



Ingeniería de software

Laboratorio 2: Uso básico de Github

Pre requisitos

1. Visual studio 2019 o 2022 instalado
2. Tener una cuenta en GitHub
3. Laboratorio 1 completado

Resumen

Tal y como se vio en el primer laboratorio, el uso de repositorios es primordial para el control de versiones. Sin embargo, también es importante conocer las herramientas que nos ofrece git para mantener el orden de las versiones y trabajar con otros desarrolladores de una forma fácil y evitando los conflictos.

Existen una serie de comandos básicos que ayudan a llevar a cabo lo antes mencionado. Para este laboratorio va a necesitar un repositorio previamente creado, así como un proyecto ASP.NET (como el que se creó en el primer laboratorio).

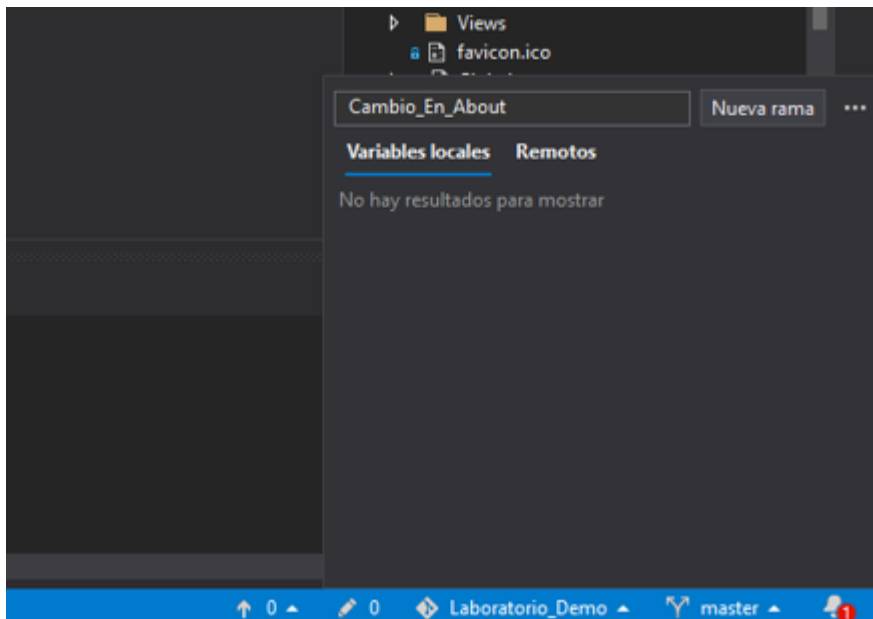
Primera parte - Creación de ramas (*branches*)

El funcionamiento de las ramas en control de versiones sirve para mantener un mejor orden ya sea por “*feature*” o por “*developer*”. Lo anterior va a depender de la estrategia git que se decida utilizar, sin embargo en este laboratorio vamos a trabajar las ramas por “*feature*” (característica), es decir, por cada nueva funcionalidad que se desee desarrollar.

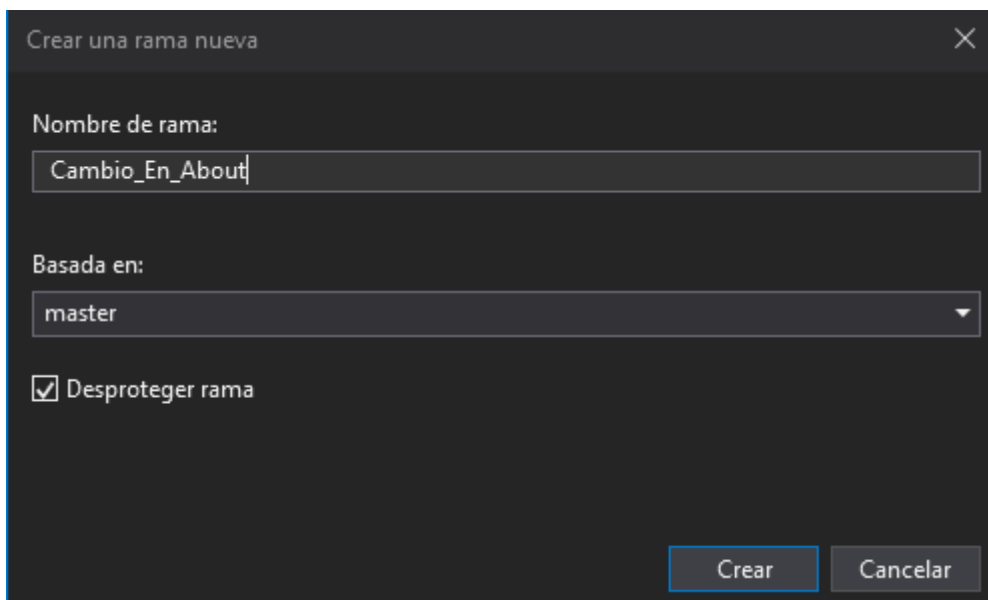
Dicho lo anterior, comenzaremos con la creación de una rama a partir de otra. Es importante saber que cuando creamos una rama, esta va a ser una copia de la rama origen desde la cual decidimos crearla. Nuestro origen en este caso es la rama “*master*” (principal).

