

Guía Resolución de Conflictos

BITBUCKET

Empresa: CMPC



Autores: Carlos Flores

Luis Sandoval

Versión del documento: 1.0

Fecha Primera Versión: 14/09/2021

Contenido

1	Introducción	2
2	Contexto: Creación Rama RE y Clonacion Local.....	3
3	Generacion de Conflicto para Ejemplificar Solución.....	7
4	Resolución Conflicto en develop	12
5	Resolución Conflicto en master.....	21

1 Introducción

Una parte fundamental del trabajo con una herramienta que permite el manejo de versionamiento de código fuente es la resolución de conflictos.

Los conflictos son una medida de control para evitar la pérdida de código fuente y gatillar la correcta integración de las piezas de código.

Para resolver un conflicto en GIT, la forma más sencilla es a través de VS ya que nos brinda una cómoda interfaz para realizarlo, teniendo en cuenta la cantidad de código o páginas que pueden estar en conflicto.

Cuando se genera un conflicto se debe tener en cuenta los archivos en conflicto y el código en conflicto, es necesario comunicar a las personas involucradas en las ramas de conflictos para identificar que código es válido y cual no.

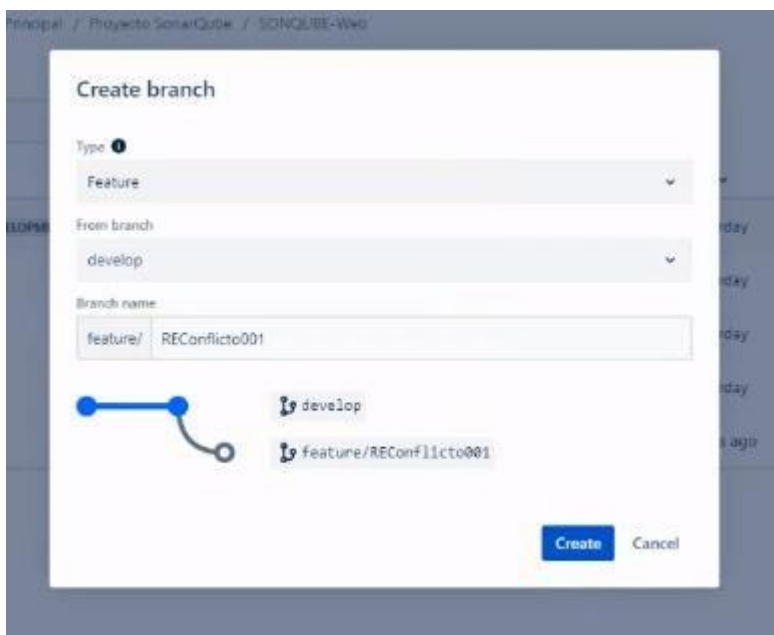
No debemos olvidar que parte del procedimiento adoptado en el uso de Bitbucket con CMPC, es que nuestros commit desde local van siempre directamente a nuestra rama feature o hotfix dependiendo de si estamos manejando requerimientos(RE) o incidentes (IN), respectivamente; es así como para traspasar código desde nuestra rama a la rama develop o a la rama master, pasamos a través de un pull request, si nos saltamos este procedimiento y operamos de distintas formas podemos llegar a tener conflictos con nuestro propio código, lo que sin duda provocará trabajo adicional, que pudo ser evitado.

En el siguiente ejemplo se explica cómo resolver un conflicto, para ello contextualizaremos la creación en local de nuestra rama, generaremos un conflicto, el cual resolveremos.

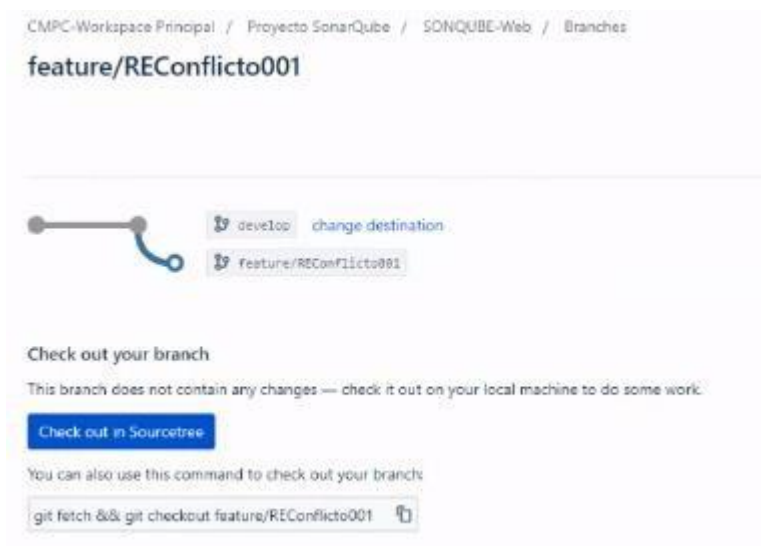
2 Contexto: Creación Rama RE y Clonación Local

Para nuestro ejemplo utilizaremos el repositorio SonQube-Web.

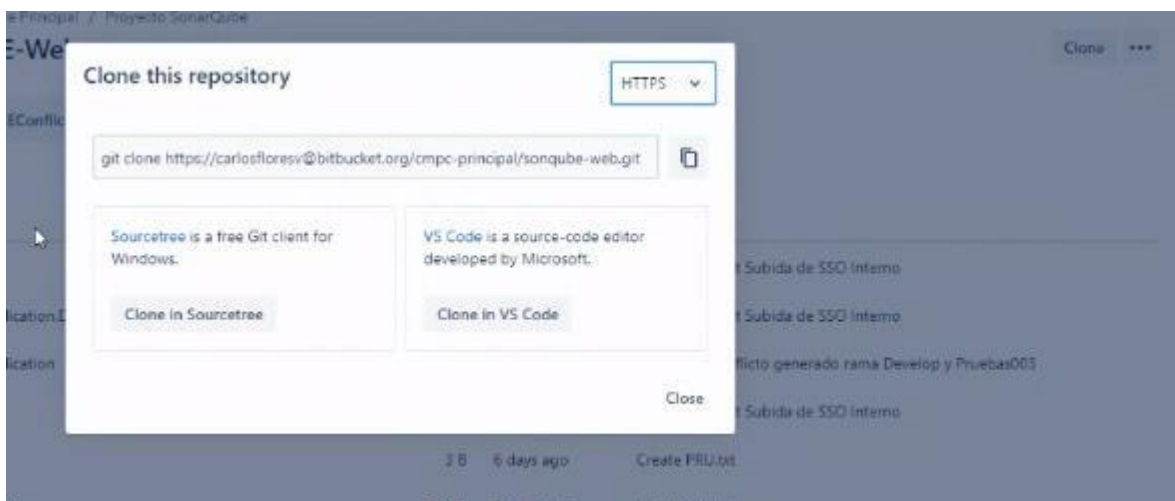
1. En primer lugar crearemos nuestra rama REConflicto001, desde “develop”.



