

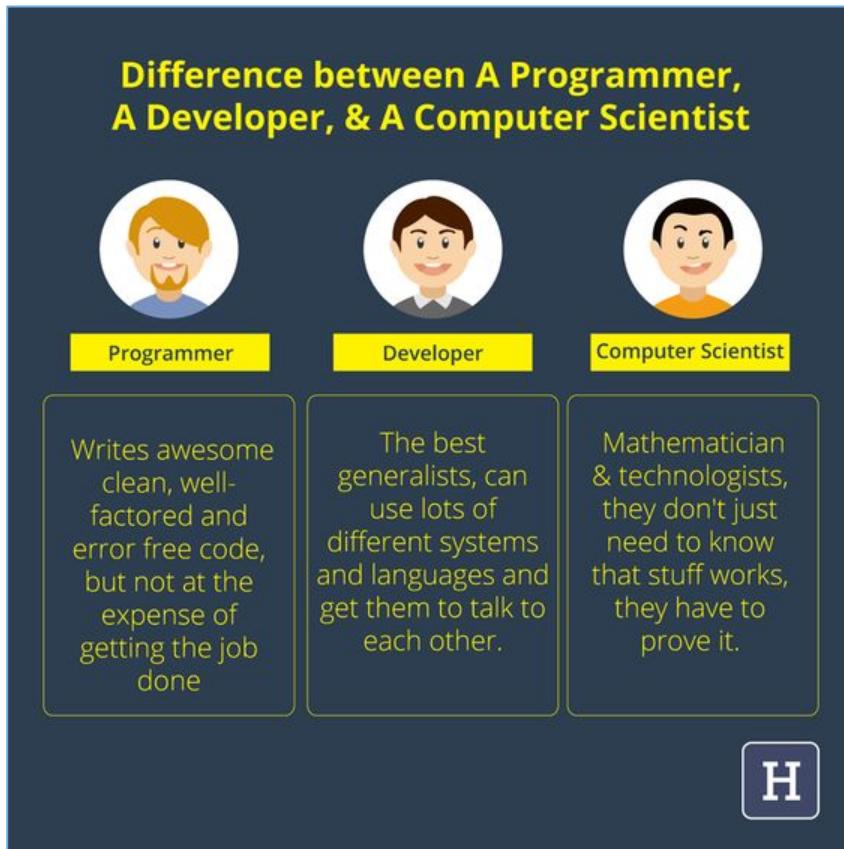
Sommario

Introduzione	2
Le figure professionali nel settore informatico	2
Il concetto di Versione	2
IDE Integrated development environment	3
Esempio di IDE.....	3
Utilizzo della rete e dell'ambiente di sviluppo	4
Git and GitHub for Beginner.....	12
Iscrizione GitHub.....	19
Clonare Repository in locale.....	26
Branches.....	30
Main Only Strategy.....	30
Development Isolation	30
Unit Test	33
Unit Test in Visual Studio	33

Introduzione

Le figure professionali nel settore informatico.

Introduciamo le figure fondamentali informatiche.



Programmer: scrive codice, senza errori entro una data scadenza. E' esperto su una specifica **tecnologia**.

Developer: è lo sviluppatore, non specializzato, generalista, ha una visione d'insieme dell'intero **progetto**. Può usare sistemi e linguaggi differenti e sa dare le specifiche ai programmatore. E' il team manager, cioè gestisce un gruppo di progettatori. Lo sviluppatore conosce tutti i componenti **della progettazione dell'applicazione**.

Computer scientist: risolve problemi. Decide come devono essere risolti i problemi, come devono essere fatte le applicazioni ed i **sistemi informatici**.

Il concetto di Versione

10.4.1.1325

Le versioni si indicano attraverso dei numeri separati da un punto. Vediamone il significato.