El programa Visual Studio nos permite crear estas ramas de forma sencilla.

***Importante:** debemos asegurarnos de estar en la rama que deseamos para ramificar a partir de esta.

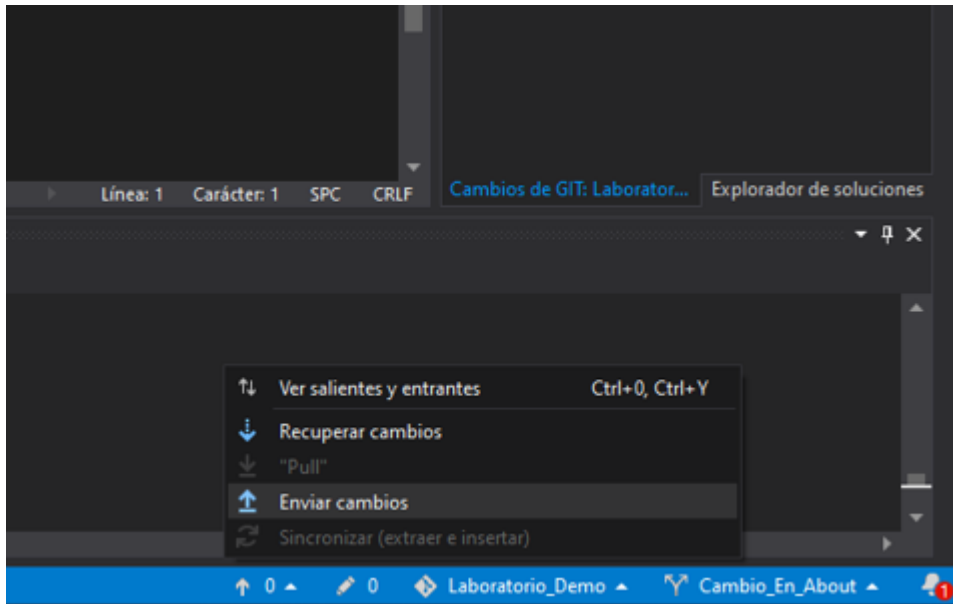


Damos click en la palabra “*master*” y se nos desplegará esta ventana. Deben colocar el nombre de la rama que desean, y seleccionar “Nueva Rama”.



Esta ventana se desplegará en donde deben confirmar el nombre de rama, y verificar que la rama sobre la cual está basada la misma sea la que queremos. De ser así, le damos a “Crear”.

Esta rama se encuentra ahora en nuestro repositorio local, no obstante no está en el repositorio remoto. Para enviarla a nuestro repositorio de git debemos seleccionar “Enviar cambios”.



A partir de aquí, nuestra rama será parte de las ramas del repositorio remoto y podremos comenzar a trabajar. Podemos verificar que se subió correctamente al revisar nuestro repositorio desde un explorador y ver que la rama ya está arriba.

Implementando funcionalidad en la rama 1

Ahora, comenzaremos a implementar el *feature* que queremos. Nos iremos a **Views -> Home**, al archivo *about.cshtml*. Aquí agregaremos otra caja de “input” en donde podremos colocar el nombre de nuestro equipo de trabajo. Tome el código de abajo y colóquelo encima del código de la “Caja para texto” en el “About”.

```
<div>
  <label>El equipo de PI del que soy parte se llama:</label>
  <input type="text" placeholder="Nombre" />
  <hr />
</div>
```

Y una vez agregado, podemos compilar y correr el programa. Se tiene que ver de la siguiente forma:



Mi primera vista

¡Hola Mundo!

Yo soy: Christian Rojas

Mi nota el día de hoy es:

El equipo de PI del que soy parte se llama:

Caja para texto:

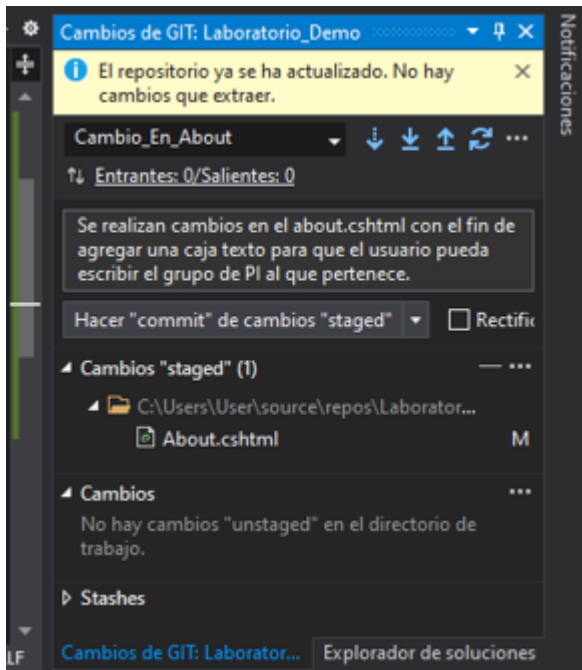
[Volver a la pagina de inicio](#)

© 2022 - Mi aplicación ASP.NET

Realizamos *commit* con un comentario significativo del cambio como se realizó en el primer laboratorio.

