiene los cambios del repositorio remoto (realiza un "Fetch") y luego los fusiona automáticamente en tu rama de trabajo actual.*
- *Es una forma más conveniente de actualizar tu rama local con los cambios del repositorio remoto en comparación con un "Fetch" seguido de una fusión manual.*

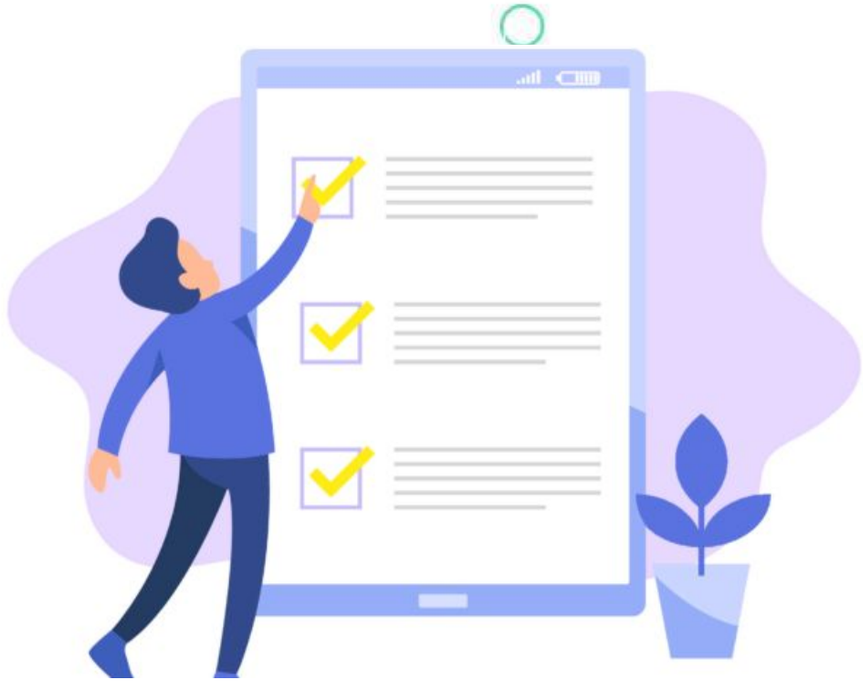
Administrando Pull Request

Administrando Pull Request

¿Qués es un pull request?

Un **pull request** o **solicitud de extracción** es una función clave en plataformas de alojamiento de repositorios como GitHub. Es un mecanismo que permite a los desarrolladores colaborar y revisar los cambios realizados en un repositorio antes de que sean fusionados en la rama principal o en otra rama.

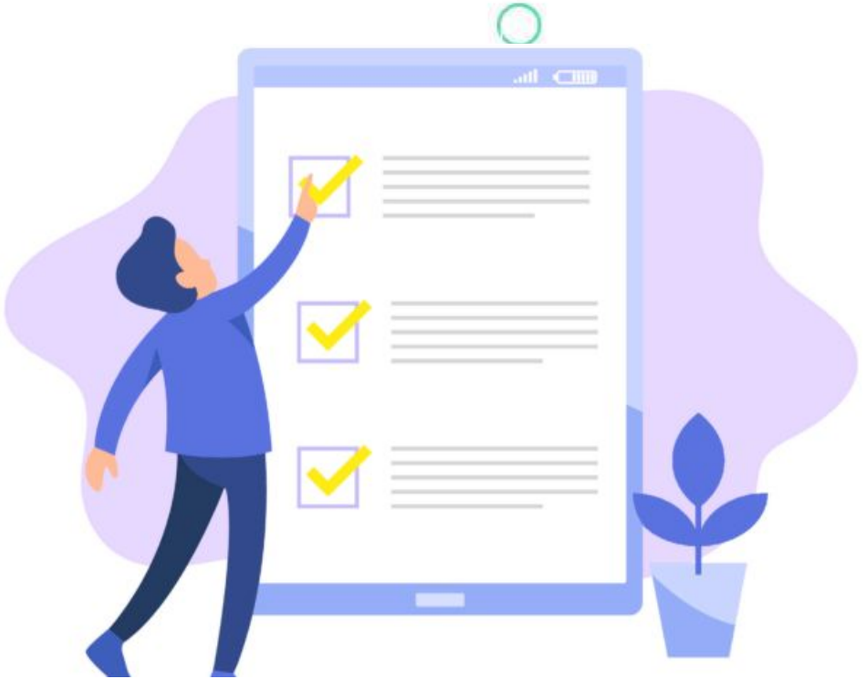
Podemos verlo como una solicitud que un desarrollador envía al propietario o colaboradores del repositorio para que revisen y consideren los cambios propuestos.



Administrando Pull Request

El flujo típico de trabajo con pull requests:

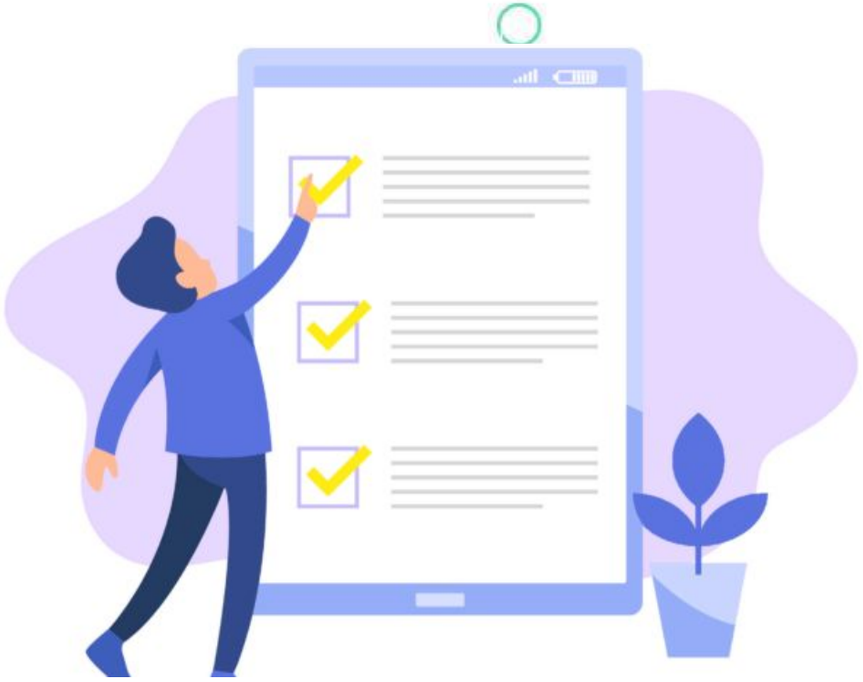
- Un desarrollador crea una rama separada en el repositorio para trabajar en una nueva función, solucionar un error o realizar cualquier otro tipo de modificación.
- Después de realizar los cambios y realizar los commits en su rama, el desarrollador envía un pull request al repositorio principal o a la rama base a la que desea fusionar sus cambios.



Administrando Pull Request

El flujo típico de trabajo con pull requests:

- Otros colaboradores pueden revisar el pull request, ver los cambios realizados, hacer comentarios y sugerencias, y solicitar modificaciones adicionales si es necesario.
- El autor del pull request puede realizar las modificaciones solicitadas, agregar más commits a su rama y actualizar automáticamente el pull request para que los revisores vean los cambios actualizados.

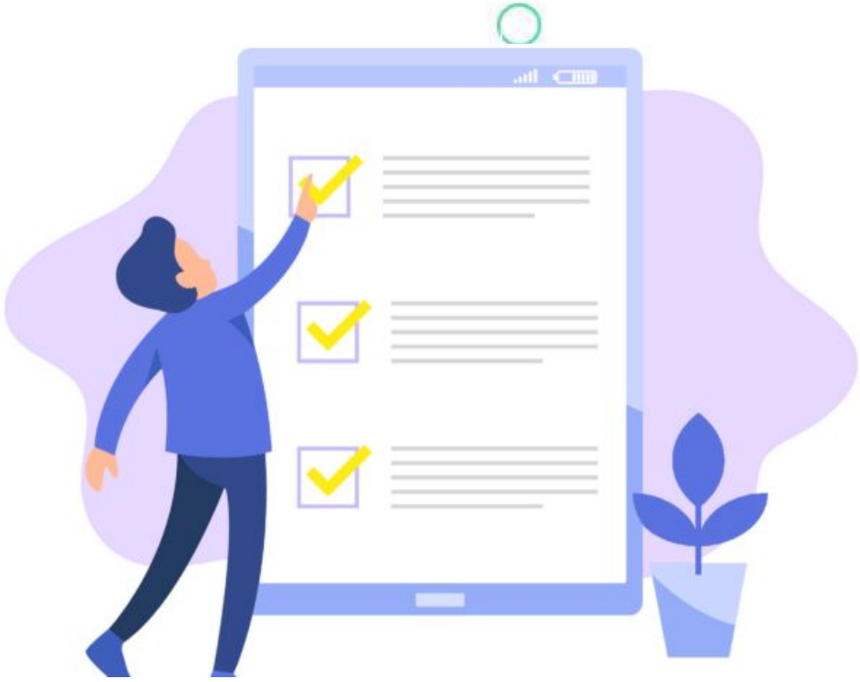


Administrando Pull Request

El flujo típico de trabajo con pull requests:

- Una vez que los cambios en el pull request han sido aprobados y revisados adecuadamente, el propietario del repositorio o un colaborador con permisos suficientes puede fusionar los cambios en la rama base.