Una vez creada se verá de la siguiente forma:

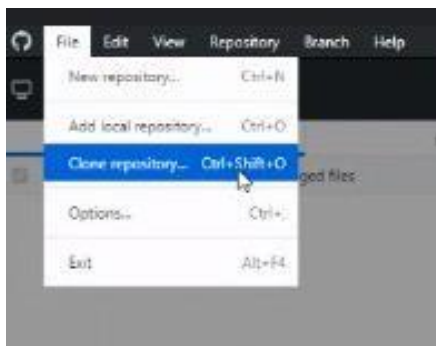


A continuación, estando en la rama REConflicto001, realizaremos la clonación de esta a local para trabajar en ella (esto es parte del procedimiento de trabajo definido con bitbucket), para ello presionamos “clone”, con lo que se verá lo siguiente:

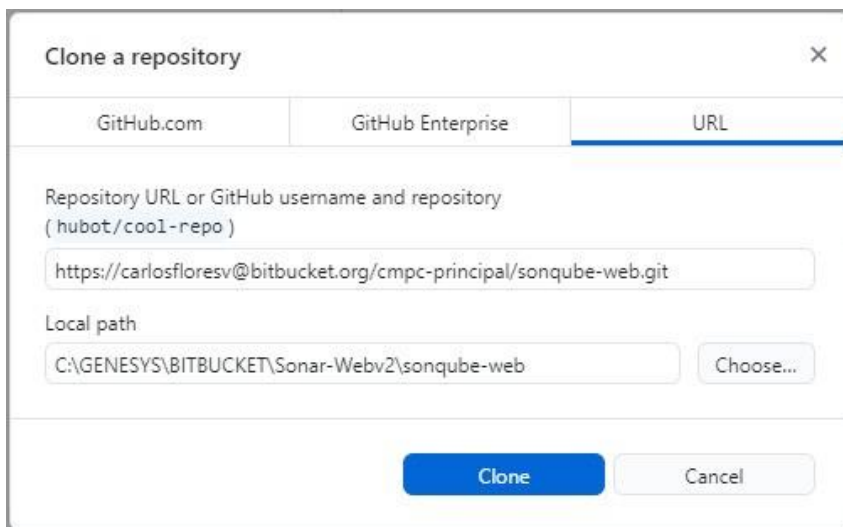


Copiamos la URL, y la llevamos a GITHUB Desdktop.

Estando en GitHub Desktop, iremos a la opción “Clone Repository...”



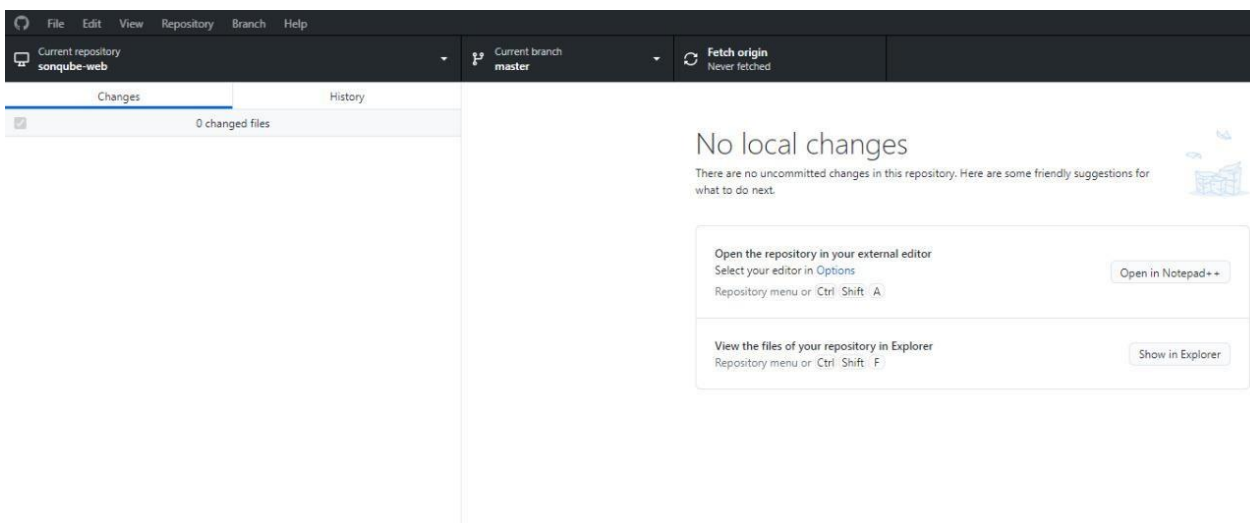
Con lo cual pegaremos la ruta en el campo superior (eliminando el comando git), en el campo inferior de la ventana, pondremos la ruta local donde dejaremos el repositorio de nuestra rama REConflicto001, tal como se muestra en la siguiente imagen:



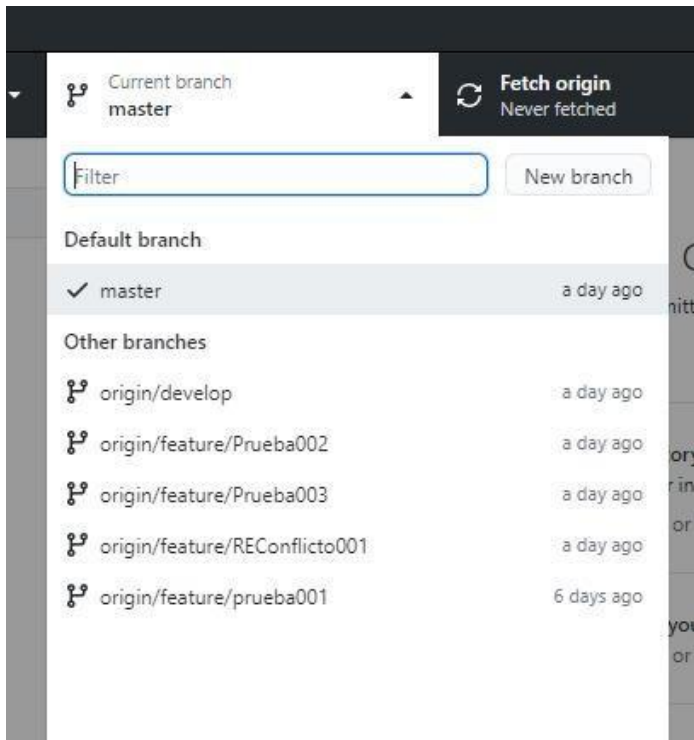
Git Desktop, comenzará a clonar nuestra rama RE, mostrando la siguiente ventana:



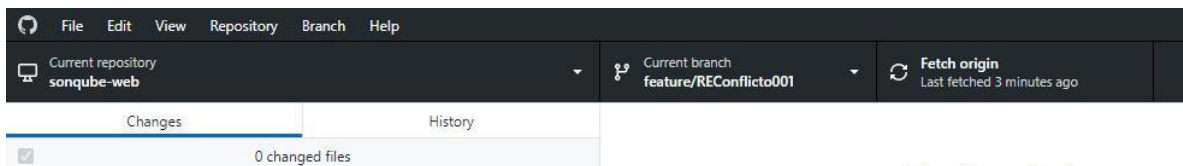
El resultado final, será el siguiente:



A Continuación, en “Current Branch”, seleccionaremos la rama “REConflicto001” (el usuario deberá seleccionar la rama de trabajo del ticket respectivo), que será en la cual trabajará.



Al seleccionarla sabremos que estamos trabajando en ella dado que se verá similar a la siguiente forma:



En este punto el usuario ya podrá abrir la aplicación en local, a través del IDE de trabajo, en este caso: visual studio.