Nota: Recuerde que por buenas prácticas es importante siempre hacer primero *pull* antes de un *commit* ya que otro desarrollador podría haber realizado cambios en esta rama, y debemos de asegurar SIEMPRE que nuestros commits no vayan a comprometer la integridad del repositorio. Por ello, recuerde siempre el orden: pull, commit, resolución de conflictos (de haber), y luego push.

Cuando ya aseguramos haber *pulleado* y tener el repositorio al día, podemos proceder a realizar el *commit*. Recuerde poner siempre comentarios significativos respecto al cambio realizado con el fin de saber que se hizo.



A partir de aquí, ya podemos subir nuestro cambio realizando un *push*.

Implementando funcionalidad en la rama 2

Una vez realizado este paso, procedemos a crear otra rama desde la rama *master*. Esta tendrá el nombre "Caja_En_About". En esta rama haremos el siguiente cambio en el archivo "about.cshtml", en el mismo lugar donde colocamos la caja anterior de la primera rama.

```
<div>
  <label>Mi numero de carne es:</label>
  <input type="text" placeholder="Numero" />
  <hr />
</div>
```

En este punto tenemos dos ramas con dos funcionalidades distintas en el mismo archivo. Ellas por sí solas no tienen conflictos, sin embargo para integrar nuestras funcionalidades a la rama principal, debemos de mezclarlas con la misma. En este caso tendremos que lidiar entonces con un proceso llamado "**Pull request**" el cual consiste en integrar de forma ordenada y segura las funcionalidades que tenemos, al código ya funcional de la rama principal.



Segunda parte - Generación de un pull request

El “*pull request*” consiste en una forma de enviar el trabajo realizado en una rama para unificarlo ya sea con otras ramas o con la rama principal del proyecto, y que este envío pase por una aprobación ya sea de uno o varios compañeros de trabajo. Esto con el fin de que pase por revisiones y así se eviten conflictos o peor aún pérdida de funcionalidades, que podría pasar si solo se hace una merge a la rama principal sin estas revisiones.

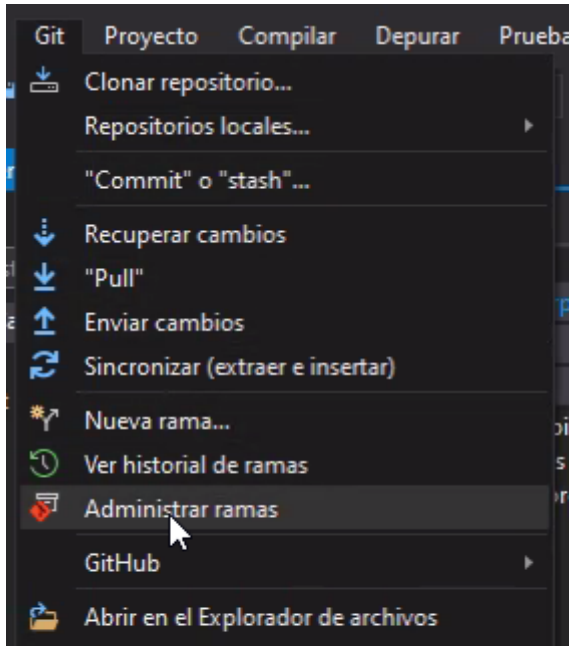
Una vez revisado y aceptado este *pull request*, la persona que envía el mismo puede decidir si desea que la rama (después de realizar el merge) sea eliminada o que quede en caso de que se quiera continuar trabajando en la misma.

En este ejercicio vamos a realizar un *pull request* a la rama principal desde la primera rama creada, la cual no debería darnos ningún tipo de conflicto. Para esta rama vamos a seleccionar la opción de que la misma permanezca (que no sea eliminada).

