UNIVERSIDAD
POLITECNICA
DE VALENCIA

Laboratory

Lab Session 1

Introduction to Visual Studio and Azure DevOps

Software Engineering

ETS Computer Science
DSIC – UPV

Year 2024-2025

1. Goal

The goal of this lab session is to introduce the student to the main functionalities of *Visual Studio 2022* and its relationship with the team project management and control version tool Azure DevOps. It is mandatory to complete the tasks described until section 5 (included). Section 6 is optional.

2. Creating a team project using Azure DevOps and an initial solution in the server

In order to correctly fulfil this lab session, you should follow the steps to create a team project with Azure DevOps in the cloud, register your team members in that project, and create a solution inside it. It is mandatory to name the Azure DevOps organization using the following schema: 2024-upv-isw-3EL1-<team name>. For instance, the server could reside at **dev.azure.com/2024-upv-isw-3EL1-team03**.

These preliminary tasks will be performed by only one member of the team, the responsible or Team Master.

The realization of the activities stated above has been described in the theory seminars of chapters 3 about Visual Studio and Azure DevOps. You must finish them before continuing to the next steps (consult the existing material in *PoliformaT* related to these theory seminars). At the completion of these activities, the development team will have a team project with a *Visual Studio* solution containing the *Presentation*, *Library* (with the *BusinessLogic* and *Persistence* subfolders) and *Testing* solution folders, as shown in Figure 1.

Please, remember that all newly created folders must have at least one child element inside them for Git to correctly synchronize them with the remote repository. When a local folder is empty, Git

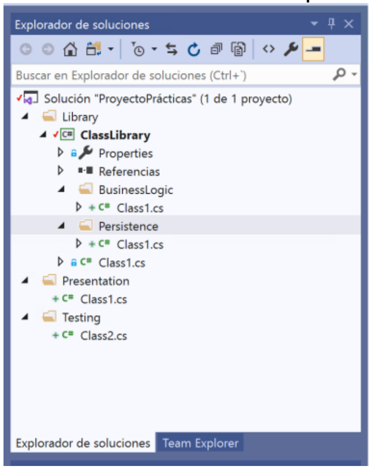

```
Explorador de soluciones
Buscar en Explorador de soluciones (Ctrl+')
Solución "ProyectoPrácticas" (1 de 1 proyecto)
  Library
    ClassLibrary
      Properties
      Referencias
      BusinessLogic
        Class1.cs
      Persistence
        Class1.cs
      Class1.cs
  Presentation
    Class1.cs
  Testing
    Class2.cs
Explorador de soluciones   Team Explorer
```

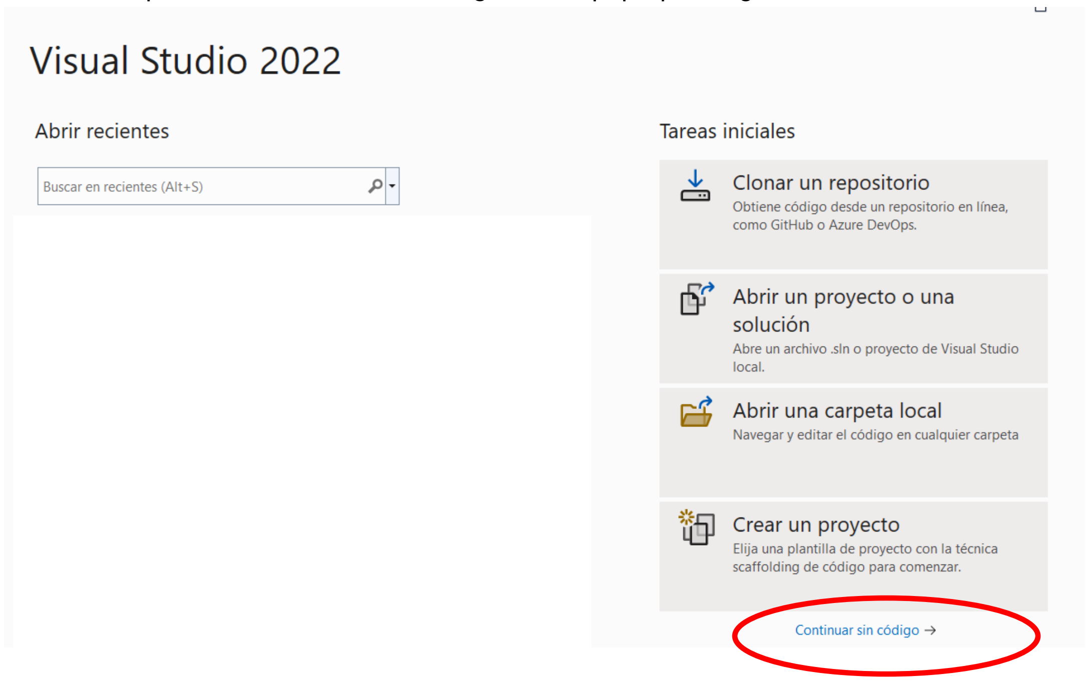
Figure 1. Initial Solution folders and modelling project

does not correctly store it, so the next time the repository is cloned on your machine, it will not be possible to add new classes or content to those empty folders and it will be necessary to remove and create them again.

3. Connection to the existing project from Visual Studio

At this moment, each member of the team can connect **individually** from *Visual Studio* to the *Azure DevOps* project, download a copy of the solution from the cloud server and assign it to a local workspace (local folder in C:\[1]) to work with it. To do so, follow the next steps (you can do it **individually**):

a. Start the development tool *Microsoft Visual Studio*.
b. Select the option `Continuar sin código` in the pop-up dialog:



Visual Studio 2022

Abrir recientes

Buscar en recientes (Alt+S)

Tareas iniciales

Clonar un repositorio
Obtiene código desde un repositorio en línea, como GitHub o Azure DevOps.

Abrir un proyecto o una solución
Abre un archivo .sln o proyecto de Visual Studio local.

Abrir una carpeta local
Navegar y editar el código en cualquier carpeta

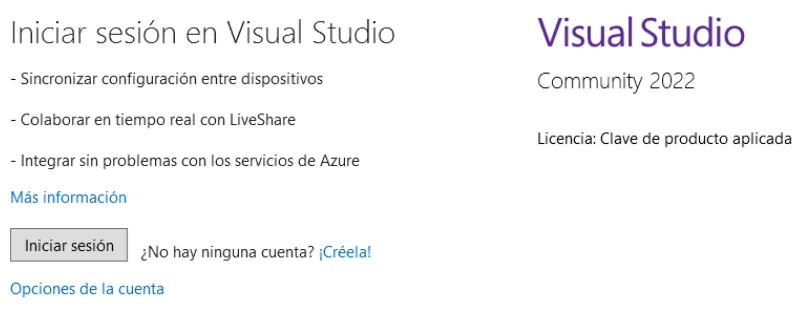
Crear un proyecto
Elija una plantilla de proyecto con la técnica scaffolding de código para comenzar.

Continuar sin código →

Figure 2: initial window

c. Login with your Microsoft user account:
   - *Archivo->Configuración de la cuenta->Iniciar Sesión*

[1] In DSIC laboratories and accesses to the virtual environment, using W personal unit is not recommended, because is a network resource and the connection to it can be lost in the middle of the work session, causing errors in Visual Studio.



Iniciar sesión en Visual Studio

- Sincronizar configuración entre dispositivos
- Colaborar en tiempo real con LiveShare
- Integrar sin problemas con los servicios de Azure

Más información

Iniciar sesión ¿No hay ninguna cuenta? ¡Créela!

Opciones de la cuenta

Visual Studio

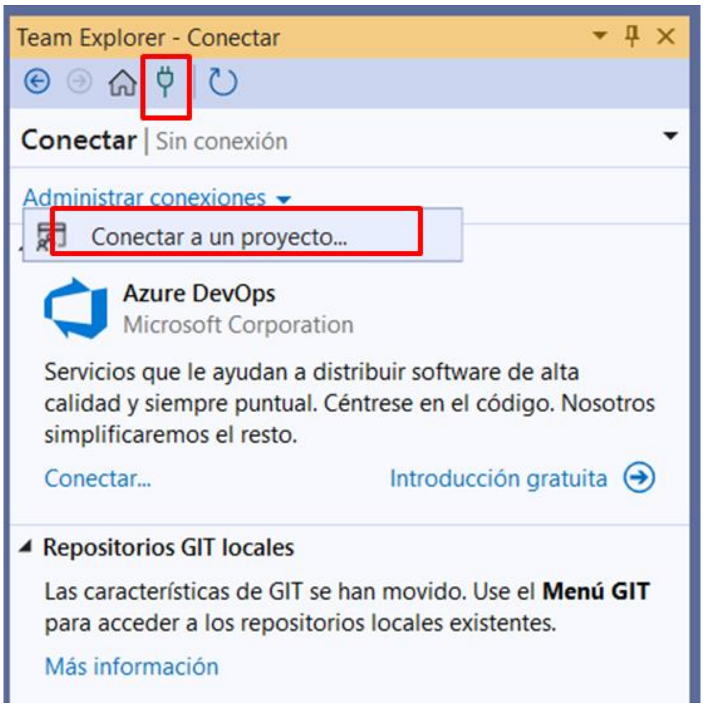
Community 2022

Licencia: Clave de producto aplicada

Figure 3: log-in in Visual Studio

d. Connect to your team project:

- Choose Team Explorer (right frame) and then click on the *Team Services connection button* (green plug, Figure 4).



Team Explorer - Conectar

Conectar | Sin conexión

Administrar conexiones

Conectar a un proyecto...

Azure DevOps

Microsoft Corporation

Servicios que le ayudan a distribuir software de alta calidad y siempre puntual. Céntrese en el código. Nosotros simplificaremos el resto.

Conectar... Introducción gratuita

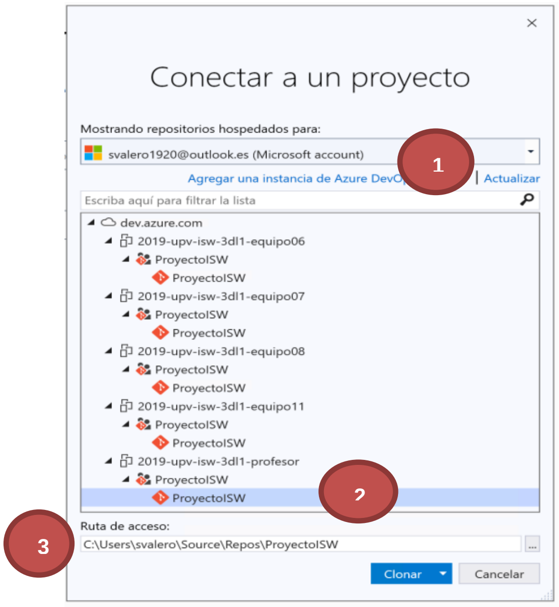
Repositorios GIT locales

Las características de GIT se han movido. Use el **Menú GIT** para acceder a los repositorios locales existentes.

Más información

Figure 4: connect to a project from "Administrar conexiones"

- In the dialog that appears (see Figure 5), you can see the Microsoft account used to login (Figure 5-1). Under this drop-down list, the list of organizations to which you belong appears (in your case, it is normal that only one appears). Also, for each organization, the projects you have appear (again, only one in your case). Selecting the git symbol of your project (Figure 5-2), confirm the route in which you want to save the project (Figure 5-3 and click on clone (Figure 5-4). You may have to confirm the account you are going to use to connect to the remote repository.



Conectar a un proyecto

Mostrando repositorios hospedados para:

svalero1920@outlook.es (Microsoft account)

Agregar una instancia de Azure DevO | Actualizar

Escriba aquí para filtrar la lista

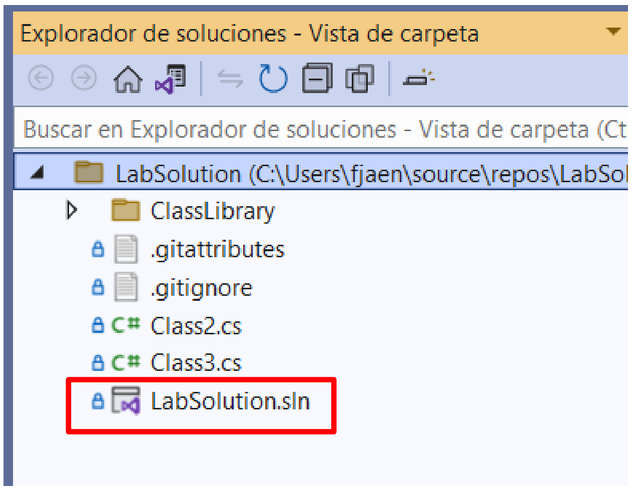
```
dev.azure.com
  2019-upv-isw-3dl1-equipo06
    ProyectoISW
      ProyectoISW
  2019-upv-isw-3dl1-equipo07
    ProyectoISW
      ProyectoISW
  2019-upv-isw-3dl1-equipo08
    ProyectoISW
      ProyectoISW
  2019-upv-isw-3dl1-equipo11
    ProyectoISW
      ProyectoISW
  2019-upv-isw-3dl1-profesor
    ProyectoISW
      ProyectoISW
```

Ruta de acceso:

C:\Users\svalero\Source\Repos\ProyectoISW

Figure 5: connect to an organization's project

e. Open the solution created by the Team Master. Once connected to the team project and the local repository has been obtained and assigned, you can open the solution that the Team Master created previously (step 2 in this guide). From the "Solutions Explorer" (*"Explorador de soluciones"* tab on the right-hand side of Visual Studio), double-click on the file with the .sln extension if you are in the folder view mode (Figure 6).



Explorador de soluciones - Vista de carpeta

Buscar en Explorador de soluciones - Vista de carpeta (Ct

```
LabSolution (C:\Users\fjaen\source\repos\LabSol
  ClassLibrary
  .gitattributes
  .gitignore
  Class2.cs
  Class3.cs
  LabSolution.sln
```

Figure 6: open a solution from the explorer

f. Check that you have a solution as the one the *Team Master* has originally created and stored in the repository, as shown in Figure 1.

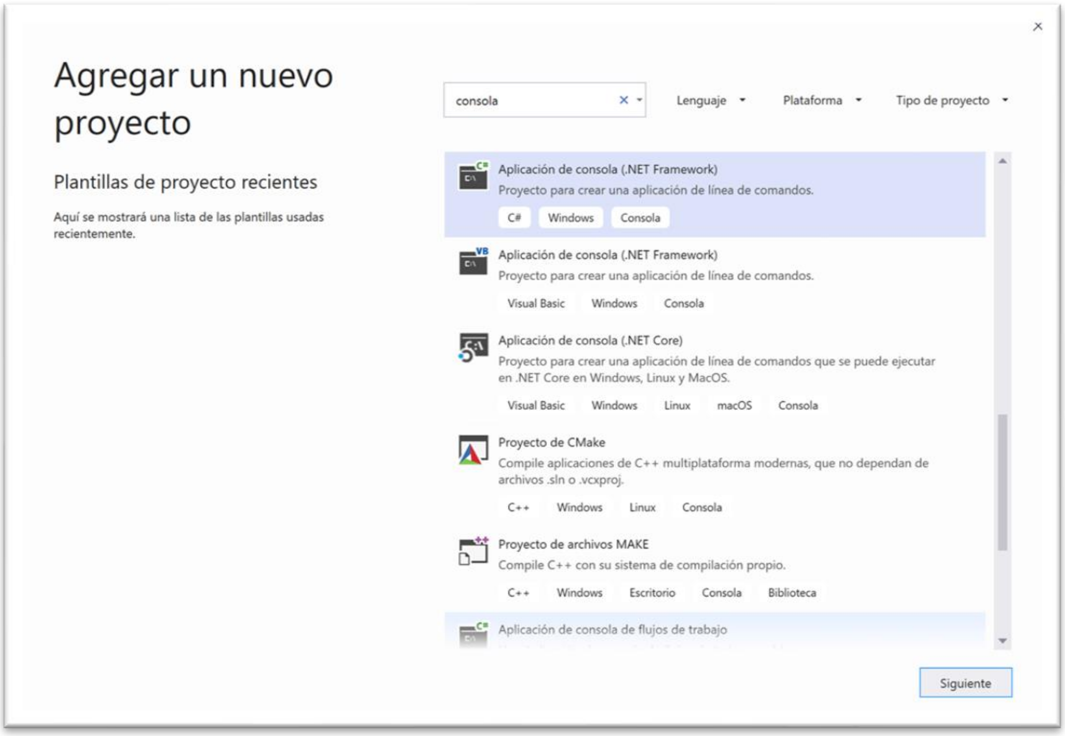
4. Introduction to the Visual Studio debugging tools

The Visual Studio debugging tools are a key element for the development of applications in Visual Studio and the detection of bugs in code. These tools allow to set break points in your code, run the code step by step, inspect the values of variables during the execution, etc.

4.1. Creating a console project

In order to get started with debugging tasks, we are going to **create a console project in the solution folder *Testing***:

a. Create a subfolder named *Lab1* inside the *Testing* folder.
b. Right click on the folder *Lab1*, and then select *Agregar -> Nuevo Proyecto-> Aplicación de consola (.NET Framework)* and name the application ***HelloWorldApp*** (select the console project type shown in Figure 7)**.**



Agregar un nuevo proyecto

Plantillas de proyecto recientes

Aquí se mostrará una lista de las plantillas usadas recientemente.

consola

Aplicación de consola (.NET Framework)
Proyecto para crear una aplicación de línea de comandos.
C# Windows Consola

Aplicación de consola (.NET Framework)
Proyecto para crear una aplicación de línea de comandos.
Visual Basic Windows Consola

Aplicación de consola (.NET Core)
Proyecto para crear una aplicación de línea de comandos que se puede ejecutar en .NET Core en Windows, Linux y MacOS.
Visual Basic Windows Linux macOS Consola

Proyecto de CMake
Compile aplicaciones de C++ multiplataforma modernas, que no dependan de archivos .sln o .vcxproj.
C++ Windows Linux Consola

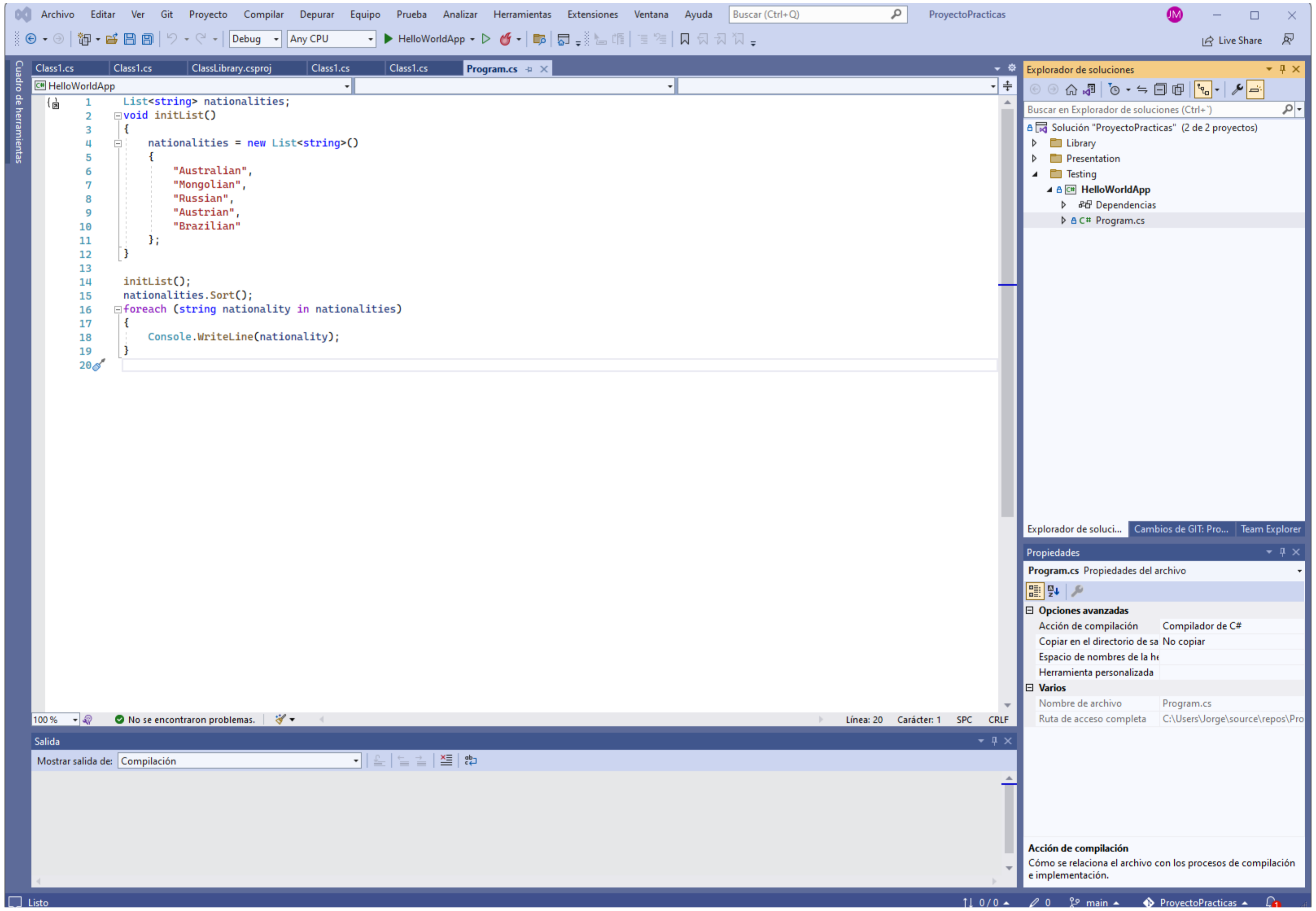
Proyecto de archivos MAKE
Compile C++ con su sistema de compilación propio.
C++ Windows Escritorio Consola Biblioteca

Aplicación de consola de flujos de trabajo

Figure 7 Creation of HelloWorldApp

c. Once the application has been generated, observe the different windows that Visual Studio provides (see **¡Error! No se encuentra el origen de la referencia.**): *Cuadro de Herramientas* (tool box, left panel); *Explorador de Soluciones* (solution explorer, upper-right panel); *Propiedades* (properties, lower-right panel); *Editor de Código* (code editor, upper-centre panels); *lista de errores y salida* (error lists and output, lower-centre panels).
d. The *HelloWorldApp* project should be marked as the start-up project (being highlighted in bold). Otherwise it should be marked as such before compiling and executing it. To

mark the project as the start-up one, click the mouse right button on the *HelloWorldApp* project and select the option ***Establecer como proyecto de Inicio.*** Observe that the name of the project is in bold now.



```
List<string> nationalities;
void initList()
{
    nationalities = new List<string>()
    {
        "Australian",
        "Mongolian",
        "Russian",
        "Austrian",
        "Brazilian"
    };
}

initList();
nationalities.Sort();
foreach (string nationality in nationalities)
{
    Console.WriteLine(nationality);
}
```

No se encontraron problemas.

Línea: 20 Carácter: 1 SPC CRLF

Program.cs Propiedades del archivo

Opciones avanzadas	
Acción de compilación	Compilador de C#
Copiar en el directorio de sa	No copiar
Espacio de nombres de la he	
Herramienta personalizada	
Varios	
Nombre de archivo	Program.cs
Ruta de acceso completa	C:\Users\Jorge\source\repos\Pro

Acción de compilación
Cómo se relaciona el archivo con los procesos de compilación e implementación.

Figure 8: working area in Visual Studio

4.2. Compile and run

To compile and run this application, click on the green triangle button *Iniciar* on the upper area of the development tool (▶ Iniciar ▾). The output window will show the result of the compilation and the executing, although the current program will not perform any action.

4.3. Exploring compilation errors

Open the file *Program.cs* and type an error artificially in the main method, for example, by writing the sentence *`fadsfsde`*. If you compile again the program you will observe the list of errors in the corresponding window. If you click on the error, the cursor will appear where it is in your code.

4.4. Creating a HelloWorld application in C#

For practicing the debugging abilities, we are going to create a simple program as an example. Open the file *Program.cs* and modify its contents to make it look as follows:

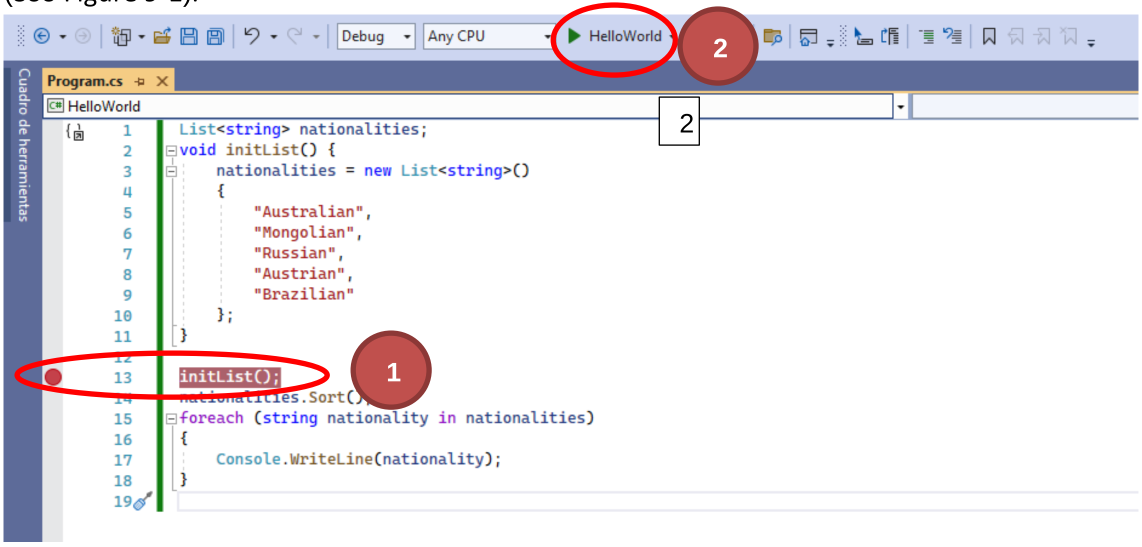
```
List<string> nationalities;
void initList() {
    nationalities = new List<string>()
    {
        "Australian",
        "Mongolian",
        "Russian",
        "Austrian",
        "Brazilian"
    };
}

initList();
nationalities.Sort();
foreach (string nationality in nationalities)
{
    Console.WriteLine(nationality);

}
```

The previous program takes advantage of the new .NET 6.0 features, which removes the *using* sentences, which act as compiler directives, as it uses them implicitly at compilation time. These directives make possible to use types defined in the imported namespace without requiring us to specify them again explicitly in the code. For this class, the namespace is also implicit and is set to the global namespace. Also, the definition of the class *Program* is obviated, but the code above belongs to that class, which contains the definition of a list, its initialization function *initList* and the rest of the code of the application. The program creates a list of *strings*, orders it alphabetically and then goes through this list and shows its contents on the console. It is important to note that, until the previous version of .NET, the code above would be inside a static *Main* method, with the *initList* method being outside that Main method but within the class. From .NET 6.0, the code of in the file Program.cs is assumed to be inside a *Main* method, so it executes directly. It also allows the creation of functions (such as initList) in it. This code will be used to practice our debugging abilities. Compile and run the program. You will see a console window in which the program executes. It is possible that you do not see the output. That is because the execution finishes so quickly that it does not give you time to see it.

4.5. Run a program step by step

Most IDEs currently allow including break points in which the execution is stopped in order to run the code sentence by sentence from that point and observe its behavior in a controlled manner. In order to practice this aspect, we will perform the following actions:

a. Insert a break point in the code line *initList(),* clicking on the grey column which appears left to the code, at the same level of the line in which you would like to stop the execution (see Figure 9-1).



```
List<string> nationalities;
void initList() {
    nationalities = new List<string>()
    {
        "Australian",
        "Mongolian",
        "Russian",
        "Austrian",
        "Brazilian"
    };
}

initList();
nationalities.Sort();
foreach (string nationality in nationalities)
{
    Console.WriteLine(nationality);
}
```

Figure 9: adding a breakpoint

b. Compile and start debugging the code (click on the green triangle, Figure 9-2) and observe how the execution is stopped at the line in which you previously set the red point (a yellow arrow appears indicating where the execution is). If you observe the console window created, this would be empty.

c. Run the application step by step. Use the step-by-step execution controls (Figure 10). You can stop the execution (red square), run the next instruction (Figure 10-1 or F11), run a whole procedure/method completely without stepping into it (Figure 10-2 or F10), or run all the sentences in a procedure to go out of Figure 10-3 or Shift+F11).

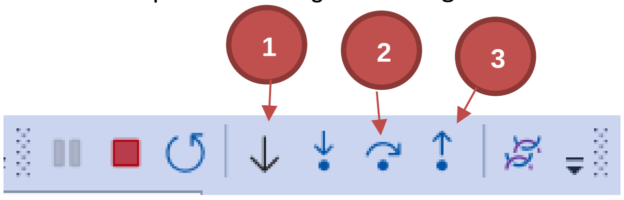
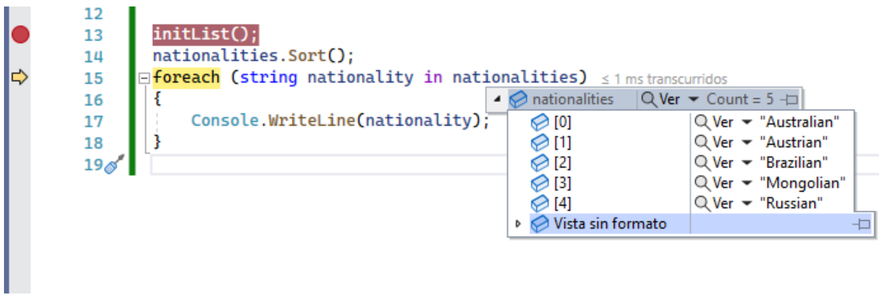


Figure 10: step by step execution

4.6. Observing the data values of the program

While the program is being executed, it is possible to observe the values of objects, attributes, variables, etc. This is **extremely useful** when debugging a program. To observe the value of any element, place the cursor over it.

a. When your program arrives to the execution of the sentence *nationalities.Sort(),* place the cursor over the variable *nationalities* and observe its content (Figure 11).

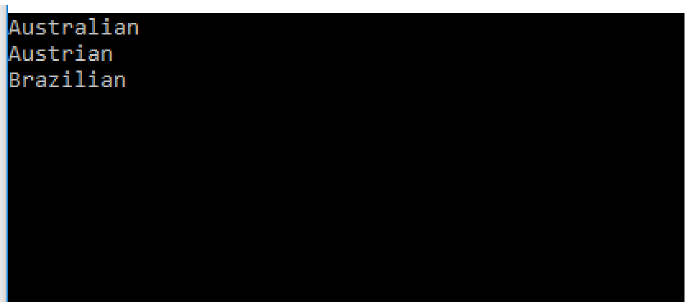


```
initList();
nationalities.Sort();
foreach (string nationality in nationalities)  ≤ 1 ms transcurridos
{
    Console.WriteLine(nationality);
}
```

Figure 11. Inspecting data values of variables

b. Execute this sentence step by step (F11) and observe again the variable *nationalities*, which has been ordered alphabetically.

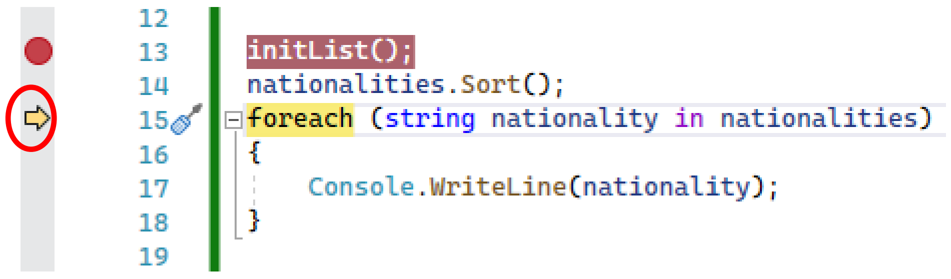
c. Keep running the program step by step and observe how the console window shows the content of the list (Figure 12).



```
Australian
Austrian
Brazilian
```

Figure 12. Console output

d. You can go back in the execution just by dragging the yellow arrow (Figure 13) up towards the line of code in which you would like to repeat the execution of your program.



```
initList();
nationalities.Sort();
foreach (string nationality in nationalities)
{
    Console.WriteLine(nationality);
}
```

Figure 13: debugging arrow

5. Version Control Basic Operations

An advantage of using a collaborative development environment like Azure DevOps is being able to control and manage the different versions of the project as it is being generated. In fact, when you created your project, you indicated you wanted to use **Git** as the version control system. A version control system is a combination of technologies and practices to control the changes made to the different elements that comprise a project, especially the source code, the documentation, the sets of tests, etc.

It is convenient to know some nomenclature associated to Git, as it will be the version control system we will use:

- **Remote repository**: a repository in which all the team members will store its changes.
- **Local repository**: an instance of the remote repository in which each member will work locally in its machine, and not always is synchronized with the remote one.
- **commit**: Stores the last changes made in your repository. It not only stores which files were changed and the lines affected, because also is linked who makes the changes, in which data and a user message to explaining the changes. This message is mandatory, and it is convenient also writing in it some references to the related tasks from the plan project.
- **clone**: Obtaining a copy of the project from the remote repository. A local repository is created, in which is possible to see all the commits made from the beginning of the project.
- **pull**: Retrieving the last commits stored in the remote repository to your local repository.
- **push**: upload the las commits stored in the local repository to the remote one.
- **merge**: it is the process in which the different commits (and their changes) are combined to obtain a new consistent version of the project.
- **conflict**: It happens when two or more people try to perform different changes in the same piece of code. Thus, when two different commits are combined in a merge process (for example during a pull) and it is detected that they contain changes in a same file, the system request the user to confirm the final change which will be stored. As a result of resolving a conflict, we will get a new commit and a coherent version of the project.

Git version control follows a workflow that most developers follow when they write code and share it with the development team. The steps are the following:

1. Obtain a local copy of the project code
2. Make changes to the code
3. Once the changes have been completed, publish the modified code to the rest of the team
4. Once accepted, merge with the code in the team repository.