3 Generación de Conflicto para Ejemplificar Solución

Paso 1. En este caso tomaremos un archivo de la rama develop, el cual se modificará para causar el conflicto:

Modificaremos el archivo “SAMLController.cs” que se muestra a continuación:

```

20 using System.Security.Cryptography;
21 using OpenSSL.X509Certificate2Provider;
22 using System.Text;
23 using System.EnterpriseServices;
24 using System.Security.Claims;
25 using System.Security.Cryptography;
26
27 namespace SAMLApplication.Controllers
28 {
29
30     [Authorize]
31     public class SAMLController : BaseController
32     {
33         #region Public Functions Actions
34         [HttpGet, AllowAnonymous]
35         public async Task

```

A este archivo incorporaremos el texto: “//TEXTO INCORPORADO DIRECTAMENTE EN BITBUCET DEVELOP”, quedando de la siguiente forma:


```
Editing SAMLApplication/Controllers/SAMLController.cs on branch: develop
19 using SAMLApplication.Models;
20 using System.Data.Entity;
21 using OpenSSL.X509Certificate2Provider;
22 using System.Text;
23 using System.EnterpriseServices;
24 using System.Security.Claims;
25 using System.Security.Cryptography;
26
27 namespace SAMLApplication.Controllers
28 {
29
30 // TEXTO INCORPORADO DIRECTAMENTE EN BITBUCKET DEVELOP
31
32 [Authorize]
33 public class SAMLController : BaseController
34 {
35     #region Public Functions Actions
36     [HttpGet, AllowAnonymous]
37     public async Task<ActionResult> SingleSignInService(BaseModel model) {
38         Logger.WriteInformation("Init SAML Request Authenticon.");
39         AsymmetricAlgorithm asymmetricAlgorithm = null; ActionResult returnR
40         Application application = SSODbContext.Applications.SingleOrDefault(
41
42         if (application != null) {
```

Destacamos, que esta modificación, al ser realizada en bitbucket directamente, no es una modificación que tendremos en el repositorio clonado en la rama RE en la cual estamos trabajando, ya que esta actividad de clonación se realizó directamente en el repositorio, actividad que puede haber sido realizada por otro desarrollador.

Esto simula que otro desarrollador modifique la rama develop, mientras nos encontramos trabajando en local.

Paso 2: Ahora en local, es decir en nuestra rama REConflicto001, realizaremos la siguiente modificación al archivo original **SAMLController.cs**

Archivo original en local:

```

using SAMLApplication.DataAccess.Entities;
using SAMLApplication.Helpers;
using SAMLApplication.Helpers.Common;
using SAMLApplication.Models;
using System.Data.Entity;
using OpenSSL.X509Certificate2Provider;
using System.Text;
using System.EnterpriseServices;
using System.Security.Claims;
using System.Security.Cryptography;

namespace SAMLApplication.Controllers
{
    [Authorize]
    public class SAMLController : BaseController
    {
        #region Public Functions Actions
        [HttpGet, AllowAnonymous]
        public async Task<ActionResult> SingleSignInService(BaseModel model) {
            Logger.WriteInformation("Init SAML Request Authentication.");
            AsymmetricAlgorithm asymmetricAlgorithm = null; ActionResult returnResult;
            Application application = SSODBContext.Applications.SingleOrDefault(m => m.Identifier.Equals(model.Identifier));

            if (application != null) {
                if (application.SAMLSPConfig != null && application.SAMLSPConfig.WantAuthnRequestSigned) {
                    byte[] rawCert = Convert.FromBase64String(application.SAMLSPConfig.X509Cert);
                    var cert = new X509Certificate2(rawCert);
                    asymmetricAlgorithm = cert.PublicKey.Key;
                }
                IdentityProvider.ReceiveAuthnRequestByHTTPRedirect(Request, out XmlElement xmlElement, out string relayState, out bool signed, asymmetricAlgorithm);
                //HTTPRedirectBinding.ReceiveRequest(Request, out XmlElement xmlElement, out string relayState, out string signed, out string keyAlgorithm);
                Logger.WriteVerbose($"SAML Request Authn: \"{xmlElement.OuterXml}\"");
                var authRequest = new AuthnRequest(xmlElement);
            }
        }
    }
}

```

Archivo Modificado en local:

```

// TEXTO MODIFICADO EN LOCAL EN RAMA REConflicto001

#region Public Functions Actions
[HttpGet, AllowAnonymous]
public async Task<ActionResult> SingleSignInService(BaseModel model) {
    Logger.WriteInformation("Init SAML Request Authenticon.");
    AsymmetricAlgorithm asymmetricAlgorithm = null; ActionResult returnResult;
    Application application = SSODBContext.Applications.SingleOrDefault(m => m.Identifier.Equals(m

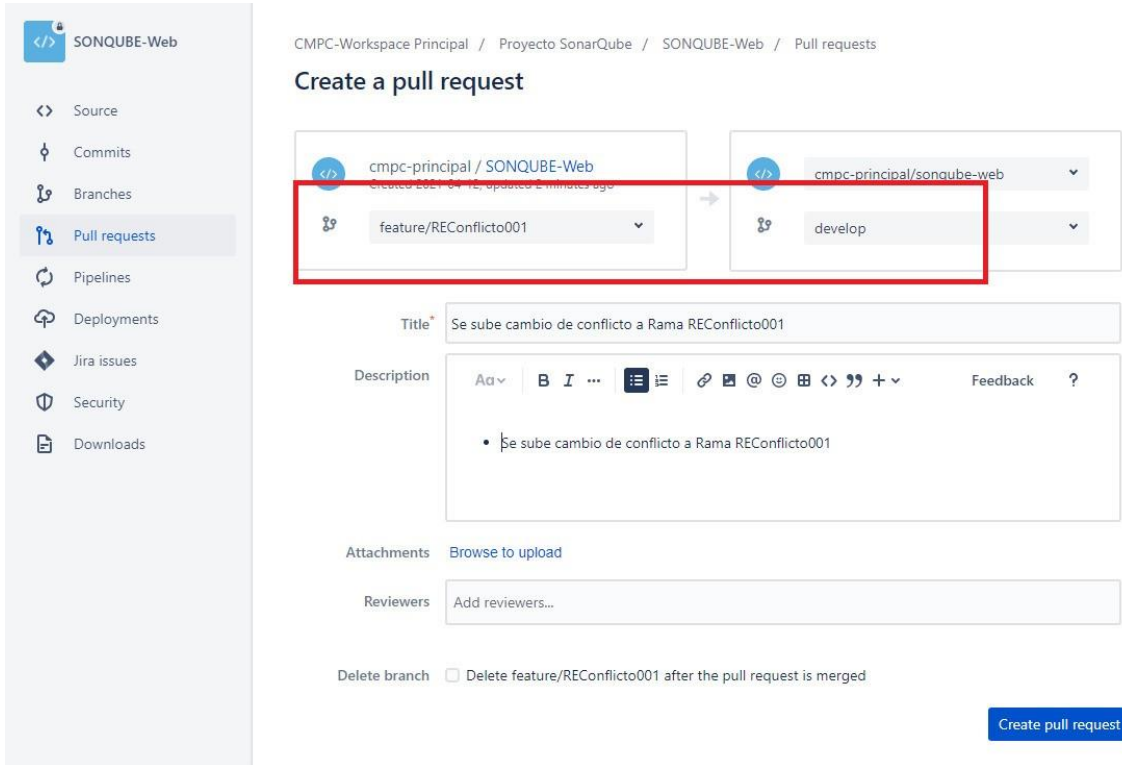
    if (application != null) {
        if (application.SAMLSPConfig != null && application.SAMLSPConfig.WantAuthnRequestSigned) {
            byte[] rawCert = Convert.FromBase64String(application.SAMLSPConfig.X509Cert);
            var cert = new X509Certificate2(rawCert);
        }
    }
}