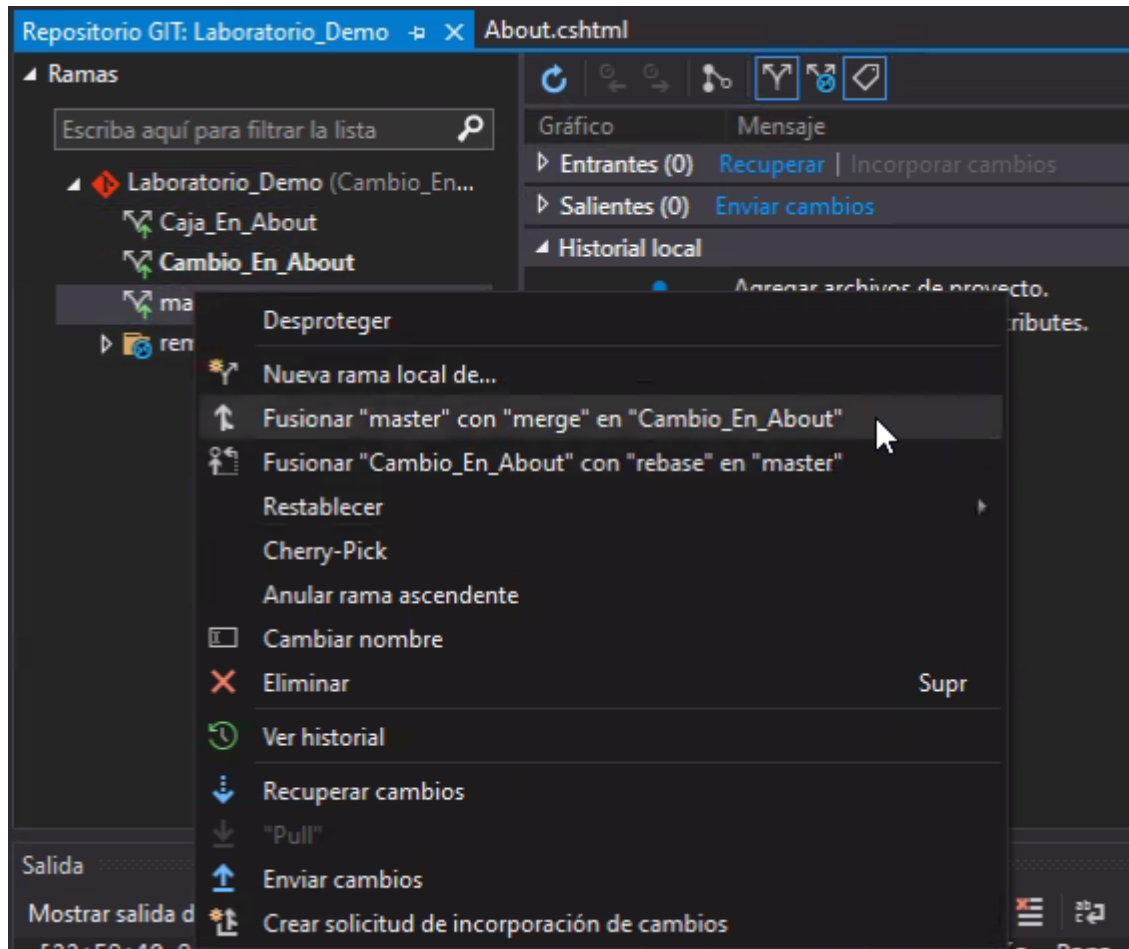
Paso 1: Unir la rama “Cambio_En_About” con master

El primer paso es ir a la rama que deseamos enviar a la rama principal. En nuestro caso, sería la rama “Cambio_En_About”. Aquí verificamos que esté actualizada en el repositorio remoto realizando un *pull*, y luego un *push*.

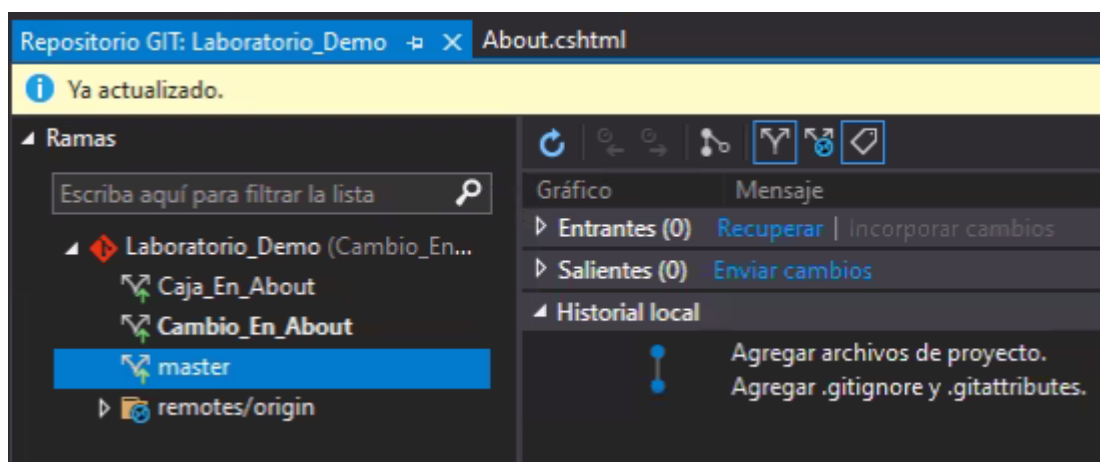
Seguidamente, seleccionamos la opción “Administrar ramas” en el menú Git.



