¿Qué es y para qué sirve Github?

Programación y Diseño Web

Hoy os quiero hablar de una herramienta desconocida por muchos desarrolladores principiantes y que sin duda es de lo mejor que he podido aprender en mi tiempo que llevo programando, y se trata ni más ni menos que de los controladores de versiones, más concretamente Git, y un servicio público llamado GitHub. Es importante que echéis un vistazo a esta herramienta porqué os va a salvar de muchos dolores de cabeza (probablemente también os dará otros) y sobretodo os permitirá mantener el código de una forma ordenada, por usuarios y sobretodo tener versiones por si alguna vez todo va mal y el proyecto se va al garete. De modo que vamos a ver en qué consiste todo esto.

GitHub, un servicio para mantener tu código a salvo de peligro

Todos sabemos que las copias de seguridad de nuestros datos son importantes, tanto si son fotografías, ficheros o código de programación. Las alternativas siempre son las mismas: la nube usando Dropbox o Drive, en local usando discos duros aparte de los que empleamos para nuestro uso diario, pero **en el caso del código tenemos una alternativa mucho mejor: los repositorios Git.** Esta clase de repositorios son una copia local del código generado con una característica muy importante, y es que **podemos hacer varias versiones para poder recular si nos hemos equivocado y nuestra aplicación ya no funciona, o para trabajar en funcionalidades nuevas sin necesidad de modificar la versión funcional y así no romper el proyecto.** Esta es la premisa más básica de los repositorios de código, pero seguimos sin solucionar el tema de que se mantiene en local. Si nuestra máquina se estropea y deja de funcionar, corrompiendo el disco duro, no podemos recuperar todo el trabajo realizado. Es por ello que nacen servicios como GitHub, BitBucked u otros similares que pretenden llenar ese vacío.

Entonces, ¿cómo trabajaremos con repositorios Git y más concretamente sincronizando con Github? Al principio del proyecto, cuando estemos preparando todo el entorno, nosotros haremos una base de la que vamos a partir, ya estemos en solitario o con compañeros. Esta base la vamos a tener como **algo funcional pero no definitivo.** Esto significa que, por poner un ejemplo, si tenemos una web con unos estilos concretos y una estructura que ya hemos definido, vamos a crear algo parecido a una plantilla de la que todos los integrantes del grupo de trabajo podrán partir. Esto no significa que debe ser el resultado final de nuestro producto, por supuesto, porqué entonces no tendría sentido usar git. Debemos

presentarlo como el esqueleto de donde cada uno podrá dar a luz las distintas partes necesarias para el producto final mencionado.

Una vez tenemos ese esqueleto lo subimos en la rama **master**, y no la volvemos a modificar hasta que queramos integrar una funcionalidad nueva. Un momento, ¿ramas? Si, en Git todo funciona en ramas ya que tiene una estructura de árbol, y vamos a crear tantas ramas a partir del master como necesitemos. Normalmente se crea una rama por funcionalidad tal que así:

"Imagen 1"

Tal y como podemos observar, existe una rama master, de color verde, de la cual se han ido creando unas divisiones para desarrollo o características, y si miramos más adelante en la línea del tiempo observamos cómo se acaban uniendo de nuevo al master. Estas uniones, o *"merge"*, se emplea cuando una característica ya está terminada y funcional, sin errores y sin posibilidad de que rompa nuestro programa, y se desea subir a master. **Recordemos que nosotros hacemos un clon de master y trabajamos sobre él, pero el original sigue sin ser modificado**, de modo que si queremos añadir los cambios, debemos hacer dicho *merge.*

Si analizamos este modo de trabajo podemos darnos cuenta de que es muy potente a la hora de organizarnos con todo el equipo, ya que nos podemos sincronizar en cualquier sitio, con cualquier máquina. De hecho herramientas de gestión de equipos que se conecten con Git nos van a permitir incluso ver una evolución en el desarrollo del software para poder identificar si han habido problemas, o *showstopers*, y aplicar soluciones; o si queremos ir más allá, varias personas pueden trabajar en el mismo código sin necesidad de estar delante de la misma máquina, ya que con un gestor de Git, como puede ser ***SourceTree* o el mismo *GitHub Desktop*,** nos resalta con colores qué partes hemos modificado nosotros y qué otras las ha hecho nuestro compañero o compañera de trabajo, para que así poder decidir qué código implementar.

En definitiva, tanto si trabajáis solos como acompañados, es una buena idea aprender a usar un gestor de repositorios para tener un buen control del código, de sus versiones y de otros componentes del proyecto y así tener la posibilidad de volver atrás en caso de error o estudiar su evolución productiva si usamos un gestor de equipos. (Miro, s.f.)

Bibliografias

https://www.deustoformacion.com/blog/programacion-diseno-web/que-es-para-que-sirve-github (Miro, s.f.)