```

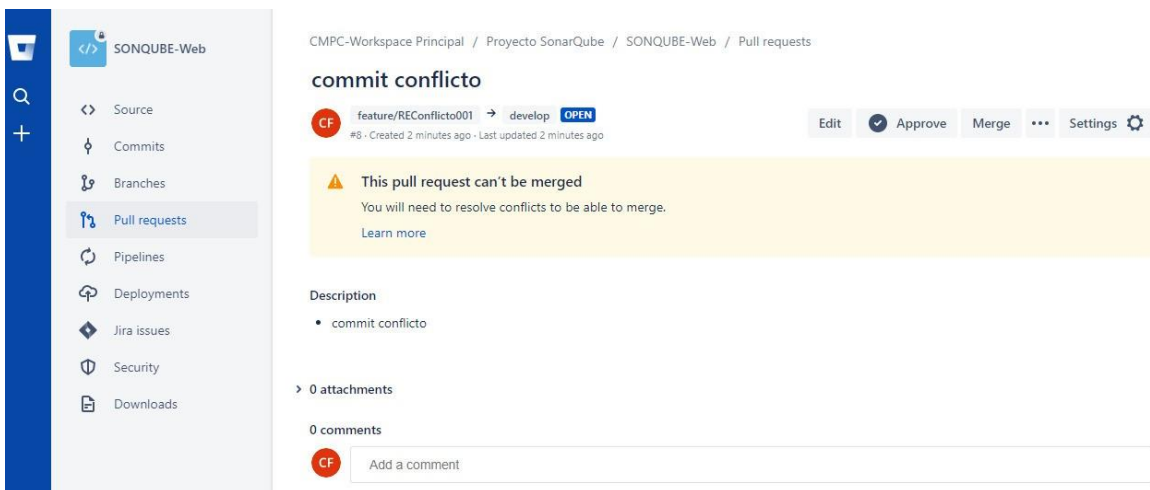
En este punto tenemos que en local la rama REConflicto001, tiene una versión distinta del archivo que también fue modificado por otra persona directamente en la rama develop, como podemos ver en el paso1.

Por lo tanto, se realizará commit en github desktop para que los códigos suban a la rama REConflict001, hasta ahora todo está normal.

Paso 3: A continuación, realizaremos un PULL REQUEST desde nuestra rama REConflicto001 a la rama develop.



Posterior a crear pull request, nos entregará un mensaje de **conflicto** como se ve en la siguiente imagen:



Mas abajo en la misma ventana se verá el conflicto existente:

feature/REConflicto001 → develop **OPEN** Edit Approve Merge ... Settings

> 1 commit 4 commits behind "develop", Sync now

1 file

FILTER BY COMMENTS SORT BY File tree

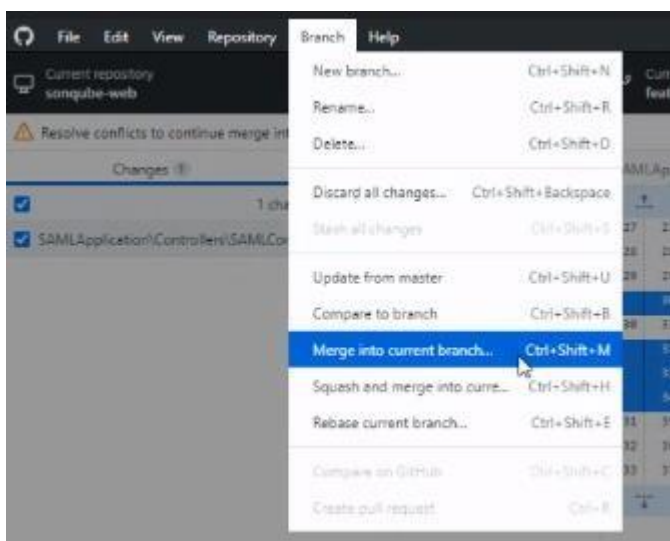
SAMLApplication / Controllers / SAMLController.cs **CONFLICTED**

```
@@ -27,7 +27,11 @@ using System.Security.Cryptography;
27 27 namespace SAMLApplication.Controllers
28 28 {
29 29
30 30 + <<<<<< destination:69b40ce1a7151413d16ddb39fc5356923db4abe
31 31 // TEXTO INCORPORADO DIRECTAMENTE EN BITBUCKET DEVELOP.
32 32 + =====
33 33 + // TEXTO MODIFICADO EN LOCAL EN RAMA REConflicto001
34 34 + >>>>>> source:208fa67c06b55ab9ea0430664a63a78acdd62aac
35 35
36 36 [Authorize]
37 37 public class SAMLController : BaseController
```

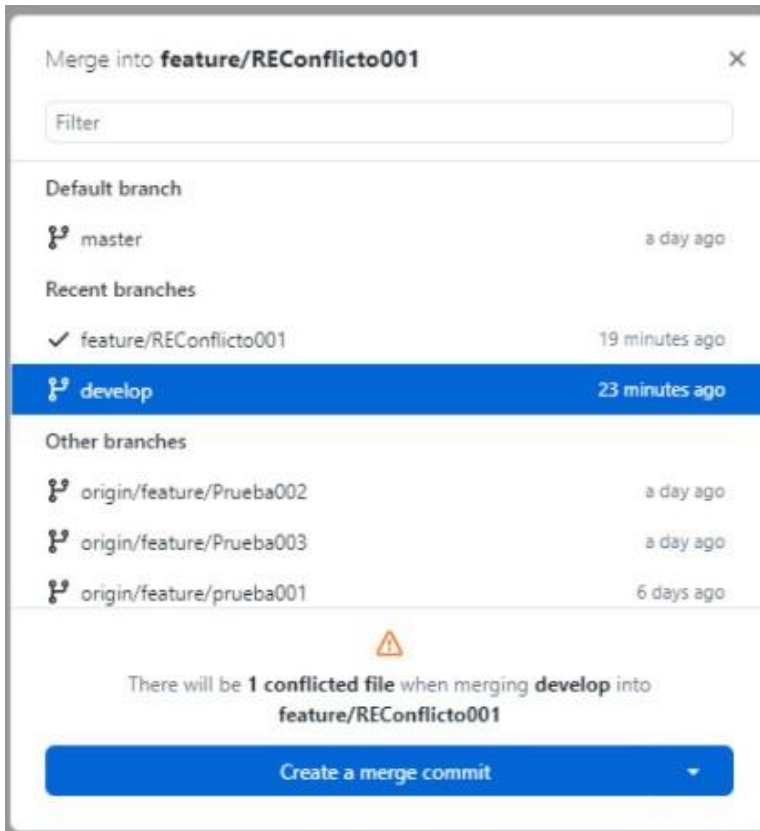
4 Resolución Conflicto en develop

En el paso anterior se ha generado un conflicto de archivos por cambios simultáneos al llegar al repositorio “develop”. Este conflicto debe ser solucionado en la rama “feature” respectiva, a continuación, los pasos que permitirán su corrección.