Ahora, debemos de mezclar la rama principal sobre nuestra rama. Para esto haremos un *merge* de *master* a “Cambio_En_About”, para verificar que no hayan conflictos, o de haberlos, resolverlos y así facilitar la revisión de la solicitud del PR.



Nos va a consultar si estamos seguros, y le decimos que sí.



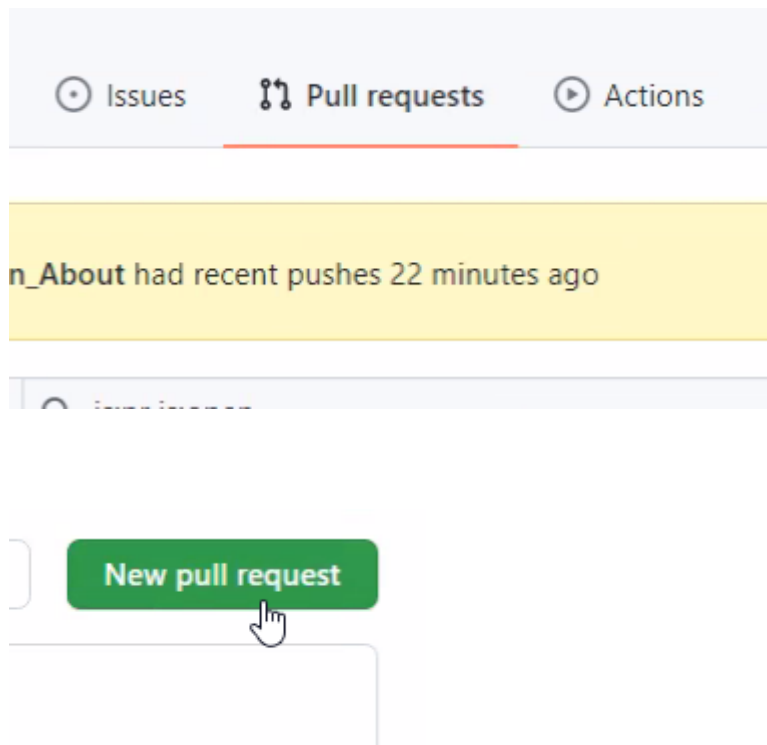
Como no existía un conflicto entre ninguna línea de código entre las dos ramas, entonces el *merge* se lleva a cabo sin problemas. A partir de aquí sabemos que






podemos enviar nuestra rama a *master* de forma correcta. Procedemos entonces a crear el *Pull Request*.

***Nota:** Antes de generar el PR, verifique que el proyecto compile y corra de forma debida. Si no es así, debe revisar a qué se debe el fallo.

Primero nos vamos a GitHub, en la pestaña “Pull requests” y seleccionamos la opción “New pull request”.



Aquí nos preguntará a quien deseamos comparar, entonces seleccionamos la Rama que queremos enviar a *master*. En este caso, “Cambio_En_About”.

Example comparisons		
	Caja_En_About	23 minutes ago
	Cambio En About	3 hours ago
	master	5 hours ago

El Git nos dirá que estas ramas sí se pueden mezclar, entonces aceptaremos la creación de nuestro PR.



✓ Able to merge. These branches can be automatically merged.

Create pull request

Aquí debemos colocar un título, una breve descripción del motivo de nuestro PR, así como colocar a los que queremos que revisen la solicitud.

Se realizan cambios en el about.cshtml con el fin de agregar una caja...