Graphically, the previous workflow can be represented as Figure 14 shown:

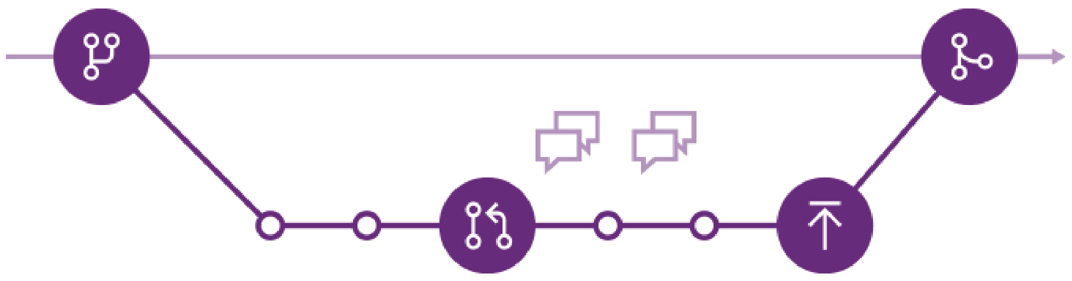


Figure 14. Git workflow

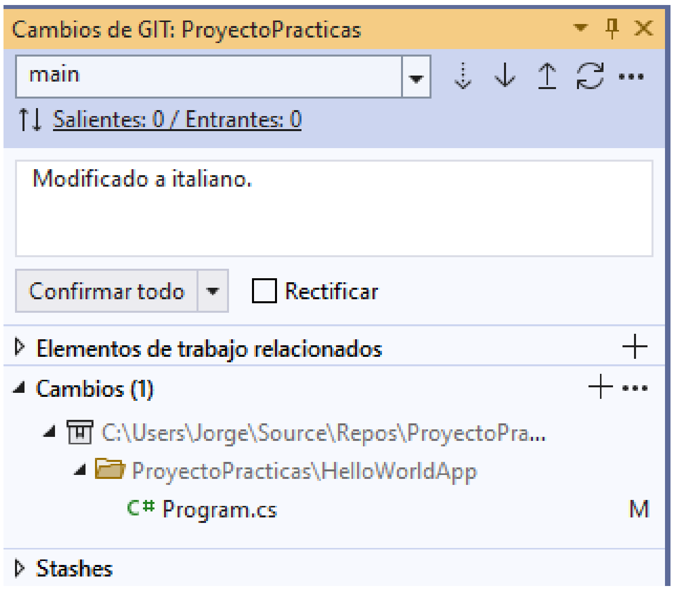
5.1. Committing changes

We are going to simulate the normal workflow in a developing team:

a) Modify any part of your code. For instance, the *Program* class so that the first element in the list is different. Each member of the team may use a different string (*"Italian", "spanish", etc.)*

b) Go to *Cambios de GIT,* and then add a descriptive comment for the new version.

c) Click on *Confirmar todo* button (Figure 15). A new commit with the last changes will be created.


Cambios de GIT: ProyectoPracticas
main
Salientes: 0 / Entrantes: 0
Confirmar todo
Rectificar
Elementos de trabajo relacionados
Cambios (1)
Stashes

Modificado a italiano.

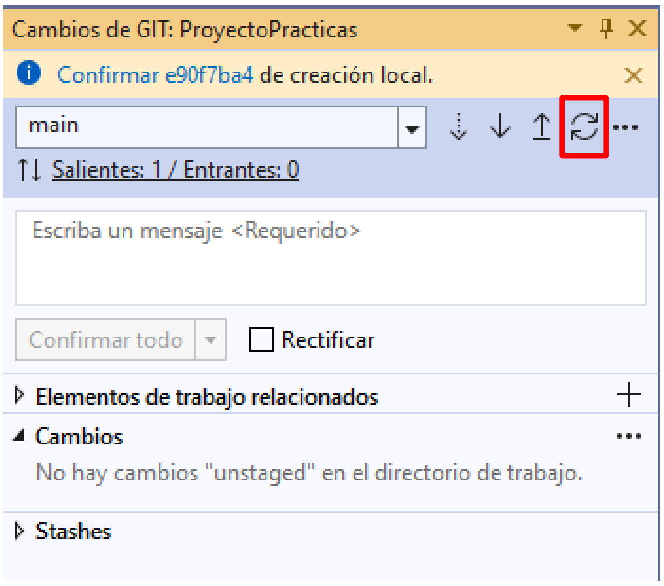
C:\Users\Jorge\Source\Repos\ProyectoPra...
ProyectoPracticas\HelloWorldApp
Program.cs M

Figure 15: window to commit changes

5.2. Pushing changes and resolving conflicts

After working locally, you will share your work with your development team, in this process conflicts can appear, and you will need to resolve them.

a) Select the option *Cambios de GIT.* In the new dialog, clicks on the link *Sincronizar*. All pending commits will be added to the remote repository.


Cambios de GIT: ProyectoPracticas

Confirmar e90f7ba4 de creación local.

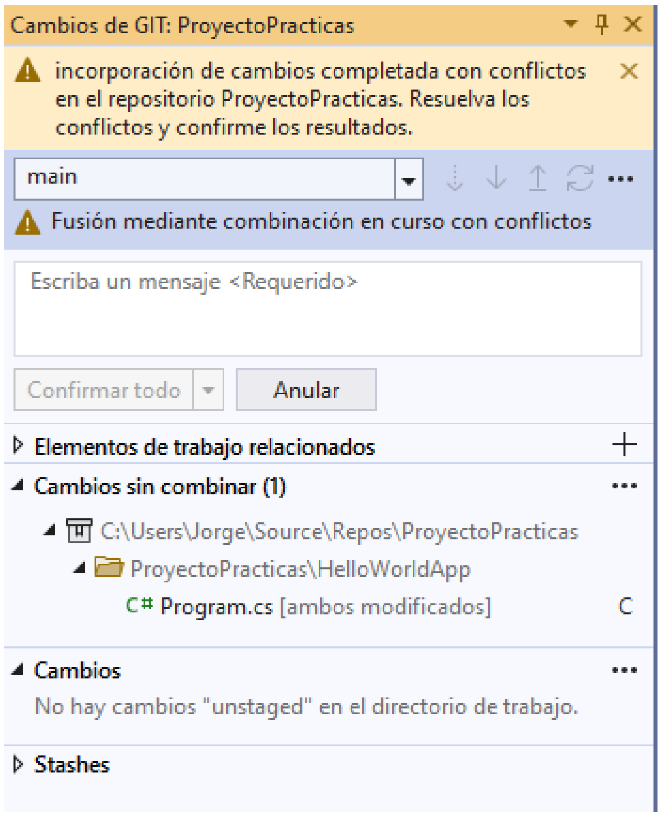
main

Salientes: 1 / Entrantes: 0

Escriba un mensaje <Requerido>

No hay cambios "unstaged" en el directorio de trabajo.

Figure 16: Synchronize button


Cambios de GIT: ProyectoPracticas

incorporación de cambios completada con conflictos en el repositorio ProyectoPracticas. Resuelva los conflictos y confirme los resultados.

main

Fusión mediante combinación en curso con conflictos

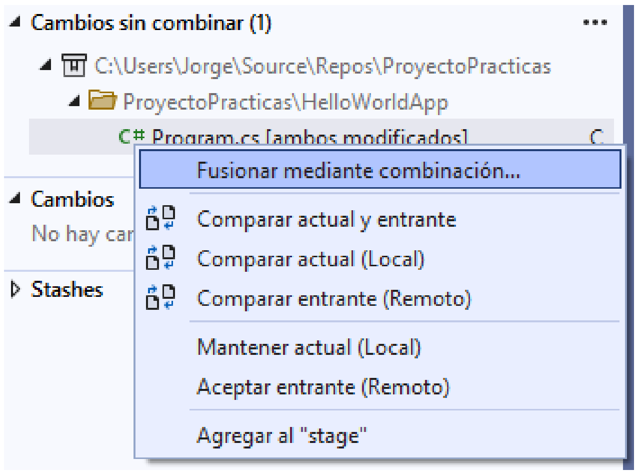
Escriba un mensaje <Requerido>

C:\Users\Jorge\Source\Repos\ProyectoPracticas
ProyectoPracticas\HelloWorldApp
C# Program.cs [ambos modificados] C

No hay cambios "unstaged" en el directorio de trabajo.

Figure 17: conflicts when trying to synchronize

b) **Resolving conflicts.** It is possible that conflicts arise if another user generated a new version before and you do not have the latest version stored in your local repository (see Figure 17 for an example). Conflict resolution may be done in a manual way, by using a tool for conflict resolution, to decide the code version to be kept when the solution is uploaded to the repository. In Figure 17, we can initiate the process clicking on the file with conflicts or by right-clicking on that file and selection either the local or the remote version.