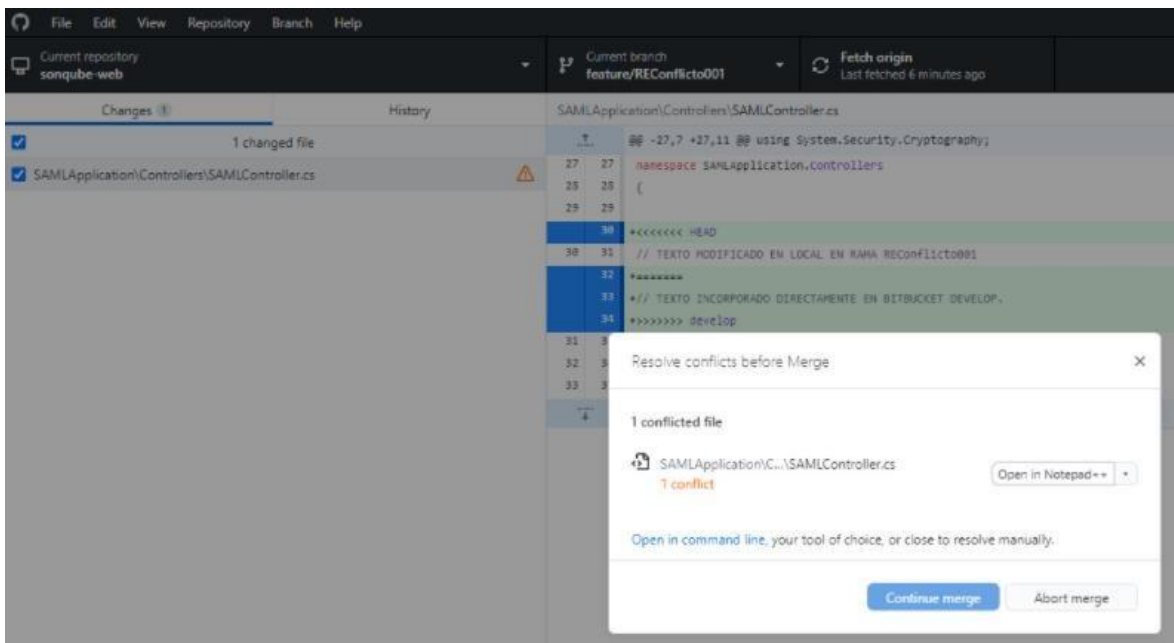
Paso 4: Ir a GitHub Desktop y seleccionar la opción: Branch -> Merge into current Branch (la selección de esta opción debe ser posterior a estar parados en la rama “REConflicto001”).



Paso 5: Con la selección del paso anterior se abrirá la siguiente pantalla donde debemos seleccionar como origen develop, debemos asegurarnos que en la parte superior diga: “Merge into <nuestra rama>” en este caso nuestra rama es REConflicto001.

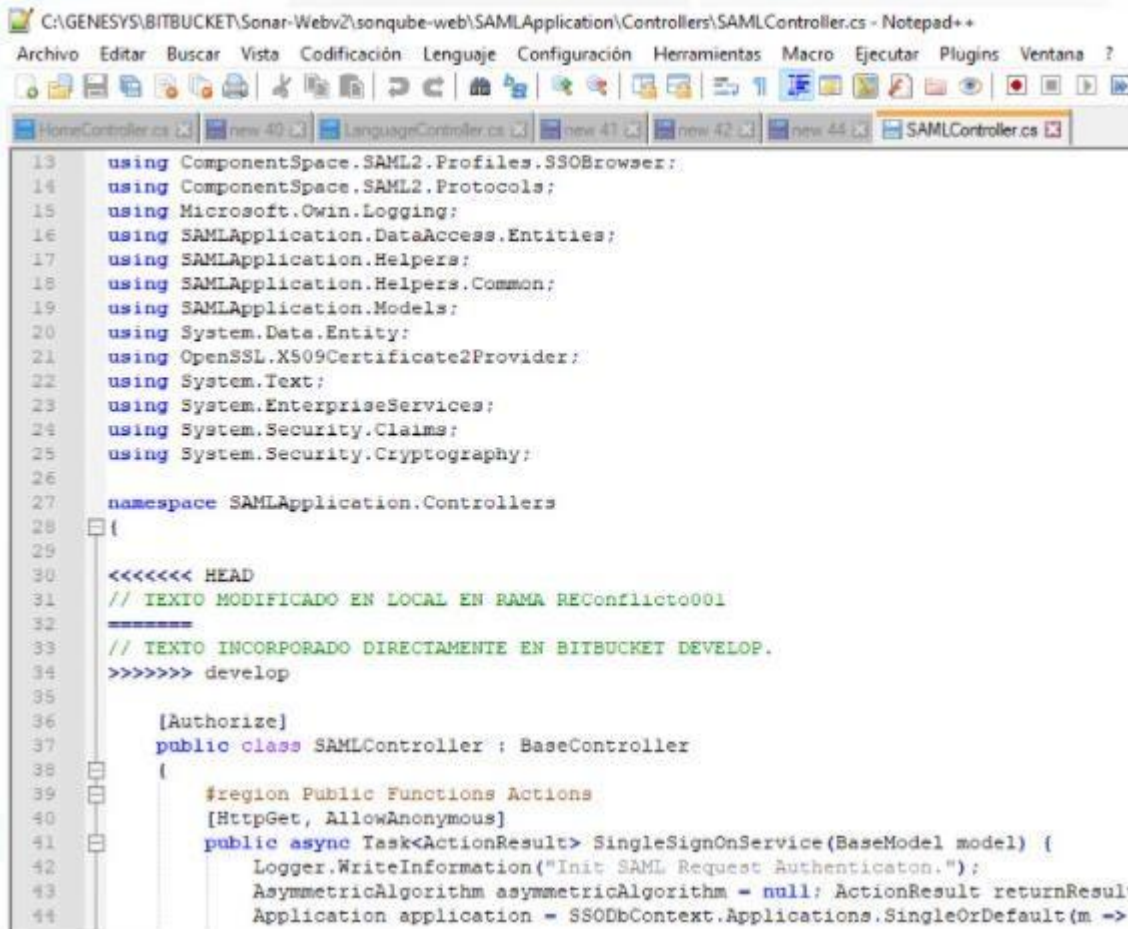


Al presionar “Crear a merge commit”, con la rama origen (donde queremos llegar) seleccionada (en este caso develop), se abrirá la siguiente ventana:



Aquí tenemos dos opciones, presionar el botón “open in Notepad” para corregir el conflicto en un Notepad u otro programa, o bien ir directamente a visual studio, como se mostrará a continuación:

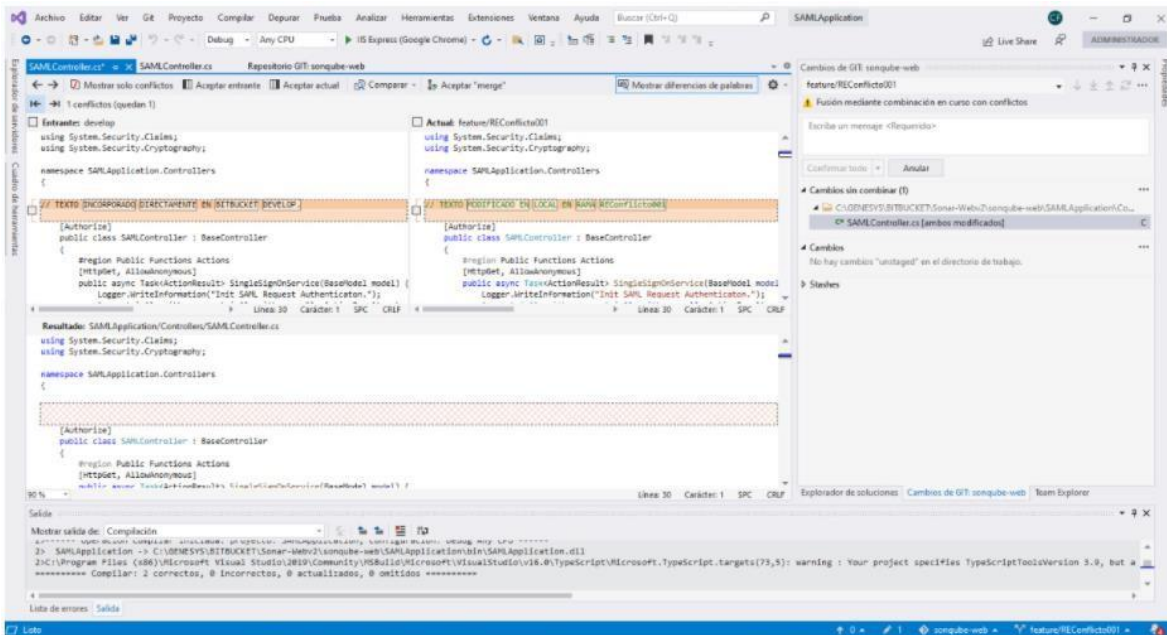
Paso 6: En el caso de que se decida solucionar el conflicto al acceder a Notepad u otra aplicación, se verá de la siguiente forma, donde deberemos decidir manualmente cual es la línea que se requiere conservar, esto es una decisión caso a caso, por lo cual muy probablemente deberemos contactarnos con el desarrollador que realizo los cambios que en este momento están provocando el conflicto.



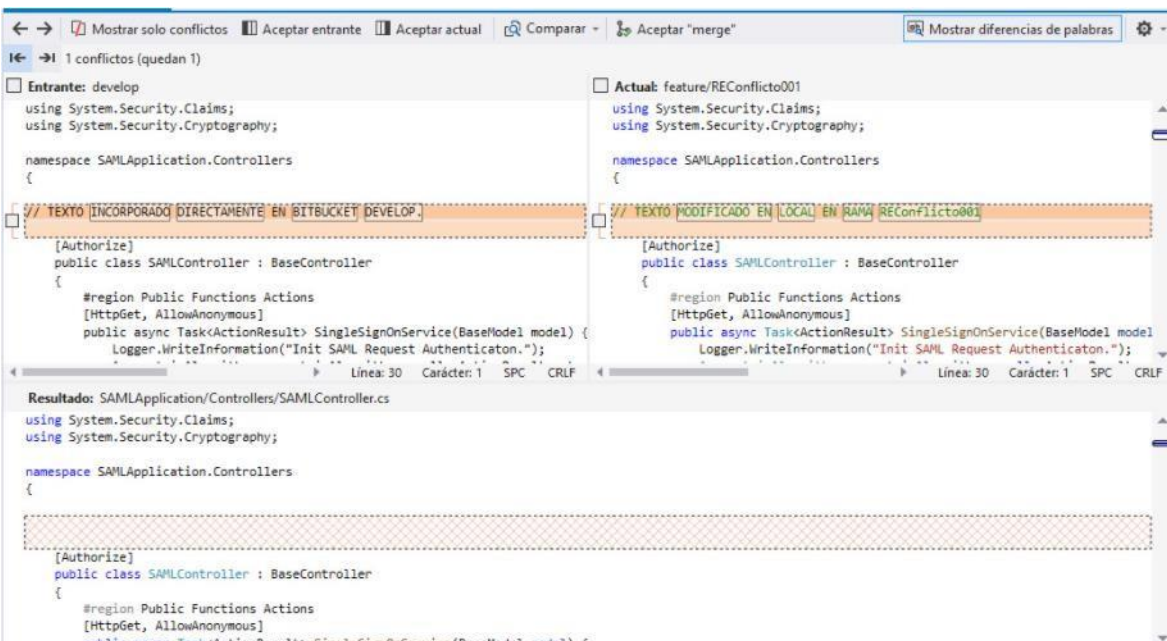
```
C:\GENESYS\BITBUCKET\Sonar-Webv2\sonqube-web\SAMLApplication\Controllers\SAMLController.cs - Notepad++
Archivo  Editar  Buscar  Vista  Codificación  Lenguaje  Configuración  Herramientas  Macro  Ejecutar  Plugins  Ventana  ?
HomeController.cs  new 40  LanguageController.cs  new 41  new 42  new 44  SAMLController.cs
13  using ComponentSpace.SAML2.Profiles.SSOBrowser;
14  using ComponentSpace.SAML2.Protocols;
15  using Microsoft.Owin.Logging;
16  using SAMLApplication.DataAccess.Entities;
17  using SAMLApplication.Helpers;
18  using SAMLApplication.Helpers.Common;
19  using SAMLApplication.Models;
20  using System.Data.Entity;
21  using OpenSSL.X509Certificate2Provider;
22  using System.Text;
23  using System.EnterpriseServices;
24  using System.Security.Claims;
25  using System.Security.Cryptography;
26
27  namespace SAMLApplication.Controllers
28  {
29
30  <<<<<<< HEAD
31  // TEXTO MODIFICADO EN LOCAL EN RAMA REConflicto001
32  =====
33  // TEXTO INCORPORADO DIRECTAMENTE EN BITBUCKET DEVELOP.
34  >>>>>>> develop
35
36  [Authorize]
37  public class SAMLController : BaseController
38  {
39
40      #region Public Functions Actions
41      [HttpGet, AllowAnonymous]
42      public async Task<ActionResult> SingleSignInService(BaseModel model) {
43          Logger.WriteInformation("Init SAML Request Authenticaton.");
44          AsymmetricAlgorithm asymmetricAlgorithm = null; ActionResult returnResult
45          Application application = SSODbContext.Applications.SingleOrDefault(m ->
```

Paso 7: en caso de que decidamos optar por ir a visual studio, este nos entregará una interfaz que permitirá simplificar la actividad de unión de los códigos (resolución de conflicto), por lo tanto accedemos a visual studio.

Paso 8: seleccionar en visual studio en la sección “cambios de GIT”, el archivo con conflicto, con lo que se abrirá la siguiente vista:



Ampliaremos la sección donde se muestra el cambio en la siguiente imagen:



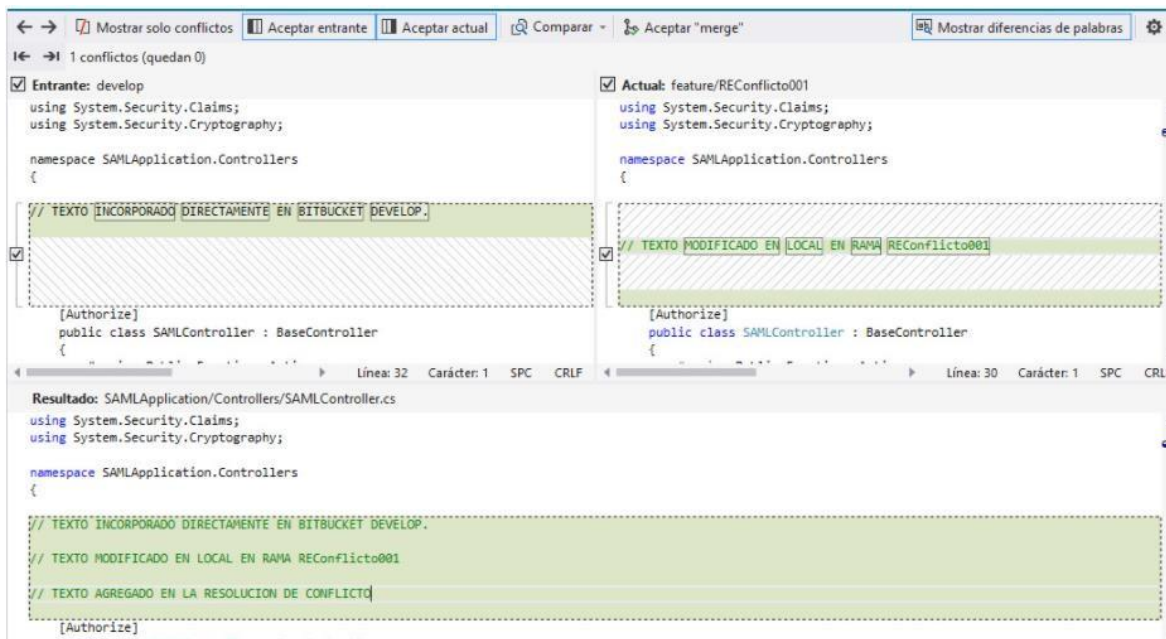
Como se puede ver en la parte inferior de la ventana, muestra el archivo resultante, en la parte izquierda muestra el archivo existente en develop, y en la parte derecha muestra el archivo de nuestra rama local.

En esta ventana podremos seleccionar los cambios que queremos traspasar al archivo resultante.

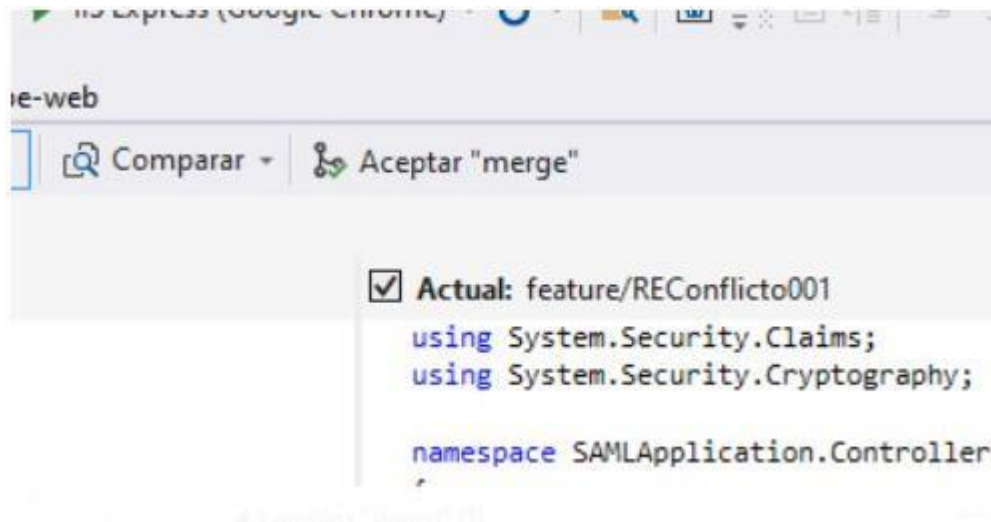
Para el ejemplo seleccionaremos ambos cambios, para ver el resultado que tendremos y adicionalmente agregaremos una línea de código, lo cual podría ser necesario en el caso a caso que se presente en la práctica.

Nota: los textos se agregarán al archivo resultante según el orden en que realicemos la selección.

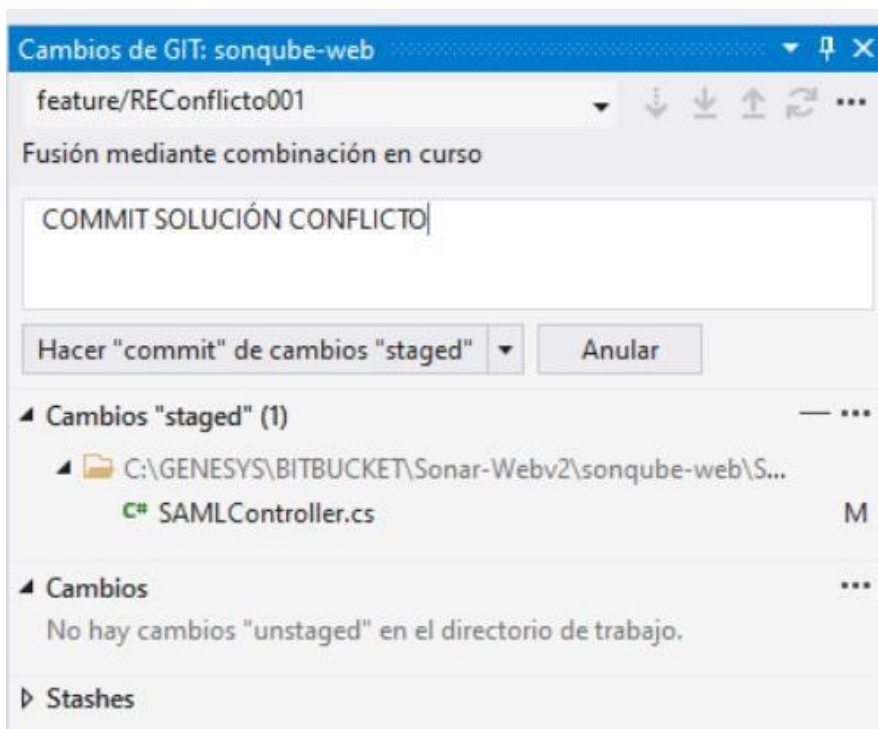
A continuación, podemos ver el archivo resultante una vez seleccionados ambos cambios e incorporada una tercera línea.



Paso 9: Una vez que realicemos los cambios necesarios debemos hacer clic en “aceptar Merge” en la parte superior:

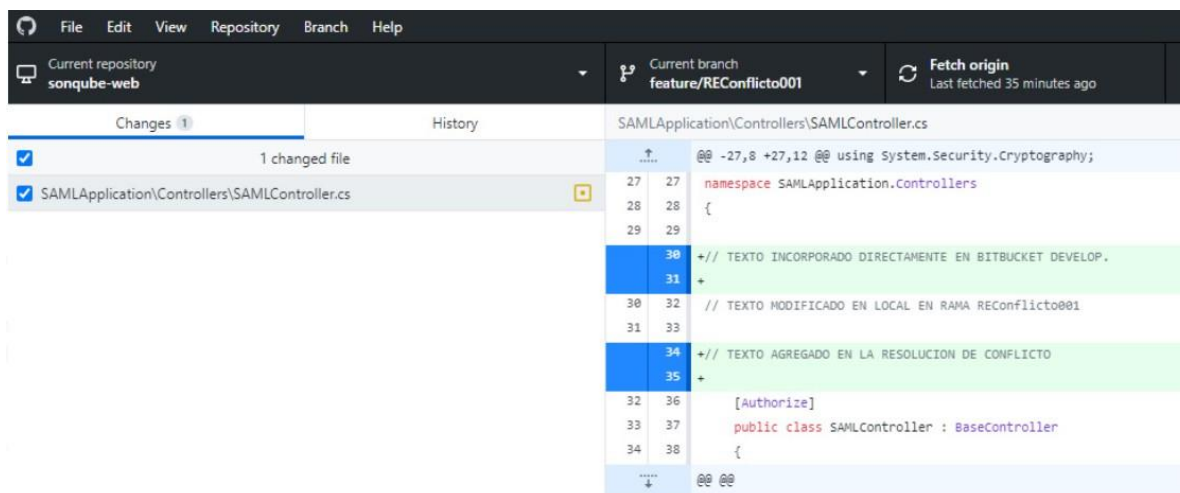


Una vez aceptado el Merge, se abrirá la siguiente ventana:



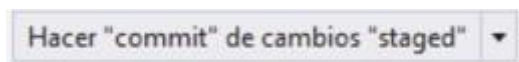
Si hacemos clic en “Hacer “commit” de cambios “staged”, realizaremos commit en nuestra rama.

Este commit, también se puede realizar al ir al GitHub Desktop donde nos mostrará la siguiente vista:

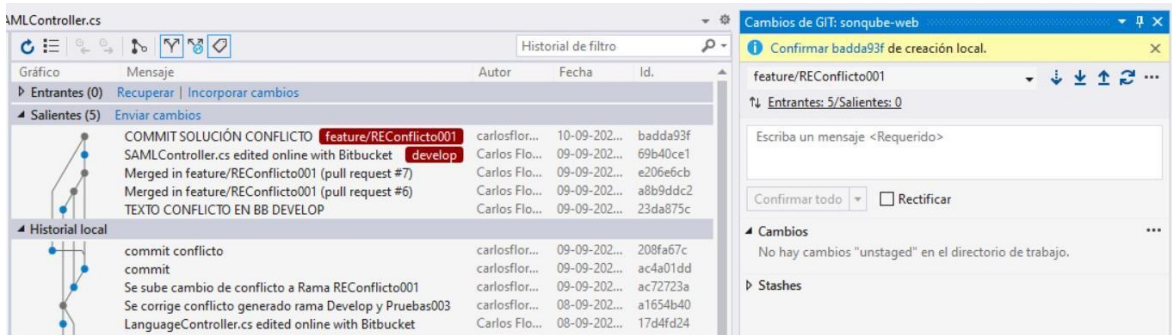


Podemos ver en la imagen anterior que ya no se ve el conflicto, y podemos ver nuestro archivo resultante.

Paso 10: Realizamos commit en Visual Studio, presionando el botón:

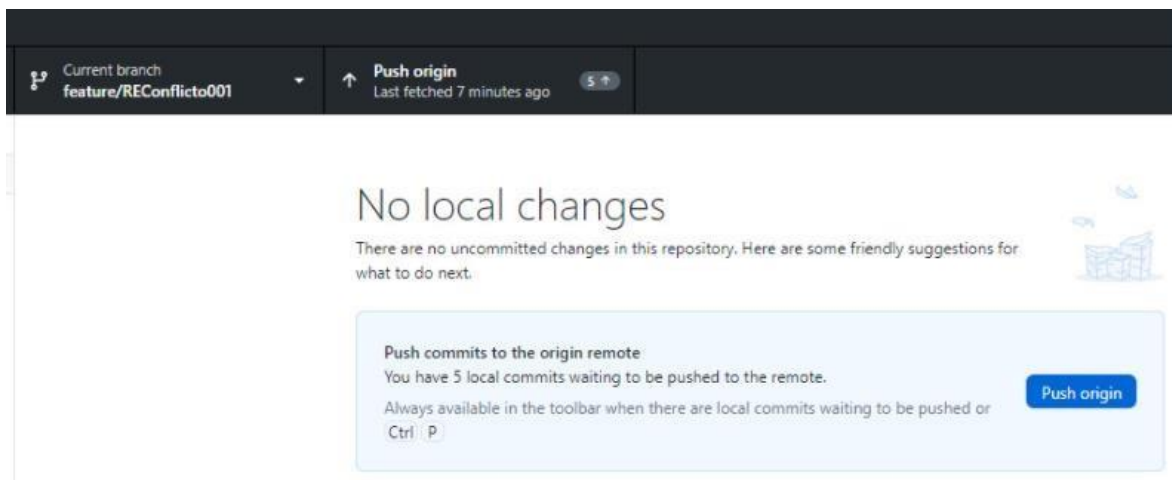


Con esto veremos algo similar a lo siguiente:



Con esta actividad tenemos los códigos de la rama develop y la rama REConflicto001 en nuestra rama REConflicto001 **en form LOCAL**.

Paso 11: Push para subir cambio a la rama REConflicto001 en bitbucket, por lo tanto vamos a “Github”, y realizmos push en la siguiente ventana:



Presionamos el botón “push Origin”, con lo cual ya tenemos los códigos en bitbucket.

Subir fuentes a develop

Paso 12: Pull request a develop. Una vez los códigos estén en rama bitbucket REConflicto001, Realizamos pull request a develop, tal como se ha realizado siempre.

CMPC-Workspace Principal / Proyecto SonarQube / SONQUBE-Web / Pull requests

Create a pull request

cmppc-principal / SONQUBE-Web
Created 2021-04-12, updated 2 minutes ago

feature/REConflicto001

cmppc-principal/sonqube-web

develop

Title*

Description

Ad **B** **I** ... Feedback ?

- Conflicto Resuelto

Attachments [Browse to upload](#)

Reviewers

Delete branch Delete feature/REConflicto001 after the pull request is merged

[Create pull request](#)

Al hacer click en crear pull request, veremos que ya no existe conflicto, tal como se puede ver en la siguiente pantalla, podemos ver que ya no existe un mensaje de conflicto como se vio en la misma pantalla del paso 3:

commit conflicto

feature/REConflicto001 → develop **OPEN**

#8 · Created 23 hours ago · Last updated 2 minutes ago

Edit **Unapprove** Merge ... Settings

Description

- Conflicto Resuelto



0 attachments

0 comments

Add a comment

2 commits

1 file

FILTER BY COMMENTS  

SORT BY File tree

```

SAMLApplication / Controllers / SAMLController.cs
@@ -29,6 +29,10 @@ namespace SAMLApplication.Controllers
29 29
30 30 // TEXTO INCORPORADO DIRECTAMENTE EN BITBUCKET DEVELOP.
31 31
32 + // TEXTO MODIFICADO EN LOCAL EN RAMA REConflicto001
33 +
34 + // TEXTO AGREGADO EN LA RESOLUCION DE CONFLICTO
35 +
32 36 [Authorize]
33 37 public class SAMLController : BaseController
34 38 {
    
```

Paso 13: realizamos Approve y Merge, normalmente. Fin.

CMPC-Workspace Principal / Proyecto SonarQube / SONQUBE-Web / Pull requests

commit conflicto

feature/REConflicto001 → develop **MERGED**

#8 · Created 23 hours ago · Last updated 2 minutes ago

... Settings

Merged pull request

Merged in feature/REConflicto001 (pull request #8)

[edaaf99](#) · Author: Carlos Flores · Closed by: Carlos Flores · 2 minutes ago

Description

- Conflicto Resuelto

5 Resolución Conflicto en master

El procedimiento para resolver un conflicto en master, es equivalente a la resolución en “develop”, por lo cual en el **paso 4**, deberemos seleccionar como origen, la rama master, en lugar de la rama develop. Esto, con la precaución que nuestro cambio deberemos subirlo primero a la rama **develop (con un pull request del desarrollador)** y posteriormente a la rama **master (con un pull request del JP cmppc)**.

Esta situación de **conflicto al ir a la master, no debiera existir**, dado que un desarrollador que resolvió un incidente, y por ende fue directo a la master, inmediatamente después debería haber llevado sus códigos a la develop, por lo cual el conflicto al subir un requerimiento sería en la develop y no en la master (tal como se describe este documento),

PERO, si el desarrollador que resolvió el incidente (hotfix), no llevó sus códigos inmediatamente a la develop (operando fuera del procedimiento definido) Y SOLO lo llevó a la master, a quien está pasando un requerimiento se le producirá el conflicto al llegar a la master.

Resolver este conflicto a la master de la forma expresada en este manual, impedirá que el desarrollador que pasó previamente un incidente (IN) a la master, pueda llevar sus códigos desde su rama IN a la develop¹ porque ya lo habremos realizado al solucionar el conflicto en “**Resolución del conflicto en master**”.

Nota: Un conflicto en la master, **podría ser válido** y dentro del procedimiento, si se produce entre dos usuarios que están resolviendo un incidente dentro de los mismos códigos (situación poco probable), o quien resuelve un incidente y en el intertanto un requerimiento subió a la master (poco probable dado que un incidente debiera ser solucionado en poco tiempo).

¹ Considerar qué, si bien no podrá llevar sus códigos, otro desarrollador lo realizará por él, no es que los códigos se pierdan pero su falta en llevar los códigos a la develop en forma oportuna, provocó trabajo adicional a otro desarrollador que tendrá que realizar esta actividad por él.