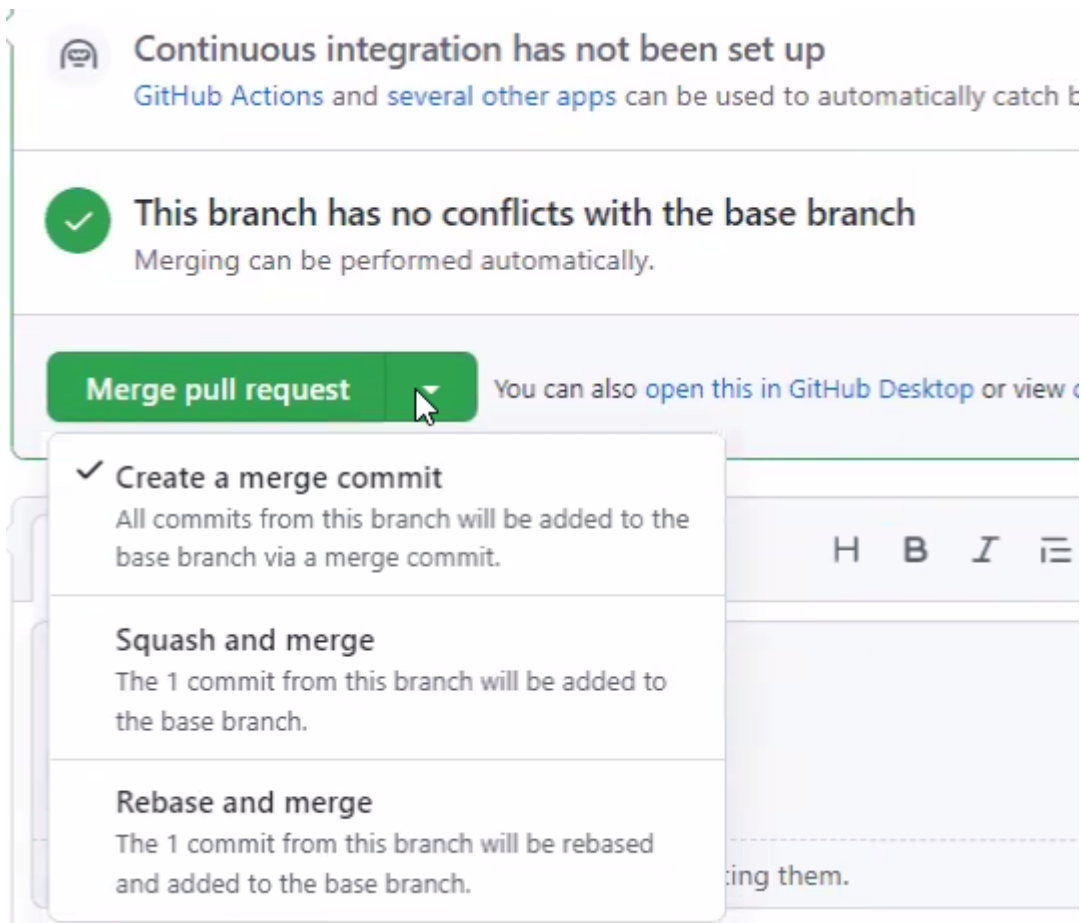
Write Preview

H B I

Pull request, con cambios en el archivo about.cshtml con la inclusión de una nueva caja de texto.

Por motivos del laboratorio, no colocaremos a ningún revisor, no obstante para temas de proyecto usted deberá colocar los usuarios que estén encargados de revisarlos. Por esto, dejamos esta parte en blanco, y seleccionamos “Create pull request”.

Esto va a generar un nuevo *issue*, y aquí el revisor podrá aceptar o denegar el PR. En este caso como usted es su propio revisor, podrá aceptarlo.



El *merge* nos da distintas opciones de integración. La elegida va a depender de la estrategia git que se esté utilizando que debe ser establecida para fines del proyecto. En este caso vamos a seleccionar la primera. Le instamos a que investigue más sobre cada una de estas opciones.





Aquí confirmamos el *merge*, y seguidamente nos informará que este ha sido exitoso al integrarse con la rama principal. En este paso podemos decidir si queremos cerrar la rama, sin embargo vamos a dejarla en el repositorio.

Pull request successfully merged and closed

You're all set—the `Cambio_En_About` branch can be safely deleted.

Delete branch

Volvemos a nuestro Visual Studio, nos cambiamos a la rama *master* y haremos *pull* para bajar los cambios realizados. Volvemos a compilar el proyecto para revisar que todo esté correcto, y observamos como la funcionalidad integrada ya está funcional en la rama principal.

Mi primera vista

¡Hola Mundo!

Yo soy: Christian Rojas

Mi nota el día de hoy es:

El equipo de PI del que soy parte se llama:

Caja para texto:

[Volver a la pagina de inicio](#)

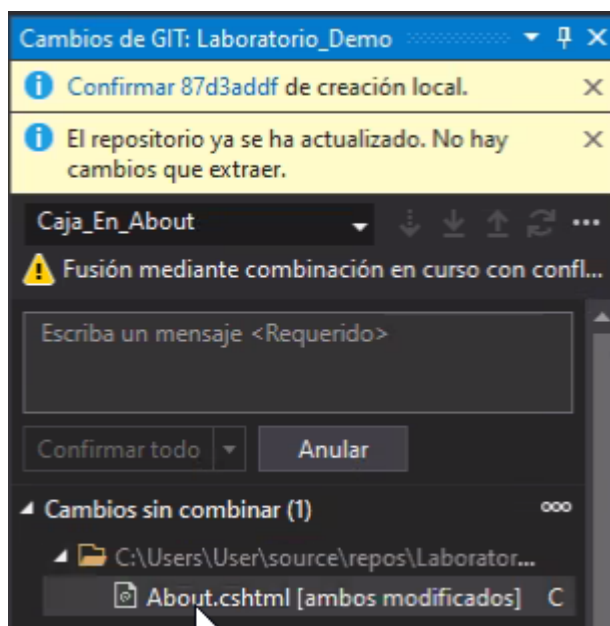
Paso 2: Unir la rama “Caja_En_About” con master

Repetiremos el proceso de generar un PR para la segunda rama, la que incluía el cambio de generar un campo para el carné universitario.