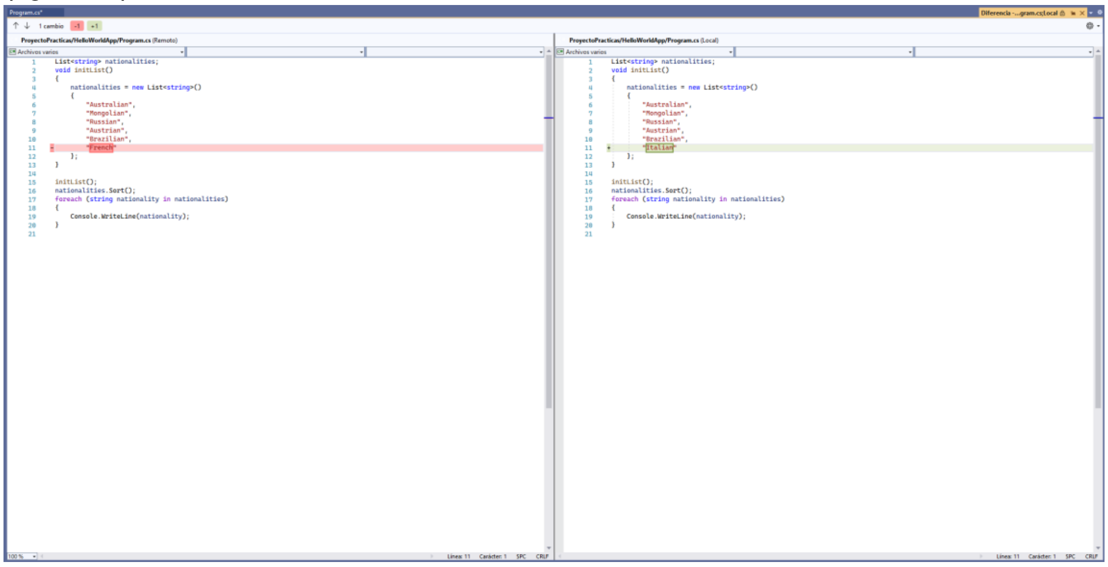


Cambios sin combinar (1)
C:\Users\Jorge\Source\Repos\ProyectoPracticas
ProyectoPracticas\HelloWorldApp
Program.cs [ambos modificados]
Cambios
No hay car
Stashes
Fusionar mediante combinación...
Comparar actual y entrante
Comparar actual (Local)
Comparar entrante (Remoto)
Mantener actual (Local)
Aceptar entrante (Remoto)
Agregar al "stage"

Figure 18: conflict resolution

c) In the dialog that appears when double-clicking, select the link *Fusionar mediante combinación* (see Figure 18).

d) The tool to compare the conflict appears (Figure 19). It is possible to compare the commit in the remote repository (left side) which has conflicts with the local commit in conflict (right side)



ProyectoPracticas/HelloWorldApp/Program.cs (Remoto)

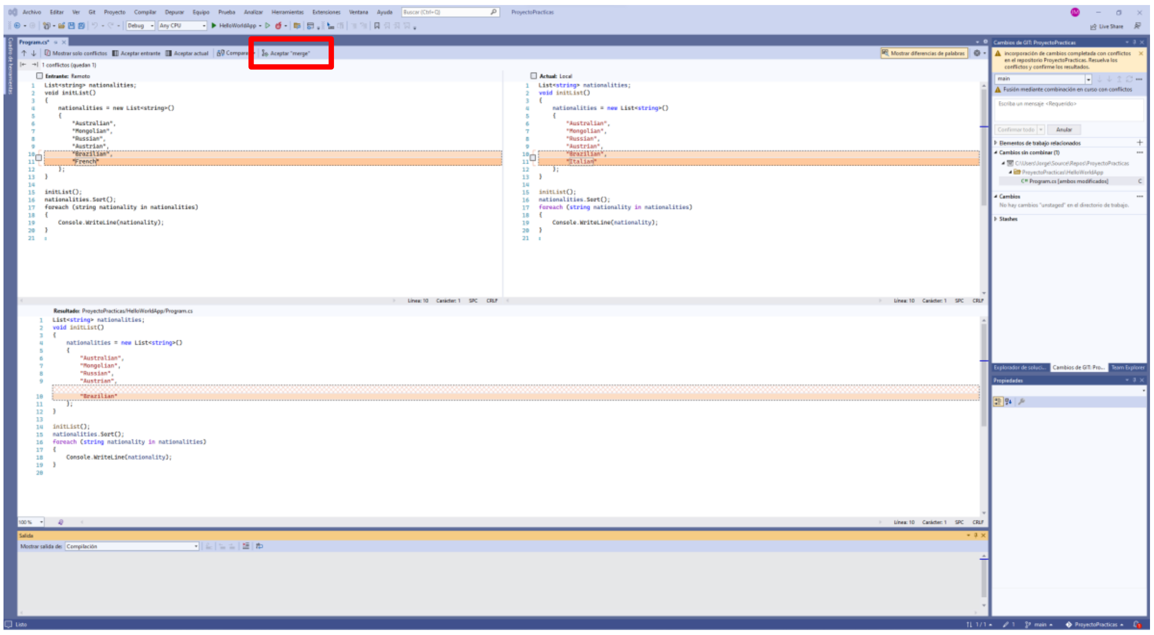
```
1   List<string> nationalities;
2   void initList()
3   {
4       nationalities = new List<string>()
5       {
6           "Australian",
7           "Mongolian",
8           "Russian",
9           "Austrian",
10          "Brazilian",
11  -       "French"
12      };
13  }
14
15  initList();
16  nationalities.Sort();
17  foreach (string nationality in nationalities)
18  {
19      Console.WriteLine(nationality);
20  }
21
```

ProyectoPracticas/HelloWorldApp/Program.cs (Local)

```
1   List<string> nationalities;
2   void initList()
3   {
4       nationalities = new List<string>()
5       {
6           "Australian",
7           "Mongolian",
8           "Russian",
9           "Austrian",
10          "Brazilian",
11  +       "Italian"
12      };
13  }
14
15  initList();
16  nationalities.Sort();
17  foreach (string nationality in nationalities)
18  {
19      Console.WriteLine(nationality);
20  }
21
```

Figure 19. Tool to compare commits

e) In this moment, you can decide to maintain your local changes clicking the option *Mantener actual (local)*, or the remote ones by selecting the link *Aceptar entrante (remoto)*. In this example, you can see a conflict due to different values at line 11.



1 conflictos (quedan 1)

Entrante: Remoto

```
List<string> nationalities;
void initList()
{
    nationalities = new List<string>()
    {
        "Australian",
        "Mongolian",
        "Russian",
        "Austrian",
        "Brazilian",
        "French"
    };
}

initList();
nationalities.Sort();
foreach (string nationality in nationalities)
{
    Console.WriteLine(nationality);
}
```

Actual: Local

```
List<string> nationalities;
void initList()
{
    nationalities = new List<string>()
    {
        "Australian",
        "Mongolian",
        "Russian",
        "Austrian",
        "Brazilian",
        "Italian"
    };
}

initList();
nationalities.Sort();
foreach (string nationality in nationalities)
{
    Console.WriteLine(nationality);
}
```

Resultado: ProyectoPracticas/HelloWorldApp/Program.cs

```
List<string> nationalities;
void initList()
{
    nationalities = new List<string>()
    {
        "Australian",
        "Mongolian",
        "Russian",
        "Austrian",

        "Brazilian"
    };
}

initList();
nationalities.Sort();
foreach (string nationality in nationalities)
{
    Console.WriteLine(nationality);
}
```

Línea: 10 Carácter: 1 SPC CRLF

Cambios de GIT: ProyectoPracticas

Incorporación de cambios completada con conflictos en el repositorio ProyectoPracticas. Resuelva los conflictos y confirme los resultados.

Fusión mediante combinación en curso con conflictos

Escriba un mensaje <Requerido>

Cambios sin combinar (1)
C:\Users\Jorge\Source\Repos\ProyectoPracticas
ProyectoPracticas\HelloWorldApp
Program.cs [ambos modificados]

Cambios
No hay cambios "unstaged" en el directorio de trabajo.

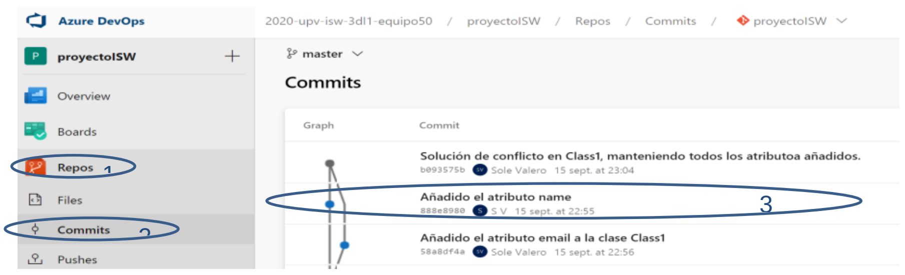
Figure 20: merge by combining both files.

f) If you need to combine both, you need to select the option *Fusionar mediante combinación*. Then, a new tool appears in which you can select which changes will be added to the result (see Figure 20). After selecting them, you need to accept the changes by clicking the option *Aceptar merge*. Finally, you can see the resulting file and, in order to finish, you commit the file and make a push to save changes on the server.

5.3. History of changes

The history of changes of any item of your project may be observed by using the code version explorer from Visual Studio. Thus, select *Explorador de Soluciones,* then click with the right button of the mouse over the element to explore and select *GIT -> Ver historial*.

