***Flujo de trabajo profesional con Pull request***

Antes de introducir el tema de ***Pull request,*** conviene bien dejar clara una noción sobre lo que son los *servidores de desarrollo* (o en ingles, ***staging develop***): básicamente pueden ser definidos como la presentación, de la última versión, de un proyecto en desarrollo (llamese sitio web o aplicación móvil, etc) que está en su última fase de prueba; es decir, es ese espacio previo, para *testear,* lo que queremos *posiblemente* presentar como proyecto final ad portas a ser presentado a un público, usuario, cliente o navegador cualquiera.

Técnicamente los *servidores de desarrollo* tienen su origen en ramas (lo que son) que están justamente por detrás de la rama *master* o *main* (las ramas principales que se suben en el servidor final, la carta de presentación: ***servidor de producción***). Entonces los servidores de desarrollo, que vendrían siendo las ramas que están justamente detrás de la rama *master* o *main,* cuentan también con un servidor que se llama: ***servidor de prueba.***

*Vealo de la siguiente forma:* el ***servidor de producción*** vendría siendo la página web o aplicación movil ya elaborada, que ya superó todas las fases de prueba y que ya está siendo presentada ante usuarios y/o clientes. En cambio, el ***servidor de prueba*** es ese misma página web o aplicación movil en su, *posiblemente,* última versión; la cúal está siendo evaluada por unas últimas fases de prueba para determinar si ya es apta (si se podría considerar o no) como un proyecto finalmente elaborado y exitoso; y poder así ser cargado, posteriormente, en el *servidor de producción*.

*Ahora bien, qué importancia podría tener todo esto en relación a la herramienta de GitHub* ***Pull request?,*** pues que, justo después de que cada desarrollador independientemente haya trabajado en su versión final, según una rama de desarrollo que le haya sido otorgada; todos, sí o sí, deben cargar y fusionar, previa revisión, sus últimos avances *(los que tienen cada uno en sus ramas de desarrollo correspondientes)* primeramente a: la rama del *servidor de desarrollo.* Justo antes de estar todo el código, de todos los desarrolladores con sus respectivas ramas por desarrollador, cargados y fusionados en el servidor de desarrollo: ***se debe revisar*** (como mecanismo de filtro igual, es por eso que es cargado todo el código de todos los desarrolladores, ya fusionado, a esta rama previa a la rama *master: stagind develop*). Ese proceso de revisión que se hace, justo en el momento antes de fusionar en su totalidad (e individualmente) todas las ramas correspondientes a cada desarrollador con la rama *stagind develop,* se le conoce como: ***pull request.***

*Pull request, a modo general, es ese lugar intermedio entre una rama y otra donde se hace un sin número de revisiones (entre lo que exporta una rama e importa otra) antes de llevar a cabo un proceso de fusión entre ambas. Estado intermedio antes de enviar el “merge”.* Esas revisiones se basan o consisten en que otros miembros del equipo puedan ver los cambios que se hicieron, por cada desarrollador, en su rama particular de desarrollo; entonces, hasta que no todos estén de acuerdo en que todos los cambios elaborados era lo esperado, no se hace la fusión entre una rama y la otra (en nuestro ejemplo si una rama de desarrolo en cuestión no gusta, finalmente no podrá hacer un *merge* con *stagind develop*). Ahora bien, en caso que a todos les guste los cambios, éstos son aprobados; y, al aprobarlos, se auto-ejecuta la fusión *merge* en la rama destino; en este caso, siguiendo la línea historica del ejemplo, la rama destino sería *stagind develop.*

Ahora, para hacer la fusión entre *stagind develop* y la rama *master* también se debe pasar por un proceso de ***pull request (herramienta exclusiva de GitHub).***

*un dato: la persona que coordina todas estas revisiones y avala finalmente los cambios y las fuciones entre ramas son los lideres de equipo, muchas veces técnicamente llamados como:* ***DevOps.*** *Un administrador del entorno de desarrollo en su conjunto y que le facilita el trabajo (y lo hace más eficiente) a los desarrolladores.*

