TALLER # 1

Git, GitHub y Django

Contenido

INTRODUCCIÓN A GIT Y GITHUB

Acerca del control de versiones

Un Sistema de Control de Versiones (Version Control System - VCS) registra los cambios realizados en un archivo o conjunto de archivos a lo largo del tiempo para poder recuperar versiones específicas más adelante. Un VCS permite revertir archivos seleccionados a un estado anterior, revertir todo el proyecto a un estado estado anterior, comparar los cambios a lo largo del tiempo y ver quién modificó por última vez algo que podría estar causando un problema. Existen VCS locales, centralizados y distribuidos.

¿Qué es Git?

Git es un VCS distribuido, diseñado principalmente para el manejo eficiente de proyectos de software. Permite a los usuarios llevar un registro de los cambios realizados en los archivos a lo largo del tiempo, facilitando la colaboración entre diferentes personas en un proyecto y la gestión de diferentes versiones de este. Cada vez que se hace un *commit* o se guarda el estado de un proyecto, Git básicamente toma una foto de cómo están todos los archivos en ese momento y almacena una referencia a esa foto. Para ser eficiente, si los archivos no han cambiado, Git no almacena el archivo de nuevo, sólo un enlace al archivo idéntico anterior que ya ha almacenado. La documentación oficial se encuentra disponible en: https://git-scm.com/.

Git tiene tres estados principales en los que pueden residir los archivos:

- **Modified** (modificado): significa que el archivo ha cambiado, pero aún no se ha confirmado para ser almacenado.
- **Staged** (preparado): significa que el archivo ha sido modificado en su versión actual para ir en la próxima foto de confirmación.
- **Commited** (confirmado): significa que los datos están almacenados de forma segura en local.

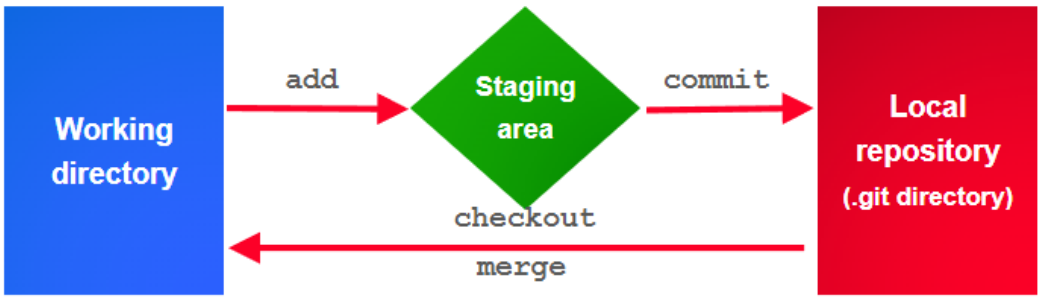
Comandos básicos de Git

- **git init:** crea un subdirectorio llamado .git que contiene los archivos necesarios para hacer el seguimiento a los archivos.
- **git clone <url>:** crea una copia de un repositorio existente, permite colaborar en el proyecto clonado.
- **git status:** muestra el estado de los archivos.
- **git push origin main:** sube los commits locales a un repositorio remoto.
- **git pull:** obtiene cambios desde un repositorio remoto y los fusiona con tu rama local.
- **git branch:** lista todas las ramas en el repositorio y muestra cuál se está utilizando actualmente.
- **git checkout <rama>:** cambia entre ramas.
- **git add <archivo>:** agrega un archivo al área de preparación (staging area) para que pueda ser incluido en el próximo commit.
- **git commit -m "<mensaje>":** crea un commit con los cambios en el área de preparación, incluyendo un mensaje que describe los cambios realizados.
- **git merge <rama>:** fusiona cambios desde otra rama a la rama actual.

La figura representa un flujo de trabajo básico en Git, ***Working directory*** es donde se realizan los cambios en los archivos. ***Staging area*** es eñl área intermedia donde se preparan los cambios antes de confirmarlos. ***Local repository*** es el almacén local donde se guardan los cambios confirmados.

Las acciones principales son:

1. **add**: mueve cambios del directorio de trabajo al área de preparación.
2. **commit**: guarda los cambios del área de preparación en el repositorio local.
3. **checkout**: actualiza el directorio de trabajo con versiones del repositorio local.
4. **merge**: combina cambios del repositorio local en el directorio de trabajo.

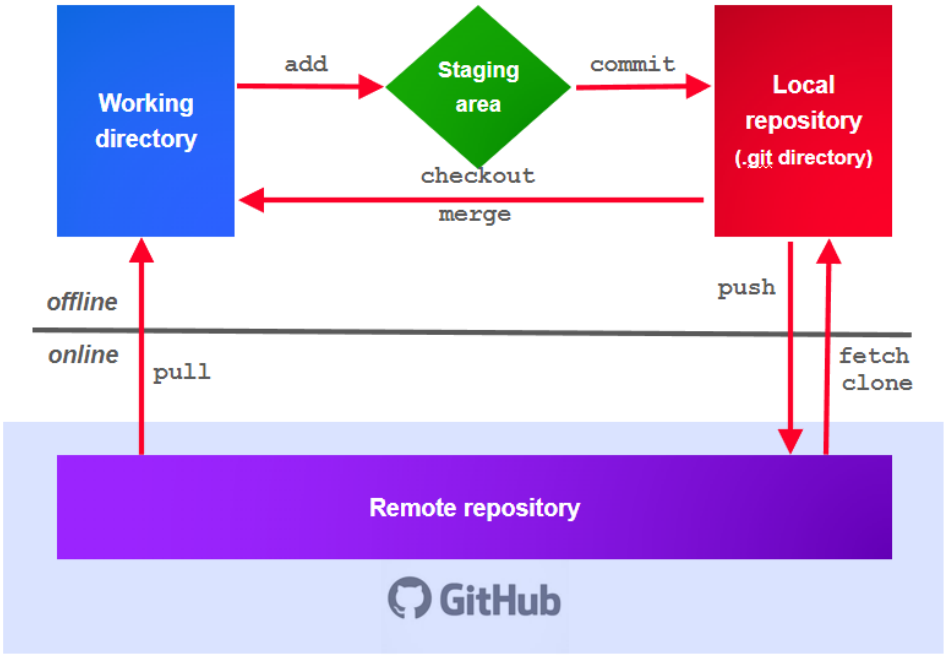


¿Qué es GitHub?

GitHub es una plataforma para alojar proyectos utilizando Git. Permite crear repositorios, los cuales son lugares virtuales alojados en la nube en donde los usuarios pueden almacenar cualquier tipo de archivos. Esta plataforma está disponible en: github.com.

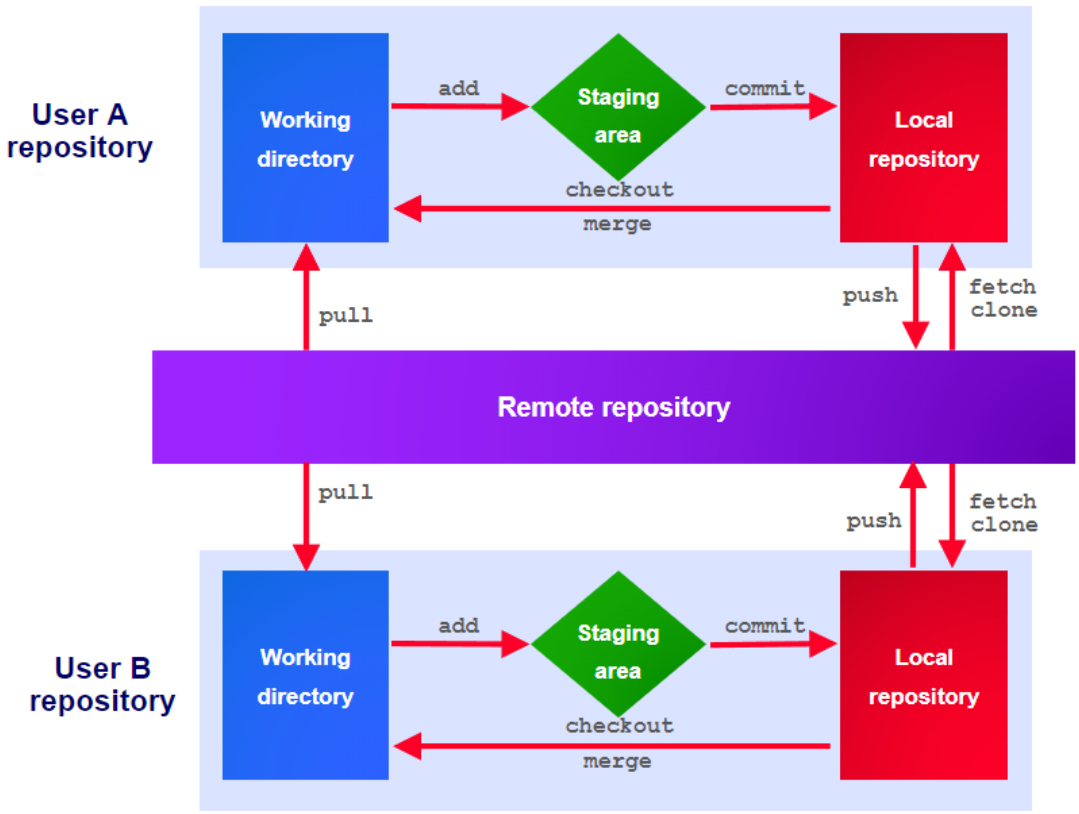
La figura amplía el flujo de trabajo en Git al incluir la interacción con el ***Remote repository***. Cuando se trabaja con un repositorio remoto en Git, las acciones principales suelen seguir este orden lógico:

1. **clone:** se copia el repositorio remoto completo a tu máquina local, incluyendo el historial de versiones.
2. **checkout:** se selecciona la rama en la que se va a trabajar.
3. **add:** se agregan los archivos modificados al área de *staging*.
4. **commit:** se guardan los cambios en el repositorio local con un mensaje descriptivo.
5. **pull:** se traen los últimos cambios del repositorio remoto y se integran con los cambios locales.
6. **merge** (si es necesario): se combinan los cambios traídos con los locales si hay diferencias.
7. **push:** se envían los cambios confirmados al repositorio remoto.



La figura ilustra el flujo de trabajo en Git cuando **dos usuarios colaboran** mediante un repositorio remoto. Cada usuario (A y B) tiene su propio entorno local. Lo más destacado es cómo se visualiza la interacción entre usuarios a través del repositorio remoto. Las acciones clave incluyen:

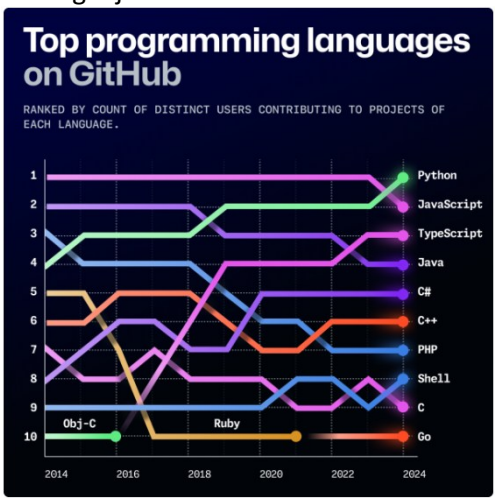
- **push**: para compartir cambios con el repositorio remoto.
- **pull**, **fetch** y **clone**: para obtener actualizaciones del repositorio remoto.
- **merge**: para integrar cambios de distintas fuentes.



INTRODUCCIÓN A PYTHON Y DJANGO

¿Qué es Python?

Python es un lenguaje de programación de alto nivel (https://www.python.org/). Python posee una licencia de código abierto, que, entre otras cosas, permite que los programadores puedan modificar su código, utilizarlo y/o redistribuirlo libremente sin tener que pagar al autor original. Python se caracteriza por ser un lenguaje de programación amigable y fácil de aprender. Durante los últimos años Python ha estado entre los lenguajes más utilizados.

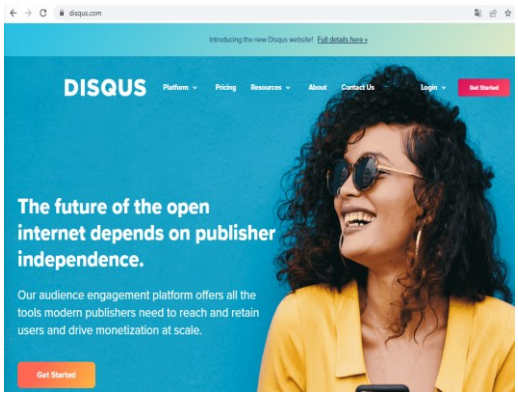


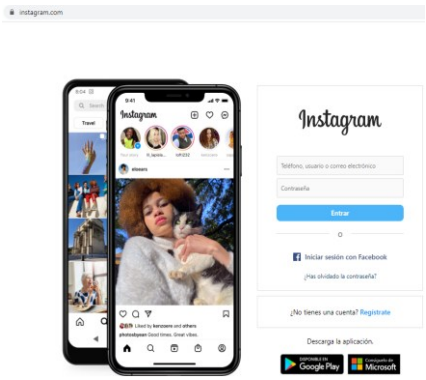
Fuente: Octoverse.

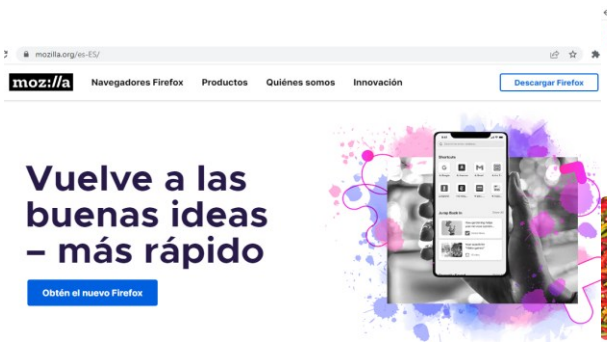
¿Qué es Django?