In a similar way, Azure DevOps allows seeing the history of changes of any item using the option *Repos>Commits.* If you select one commit, the system shows which files were modified and which were the changes made. This option also is available in Visual Studio, from T*eam Explorer> Ramas.* In this case, you should select the branch and then select from the context menu the option V*er Historial.*



Solución de conflicto en Class1, manteniendo todos los atributoa añadidos.
b093575b Sole Valero 15 sept. at 23:04

Añadido el atributo name
888e8980 S V 15 sept. at 22:55

Añadido el atributo email a la clase Class1
58a8df4a Sole Valero 15 sept. at 22:56

Figure 21. Visualize the change history

6. (Optional Task) Advanced Operations with the Version Control

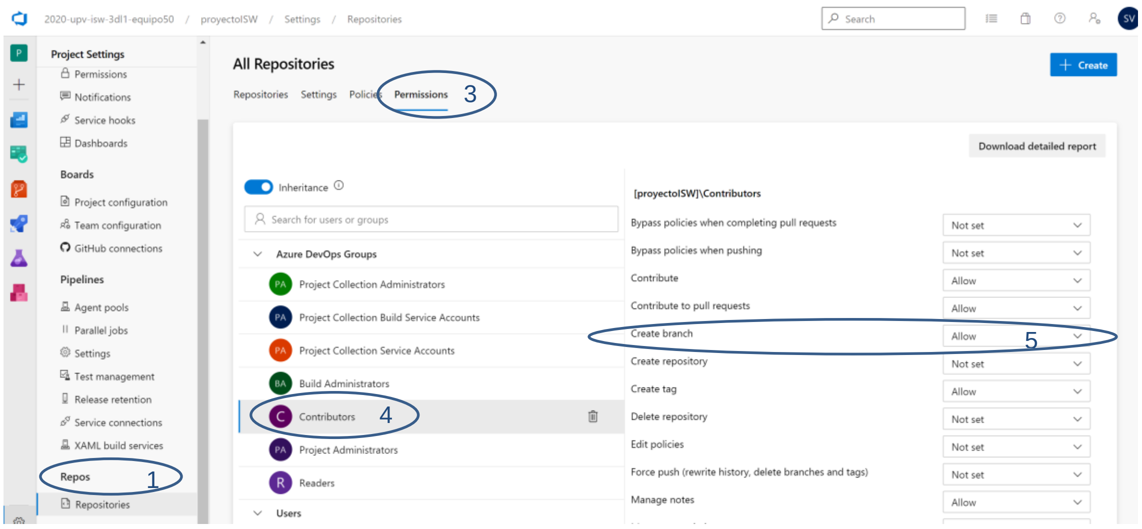
In this section we will learn some additional concepts related to version control:

- **branch**: you can see a branch as a development line, in Git as a sequence of commits. In Git, by default, a repository has only a branch named master, in which are stored all the commits. But, it is possible to create a new branch from any point, so developers can work independently, and store its changes in a new development line, without affecting the users who work in the original branch. It is convenient using consistent naming convention for your feature branches to identify the work done in the branch. For instance:
    - users/username/description
    - users/username/workitem
    - bugfix/description
    - features/feature-name
    - features/feature-area/feature-name
    - hotfix/description
- **merge**: branches can be merged, so the work can be added from one branch to another.

6.1. Generating a development line (branch)

To generate a development line in a way that it isolates the changes made by you from the ones made by others, you can proceed as follows:

a. The Team Master must grant branch creation permission to the rest of the team members at the Azure DevOps (by default, contributors already have got this permission) from *Project settings>Repos>Permissions* (see Figure 22).


2020-upv-isw-3dl1-equipo50 / proyectoISW / Settings / Repositories

Project Settings: Permissions, Notifications, Service hooks, Dashboards
Boards: Project configuration, Team configuration, GitHub connections
Pipelines: Agent pools, Parallel jobs, Settings, Test management, Release retention, Service connections, XAML build services
Repos 1: Repositories

All Repositories
Repositories | Settings | Policies | Permissions 3

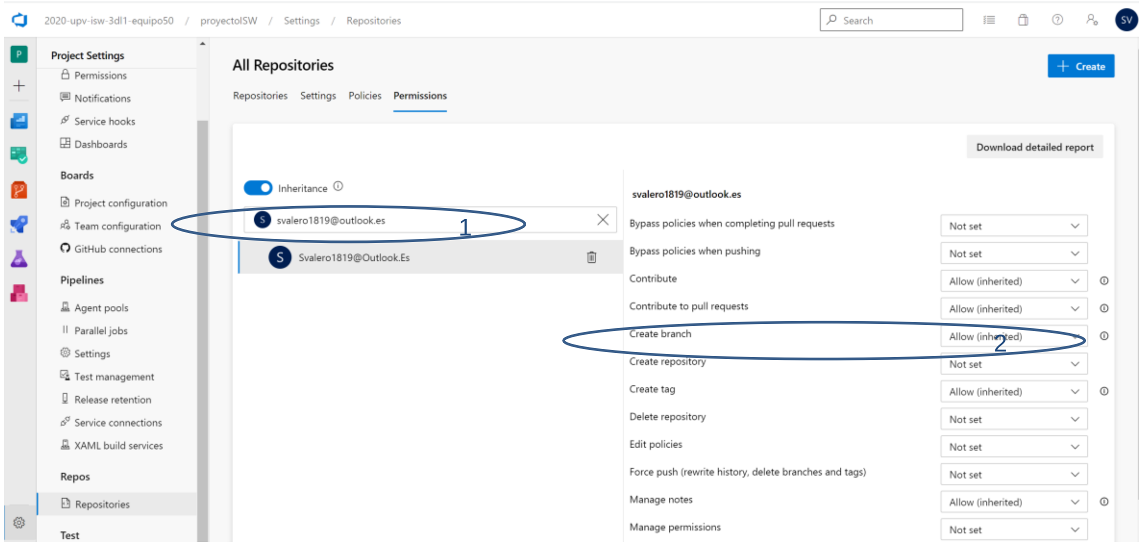
Inheritance
Search for users or groups
Azure DevOps Groups: Project Collection Administrators, Project Collection Build Service Accounts, Project Collection Service Accounts, Build Administrators, Contributors 4, Project Administrators, Readers
Users

[proyectoISW]\Contributors

Permission	Value
Bypass policies when completing pull requests	Not set
Bypass policies when pushing	Not set
Contribute	Allow
Contribute to pull requests	Allow
Create branch 5	Allow
Create repository	Not set
Create tag	Allow
Delete repository	Not set
Edit policies	Not set
Force push (rewrite history, delete branches and tags)	Not set
Manage notes	Allow

Figure 22. Project settings. Check the repository permissions

b. If needed, it is possible to assign permissions individually, although it is recommended to manage permissions through the predefined groups. To do this, select the user by indicating his or her account in the edit field *"Search for users or groups"* of Figure 22. In this example (see Figure 23), the user svalero1819@outlook.es was selected, which was previously added to team development group. After selecting a user, the permissions assigned to him/her appear. These permissions can be changed clicking on the right panel. Note that some are inherited and do not need to be modified (they are marked as inherited). Grant branch creation permissions to the users you want.


2020-upv-isw-3dl1-equipo50 / proyectoISW / Settings / Repositories

All Repositories
Repositories | Settings | Policies | Permissions

Inheritance
svalero1819@outlook.es 1
Svalero1819@Outlook.Es

svalero1819@outlook.es

Permission	Value
Bypass policies when completing pull requests	Not set
Bypass policies when pushing	Not set
Contribute	Allow (inherited)
Contribute to pull requests	Allow (inherited)
Create branch 2	Allow (inherited)
Create repository	Not set
Create tag	Allow (inherited)
Delete repository	Not set
Edit policies	Not set
Force push (rewrite history, delete branches and tags)	Not set
Manage notes	Allow (inherited)
Manage permissions	Not set

Figure 23. Grant branch creation permission to one user

c. Once create branch permission is granted, users can create new branches in the repository. In Visual Studio, select the branch icon below and click on `Nueva rama` (Figure 24).

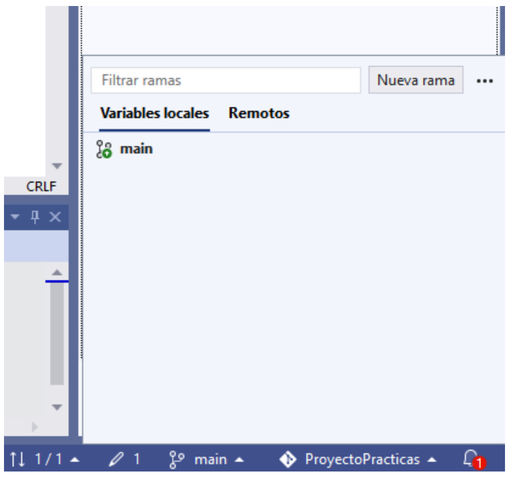
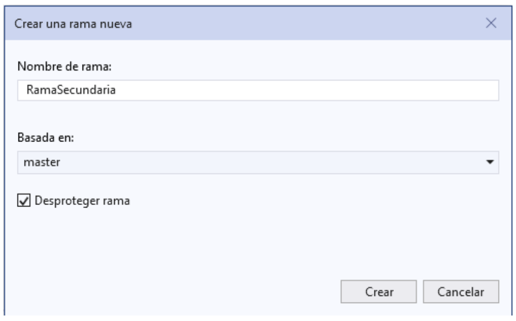


Figure 24: location of the new branch button

d. Select the branch from you want to create a new one (local branch master in Figure 24).
e. Finally, assign a new name for the branch (Figure 25) and click on the button *Crear*. Both branches are in the local repository, but only the master one is (for now) on the remote.