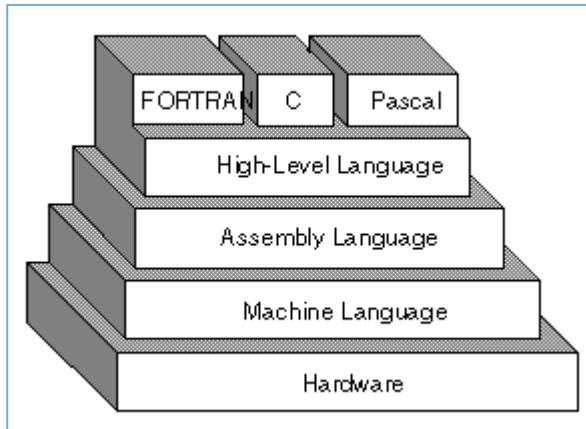
Major Version: l'applicazione ha un cambiamento radicale rispetto alla precedente.

Minor Version: rispetto alla versione precedente è cambiato un qualcosa di significativo (cambio icone, spostamento di un menù...)

Release riguarda la correzione di bug. Le release note rappresentano la descrizione di cosa è stato corretto.

Build: rappresenta la costruzione dell'applicazione. Ad ogni verifica del codice si conta una build. Spesso rappresenta anno, mese e giorno di quando è stata fatta.

IDE Integrated development environment



Queste sono le azioni fondamentali che svolge un IDE:

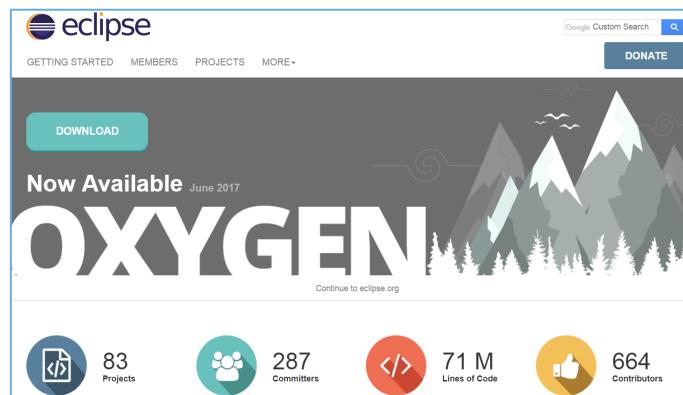
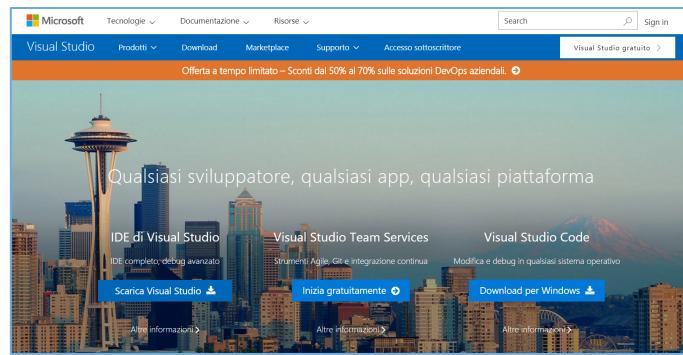
1. Scrivo codice con un linguaggio ad alto livello.
2. Compila e traduce in linguaggio macchina (linguaggio a basso livello), l'unico linguaggio comprensibile dal computer.
3. Eseguo il programma; se il programma non funziona segue un'ulteriore fase: debug.

L'IDE è integrato cioè comprende l'editor, il compiler, il runner, ed il debugger. Un ambiente integrato permette di svolgere tutte le azioni di scrittura, compilazione, esecuzione e correzione.

Esempio di IDE

The screenshot shows a web browser window for the Apple Developer website. The URL bar shows "Xcode". The main content area is titled "What's New in Xcode 9". Below the title, there is a paragraph of text describing the features of Xcode 9, mentioning Swift 4 compatibility and refactoring tools.

With everything you need to create amazing apps for Apple platforms, Xcode 9 is unbelievably quick and consistently smooth while editing even the largest files. It also understands your code better than ever, so you can select and edit the structure or even transform the selection directly in the editor. Powerful new refactoring tasks happen in place, renaming symbols across Swift, Objective-C, and even user interface files without skipping a beat. And with source compatibility in Swift 4, Xcode 9 uses the same compiler to build existing Swift 3 code and updated Swift 4 code, so you can migrate at your own pace.