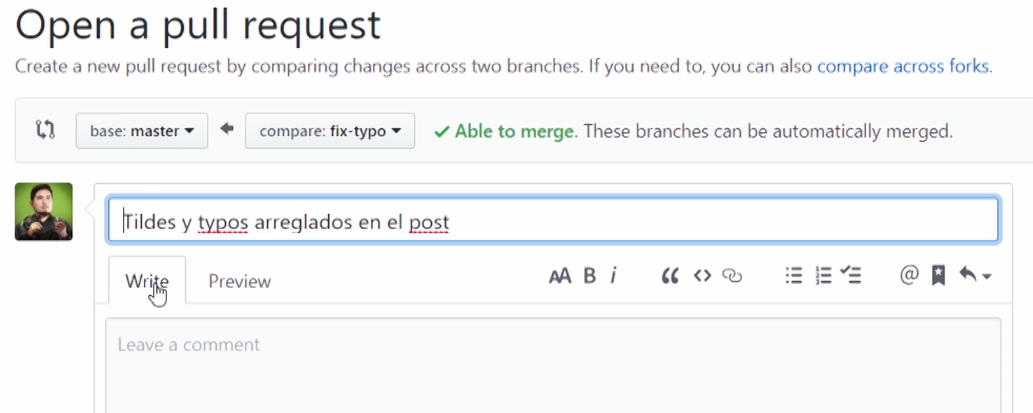
*un dato: es una buena práctica también crear, a partir de la rama base, una rama de desarrollo que consista en corregir todos los errores de todas las demás ramas de desarrollo en cuestión, a esta rama se le suele poner un nombre relacionado con la palabra “fix... ”, esta rama puede ser empujada luego a GitHub.*

***Pull request en acción***

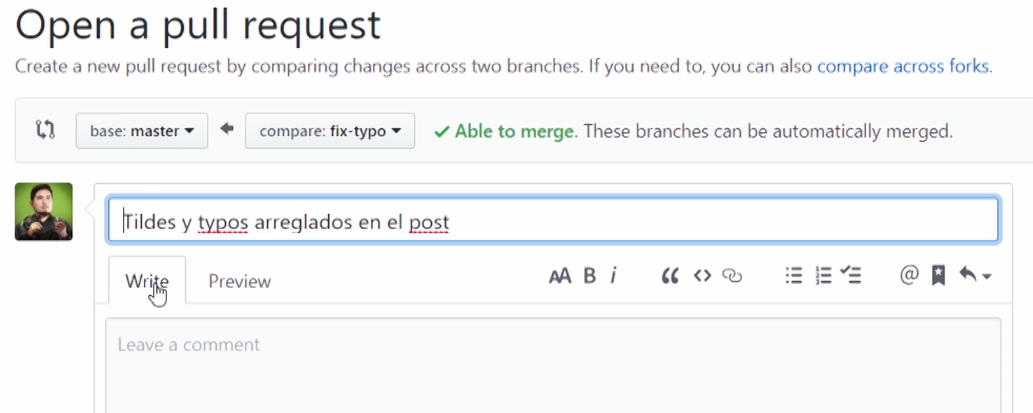
Toda vez que en GitHub registremos un nuevo cambio o se carguen nuevos *commits*, de alguna rama de desarrollo por ejemplo (por parte de un desarrollador tercero), el mismo GitHub *por un tema de buena práctica* nos preguntará si deseamos agregar esos *nuevos cambios,* recién hechos en una rama de desarrollo, a nuestra rama master; pidiendonos primero que, nosotros como propietarios del repositorio remoto en cuestión, revisemos los cambios cargados *(por otro desarrollador comúnmente)* y decidamos si estamos de acuerdo o no con ellos para agregarlos: los cambios cargados en la rama de desarrollo de interés, finalmente a la rama *master.*



Presionado ya el botón de *“Compare & pull request”*, veremos algo más o menos así:



En este ejemplo, se supone que se registraron y cargaron nuevos cambios en la rama de desarrollo *“fix-typo”;* y GitHub, siendo nosotros los propietarios del repositorio remoto, nos pregunta si estamos de acuerdo o no en que esos cambios guardados en *fix-typo* se agreguen a nuestra rama principal *master*. Justo en:



Luego, nos pone un titulo borrador de con qué nombre se guardará el registro del *pull request* que, *posiblemente*, llevaremos a cabo entre la rama *master* y la rama *fix-typo: “Tildes y typos arreglados en el post”;* se dará cuenta que este *titulo borrador* lo asigna intuitivamente GitHub en base al nombre, o mensaje refencial, con el que hemos registrado el *commit*, justo antes de ser empujado a GitHub, en la rama de desarrollo con la que estamos intentando hacer el *merge;* en este caso, esa rama de desarrollo es: *fix-typo.* Es decir, lo que le hace falta a *master* y que tiene *fix-typo* es un último *commit* que fue enviado con el mensaje referencial de: *“Tildes y typos arreglados en el post”;* y, GitHub, sólo quiere saber si usted como propietario del repositorio remoto está de acuerdo o no en agregar esos nuevos cambios a la rama principal *master.*

Posteriormente, debajo sale un recuadro en blanco que sirve para dejar un comentario *(Write).* *Dejar un comentario* es opcional y puede resultar ser una buena práctica; quizás a usted como dueño del repositorio remoto le podría interesar dejar un mensaje particular, en relación a las modificaciones hechas*,* al desarrollador que acaba de cargar cambios (commits) en su rama de desarrollo y que están esperando ser vinculados a rama *master.*

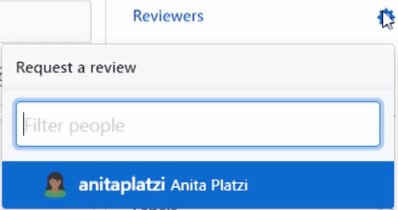
Ahora, si desea ver puntualmente cuáles fueron esos cambios realizados en la rama de desarrollo con la que se está intentando hacer un *merge* a rama *master,* debe irse más abajo. Habrá un editor de código que le señalará, en color verde, el código que fue agregado; y, en color rojo, el código que fue suprimido de una misma línea de código que haya sufrido alteraciones. Veamos el siguiente ejemplo en la línea de código 15:

  
*La palabra “parrafo” pasó a ser “párrafo” y la palabra “increéibles” pasó a ser “increíbles”. Esos fueron los cambios.*

***A usted ahora, como propietario del servidor remoto, le corresponde decidir si esos cambios son válidos o no para hacer une merge con master desde GitHub.***

Sin embargo, si también desea, antes de hacer propiamente el *merge,* puede solicitarle primero a otros desarrolladores que revisen el *pull request* para esperar la validación de todos y decidir si es conveniente o no hacer un *merge* (hasta que todos los solicitados no estén de acuerdo con el *merge,* el *merge* no pasará)*.*

Para invitar a revisión a otros desarrolladores,... eso se hace agregando a un desarrollador de su interés, *o varios*, vinculados al repositorio remoto en cuestión, tal que así: (revise la casilla *Reviewers*):



En este caso estamos suponiendo que sólo tenemos agregado un desarrollador en nuestro repositorio remoto de trabajo actual: *anitaplatzi,* quiere decir que sólo a ella podemos solicitarle una revisión del *pull request.*

Ahora, usted como propietario del repositorio remoto, si solicita una revisión del *pull request* a otro u otros desarrolladores, si bien no se concreta en definitiva el *merge,* está dejando claro que sí está de acuerdo en qué se haga el *merge* en cuestión (como para validar su voto u opinión sobre la decisión o rumbo final que tomen los desarrolladores solicitados para hacer la revisión del *pull request* en discusión). Debe hacer click en:



Los desarrolladores para revisar si les pidieron su opinión (o validación) de un *pull request,* deben situarse sobre el repositorio en cuestión y ver la notificación que le muestre la pestaña *pull request* y hacer click ahí. En caso que estén de acuerdo con el *merge;* es decir, de acuerdo con los cambios que se hicieron en la rama de desarrollo que pretende ser fusionada a *master* (estos cambios los puede visualizar al detalle en la sub-pestaña de la pestaña pull request: *commits*). Presiona click en:

