

Pràctiques

Butlletí Pràctica 1

Introducció a Visual Studio i Azure DevOps

Enginyeria del Programari

ETS Enginyeria Informàtica
DSIC – UPV

Curs 2021-2022

1. Objectiu

L'objectiu de la sessió de laboratori és introduir a l'alumne a les principals funcionalitats de *Visual Studio 2019* i la seua relació amb l'eina de gestió de projectes en equip i control de versions *Azure DevOps*¹. L'alumne haurà de desenvolupar obligatòriament totes les activitats descrites fins al punt 5 (inclusivament). El punt 6 sobre ramificació i control de branques és treball opcional.

Nota: Les captures que apareixen en aquest butlletí poden correspondre a projectes realitzats en cursos diferents.

2. Creació d'un projecte d'equip d'*Azure DevOps* i d'una solució inicial protegida en el servidor

Per a la realització d'aquesta pràctica és necessari haver seguit els passos per a la creació d'un projecte d'equip mitjançant *Azure DevOps* en el núvol, haver donat d'alta als membres de l'equip de pràctiques i haver creat una solució en aquest projecte. Es recomana utilitzar com a nom per al servidor de *Azure DevOps* el següent esquema de noms: *2020-upv-isw-<nom grup laboratori>-<nom equip>*. Per exemple, el servidor podria estar allotjat en ***dev.azure.com/2021-upv-isw-3a1-E1***.

Aquestes tasques prèvies les realitzarà un únic membre de l'equip (Team Master).

La realització d'aquestes activitats ha sigut descrita en el seminari de teoria Tema 2 i Tema 3 (*Visual Studio & DevOps & Git*). Abans de passar als següents punts s'han d'haver realitzat aquestes activitats (consulte el material existent en PoliformaT relatiu a aquestos seminaris (Part I y Part II. També es pot consultar els videos disponibles en el canal General ([Azure DevOps: Crear Organización](#) y [Visual Studio: Desarrollo del Proyecto SW](#))). En finalitzar aquestes tasques, l'equip de desenvolupament tindrà un projecte d'equip amb una solució Visual Studio que conté les carpetes de solucions *Presentation*, *Library* (amb carpetes per a *BusinessLogic* i *Persistence*), *Testing*, tal com es mostra a la Figura 1.

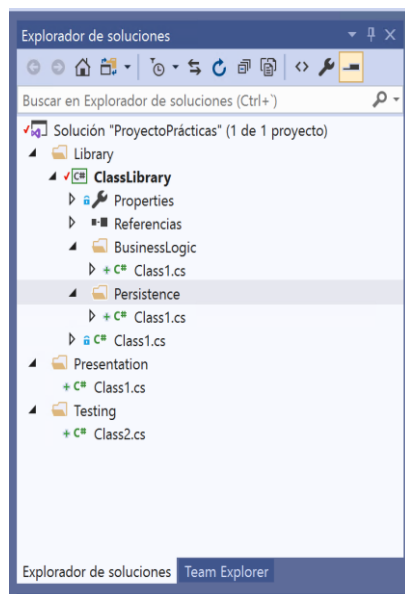


Figura 1. Solució inicial amb les carpetes de solucions.

Recordeu que totes les carpetes creades deuen tindre al menys un element per a que Git les sincronitze correctament en el repositori remot. Git no gestiona adequadament les carpetes buides, per lo que si putjeu al repositori remot una carpeta sense contingut, quan el repositori es torne a clonar, no es possible afegir contingut a la carpeta, per la qual cosa deurien ser eliminades

¹ Azure DevOps és la nova nomenclatura de Microsoft per a Team Services.

i tornales a creat. Per a evitar aquest inconvenient, les carpetes creades a la solució no deuen estar buides, tal com mostra la Figura 1 (hem afegit una classe).

3. Connexió al projecte existent des de Visual Studio

En aquest moment, cada membre de l'equip pot connectar-se des de *Visual Studio* **de forma individual** al projecte de *Azure DevOps*, descarregar del servidor en el núvol una còpia de la solució i assignar-la a un àrea de treball local (directori local privat²) per a treballar amb ella. D'aquesta manera cada alumne podrà realitzar les següents activitats **de forma individual**. Per a tot açò realitze els següents passos:

- Inicie l'eina de desenvolupament *Microsoft Visual Studio 2019*
- Apareixerà una finestra de diàleg on haurà de seleccionar l'opció *Continuar sin código*

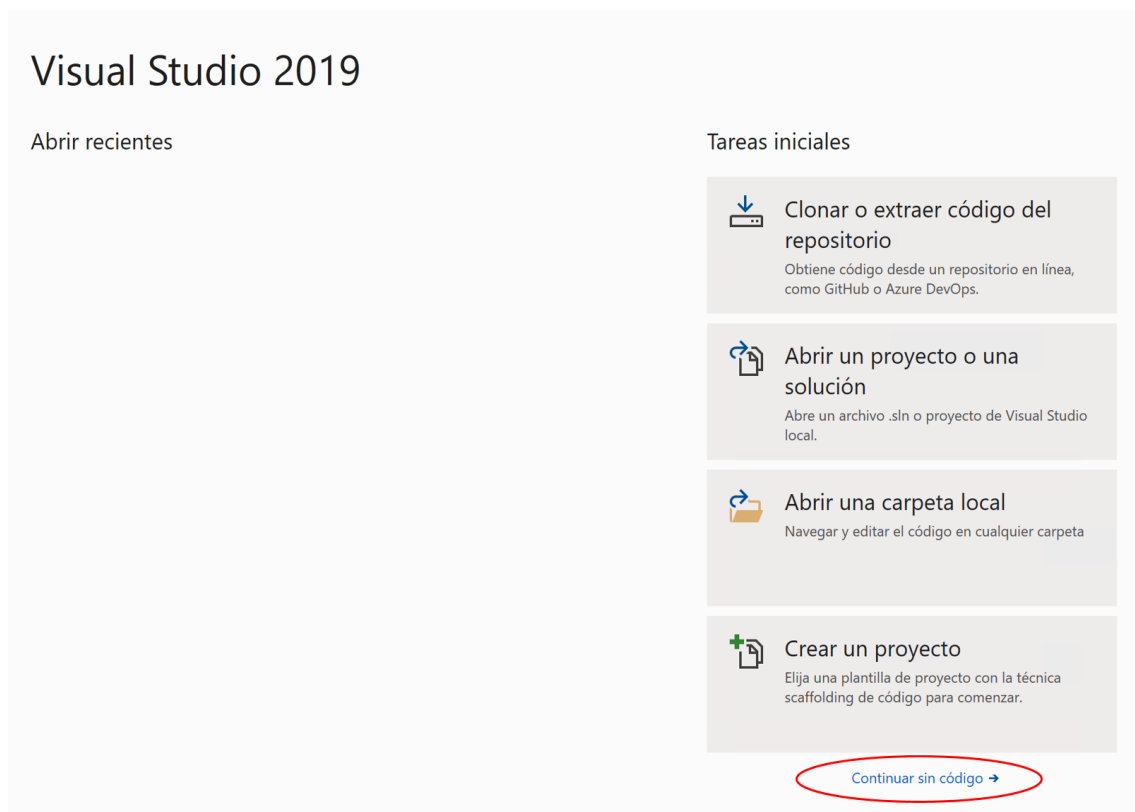


Figura 2. Seleccionar “continuar sin código” des de la finestra inicial

- Inicie sessió amb el seu compte d'usuari Microsoft assignat al projecte:
 - Archivo->Configuración de La cuenta->Iniciar Sesión*

² **Important:** en els laboratoris del DSIC es **recomana** utilitzar el **directori privat** de cada usuari a partir de la ruta C:\Users\nomUsuari\... però **no** utilitzar la unitat W:, que es una unitat de xarxa en la que pot pedres la connexió, produint errades en la sincronització i errades en Visual Studio.

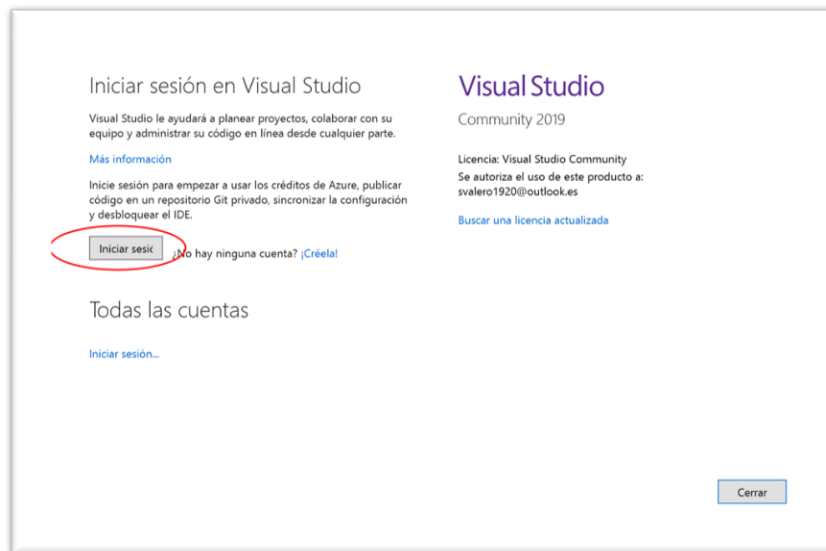


Figura 3. Iniciar sessió a Visual Studio

- d. Connecte's al seu projecte d'equip. Seleccione *Team Explorer* (finestra de la dreta) i a continuació seleccione el botó d'*Administrar conexiones* (icona de l'endoll de color verd), com es mostra a la Figura 4.

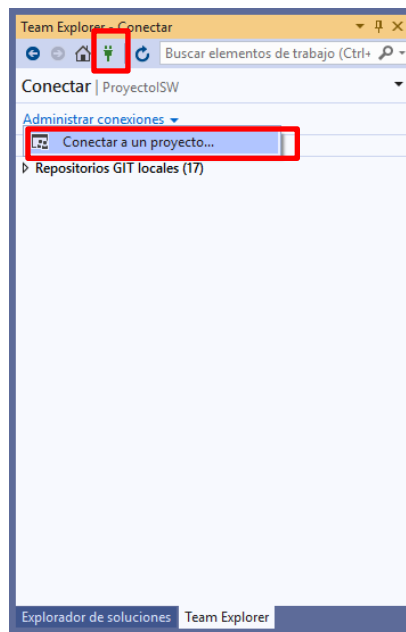


Figura 4. Conectar-se a un projecte des d'Administrar Conexiones

- e. Una vegada haja polsat en el menú contextual de la figura anterior li apareixerà una finestra similar a la de la següent figura. A la pestanya apareix el compte de Microsoft amb el qual heu iniciat sessió (Figura 5-1). Davall, apareix la relació d'organitzacions a les que pertanyeu (en el vostre cas, és normal que només hi aparega una). Per a cada organització, apareixen els projectes que té (novament, només un en el vostre cas). Seleccione el símbol de git del seu projecte (Figura 5-2), confirme la ruta a la qual vol desar el projecte (Figura 5-3) i polseu *Clonar* (Figura 5-4). És possible que necessite confirmar el compte amb el qual va a connectar-se al repositori remot.

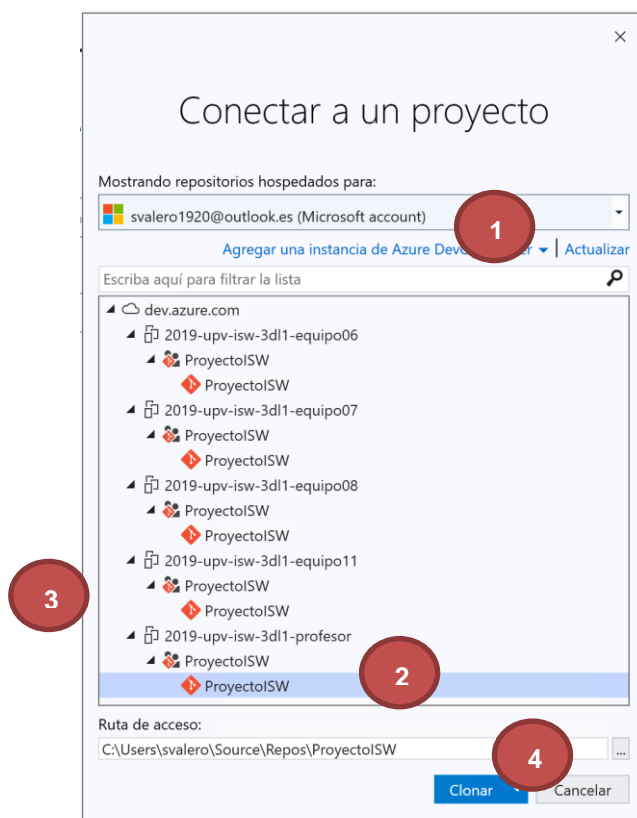


Figura 5. Connectar-se a un projecte d'una organització

- f. Obriga la solució creada pel *Team Master*. Una vegada s'haja connectat al projecte d'equip, ha obtingut i assignat la seua àrea de treball local pot obrir la solució que el *Team Master* ha creat prèviament (punt 2 d'aquesta guia). Des del *Team Explorer* faça doble clic sobre la solució que es troba a la secció de solucions (Figura 6-2).

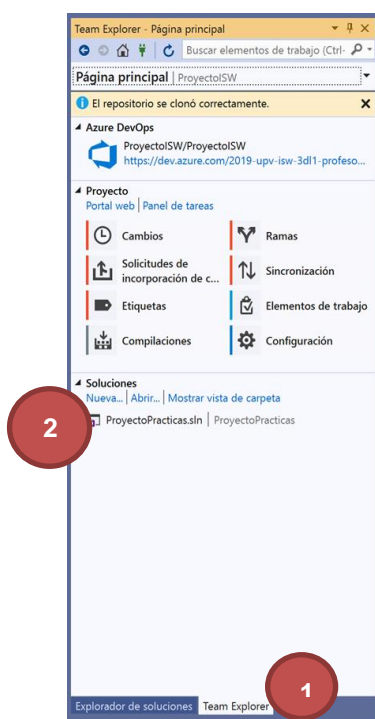


Figura 6. Obrir una solució des del Team Explorer

- g. Vaja a l'explorador de solucions (pestanya “*Explorador de soluciones*” en la part dreta inferior de Visual Studio) i comprova que té una solució com la creada pel *Team Master* en el servidor tal com es mostrava en la Figura 1.

4. Introducció a les eines de depuració de Visual Studio

Les eines de depuració de *Visual Studio* són un element fonamental per al desenvolupament d'aplicacions en *Visual Studio* i la detecció de *bugs* en el codi. Aquestes eines permeten establir punts de ruptura en el codi, executar codi pas a pas, inspeccionar valors de variables en temps d'execució, etc.

4.1 Creació d'un projecte de consola

Per a iniciar-nos en les labors de depuració anem a **crear un projecte de consola en la carpeta de solucions *Testing***. Per a açò:

- Creu una subcarpeta en *Testing* denominada *Lab1*
- Botó dret del ratolí en carpeta *Lab1* -> *Agregar* -> *Nuevo Proyecto*-> *Aplicación de consola* i anomenar a la aplicació ***HelloWorldApp***



Figura 7. Creació aplicació de consola *HelloWorldApp*

- Una vegada generada l'aplicació observe les diferents finestres que Visual Studio li proporciona (veure Figura 8): quadre d'eines (esquerra), explorador de solucions (dreta a la part de dalt), propietats (dreta a la part de baix), editor de codi (centre a la part de dalt), llistes d'errors i eixida (centre a la part de baix).
- En el cas que el projecte que acaba de crear no aparega a l'explorador de solucions ressaltat en negreta (projecte d'inici) haurà d'indicar-li a Visual Studio que aquest és el seu projecte d'inici actual a l'efecte de compilació i execució. Per a açò preme el botó dret del ratolí sobre el projecte *HelloWorldApp* i seleccione l'opció de menú ***Establecer como proyecto de Inicio***. Observe que el nom del projecte ha passat a estar ressaltat en negreta.

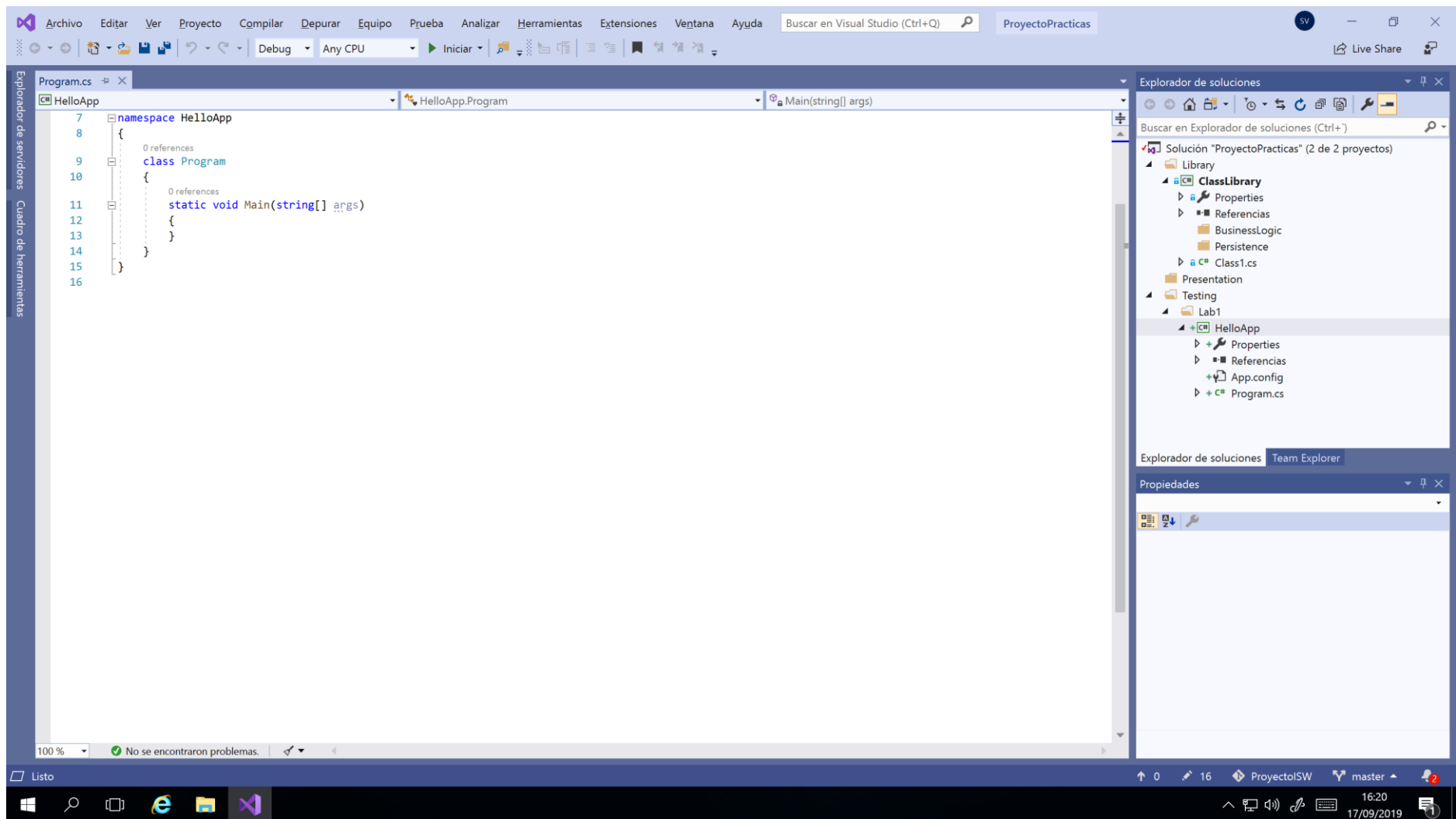
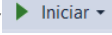


Figura 8. Finestres de treball de Visual Studio

4.2 Compilar i executar

Per a compilar i executar aquesta aplicació premeu el triangle verd () en la zona superior de l'eina de desenvolupament. Per la finestra d'eixida observareu el resultat de la compilació i de l'execució. Encara que el programa actual no realitza cap acció de moment, si se podrà observar com s'obri i com es tanca una finestra de consola.

4.3 Exploració d'errors de compilació

Obriu el fitxer *Program.cs* i introduïu un error artificialment en el mètode *main*, per exemple, escrivint la sentència *fadsfsde*. Si torna a compilar el seu programa observareu la llista d'errors en la finestra corresponent. Si feu clic sobre l'error, el cursor apareixerà en la zona del seu programa on es troba el mateix.

4.4 Creant una aplicació HelloWorld en C#

Per a poder practicar les capacitats de depuració anem a generar un senzill programa d'exemple. Obriu el fitxer *Program.cs* i modifiqueu el seu contingut perquè quede com segueix.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace HelloWorldApp
{
    class Program
    {
        private static List<string> nacionalities;
        private static void InitList()
        {
            nacionalities = new List<string>()
            {
                "Australian",
                "Mongolian",
                "Russian",
                "Austrian",
                "Brazilian"
            };
        }

        static void Main(string[] args)
        {
            InitList();
            nacionalities.Sort();
            foreach (string nationality in nacionalities)
            {
                Console.WriteLine(nationality);
            }
        }
    }
}
```

Es pot observar que el programa anterior té unes sentències *using* que actuen com a directrius del compilador, fent que es puguin utilitzar tipus definits en l'espai de noms importat sense necessitat d'especificar-los de forma explícita en el codi. A continuació, es defineix un espai de noms *HelloWorldApp* i en ell la classe *Program*. Esta classe conté la definició d'un atribut privat

del tipus llista de *strings* (*List<string>*), un mètode estàtic d'inicialització de la mateixa i un mètode estàtic *main* que és el que iniciarà l'aplicació. El codi proposat és relativament senzill d'entendre i comparteix sintaxi amb codi Java. Concretament s'està creant una llista de *strings*, s'ordena alfabèticament (mètode *Sort()*) i després es recorre (bucle *foreach*) i es visualitza el seu contingut per pantalla. Aquest codi serà usat per a practicar les capacitats de depuració. Compile i execute el programa. De nou veurà que ha aparegut una finestra de consola, s'ha executat el programa ràpidament però no ha pogut observar l'eixida generada pel mateix.

4.5 Execució d'un programa pas a pas

En la majoria de IDEs actuals és possible incloure punts de ruptura on es deté l'execució per a poder des d'aqueix moment executar sentència a sentència el codi i anar observant el seu comportament de forma controlada. Per a practicar aquest aspecte realitzarem les següents accions

- Inserisca un punt de ruptura en la línia de codi *InitList()*. Per a açò preme el ratolí sobre la columna de color gris que apareix a l'esquerra del codi, a l'altura de la línia on desitja interrompre l'execució (veure Figura 9).

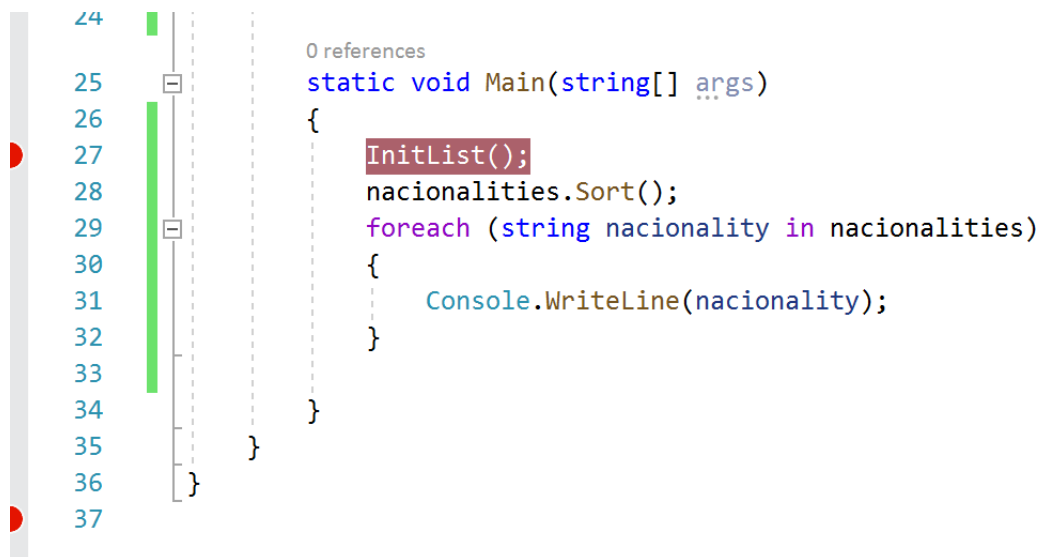


Figura 9. Inserció de punt de ruptura

- Compile i iniciï la depuració del codi (preme el triangle verd) i observe com l'execució es deté en la línia on es va introduir el cercle roig (apareix una fletxa groga indicant on es troba l'execució i quina serà la següent instrucció a executar). Si observa la finestra de consola creada, aquesta apareixerà (de moment) buida.
- Execute pas a pas l'aplicació. Per a açò utilitzi els controls d'execució pas a pas (veure Figura 10). Pot detenir l'execució (quadrat roig), executar la següent instrucció (Figura 10-1 o F11), executar tot un procediment/mètode de forma completa sense entrar en ell (Figura 10-2 o F10) o executar totes les sentències que hi ha en un procediment per a eixir d'ell (Figura 10-3 o Majúsc+F11).

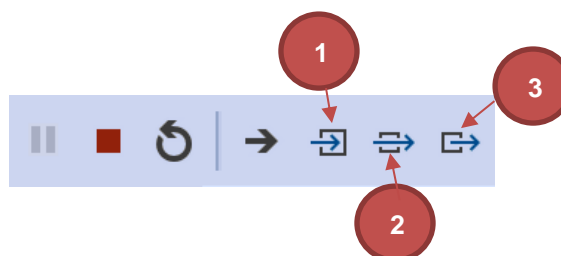


Figura 10. Execució pas a pas

4.6 Observant els valors de dades del programa

Mentre el programa es troba en execució és possible observar els valors d'objectes, atributs, variables, etc. Açò és **extremadament útil** quan està depurant el seu programa. Per a observar el valor de qualsevol element situe el cursor sobre el mateix.

- Quan el seu programa arribi a l'execució de la sentència `nacionalities.Sort()` pose el cursor sobre la variable `nacionalities` i observe el seu contingut (veure Figura 14).

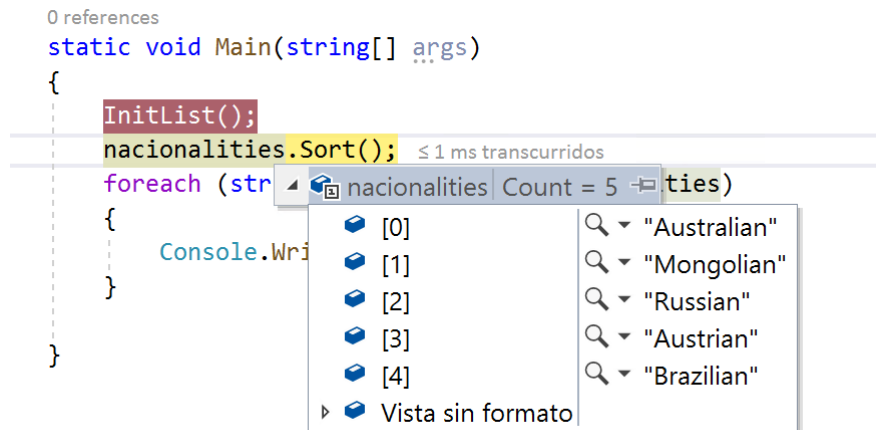


Figura 11. Inspeccionant valors de dades durant l'execució

- Execute aqueixa sentència passe a pas (F11) i torne a observar la llista comprovant que s'ha ordenat alfabèticament.
- Seguisca executant pas a pas el seu programa i observe com per la finestra de consola apareix el contingut de la llista (Figura 12).

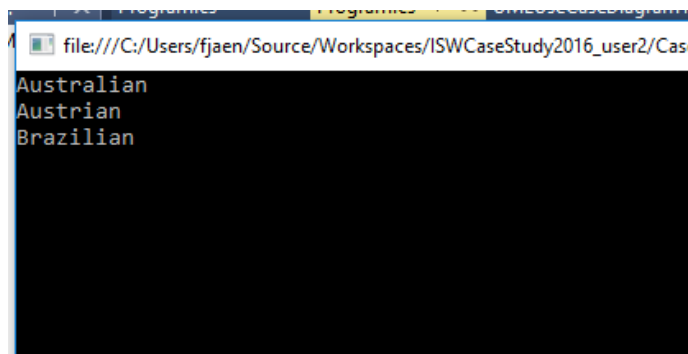


Figura 12. Eixida per consola durant l'execució

- Pot tornar cap a arrere en l'execució simplement arrossegant la fletxa groga cap amunt a la línia de codi des d'on desitge repetir l'execució del seu programa. Açò pot resultar molt útil doncs ens permet canviar el flux original del nostre codi en temps d'execució.

5. Operacions bàsiques de control de versions

Un dels avantatges d'utilitzar un entorn de desenvolupament en equip com *Azure DevOps* és la possibilitat de tenir control i gestió sobre les diferents versions del projecte que es vagen

generant. De fet, quan va crear el seu projecte va indicar què sistema de control de versions utilitzar, **GIT**. Un sistema de control de versions és una combinació de tecnologies i pràctiques per a controlar els canvis realitzats en els diferents elements que componen el projecte, en particular en el codi font, documentació, conjunts de proves, etc.

Existeix una nomenclatura específica a Git, que esmentarem a continuació, atès que és el sistema de control de versions que utilitzarem per defecte:

- h. **Repositori remot:** el repositori que resideix en el servidor de Microsoft Azure DevOps i en el qual tots els membres de l'equip poden emmagatzemar els seus canvis.
- i. **Repositori local:** conté una còpia del repositori remot on un membre de l'equip pot treballar localment en la seua màquina, podent no estar sincronitzat en tot moment amb el repositori remot.
- j. **Commit:** Emmagatzema els canvis realitzats localment en el repositori local. Guarda informació sobre els arxius que han sigut modificats, informació sobre qui ha fet el canvi i un missatge (en aquest cas obligatori) que descriga la naturalesa del canvi.
- k. **Clone:** Obté una còpia en local del Projecte a partir de repositori remot. En la còpia local és possible observar tots els canvis que s'han dut a terme des de l'inici del projecte.
- l. **Pull:** Obté els últims commits des del repositori remot i els emmagatzema en el repositori local.
- m. **Push:** Puja els commits emmagatzemats en el repositori local al repositori remot.
- n. **Merge:** És el procés mitjançant el qual diferents commits es combinen per a obtenir una nova versió del Projecte.
- o. **Conflict:** Pot sorgir quan dos o més membres de l'equip han realitzat canvis diferents en el mateix fragment de codi. Quan els diferents commits es combinen en un procés de mescla (merge), per exemple, com a part d'una operació pull, i es detecta que existeixen canvis en el mateix arxiu, el sistema li demana a l'usuari amb quin canvi es quedarà perquè siga emmagatzemat. Com a resultat de la resolució del conflicte obtindrem un nou commit i una nova versió coherent del projecte.

El control de versions **Git** segueix un flux de treball que la major part dels desenvolupadors segueixen quan escriuen codi i ho comparteixen amb l'equip de desenvolupament. Els passos són els següents:

1. Obtenir una còpia en local del codi del projecte
2. Fer canvis en el codi
3. Una vegada que s'han acabat els canvis fer públic el codi modificat a la resta de l'equip
4. Una vegada acceptat, fusionar amb el codi en el repositori de l'equip.

Gràficament l'anterior flux de treball pot representar-se de la següent forma:

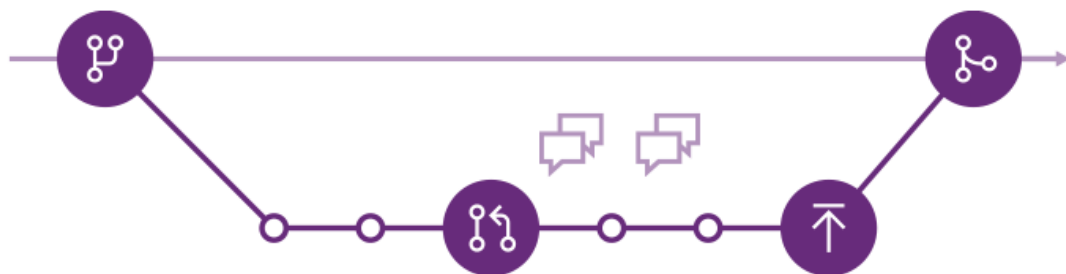


Figura 13 Flux de treball de Git

A continuació introduïrem un conjunt de termes i comandos associats com *repositories*, *branches*, *commits*, and *pull requests* que són comuns a diferents gestors de control de versions com *Team Foundation Version Control* o *Subversion*. Cal indicar que els comandos es comporten de manera lleugerament diferent en *Git*.

- Crear una branca. Les branques s'utilitzen per als canvis que el desenvolupador vol dur a terme en local i porten obligatòriament un nom.
- Commit canvis a la teua branca local. En local podem realitzar tots els canvis que considerem oportuns a la nostra branca.
- Push la branca al repositori remot.
- Crear un pull request de manera que la resta de l'equip pugui revisar els teus canvis.
- Completar el pull request i resoldre qualsevol conflicte generat per canvis que uns altres han realitzat sobre la teua branca.

La creació de branques, l'actualització i la mescla entre les mateixes s'explica al final del document.

A continuació simularem un escenari típic de funcionament, sobre la còpia local del projecte canviarem el contingut d'algun fitxer, usarem l'opció canvis i posteriorment confirmarem els mateixos. Amb açò les nostres modificacions s'incorporen al repositori local. Utilitzant l'opció *Sincronización* pujarem els canvis al repositori remot i resoldrem els possibles conflictes que sorgisquen.

Sobre el projecte de depuració recentment creat durem a terme les següents activitats:

- a) Modifique qualsevol part del seu programa. Per exemple, la classe *Program* perquè el primer element de la llista siga un altre (cada membre de l'equip pot triar una cadena diferent ("Italian", "Spanish", etc.))
- b) Vaja a Team Explorer -> Cambios (afegisca un comentari descriptiu de la nova versió o modificació introduïda).

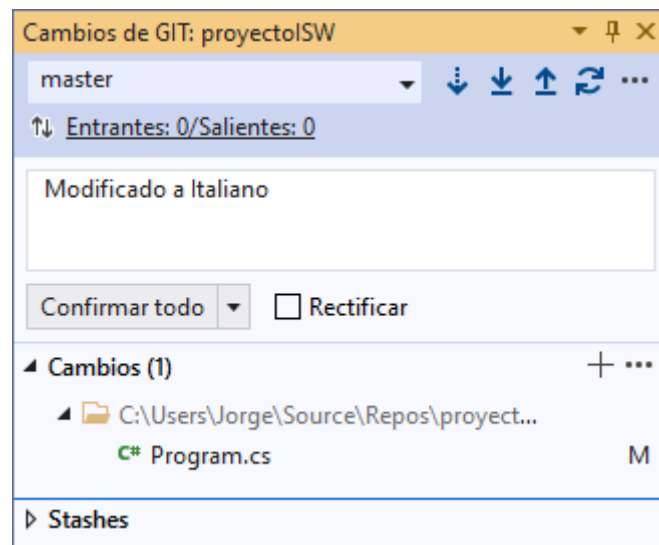


Figura 14 Finestra per a pujar modificacions al repositori local

Al pulsar **Confirmar todo** li apareixerà la següent finestra on ens indica el resultat de l'operació en el repositori local i ens dona l'opció de pujar-ho al repositori central (al pulsar en sincronitzar³).

³ No és obligatori pujar cada versió local al repositori central. Cal pujar la versió al repositori central quan siga una versió correcta. Es pot accedir a l'opció per a sincronitzar des de *Team Explorer*.

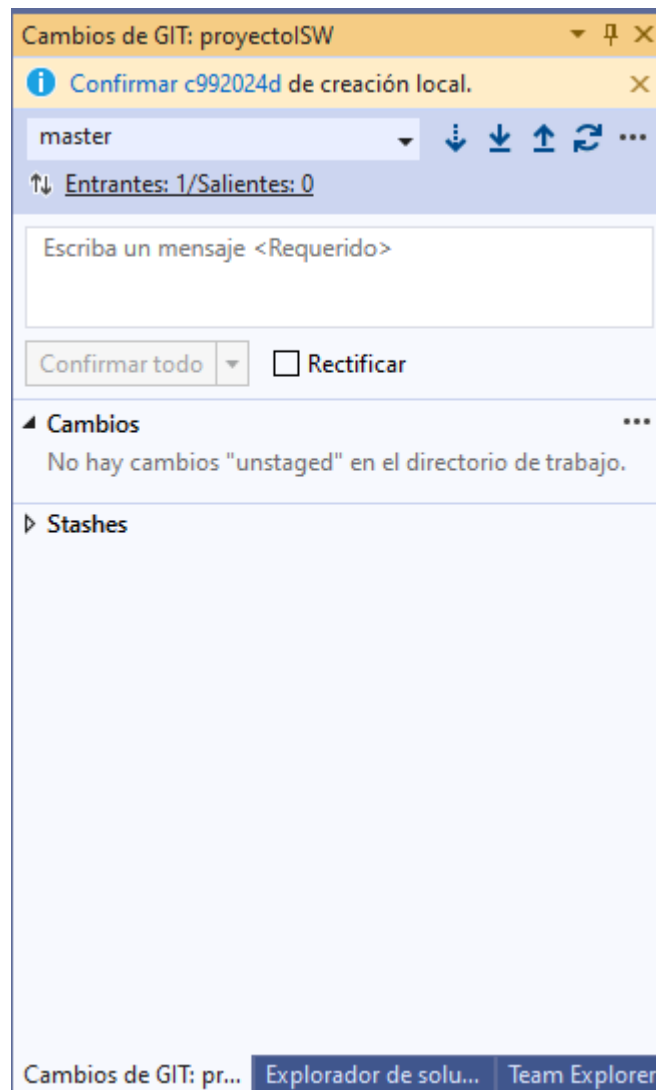


Figura 15 Finestra després de polsar Confirmar todo

Polse ara damunt Sincronizar. El resultat de l'operació es mostra en la següent figura.

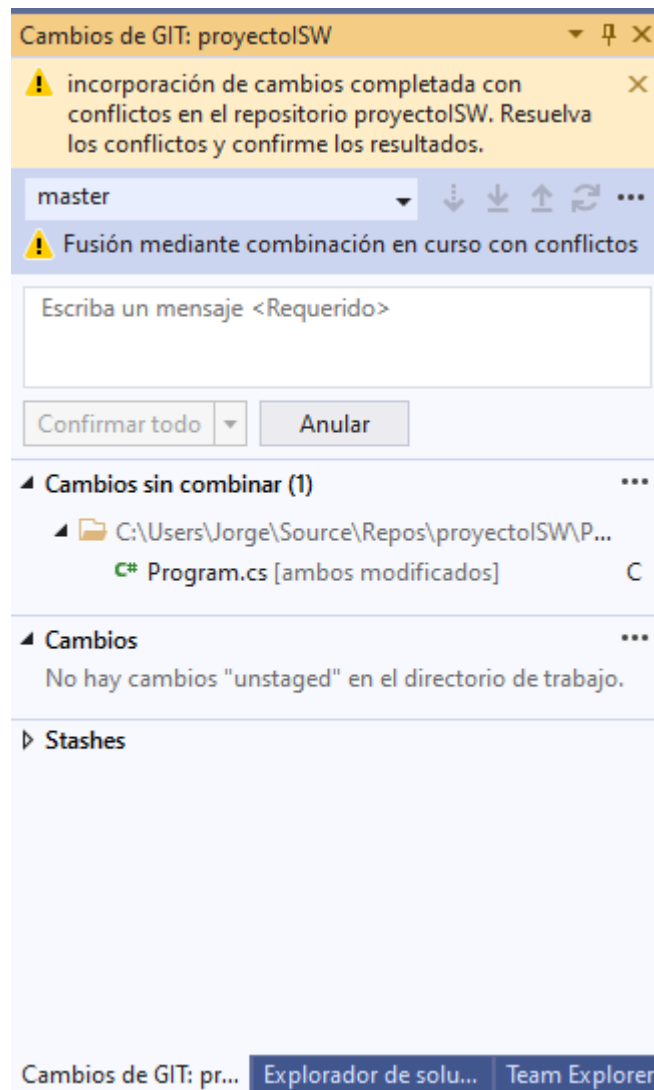


Figura 16 Conflictos al realizar push/pull (Sincronizar)

- c) Resolució de conflictes: és possible que, si un altre usuari va generar una nova versió abans que vostè o bé després i vostè no va descarregar l'última versió existent en el repositori, es produïsquen conflictes que calga resoldre. La resolució de conflictes es pot fer de forma manual decidint què codi conservar en la nova versió a generar mitjançant l'eina de resolució de conflictes (veure Figura 17). Prema en Conflictos de la figura anterior.

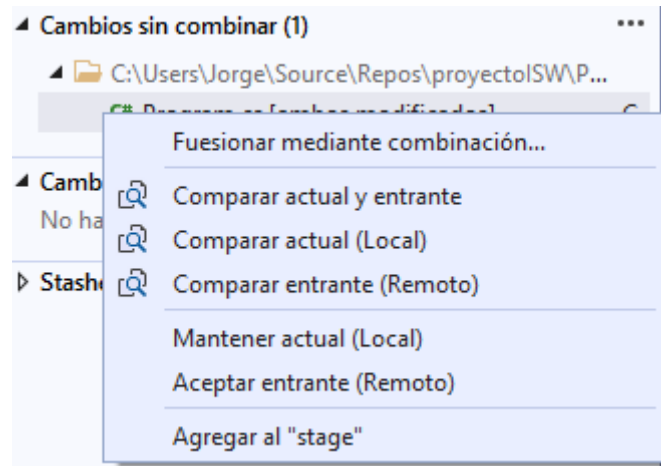


Figura 17 Resolució de conflictes

- d) Pulsant en Comparar archivos de la finestra anterior podem veure les diferències entre el arxiu remot (servidor) y el arxiu local.

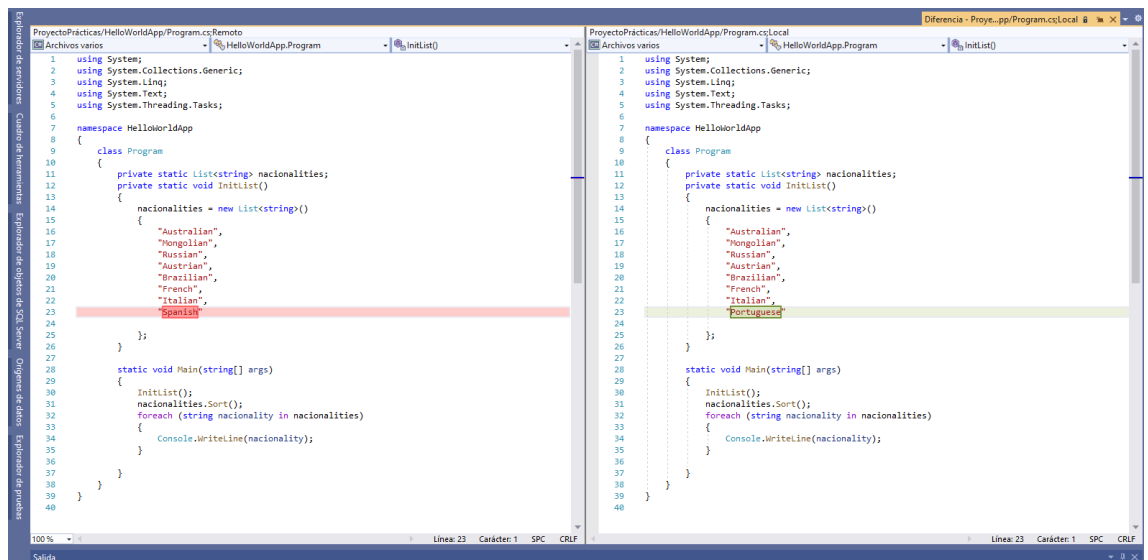


Figura 18 Finestra de resolució de conflictes

- e) Els conflictes sorgeixen per la diferència de contingut en la línia remarcada del codi. S'ha de decidir quina versió és la correcta. Com es pot observar en la Figura 18 tenim una versió amb un element en la llista ("Italiano"), i una altra versió amb un element ("French"). Seleccionem com a correcta una de les versions i finalitzem l'operació. En fer açò incorporarem els canvis al repositori central generant una nova versió del projecte. També podem seleccionar Fusionar mediante combinación, en aquest cas ens apareix una nova finestra com la de la figura següent.



Figura 19. Resolució de conflictes en les versions de codi

- f) Hem marcat en la figura precedent que desitgem mantenir el canvi “French” i el canvi “Italiano” en la llista de països. El resultat del fitxer combinat apareix en la part inferior de la figura anterior. Per a finalitzar premem sobre Aceptar fusión mediante combinación.
- g) Pot vore l’historial de canvis de qualsevol element de seu projecte mitjançant el “Explorador de Versiones de código”. Vaja a Explorador de soluciones -> Seleccione en el explorador el elemento a explorar (per exemple el fitxer Program.cs) -> Botón derecho de ratón -> Ver historial
- h) Anàlogament es poden observar els diferents conjunts de canvis generats en la Web associada al seu projecte de Azure DevOps. Per açò: Seleccione l’opció Repos -> Commits. Seleccionant un commit, es mostra quins fitxer han segut modificats i quins són els canvis.

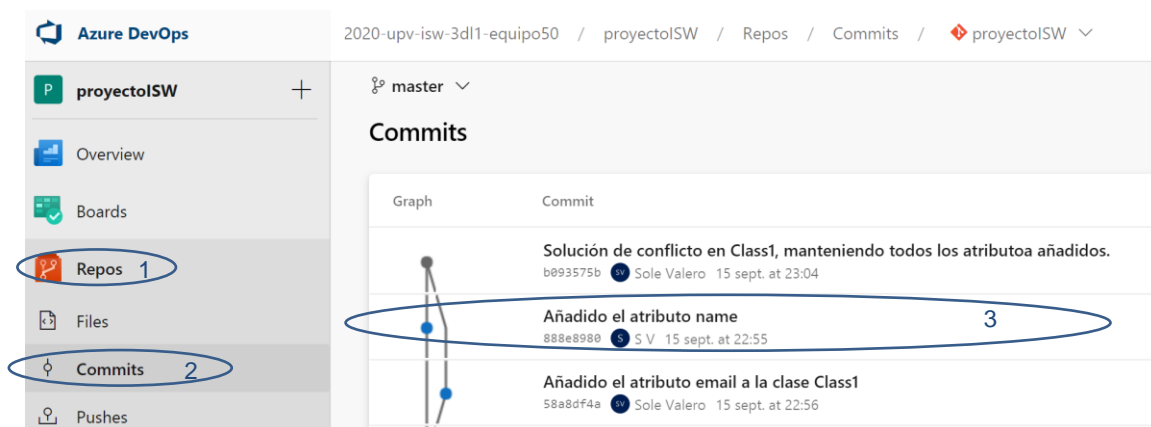


Figura 20 Visualització de la historia de canvis en el servidor

6. (Treball Opcional) Operacions Avançades amb l'Explorador de control de versions

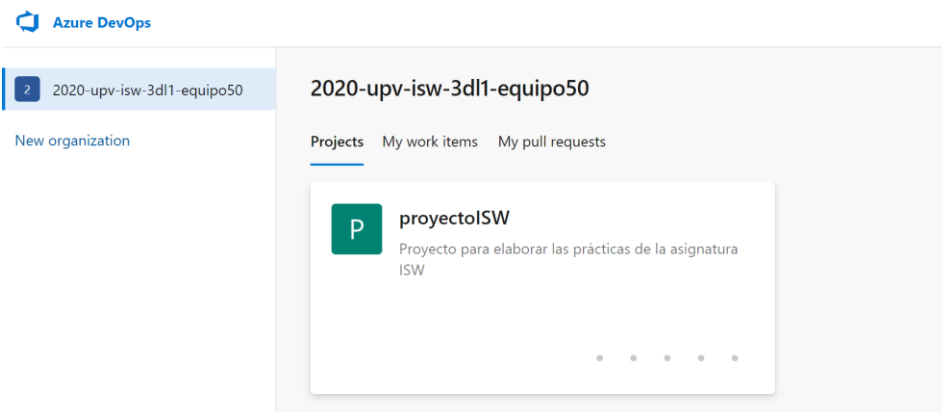
En aquesta secció practicarem dos conceptes addicionals relatius al control de versions:

- p. **branca (branch)**: és una còpia del projecte, sota el control de versions, però aïllat, de manera que els canvis realitzats en una branca no afecten a la resta del projecte i viceversa. Les branques també són conegudes com "Línies de desenvolupament".
- q. **merge**: moure un canvi d'una branca a una altra.

6.1 Generant una línia de desenvolupament (branca)

Per a generar una línia de desenvolupament pròpia de manera que independitze els canvis realitzats per vostè dels realitzats per uns altres es pot procedir de la següent manera:

- a. El *Team Master* ha de concedir permisos de creació de branques als usuaris del seu equip que estiguen autoritzats. Per a açò, en la Web de gestió del projecte de *Azure DevOps* anirà al projecte actual, preme sobre el símbol "P":



I a continuació en "Project settings" -> Repos -> Repositories per afegir permisos específics d'accés per a un membre de l'equip. Per defecte, els 5 primers membres convidats a una organització formen part de l'equip de desenvolupament que crea Azure DevOps per defecte, a la vegada que formen part del grupo "Contributors". Tots els membres d'aquest grup poden crear rames noves (vore Figura 21).

Si necessita afegir permisos de forma individual, també es possible, encara que es recomana la gestió de permisos gastant grups predefinits. Per a esta acció, primer hem de seleccionar l'usuari indicant el seu compte en el camp d'edició "Search for users or groups" de la Figura 21. Per exemple, s'ha seleccionat al usuari svalero1819@outlook.es que ja s'ha afegit amb anterioritat a l'equip de desenvolupament del projecte. Els permisos poden canviar-se prement en el panell dret. Tinga en compte que alguns s'hereten i no fa falta modificar-los. Concedisca permisos de gestió de branques als usuaris que desitge (veure 22)

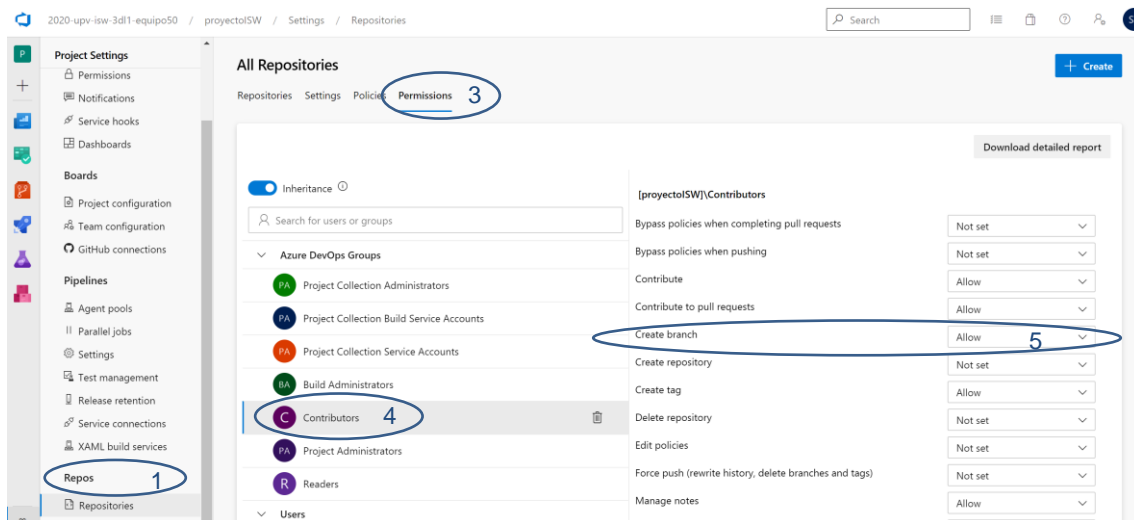


Figura 21. Concessió de permisos de gestió de branques a un grup d'usuaris

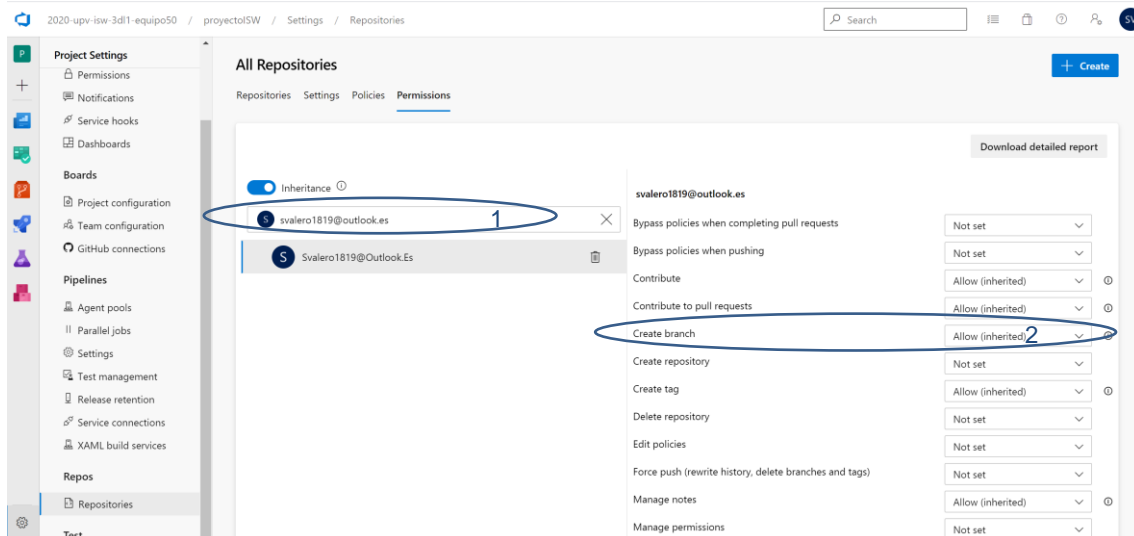


Figura 22. Concessió de permisos en el repositori a un usuari

- b. Una vegada concedits aquests permisos, l'usuari en qüestió pot crear branques. En *Visual Studio* ha de seleccionar primer la carpeta del projecte (dins de la solució) a partir de la qual es genera la ramificació i prémer en el menú emergent Nueva rama.

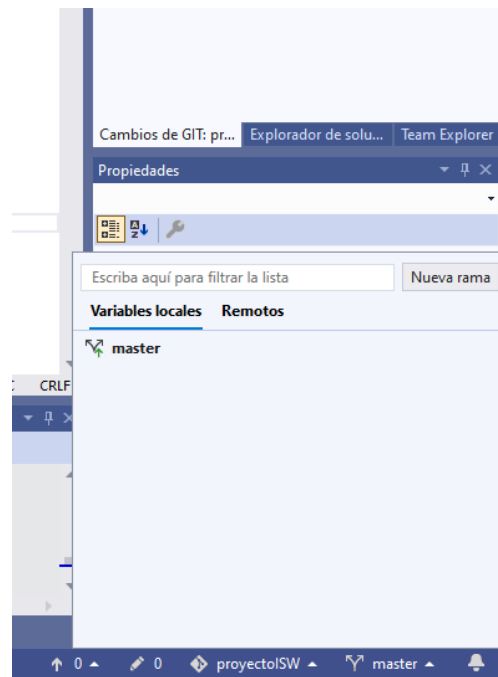


Figura 23. Obrint l'explorador de control de codi.

- c. En el nostre cas generarem la nova branca prenent com a versió de codi la que tenim en la nostra àrea de treball local (veure 24)

NOTA: En aquesta pràctica cada membre de l'equip pot crear una branca diferent amb noms "...-Feature2", "...-Feature3", etc.

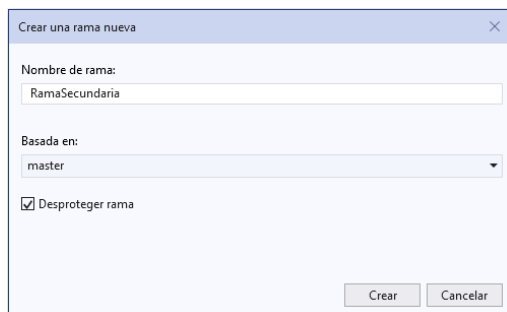


Figura 24. Creació de branca a partir de la versió de codi en àrea de treball local

Una vegada introduïda la informació, el nom de la nova rama, premeu el botó Crear

Després d'açò s'obté una nova ramificació (Figura 25), que es converteix en la rama actual de treball. Les dues branques, la principal (master) i la recentment creada estan en el repositori local mentre que la principal està en el remot.