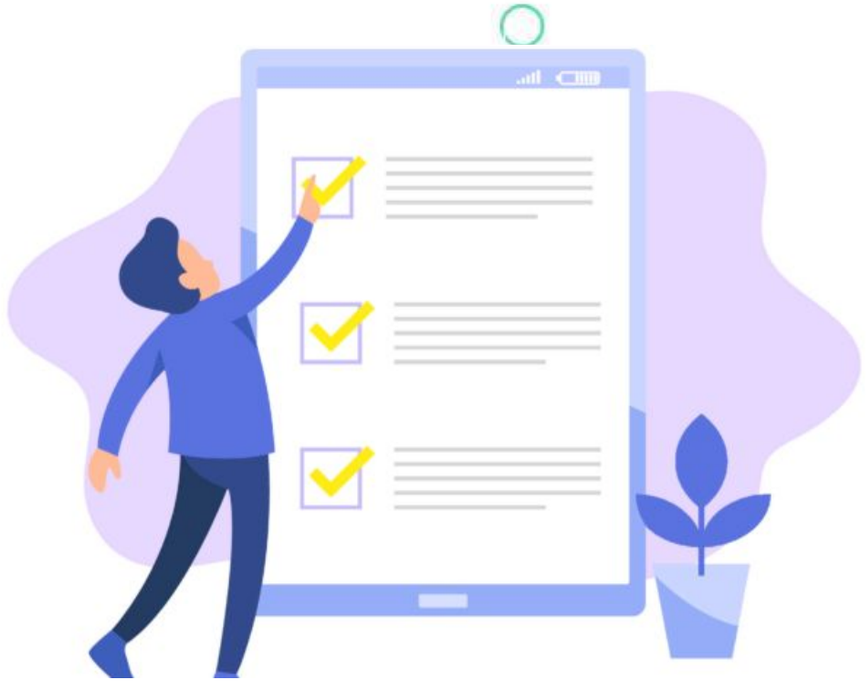
Los pull requests son valiosos en entornos colaborativos, ya que permiten una revisión de código más rigurosa y fomentan la colaboración abierta entre los miembros del equipo.



Administrando Pull Request

Administrar un Pull Request

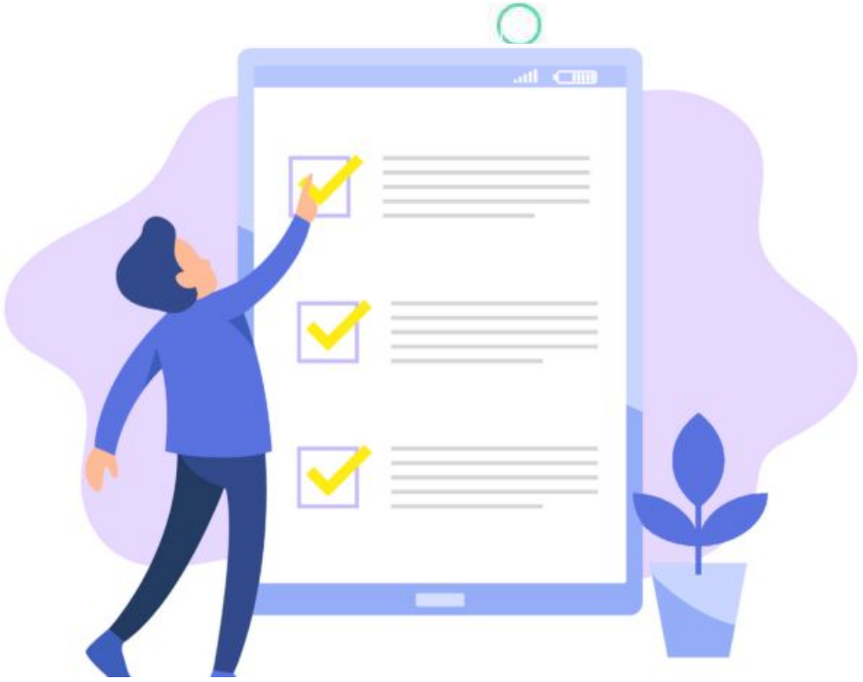
Administrar Pull Requests es una parte fundamental del flujo de trabajo colaborativo en GitHub.



Administrando Pull Request

Crear un Pull Request:

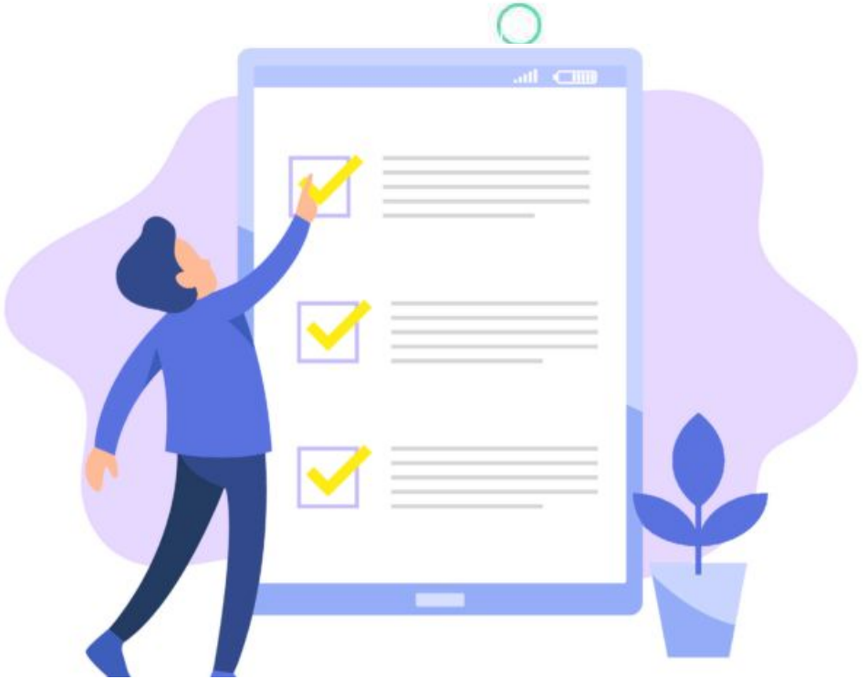
1. Desde la página principal del repositorio en GitHub, haz clic en el botón "New pull request" (Nuevo pull request).
2. Selecciona la rama de la cual deseas fusionar los cambios (rama base) y la rama que contiene los cambios que deseas fusionar (rama comparada).



Administrando Pull Request

Crear un Pull Request:

3. Proporciona un título descriptivo y una descripción detallada de los cambios realizados en la rama comparada.
4. Revisa los cambios y, si es necesario, agrega comentarios o realiza modificaciones adicionales.
5. Finalmente, haz clic en el botón "Create pull request" (Crear pull request) para enviar el Pull Request.

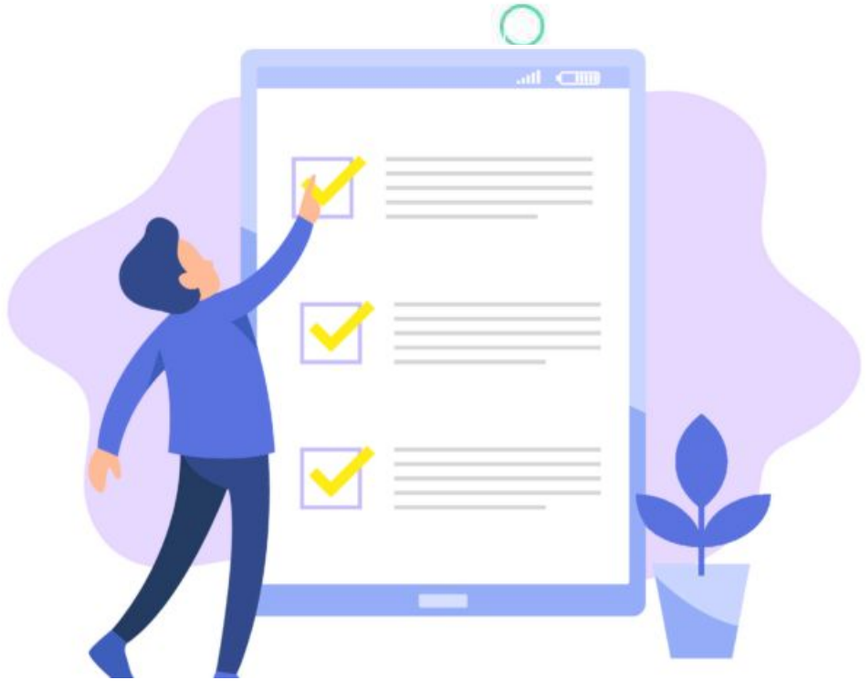


Administrando Pull Request

Revisar y comentar en el Pull Request:

Otros colaboradores pueden revisar tus cambios y dejar comentarios en el Pull Request.

Podes responder a los comentarios, hacer modificaciones adicionales o discutir los cambios propuestos antes de que se fusionen.

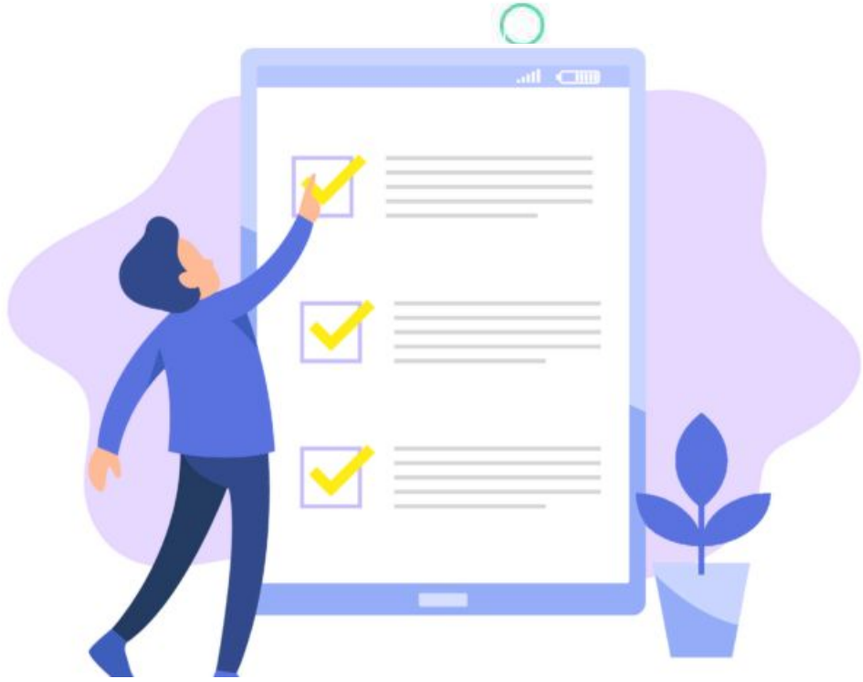


Administrando Pull Request

Fusionar el Pull Request:

Una vez que los cambios en el Pull Request hayan sido revisados y aprobados, podes fusionarlos en la rama base.

1. En la página del Pull Request, haz clic en el botón "Merge pull request" (Fusionar pull request) para incorporar los cambios a la rama base.
2. Podes seleccionar opciones adicionales, como "Squash and merge" (Fusionar y comprimir) o "Rebase and merge" (Fusionar y rebase), dependiendo del flujo de trabajo preferido.
3. Confirma la fusión del Pull Request.
4. Cerrar el Pull Request



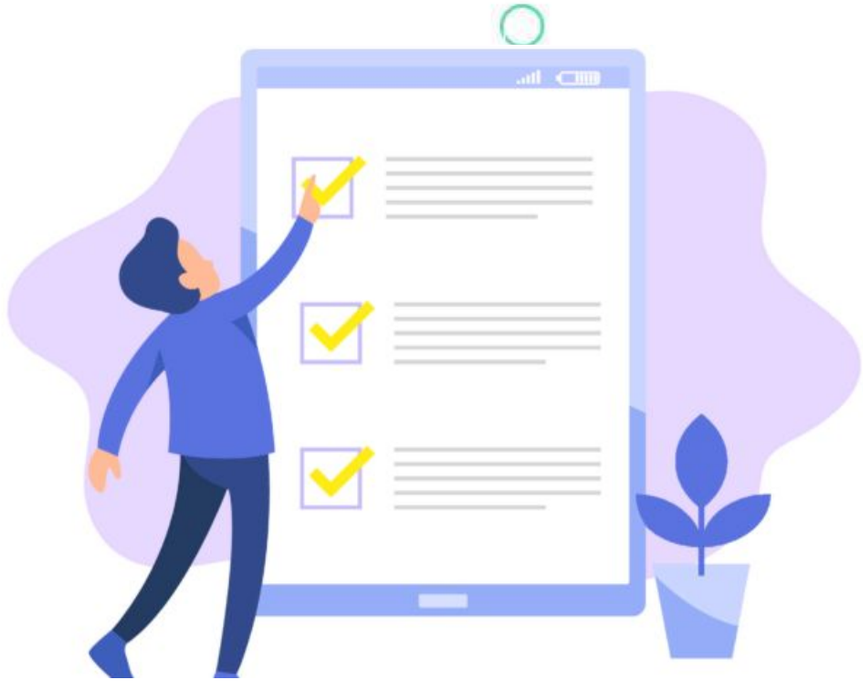
Administrando Pull Request

Fusionar el Pull Request:

Después de fusionar los cambios, podes cerrar el Pull Request para indicar que se ha completado.

En la página del Pull Request, haz clic en el botón "**Close pull request**" (Cerrar pull request) para cerrarlo.

Estos son los pasos básicos para administrar Pull Requests en GitHub. Recuerda que el flujo de trabajo puede variar según las preferencias y políticas del equipo. Además, GitHub proporciona características adicionales, como la posibilidad de asignar revisores, integración continua y otras opciones avanzadas para mejorar la colaboración en el desarrollo de proyectos.



Configuración de los editores de código

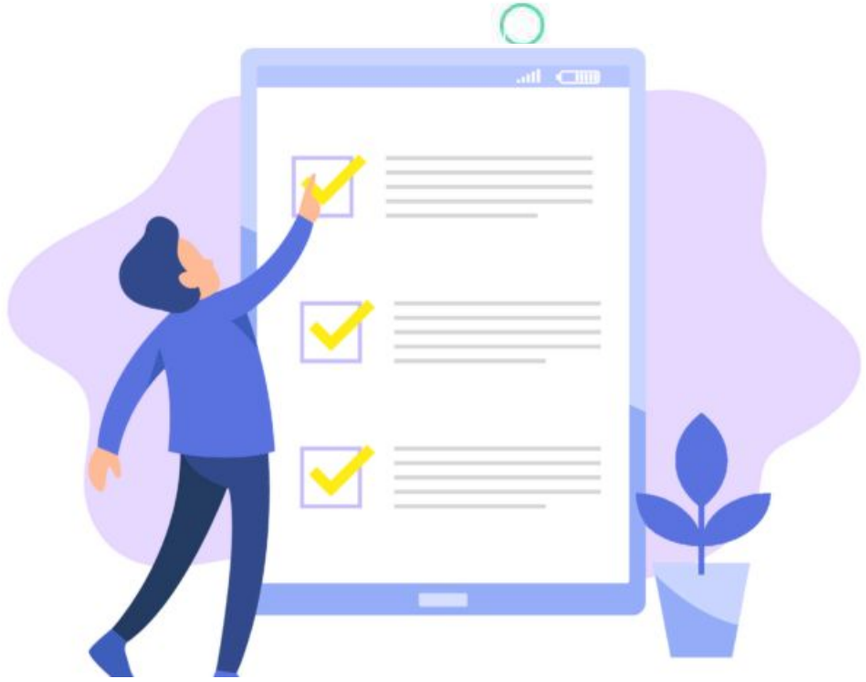
Configuración de los editores de código

¿Por qué es importante configurar el VCS?

Configurar el sistema de control de versiones (VCS), como Git, es importante por varias razones:

1. **Gestión de versiones**

El VCS te permite realizar un seguimiento de los cambios realizados en tu proyecto a lo largo del tiempo. Podes ver el historial de cambios, quién hizo cada cambio y revertir a versiones anteriores si es necesario. Esto es útil para mantener un registro de las modificaciones y facilitar la colaboración en equipos de desarrollo.

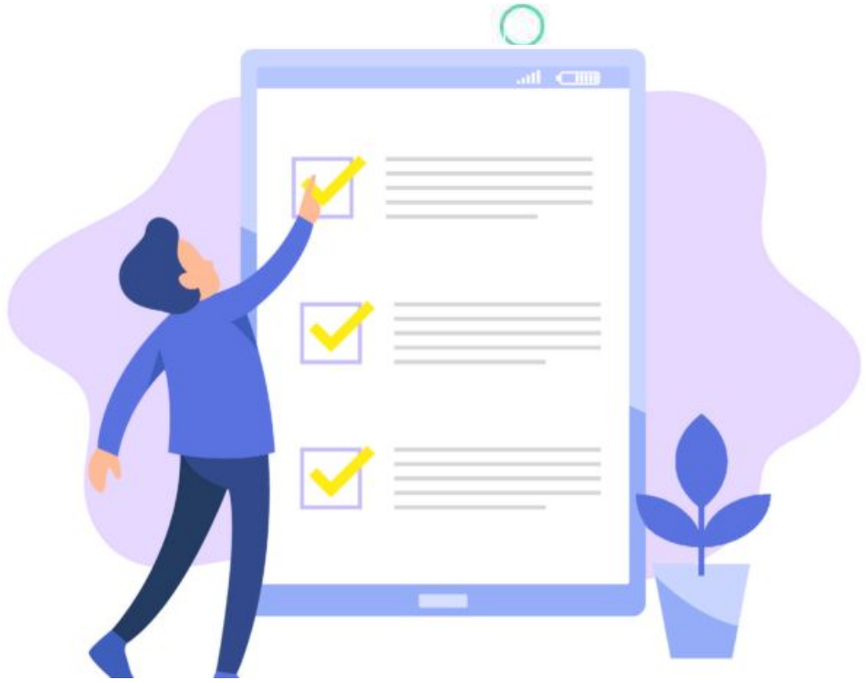


Configuración de los editores de código

¿Por qué es importante configurar el VCS?

2. Colaboración en equipo

Configurar el VCS en los editores de código te permite colaborar eficientemente con otros desarrolladores en un proyecto. Puedes trabajar en paralelo en diferentes ramas, fusionar cambios y resolver conflictos de manera efectiva. Además, puedes revisar y comentar los cambios propuestos por otros miembros del equipo antes de fusionarlos en la rama principal.

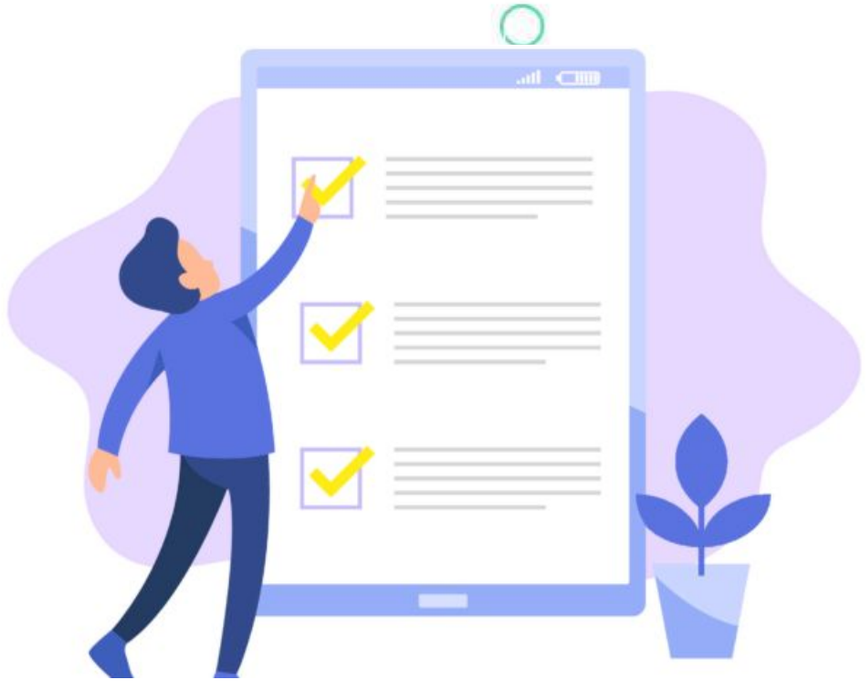


Configuración de los editores de código

¿Por qué es importante configurar el VCS?