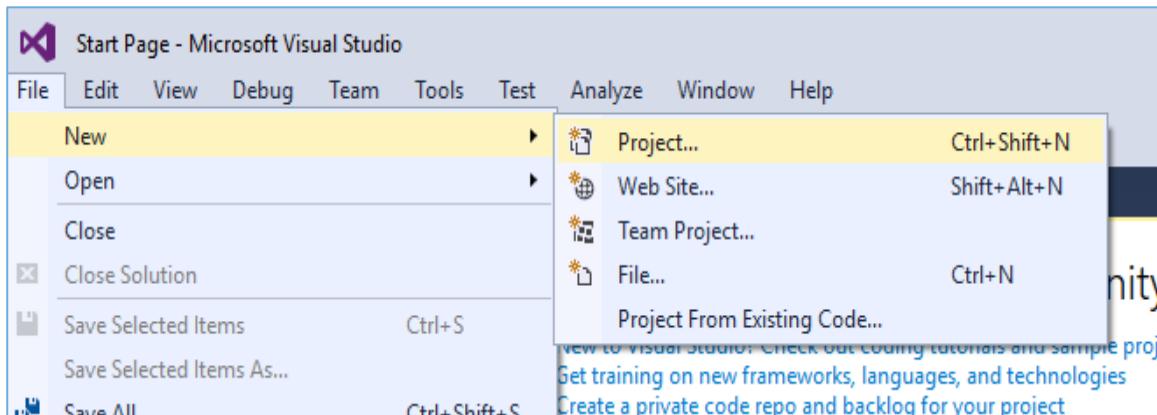
Utilizzo della rete e dell'ambiente di sviluppo

Nome del server della rete DC01srv è il nome simbolico del server della scuola.

Se non si conosce il nome si può trovare con l'indirizzo non mnemonico 192.168.5.250.

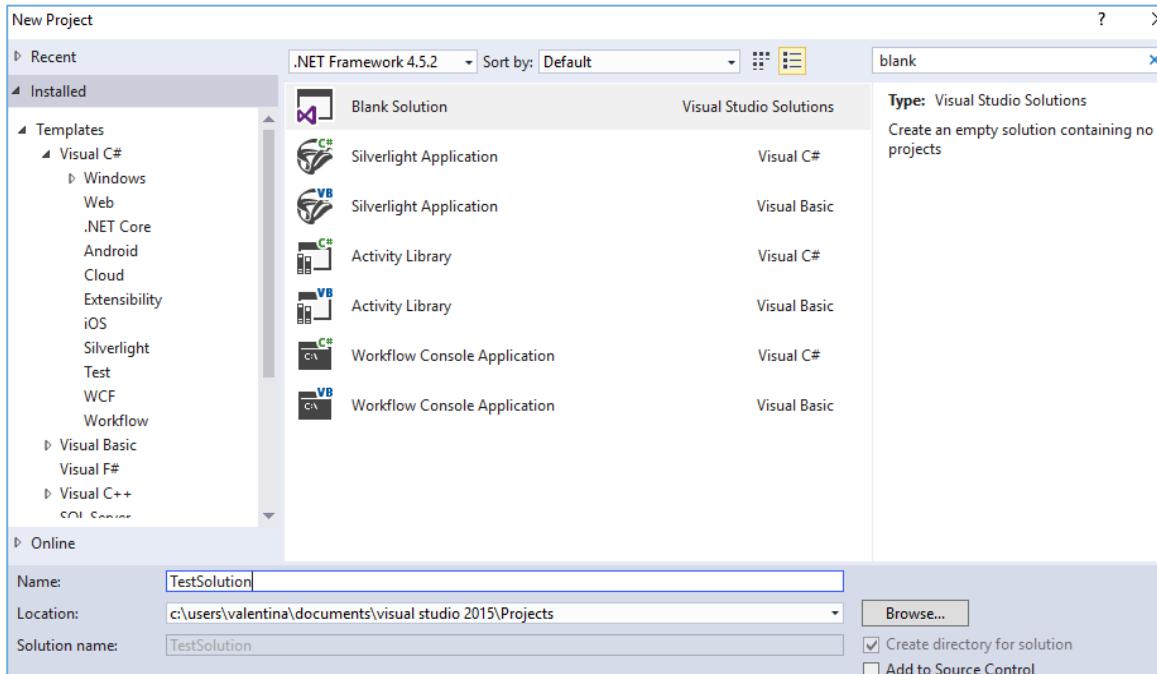
All'apertura di Visual studio è importante avere le seguenti finestre attive

- Solution Explore
- Properties
- Toolbox
- Error
- Output



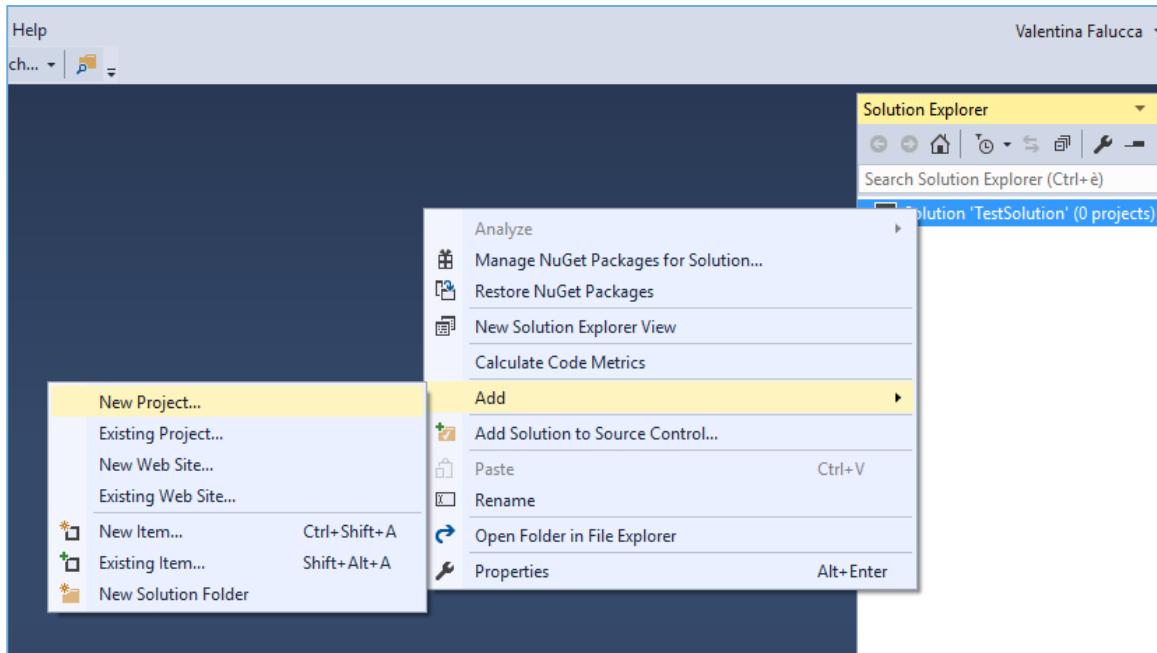
La prima cosa da creare è sempre una soluzione.

Ricordarsi di salvare nella giusta cartella e di chiamarla nel modo corretto (Iniziali di tutte le parole Maiuscole, senza spazi).