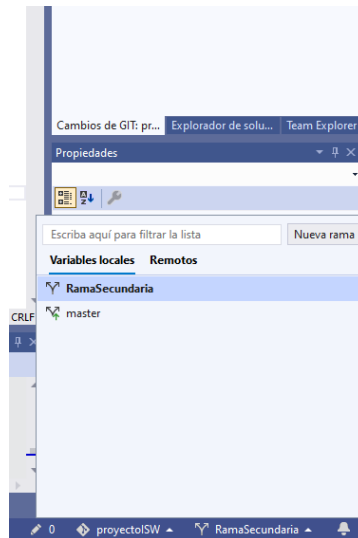


Figura 25. Estat actual de les branques

6.2 Actualitzant una branca

Com ja s'ha vist anteriorment, en qualsevol moment es pot crear una branca local associada a una branca principal, de manera que es genere un conjunt de canvis i que es puguin pujar les noves versions al control de versions, generant una nova versió del producte.

- Modifique de nou la classe *Program* perquè l'algun element de la llista siga un altre (cada membre de l'equip pot triar una vegada més una cadena diferent per als elements de la llista).
- En aquest punt anem a actualitzar els canvis en la nova branca local (veure Figura 26).

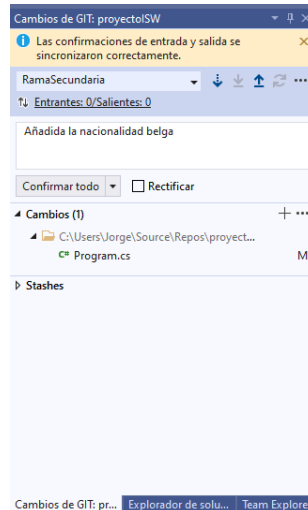


Figura 26. Canvis de la solució a partir de una nova branca

- Polse en **Confirmar todo** de la figura anterior, en la nova finestra polse **Sincronizar** (Figura 27). Perquè els canvis es pugen al servidor premeu en **Insertar**. Amb açò es pugen els canvis de la nova branca al servidor.

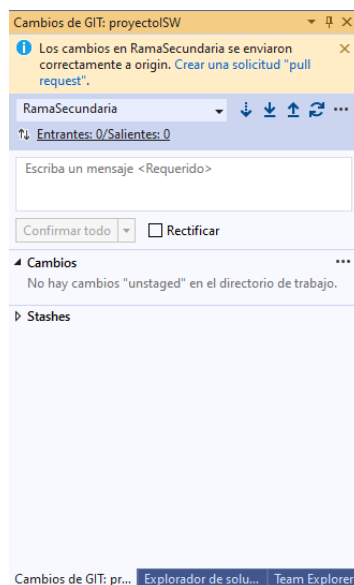


Figura 27 Pas de sincronització de canvis al servidor

- d. Pot observar les dues branques existents en el servidor a través de l'aplicació Web de *Azure DevOps* i comparar les dues versions de la classe *Program* en les dues branques creades.

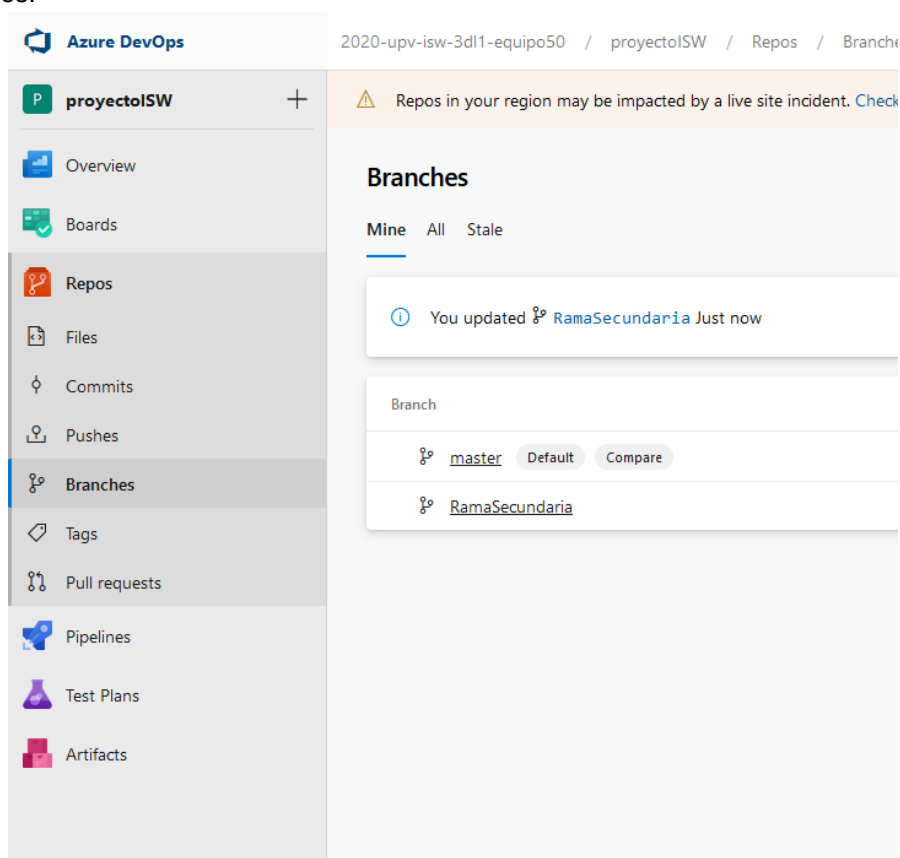


Figura 23 Branques en el servidor: la principal i la creada en local

- e. Es pot observar les branques existents en aquest moment seleccionant en Visual Studio -> Ramas Seleccione del repositori local la branca acabada de crear i amb el menú contextual utilitze Ver historial

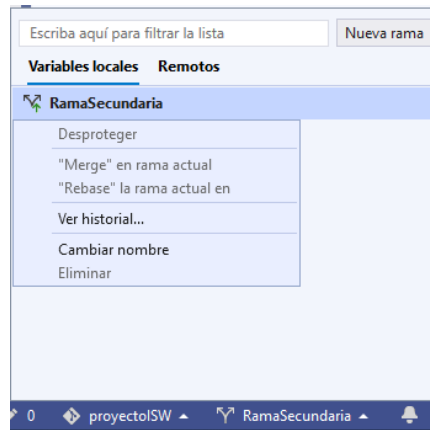


Figura 24. Explorant les branques

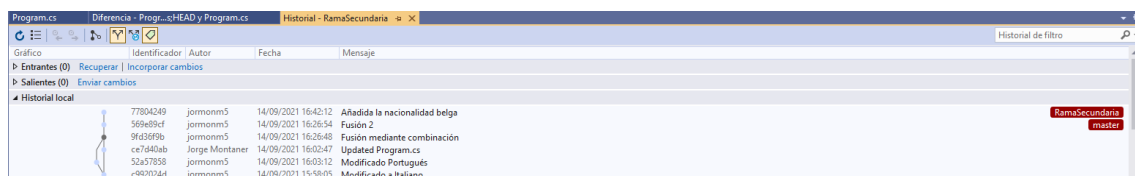


Figura 25. Historial de canvis

Nota: La captura de l'historial mostra alguns canvis realitzats en el projecte i no descrits en aquest document.

6.3 Combinació de branques (merging)

És possible combinar dues branques de manera que els canvis existents en les mateixes es combinen. Açò és útil per exemple per a incorporar els canvis d'una branca de desenvolupament a una branca de *release* o per a combinar dues branques de desenvolupament en una. En aquest cas anem a combinar (fusionar) la branca principal (master) amb la branca Feature1. Per a açò:

- Seleccionarem la branca d'origen a combinar en el Team Explorer de *Visual Studio*, botó dret de ratolí i a continuació l'opció de menú "*Merge*" en *La rama actual*

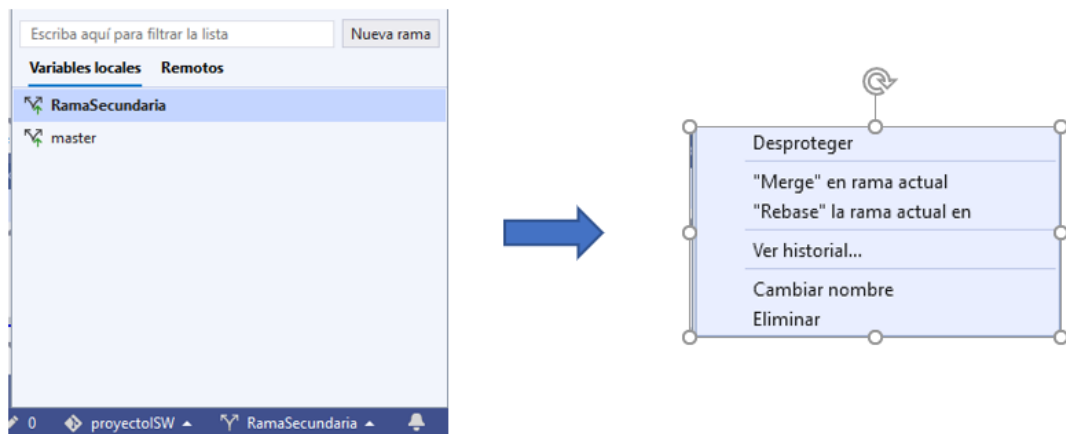


Figura 31. Combinació de dos branques

- b. PULSE en el botó Sí. En realitzar aquesta operació poden aparèixer conflictes. El conflicte es podrà resoldre automàticament o mitjançant l'eina de resolució de conflictes com vam veure anteriorment.
- c. Recordar que es deuen confirmar els canvis i sincronitzar el nou commit que han sigut creats, per a que la fusió de les rames quede reflectida també en el repositori remot.
- d. Es pot comprovar mitjançant l'aplicació Web que en el repositori central ambdues branques contenen ara la mateixa versió de la classe *Program*.

Per a més informació sobre el model de control de versions en forma de branques en Team Services pot consultar la informació disponible en:

<https://www.visualstudio.com/nl-nl/docs/tfvc/use-branches-isolate-risk-team-foundation-version-control>

<https://docs.microsoft.com/en-us/azure/devops/repos/git/git-branching-guidance?view=azure-devops>

NOTA: L'equip de desenvolupament haurà de decidir quantes branques tenir, només una principal, una de desenvolupament i una altra principal, etc. Es recomana llegir el següent document <https://www.visualstudio.com/nl-nl/docs/tfvc/branch-strategically> per a decidir una adequada estratègia de ramificació.