3. Seguridad y respaldo

Al utilizar un VCS, como Git, tus cambios se almacenan de forma segura en el repositorio remoto. Esto proporciona un respaldo para tu código y evita la pérdida de trabajo en caso de fallos o errores. Además, puedes configurar reglas de acceso y permisos para proteger tu repositorio y controlar quién puede realizar cambios y revisiones.

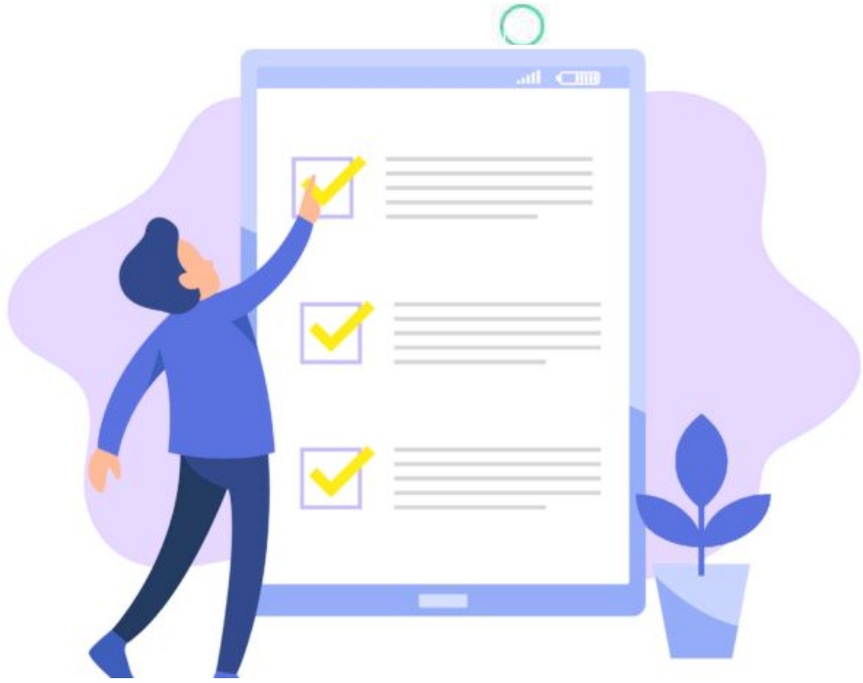


Configuración de los editores de código

¿Por qué es importante configurar el VCS?

4. Gestión de ramas y características

El VCS te permite crear ramas separadas para trabajar en nuevas características, solucionar errores o experimentar con cambios. Puedes trabajar en una rama sin afectar directamente la rama principal y fusionar los cambios cuando estén listos. Esto facilita la gestión de diferentes versiones de tu proyecto y te permite probar y desarrollar nuevas funcionalidades de forma aislada.

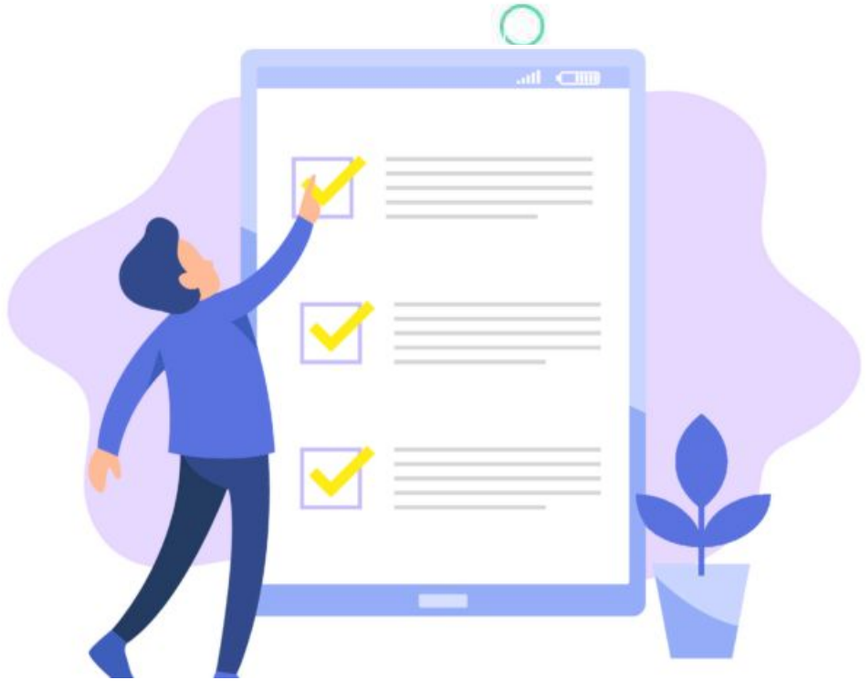


Configuración de los editores de código

¿Por qué es importante configurar el VCS?

5. Historial y análisis de cambio

Configurar el VCS te permite analizar el historial de cambios en tu proyecto, lo que puede ser útil para comprender cómo ha evolucionado tu código, identificar y solucionar problemas, y realizar análisis de rendimiento. Puedes ver quién hizo cada cambio, cuándo y por qué, lo que facilita el seguimiento de la evolución del proyecto.



Configuración de los editores de código

¿Cómo configurarlo?

La configuración del sistema de control de versiones (VCS, por sus siglas en inglés) en los editores de código te permite integrar y utilizar Git para gestionar tu proyecto.

Configuración de los editores de código

Iniciar un repositorio usando Visual Studio Code

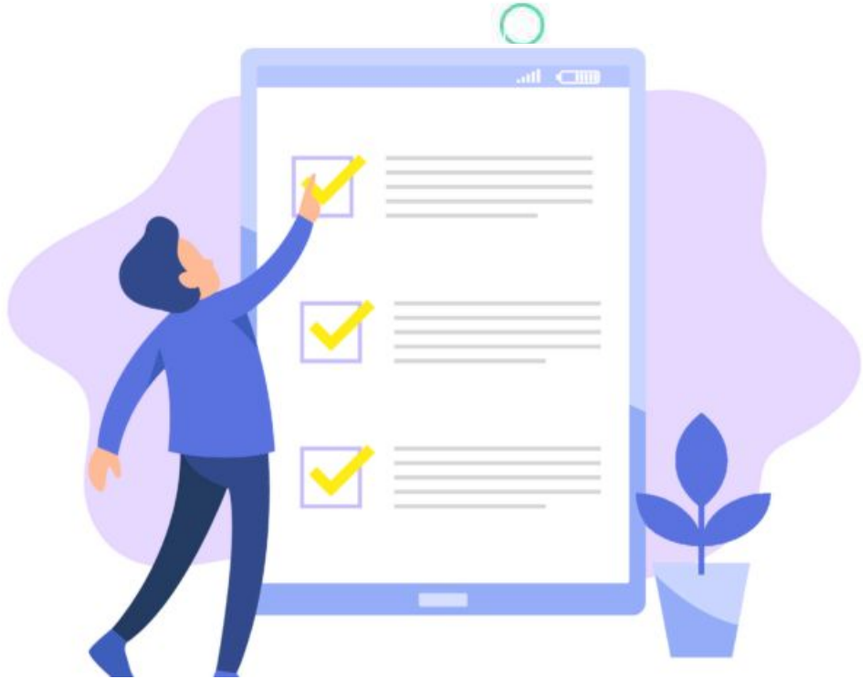
Para configurar Git y GitHub en Visual Studio Code, sigue estos pasos:

1. Instalar Visual Studio Code:

Si no tienes Visual Studio Code, puedes descargarlo e instalarlo desde el sitio oficial: https://code.visualstudio.com/

2. Abrir Visual Studio Code:

Una vez que hayas instalado Visual Studio Code, ábrelo en tu computadora.

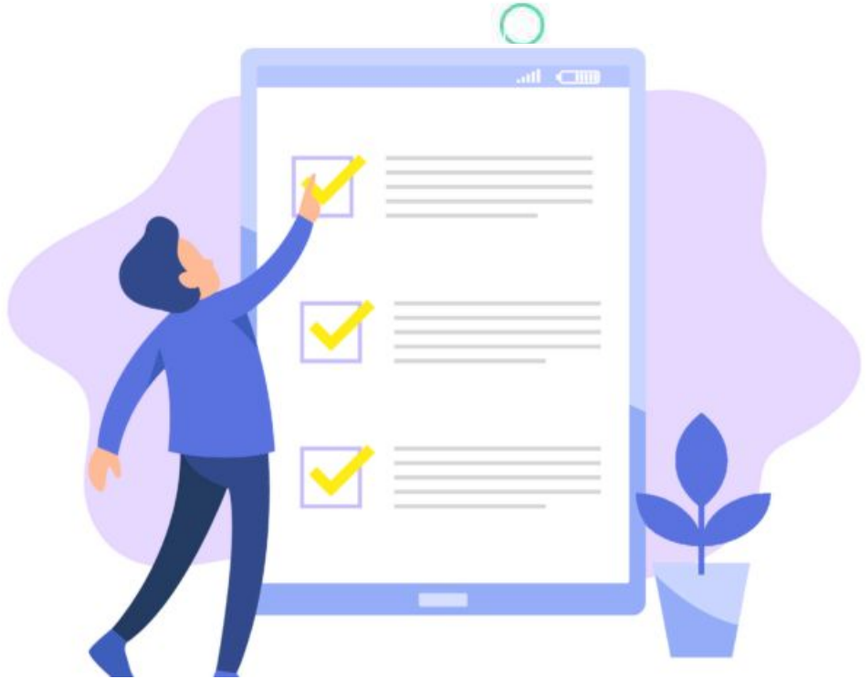


Configuración de los editores de código

Iniciar un repositorio usando Visual Studio Code

3. Instalar la extensión de Git en Visual Studio Code:

Haz clic en el ícono de "Extensions" (cuarta opción en la barra lateral vertical, o simplemente usa el atajo Ctrl + Shift + X). Busca "Git" en el cuadro de búsqueda y selecciona "Git" de la lista de resultados. Luego, haz clic en el botón "Install" para instalar la extensión.