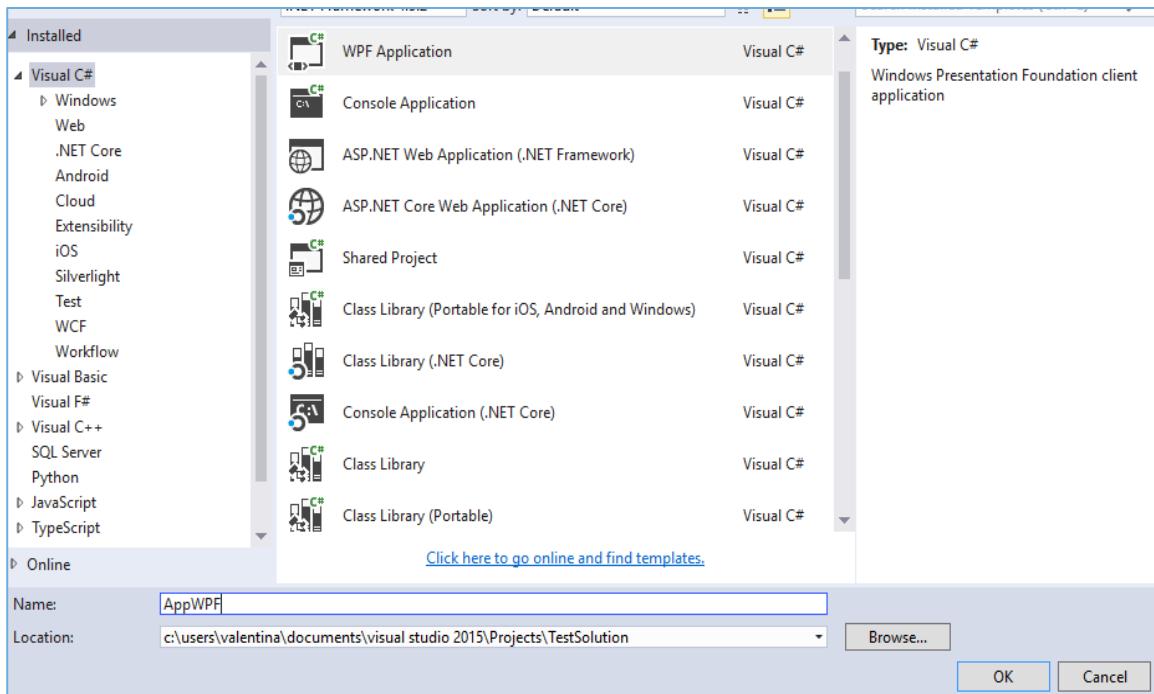


Successivamente si passa a creare i progetti.

Bisogna scegliere il tipo di progetto ed il linguaggio di programmazione.

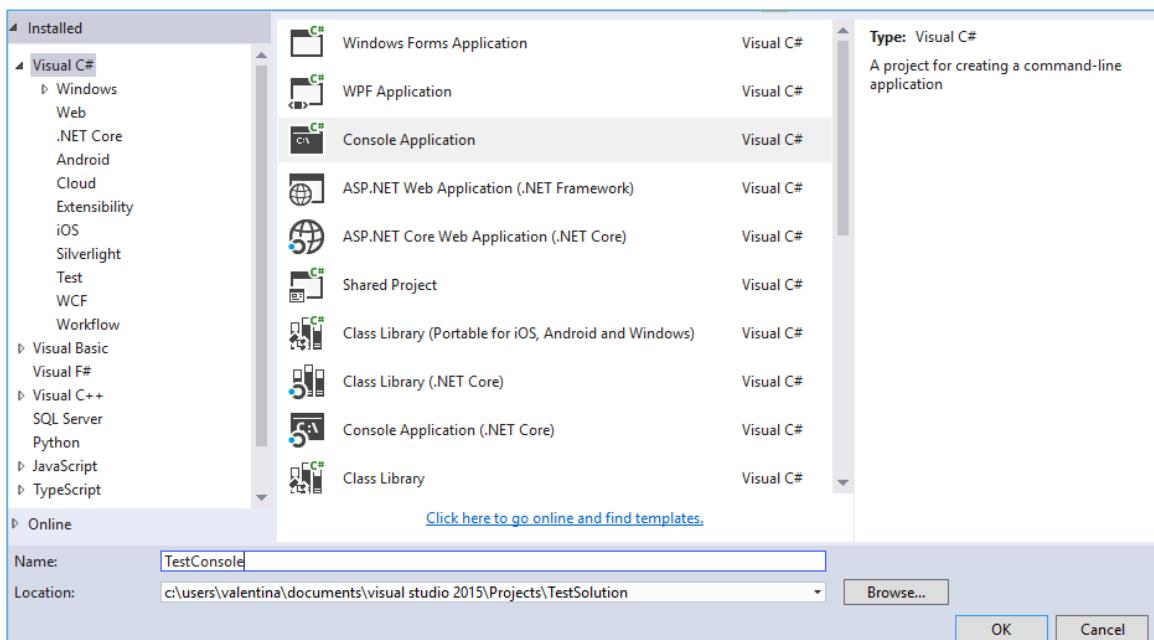


Vediamo il primo progetto da inserire: WPF Application:

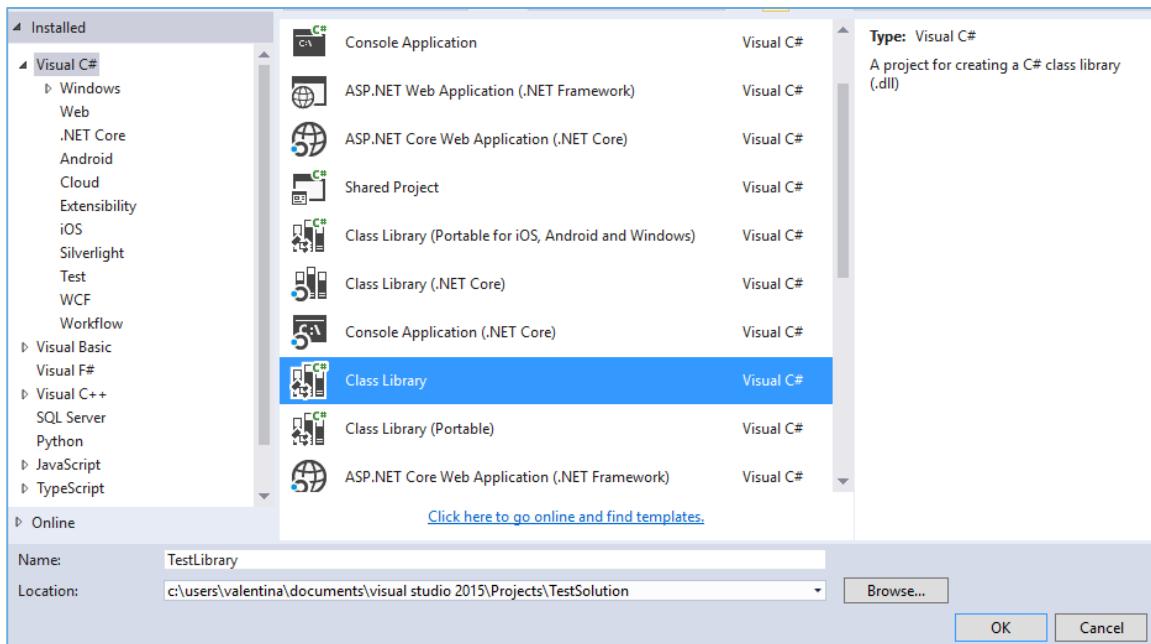


E' necessario sottolineare che se si commette un errore non si può rinominare.

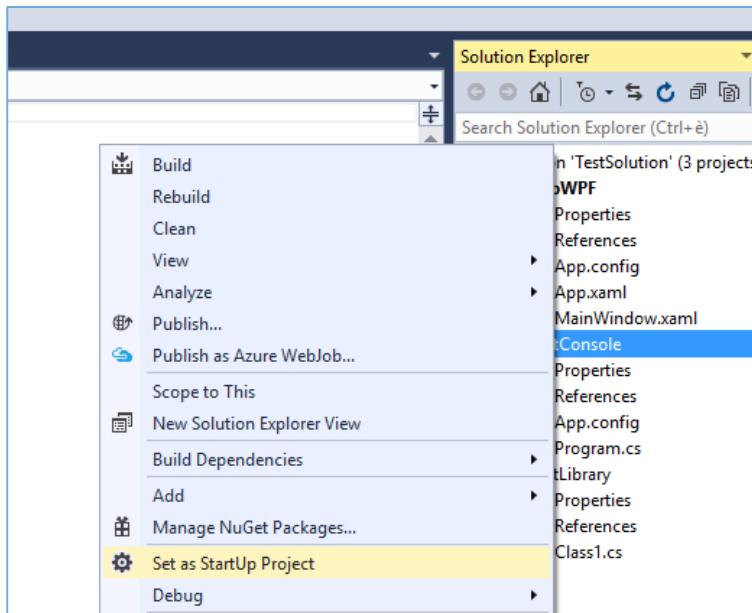
Inseriamo un secondo progetto: Console Application.



