GIT

Taller rápido con Midu

- **Git es un control de versiones**
- Registra cada cambio en el código de un proyecto. Hace un histórico, pero no es solo una linea del tiempo
- Hay que pensar más en el multivers cuando hablamos de ramificaciones
- Puedes clonar un árbol y evolucionar el árbol totalmente diferente
- **Github es dónde hospedar ese control de versiones del código en la nube**
- Github tiene a Microsoft detrás
- **Branches** son las ramas
- **Issues** son los problemas encontrados en el código
- Pull requests son las famosas **PR**
    - Son peticiones de cambios (que se integren los cambios)
    - De una rama que sale del main (feature-1) puedo pedir una pull request para que integre los cambios en el main
    - O incluso puedo sacar una rama de feature-1 y pedir una pull request para que integre los cambios en feature-1 que integrará los cambios en el main

Tipica estrategia en github (para contribuir al código abierto)

- Dónde contribuir (para solucionar problemas)
    - **ForGoodFirstIssue.github.com**
    - **GoodFirstIssue.dev/language/typescript**
    - **GoodFirstIssues.com**
- Como creador/mantenedor de un repositorio, puedo aplicar una label como **good-first-issue** a una issue presentada en mi repo
- También se puede colaborar y contribuir en proyectos que te molen
- El documento CONTRIBUTING.md del repo da unas guías para poder contribuir y si aceptan o no contribuciones
- Ir al archivo CONTRIBUTING de tu repo favorito solo a aprender puede ser MUY INTERESANTE
    - Por ejemplo, si quieres que te hagan una PR vas a tener que demostrar que funciona con un test
- Hay 3 formas de clonar un proyecto
    - HTTPS: no recomendada. Te pide usuario y contraseña
    - Github CLI: **gh repo clone user/repo** no hace lo mismo que ssh
        - git remote --verbose Me da los origenes, incluido el upstream (el repo original)
        - Con git clone si hago el git remote no me dará el upstream, debo configurarlo manualmente
        - Con el upstream podremos traernos los cambios posteriores al clone a nuestro repo clonado
    - SSH: hay que tener configurada la llave SSH.
    - Puedo añadirle **--depth=1** para descargar solo desde el último commit

```
git clone git@github.com:user/jkahsas.git
```

- Uso git status para ver los archivos que he modificado
- Si hago el .add y el commit luego **no puedo hacer un git push**
- Necesito hacer un **fork**, una bifurcación. EL fork es mio, hago con él lo que quiero
- Fotocopio el árbol, hago los cambios, y luego hago una PR
- Se pueden hacer forks de forks ad infinitum. Un fork es una fotocopia de la cual yo me hago dueño
- Un clone sería copiarlo con los mismos permisos (yo no soy el propietario)
- El fork se hace en github (no en local). Copia todo el histórico
- Pongamos que copio un repo de una web/app
- Corro npm run dev (o lo que sea, con las .env configuradas,etc)
- En el navegador busco en la pestaña Lighthouse de la consola problemas de accesibilidad (Analyze page load)
- Busco en el código fuente donde están los problemas que me indica
- **Hay diferentes formas de crear una rama**
- Puedo hacerlo desde vscode desde el icono de git main (poner nombre y ya está)
- Cuando se hacen cambios, estos deben ser quirúrgicos. Solo tocar lo mínimo indispensable
- No usar . en los commits "commit con final en punto.". El punto al final sobra! Son titulos
- Con el icono de git lateral confirmo los cambios. Le doy al + para añadir etiquetas a los commits (confirmaciones)
- botón Publicar Branch o escribir en consola git push.
    - En upstream no tenemos acceso, en origin (que es mi rama) si
    - Puedo confirmarlo desde VSCODE o ir a Github, dónde encontraré un banner con Compare & pull request
- Normalmente en las empresas no se hacen forks, pero puedes hacerlo
- En código abierto es muy típico hacer forks
- Las pull request (PR) tienen que ser cortas y muy explicativas (con una imagen del antes y el después, un video explicativo)

Todo lo que acabamos de hacer se le llama GITHUB FLOW

- Es una estrategia muy utilizada (oficial de github)
- Realizar un fork e ir haciendo pull requests pequeñas pero frecuentes al main
- Puedes eliminar la rama una vez hecho el merge en el main
- Yo la tendré en local pero no estará en el repo remoto
- Si han habido varios commits en el upstream desde que hiciste tu fork puedes darle al botón (desde tu fork en github) de **Sync fork**
- Cuando creamos el fork a traves del CLI, con git remote --verbose podiamos ver el **upstream** que es el repo del que hicimos el fork
- upstream = la fuente original
- Con **git pull upstream main** obtenemos el mismo resultado que con el botón Sync fork para actualizar esos cambios en la fuente original
- Debo hacer un git push a mi repo para que los cambios se hagan
- Si no he usado el CLI y he usado git clone con ssh, voy a tener que añadir el upstream manualmente
- Se pueden tener todos los repos remotos como quieras

```
git remote add _nombre_cualquiera ggit@github,com:user/repo.git
```