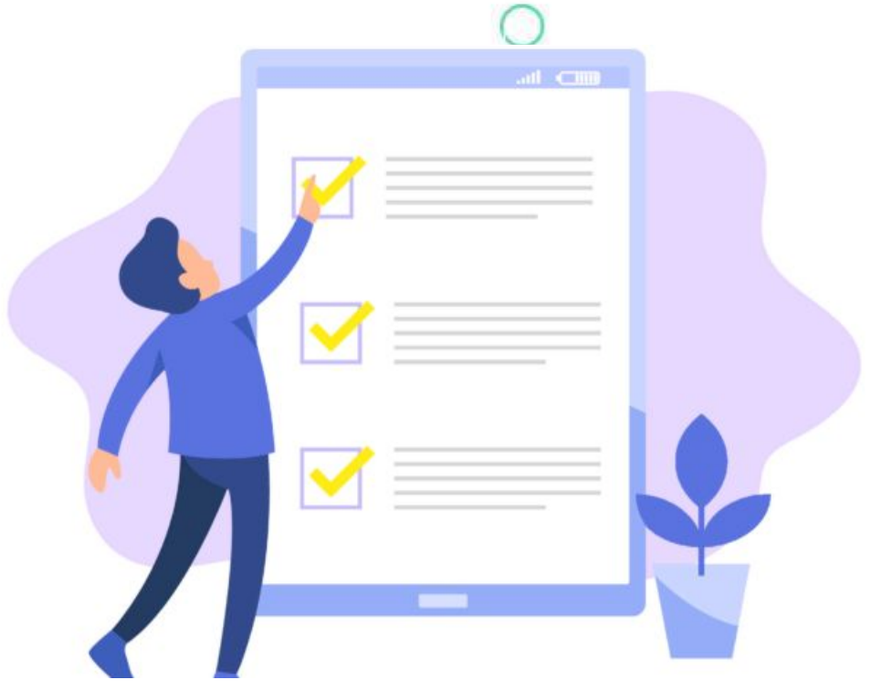
Configuración de los editores de código

Iniciar un repositorio usando Visual Studio Code

4. Configurar Git en Visual Studio Code:

Asegúrate de que Visual Studio Code esté configurado para usar Git correctamente. Para ello, sigue estos pasos:

- Abre Visual Studio Code y navega a "View" (Vista) en la barra de menú.
- Selecciona "Command Palette..." (Paleta de comandos) o usa el atajo Ctrl + Shift + P.
- Escribe "Git: Set Global Git Configuration" en el cuadro de búsqueda y selecciona la opción.
- Se te pedirá que ingreses tu nombre de usuario de GitHub y tu dirección de correo electrónico. Esto configurará Git globalmente en tu computadora.

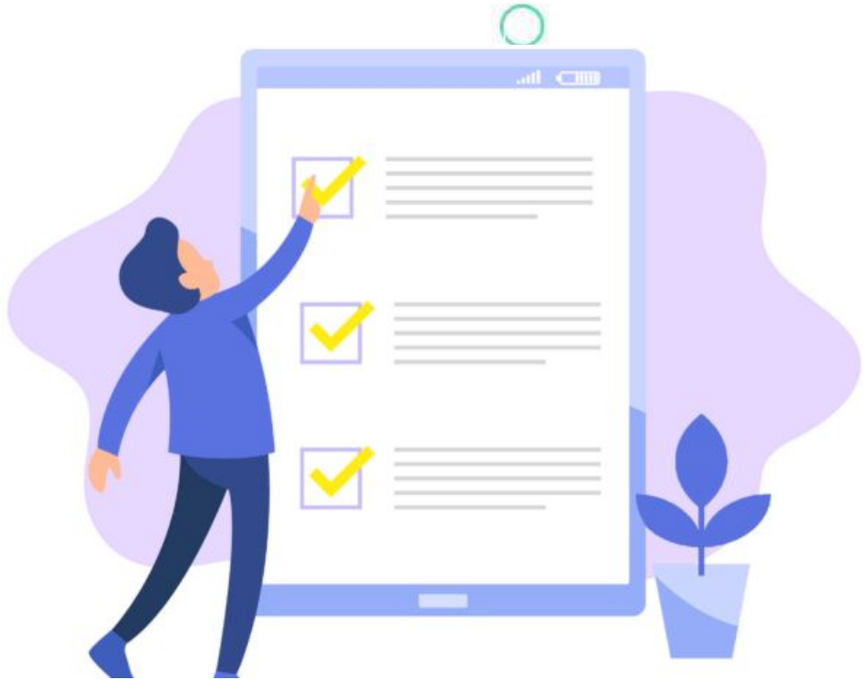


Configuración de los editores de código

Iniciar un repositorio usando Visual Studio Code

5. Clonar un repositorio desde GitHub:

- Ve a la página del repositorio en GitHub que deseas clonar en Visual Studio Code.
- Haz clic en el botón verde "Code" y copia la URL del repositorio.
- Vuelve a Visual Studio Code y selecciona "View" (Vista) en la barra de menú.
- Elige "Command Palette..." (Paleta de comandos) o usa el atajo Ctrl + Shift + P.
- Escribe "Git: Clone" en el cuadro de búsqueda y selecciona la opción.
- Pega la URL del repositorio que copiaste



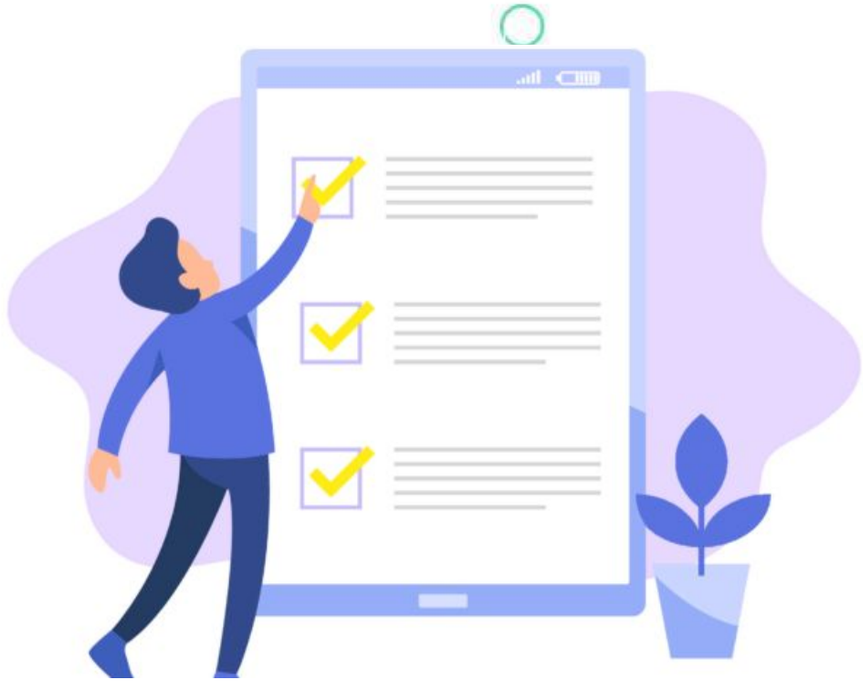
Configuración de los editores de código

Iniciar un repositorio usando Visual Studio Code

6. Configurar GitHub en Visual Studio Code:

Para que puedas trabajar con GitHub directamente desde Visual Studio Code, necesitarás configurar la integración de GitHub. Sigue estos pasos:

- Ve a "View" (Vista) en la barra de menú de Visual Studio Code.
- Selecciona "Command Palette..." (Paleta de comandos) o usa el atajo Ctrl + Shift + P.
- Escribe "Sign in to GitHub" en el cuadro de búsqueda y selecciona la opción.

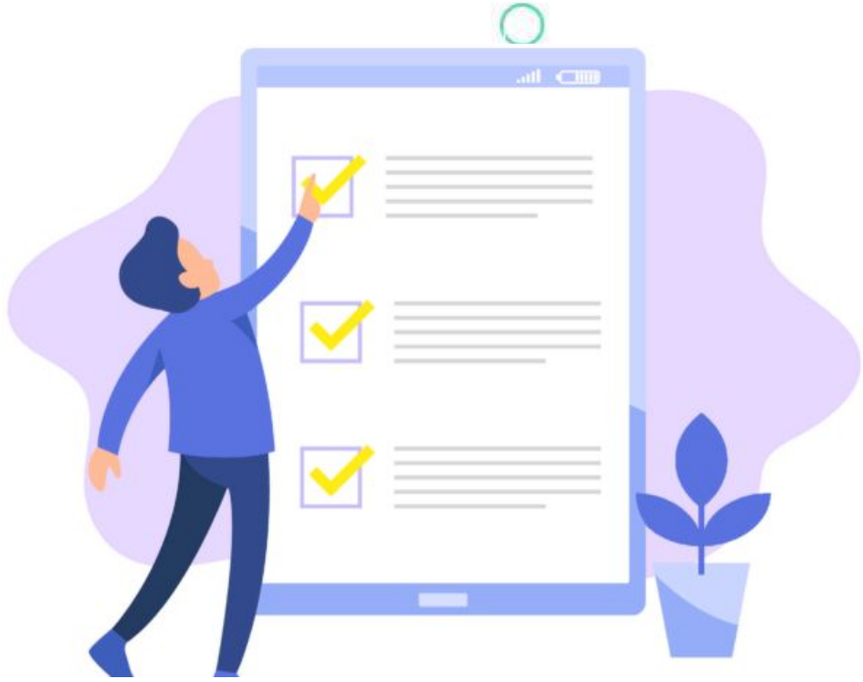


Configuración de los editores de código

Iniciar un repositorio usando Visual Studio Code

6. Configurar GitHub en Visual Studio Code:

- Se abrirá una ventana emergente solicitando tu autorización para acceder a GitHub. Haz clic en "Sign in with your browser" (Iniciar sesión con tu navegador) y sigue las instrucciones para iniciar sesión en tu cuenta de GitHub.
- Una vez que hayas iniciado sesión, recibirás un código de autorización. Copia el código y pégalo en la ventana emergente de Visual Studio Code.