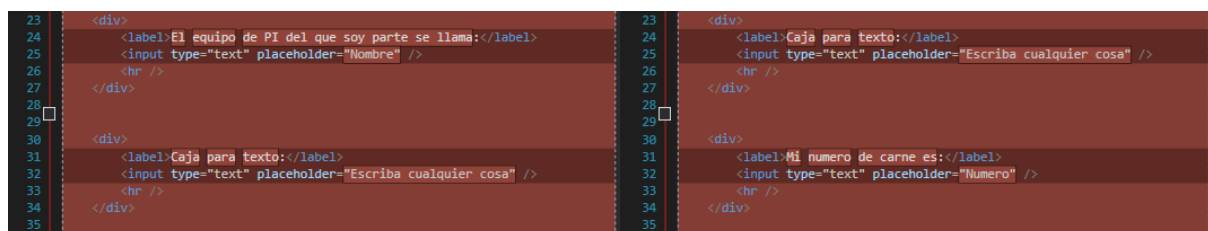
Volvemos a generar un *merge* de “master” en “Caja_En_About”, solo que esta vez vamos a notar que tenemos una diferencia: Se generó un conflicto entre algunas líneas de código.

⚠ Fusión mediante combinación completada con conflictos en el repositorio Laboratorio_Demo. [Resuelva los conflictos](#) y confirme los resultados.

Para proceder, debemos seleccionar en “Resuelva los conflictos”, la cual se resalta de color celeste en el mensaje anterior.



Se nos abrirá lo siguiente, y daremos click sobre el archivo que nos está generando problemas, en este caso el “About.cshtml”. Dependiendo de la línea en que se haya colocado el código de las cajas de input, el conflicto se producirá con líneas distintas. En este caso, ambos códigos conflictúan con la “Caja para texto”. Aquí debemos de seleccionar cuáles códigos queremos conservar, y en nuestro caso son ambos

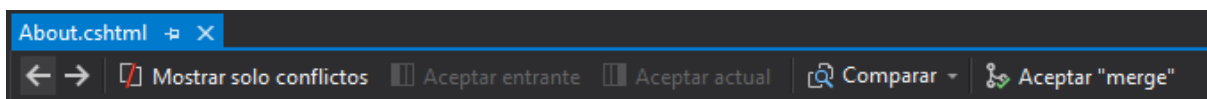


Debemos aceptar ambos cambios, ya que deseamos que ambos estén integrados en la rama principal. Esto se debe ver como la imagen de abajo.

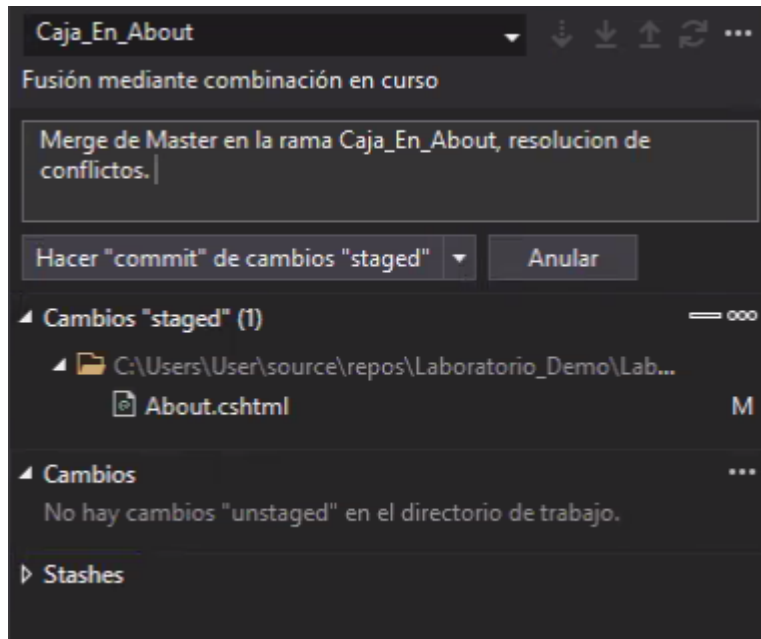


```
23 <div>
24   <label>El equipo de PI del que soy parte se llama:</label>
25   <input type="text" placeholder="Nombre" />
26   <hr />
27 </div>
28
29 <div>
30   <label>Mi numero de carne es:</label>
31   <input type="text" placeholder="Numero" />
32   <hr />
33 </div>
34
35 <div>
36   <label>Caja para texto:</label>
37   <input type="text" placeholder="Escriba cualquier cosa" />
38   <hr />
39 </div>
40
```