Nombre de rama: RamaSecundaria

Basada en: master

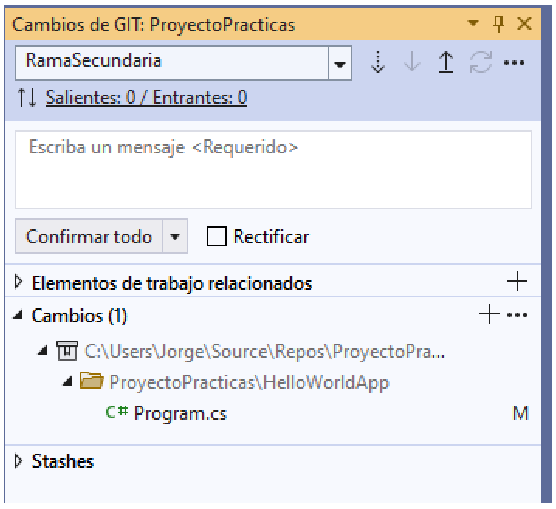
☑ Desproteger rama

Figure 25: creating a new branch from a local version of the code

6.2. Updating and pushing a local branch

As it has been discussed earlier, at any time a local branch can be created from other branch, creating a new line of development. From this point, new changes can be committed in the new branch and also pushed to the remote repository, without interfere in the original branch or line of development. Now, let is practise with this:

a. Modify again the class *Program* by changing another element in the list ("Italian", "Portuguese", etc.)
b. Create a new commit with your changes from *Cambios de GIT*, adding a text to explain the commit and select *Confirmar Todo* (see Figure 26).


Cambios de GIT: ProyectoPracticas

RamaSecundaria

Salientes: 0 / Entrantes: 0

Escriba un mensaje <Requerido>

Confirmar todo ☐ Rectificar

Elementos de trabajo relacionados

Cambios (1)

C:\Users\Jorge\Source\Repos\ProyectoPra...

ProyectoPracticas\HelloWorldApp

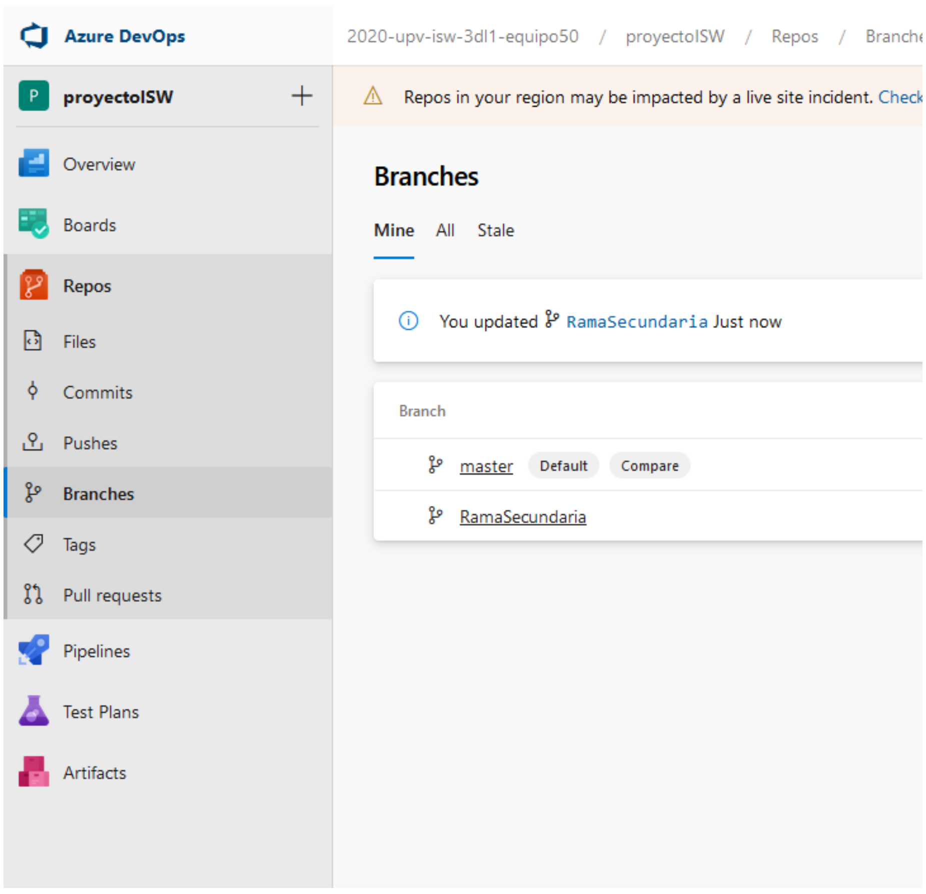
C# Program.cs M

Stashes

Figure 26: changes in the solution from a new branch

c. Then push the commit to the remote repository by clicking on the link *Sincronizar* from the dialog that appears (see Figure 16).

d. Also, you can see the two existing branches on Azure DevOps web, selecting the option *Repos>Branches* (Figure 27).



2020-upv-isw-3dl1-equipo50 / proyectoISW / Repos / Branche

Repos in your region may be impacted by a live site incident. Check

Branches

Mine All Stale

You updated RamaSecundaria Just now

Branch
master Default Compare
RamaSecundaria

Figure 27: branches on the server: main and RamaSecundaria

6.3. Branches combination by merging

It is possible to merge two branches in a way that the changes of both are combined. This is useful to add the changes from a development branch into release branch or to combine two development branches. In this case, we are going to use other example in which a new branch named *RamaSecundaria* was created. You can follow the same steps to add the commits made in your branch *RamaSecundaria /NewCountries* to the main branch *master*.

a. Click on the current branch in the bottom bar in Visual Studio, select the current branch and open the menu context by cicking on the right button. Then, select the option `Combinar en la rama actual…` from the context menu (see Figure 28).

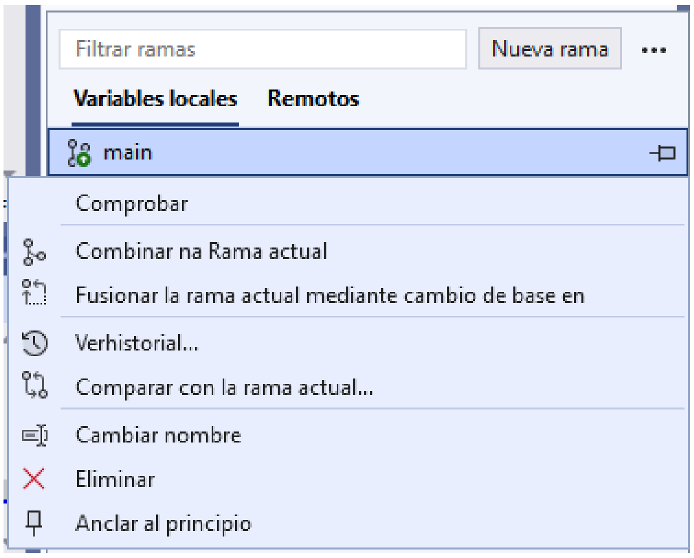


Figure 28: merging two branches

b. Finally, click on the button *Sí*. Conflicts may arise when performing this operation. The conflict can be solved automatically or by using the conflict resolution tool as we saw before (section 5.2).
c. Remember to confirm all the changes and synchronize the new commits, so the merge between both branches will appear also in the remote repository.
d. You can check via the Azure DevOps web application that in the central repository both branches now contain the same version of the class Program.

In this example there are no conflicts, but it is possible that some conflict occurs in this process. In that case, you should act as explained in section 5.2.

For more information on the version control model in branches in *Azure DevOps with Git* visit the information available in:

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

Also, knowing about GitFlow workflow to work with branches can be a good idea for your team. There are several resources in Internet to know about it, but we can suggest you the book available online in the UPV Library: Santacroce, F. (2015). Git Essentials. Olton Birmingham: Packt Publishing.

NOTE: The development team should decide how many branches they want to have; only the main branch, the main and a development branch, etc. We also recommend you read https://www.visualstudio.com/nl-nl/docs/tfvc/branch-strategically to decide a right branch strategy.