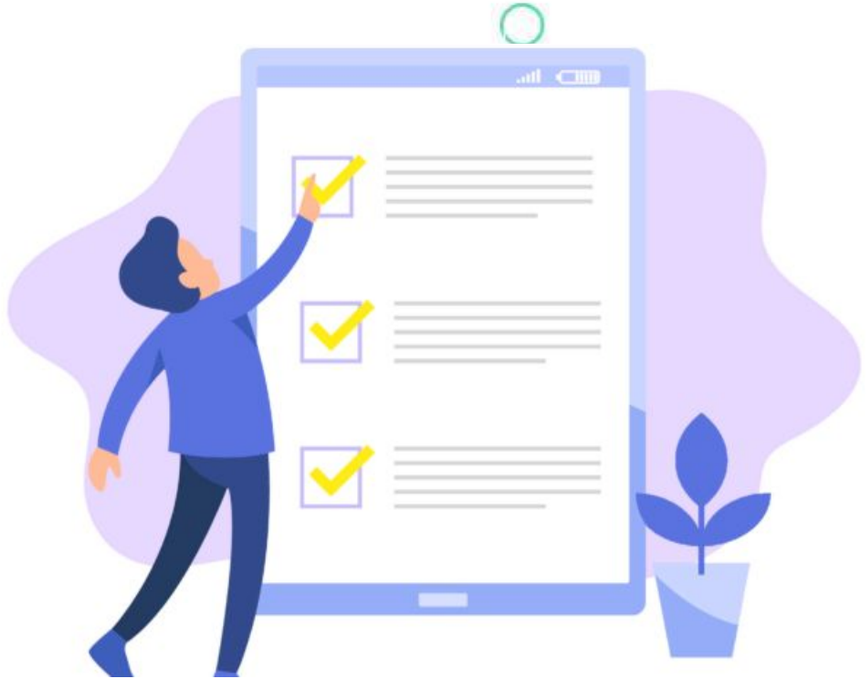
7. ¡Listo para usar!

Ahora que has configurado Git y GitHub en Visual Studio Code, podrás clonar, crear, editar y gestionar tus repositorios directamente desde el entorno de desarrollo.

Configuración de los editores de código

¿Cómo configurarlo?

En IntelliJ IDEA, también puedes configurar Git y GitHub para gestionar tus repositorios y colaborar en proyectos.

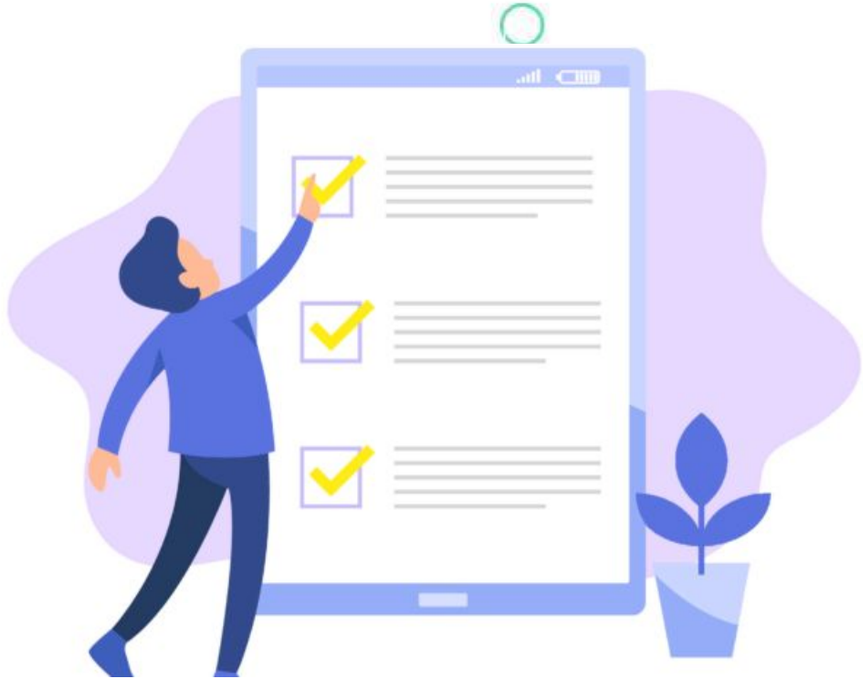


Configuración de los editores de código

1. Instalar IntelliJ IDEA:

Si aún no tienes IntelliJ IDEA, descárgalo e instálalo desde el sitio oficial:

https://www.jetbrains.com/idea/.

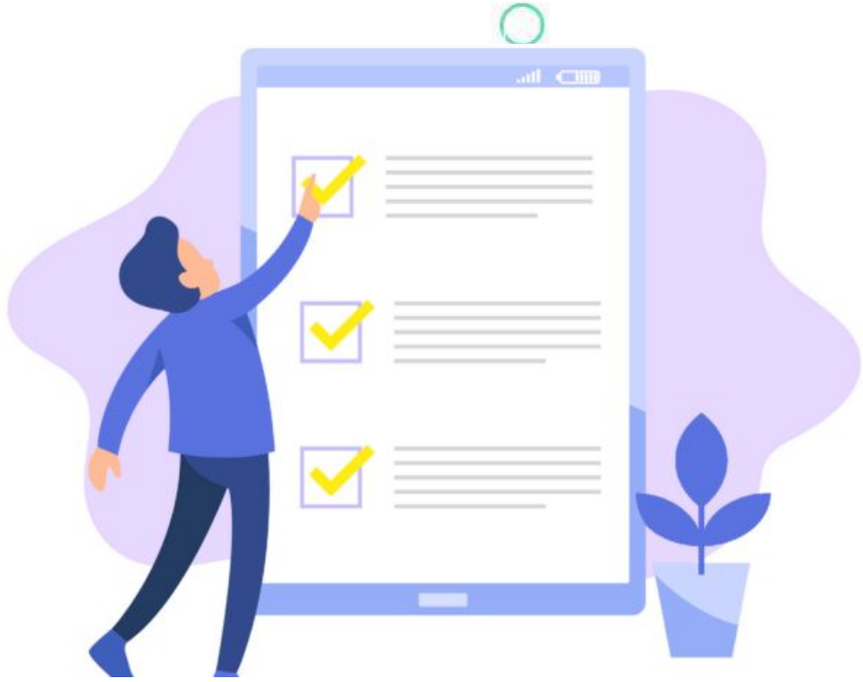


Configuración de los editores de código

2. Configurar Git en IntelliJ IDEA:

Una vez que tengas Git instalado, sigue estos pasos para configurarlo en IntelliJ IDEA:

- Abre IntelliJ IDEA en tu computadora.
- Ve a "File" (Archivo) en la barra de menú y selecciona "Settings" (Configuración) o usa el atajo Ctrl + Alt + S.
- En la ventana de configuración, busca "Version Control" (Control de versiones) en el panel izquierdo y selecciona "Git" en la lista desplegable.
- Asegúrate de que el campo "Path to Git executable" (Ruta al ejecutable de Git) apunte al archivo ejecutable de Git en tu computadora. Si Git se instaló correctamente, IntelliJ IDEA debería detectar automáticamente la ruta. Si no, deberás proporcionar la ruta manualmente.

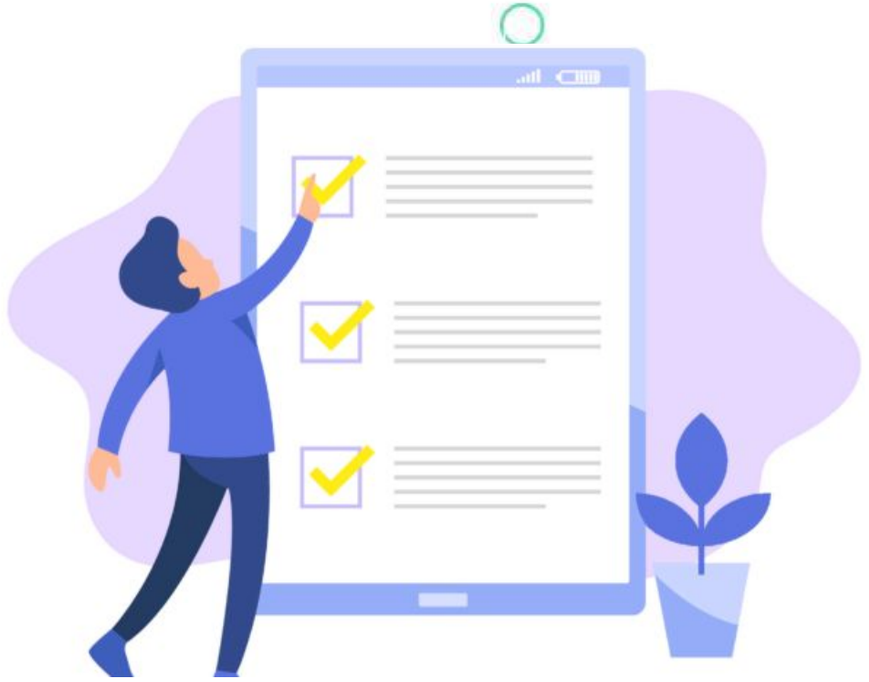


Configuración de los editores de código

3. Configurar GitHub en IntelliJ IDEA:

Para configurar GitHub en IntelliJ IDEA y trabajar con repositorios remotos, sigue estos pasos:

- Ve a "File" (Archivo) en la barra de menú y selecciona "Settings" (Configuración) o usa el atajo Ctrl + Alt + S.
- En la ventana de configuración, busca "Version Control" (Control de versiones) en el panel izquierdo y selecciona "GitHub" en la lista desplegable.
- Haz clic en "Configure GitHub" y sigue las instrucciones para iniciar sesión en tu cuenta de GitHub y autorizar IntelliJ IDEA para acceder a tus repositorios..