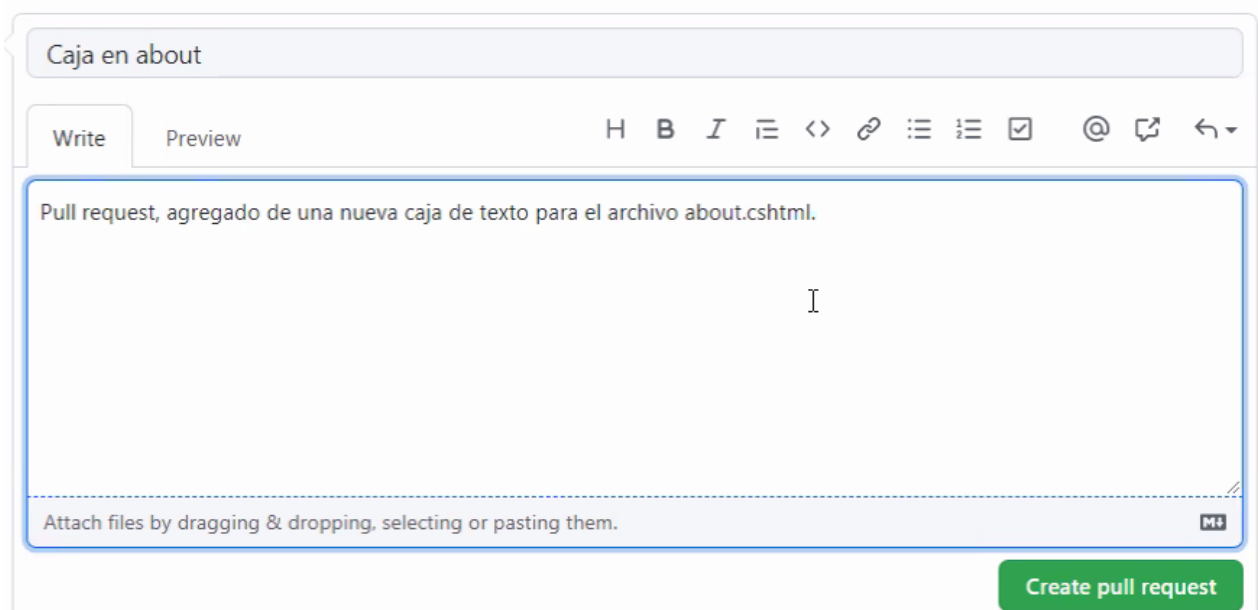
Ya con el código correcto y sin conflictos, seleccionamos “Aceptar “merge””.




Esto nos va a generar un cambio, al cual le haremos *commit* informando que solucionamos el conflicto causado. Antes de generar el commit se recomienda que verifique que todo corra como es debido.





Una vez generado el *commit*, la rama ya está lista para generar un PR hacia *master*. El PR se debe hacer exactamente igual como se mostró anteriormente, por lo tanto repita los pasos de generación del PR para la primera rama. Recuerde poner títulos y comentarios significativos para que los revisores puedan entender de qué se trata su solicitud.






**Continuous integration has not been set up**
GitHub Actions and several other apps can be used to automatically catch bugs and enforce style.

**This branch has no conflicts with the base branch**
Merging can be performed automatically.

Merge pull request  You can also [open this in GitHub Desktop](#) or view [command line instructions](#).

Confirmamos el cambio, y nos mostrará nuevamente este mensaje.

Pull request successfully merged and closed
You're all set—the `Caja_En_About` branch can be safely deleted.

Delete branch 

Aquí, vamos a borrar la rama sobre la que trabajamos, con la certeza de que nuestros cambios quedarán integrados en *master*.

Ahora, quedamos con nuestras dos funcionalidades ya integradas en master, un master funcional, y una estrategia git segura para evitar pérdida de código.



Mi primera vista

¡Hola Mundo!

Yo soy: Christian Rojas

Mi nota el día de hoy es:

El equipo de PI del que soy parte se llama:

Mi numero de carne es:

Caja para texto:

[Volver a la pagina de inicio](#)

© 2022 - Mi aplicación ASP.NET

Este sería el resultado que se encuentra ahora en la rama principal y con esto concluimos con el laboratorio.

Recuerde siempre verificar las ramas donde está, seguir los pasos y respetar la estrategia para que todo el equipo pueda trabajar de forma segura e integrada.



Entregable

Para este laboratorio se debe compartir el repositorio usado con el asistente y la profesora a los siguientes usuarios de github:

- rebecca.ov
- ChristianRojasRios

Cree un documento donde explique cada una de las siguientes acciones en git y en qué casos se deben ser usados:

- rebase and merge
- squash and merge
- git reset
- git revert
- cual es la diferencia entre un git reset and git revert

En este documento agregue el enlace del repositorio.

Material elaborado por Christian Rojas Rios.

Referencias:

1. <https://www.atlassian.com/es/git/tutorials/learn-git-with-bitbucket-cloud>