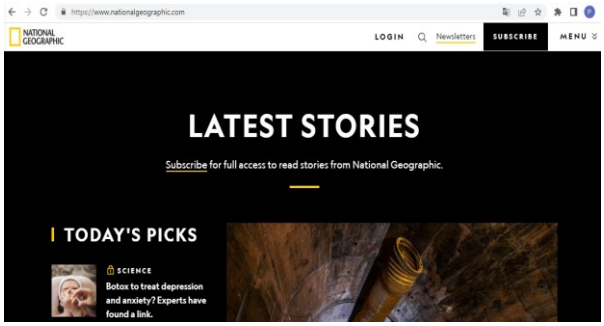
Django es un *framework* gratuito y de código abierto para desarrollar aplicaciones web en Python. Entre sus características están la rapidez para el desarrollo, la seguridad que ofrece y la escalabilidad de las aplicaciones. Django fomenta el desarrollo rápido y pragmático. Sus desarrolladores indican que Django es ridículamente rápido, tranquilizantemente seguro, extremadamente escalable e increíblemente versátil. Todos los detalles sobre el *framework* y su documentación se encuentran disponibles en línea (https://www.djangoproject.com/).

Algunos sitios web que usan Django son: Disqus (https://disqus.com/), Instagram (https://www.instagram.com/), Mozilla (https://www.mozilla.org/es-ES/), Pinterest (https://co.pinterest.com/), National Geographic (https://www.nationalgeographic.com/).







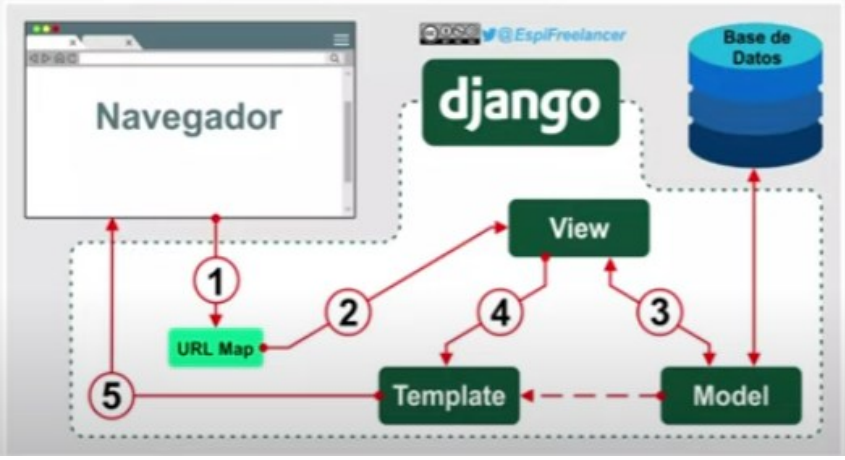


Fuente : https://djangostars.com/blog/10-popular-sites-made-on-django/

Django utiliza el patrón MVT (Model-View-Template). La capa **Model** (modelos) permite estructurar y manipular los datos de la aplicación web, representa los datos y la lógica de la base de datos. La capa **View** (vistas) se usa para encapsular la lógica responsable de procesar la petición de un usuario y devolver la respuesta. La capa **Template** (plantillas) proporciona una sintaxis de fácil diseño para representar la información que se presentará al usuario. Este enfoque ayuda a separar las responsabilidades dentro de una aplicación web, lo que facilita el mantenimiento y la escalabilidad del proyecto.

Esta figura representa el flujo de trabajo del patrón **MVT**, destacando cómo se procesa una solicitud web:

1. El **navegador** envía una solicitud al servidor Django.
2. El **URL Map** determina qué **vista (View)** debe manejar la solicitud.
3. La **vista** consulta el **modelo (Model)** para obtener o modificar datos.
4. Luego, la vista utiliza una **plantilla (Template)** para construir la interfaz que verá el usuario.
5. Finalmente, la **respuesta** se envía de vuelta al navegador.



Fuente: https://www.youtube.com/watch?v=laURuZdJDsU

BIBLIOGRAFÍA

Chacon, S., Straub, B. Pro Git. 2024. https://git-scm.com/book/en/v2

Octoverse. 2024. https://octoverse.github.com/2022/top-programming-languages

Moure, B. Curso de GIT y GITHUB desde CERO para PRINCIPIANTES. https://www.youtube.com/watch?v=3GymExBkKjE

https://twitter.com/Harsa_Dash/status/1750741820135596036?s=20

https://medium.com/@AyushAgrawal_/understanding-django-mvt-architecture-and-view-functions-django-full-course-for-beginners-lesson-39c8da093b44

https://www.youtube.com/watch?v=laURuZdJDsU