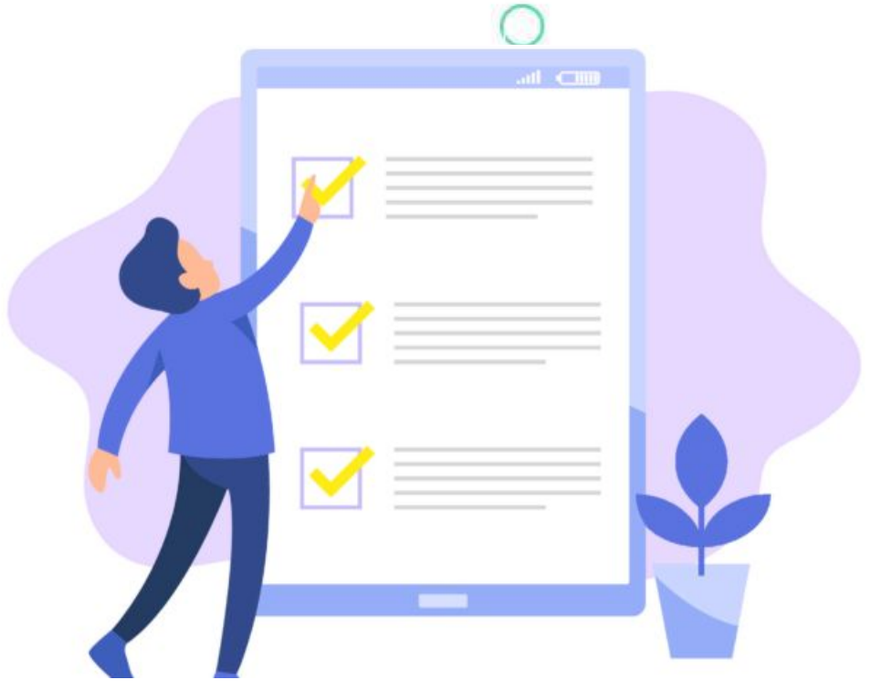


Configuración de los editores de código

4. Clonar un repositorio desde GitHub:

Para clonar un repositorio desde GitHub en IntelliJ IDEA, sigue estos pasos:

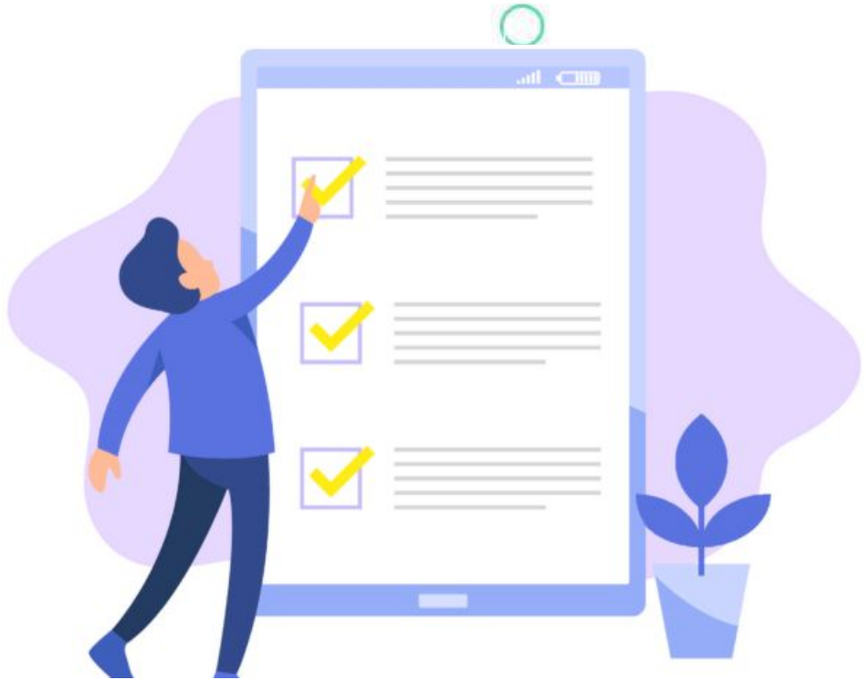
- Ve a "VCS" en la barra de menú y selecciona "Get from Version Control" (Obtener desde el control de versiones) o usa el atajo Ctrl + Alt + V.
- Pega la URL del repositorio de GitHub en el campo "URL" y elige la ubicación donde deseas clonar el repositorio en tu computadora.
- Haz clic en "Clone" para clonar el repositorio.



Configuración de los editores de código

5. Crear, editar y gestionar archivos:

Una vez que hayas clonado el repositorio, podrás crear, editar y gestionar archivos en IntelliJ IDEA. Los cambios que realices se reflejarán en tu repositorio local.

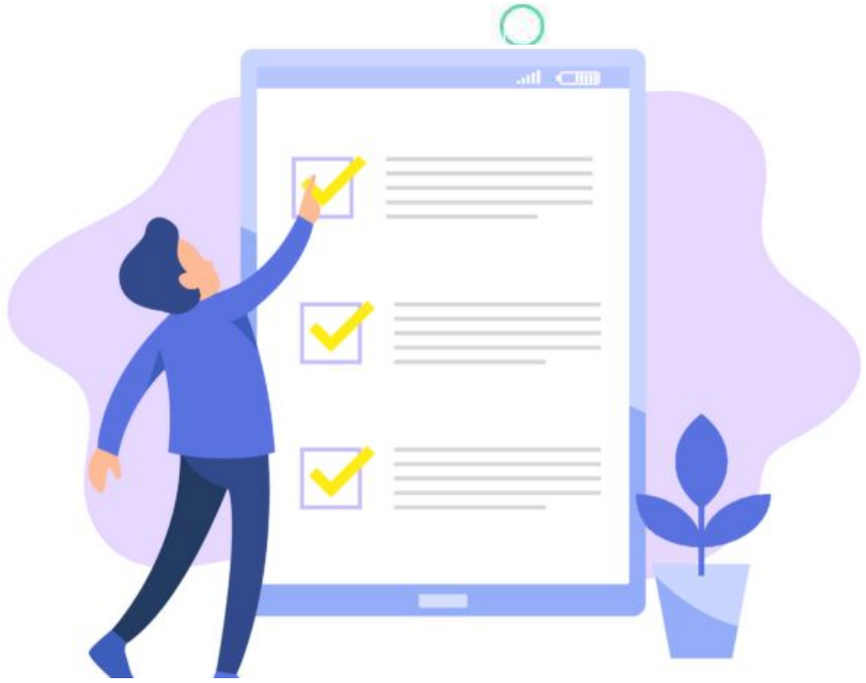


Configuración de los editores de código

6. Hacer commits y Push:

Para hacer commits de tus cambios y enviarlos al repositorio remoto en GitHub, sigue estos pasos:

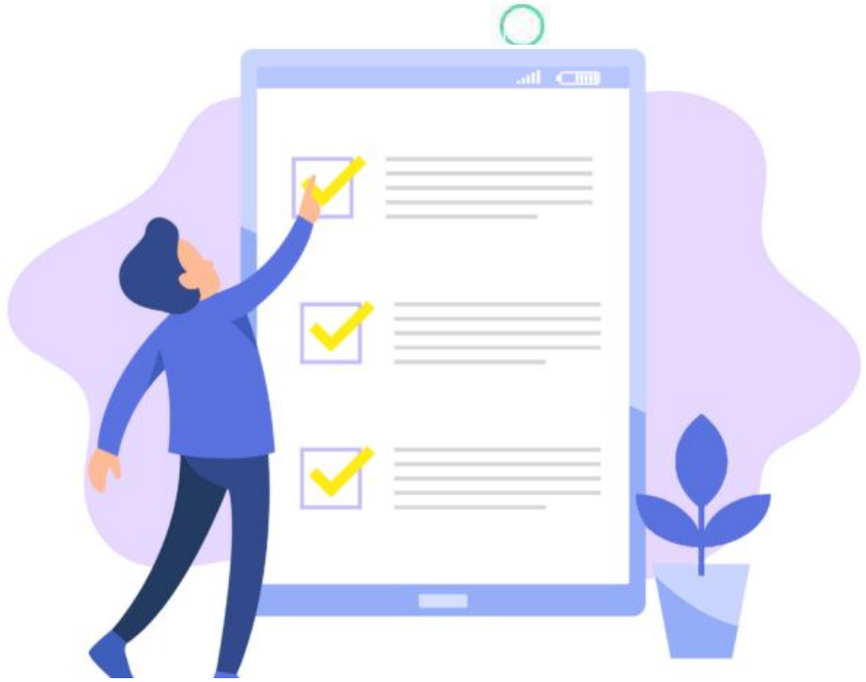
- Haz clic derecho en el archivo o en la carpeta que deseas commitear y selecciona "Git" en el menú contextual.
- Elige "Commit File" para commitear un archivo específico o "Commit Directory" para commitear todos los cambios en una carpeta.
- Escribe un mensaje de commit descriptivo y haz clic en "Commit".
- Luego, para enviar tus commits al repositorio remoto en GitHub, ve a "VCS" en la barra de menú y selecciona "Git" > "Push" o usa el atajo Ctrl + Shift + K.



Configuración de los editores de código

7. Listo:

Ahora has configurado Git y GitHub en IntelliJ IDEA y estás listo para gestionar tus repositorios y colaborar en proyectos desde el entorno de desarrollo. Disfruta de tu experiencia de desarrollo con IntelliJ IDEA y Git/GitHub.

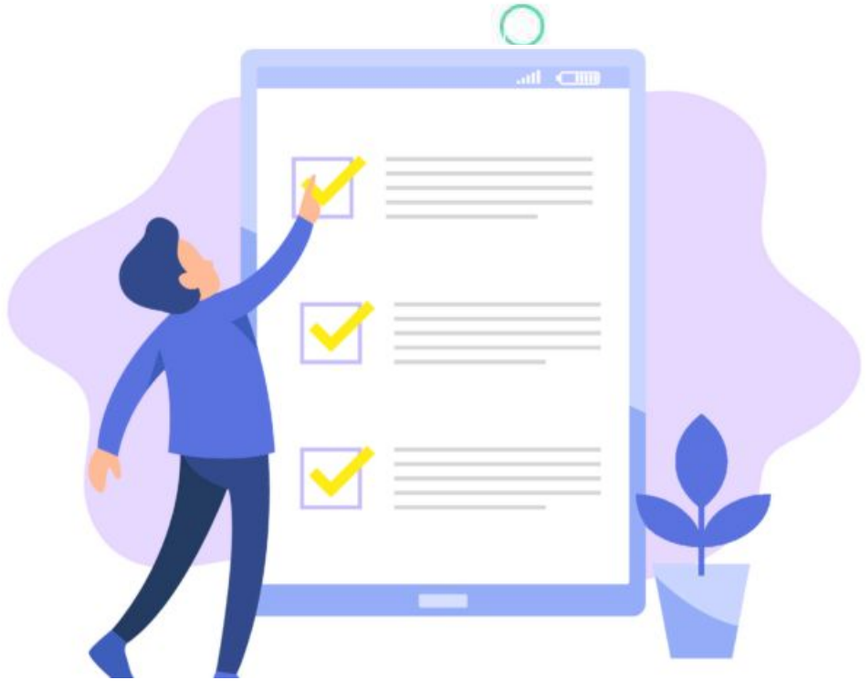


Markdown

Markdown

Markdown es una sintaxis de marcado ligera que se utiliza comúnmente para dar formato y estructurar documentos de texto plano.

Es ampliamente utilizado en la documentación de proyectos, blogs, foros y plataformas de colaboración como GitHub debido a su simplicidad y facilidad de uso.

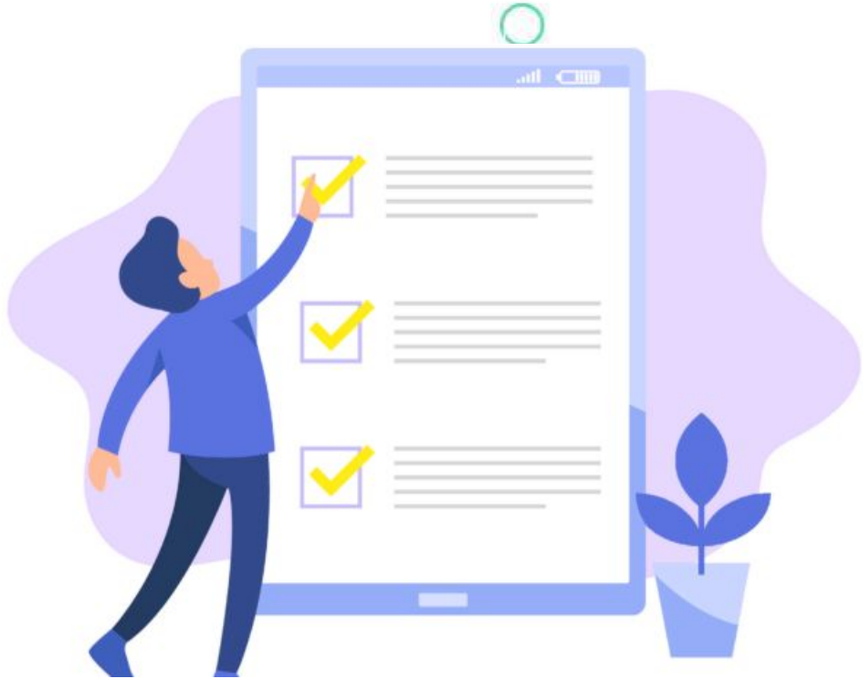


Markdown

Algunos elementos y características comunes de Markdown:

Encabezados: Se pueden crear encabezados utilizando hashtags (#). Cuantos más hashtags uses, más grande será el encabezado (# para encabezado de nivel 1, ## para encabezado de nivel 2, y así sucesivamente).

Texto en Negrita y Cursiva: Puedes enfatizar el texto utilizando asteriscos (*) o guiones bajos (_) alrededor del texto. Dos asteriscos o guiones bajos para negrita, uno para cursiva.



Markdown

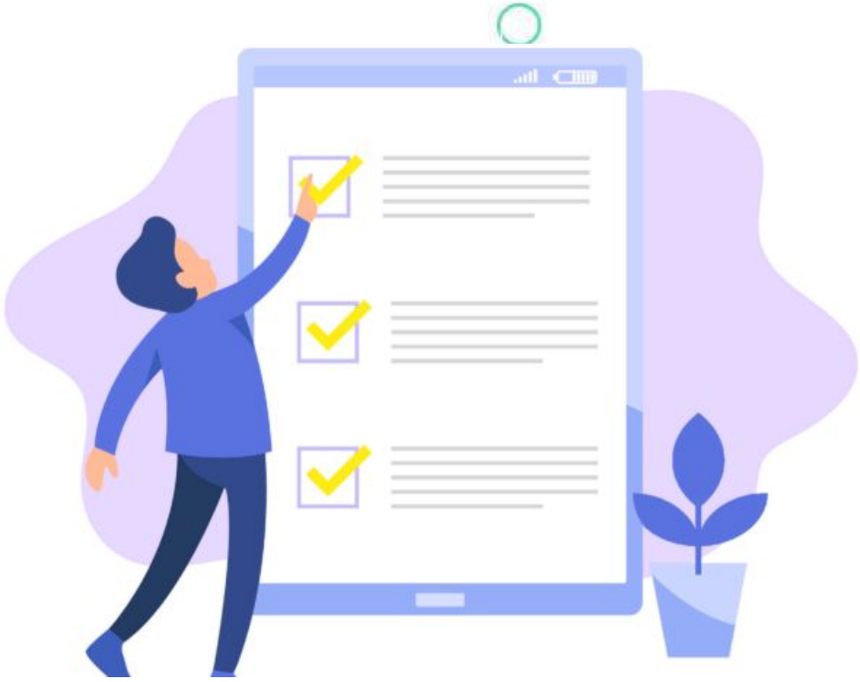
Algunos elementos y características comunes de Markdown:

Listas: Puedes crear listas ordenadas y no ordenadas utilizando asteriscos, guiones o números.

Enlaces: Los enlaces se crean con el formato [Texto del Enlace](URL).

Imágenes: Las imágenes se insertan de manera similar a los enlaces, pero con un signo de exclamación antes del corchete.

Citas: Las citas se crean utilizando el símbolo mayor que (>). Puedes anidar citas para respuestas múltiples.



Evaluación Integradora ✨

Trabajo Integrador del Módulo 💪🏽

Iremos completándolo progresivamente clase a clase

CLICK AQUÍ
para ver la consigna completa

Ejercicio

Repositorio GITHUB

Repositorio GITHUB

Breve descipción

Contexto: 🙌

Continuamos trabajando en el proyecto para la wallet digital. El objetivo es utilizar utilizar GitHub para la gestión de código y colaboración.

Consigna: ✍️

Usar los conocimientos vistos en la clase de hoy para subir y gestionar el proyecto con GitHub.

Tiempo🕛: 20 minutos

Repositorio GITHUB

Breve descipción

Paso a paso: ⚙️

Gestión en GitHub:

- Crea un repositorio en GitHub para el proyecto de la wallet digital.
- Realiza commits regulares con descripciones claras y significativas de tus cambios.
- Utiliza Pull Requests para solicitar y revisar cambios en el proyecto.
- Documenta el proyecto utilizando Markdown para proporcionar información sobre el funcionamiento y la estructura del código.

¿Alguna consulta?

RESUMEN

¿Qué logramos en esta clase?

- ✓ *Comprender los fundamentos de GitHub como una plataforma de gestión de control de versiones.*
- ✓ *Dominar la gestión de repositorios remotos y la sincronización de cambios mediante Push y Pull.*
- ✓ *Distinguir entre Fetch y Pull y aplicar el concepto apropiado según la situación.*
- ✓ *Aprender a clonar un repositorio existente para colaborar en proyectos.*
- ✓ *Adquirir habilidades para administrar Pull Requests y revisar y fusionar cambios de colaboradores.*

#WorkingTime

Continuemos ejercitando

¡Antes de cerrar la clase! Te invitamos a: 👇👇👇

1. Repasar nuevamente la grabación de esta clase
2. Revisar el material compartido en la plataforma de Moodle (lo que se vio en clase y algún ejercicio adicional)
   a. Material 1: Lección 7: Fundamentos de GIT y GITHUB: 14-25
   b. Material 2: Working Time de la Lección N° 7
   c. Material 3: Ponte a Prueba de la Lección N° 7
3. Traer al próximo encuentro, todas tus dudas y consultas para verlas antes de iniciar nuevo tema.

¡Muchas Gracias!

Nos vemos en la próxima clase 👋🏻

English Always | alkemy