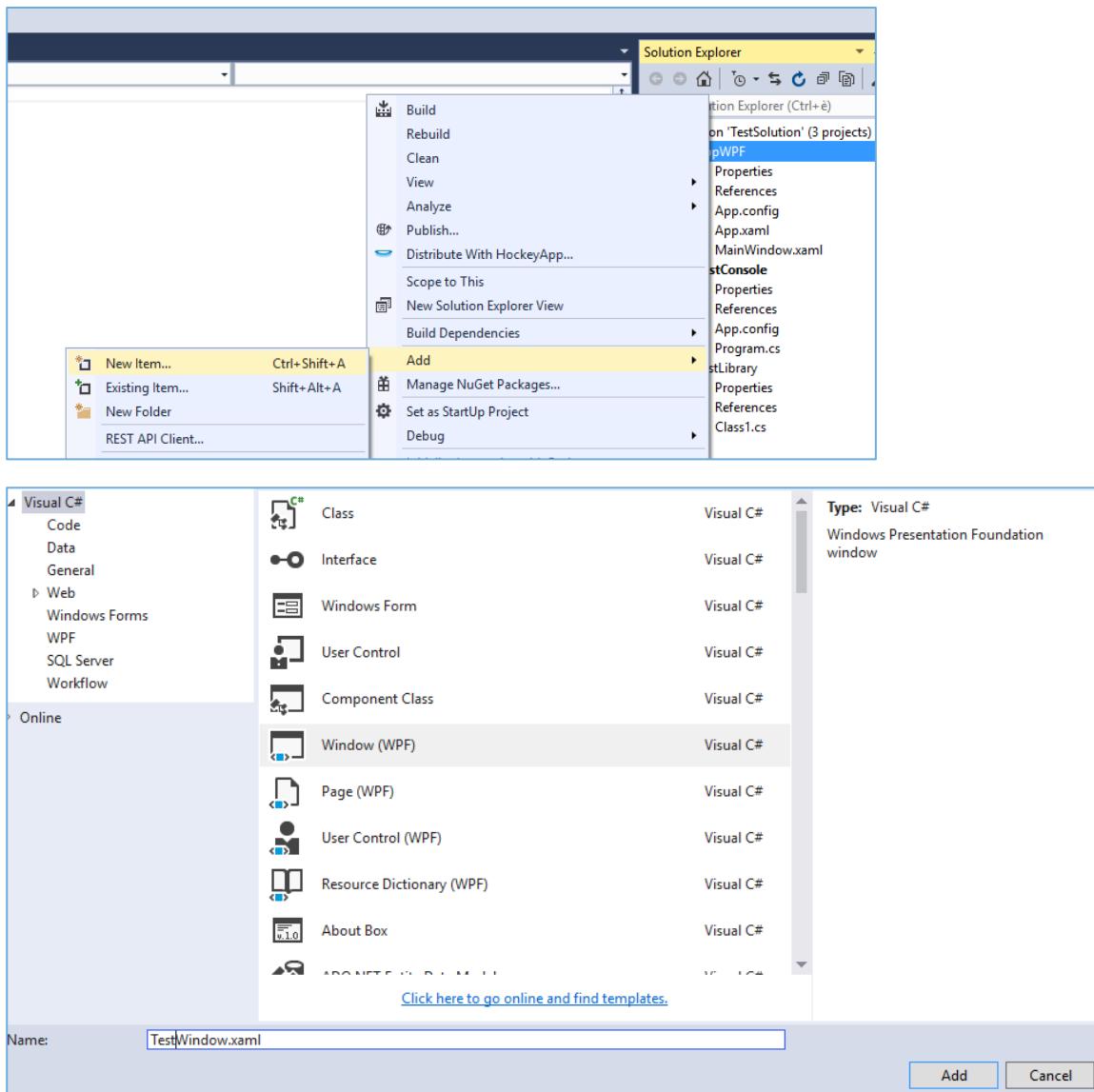
Ed infine il terzo progetto: Class Library.



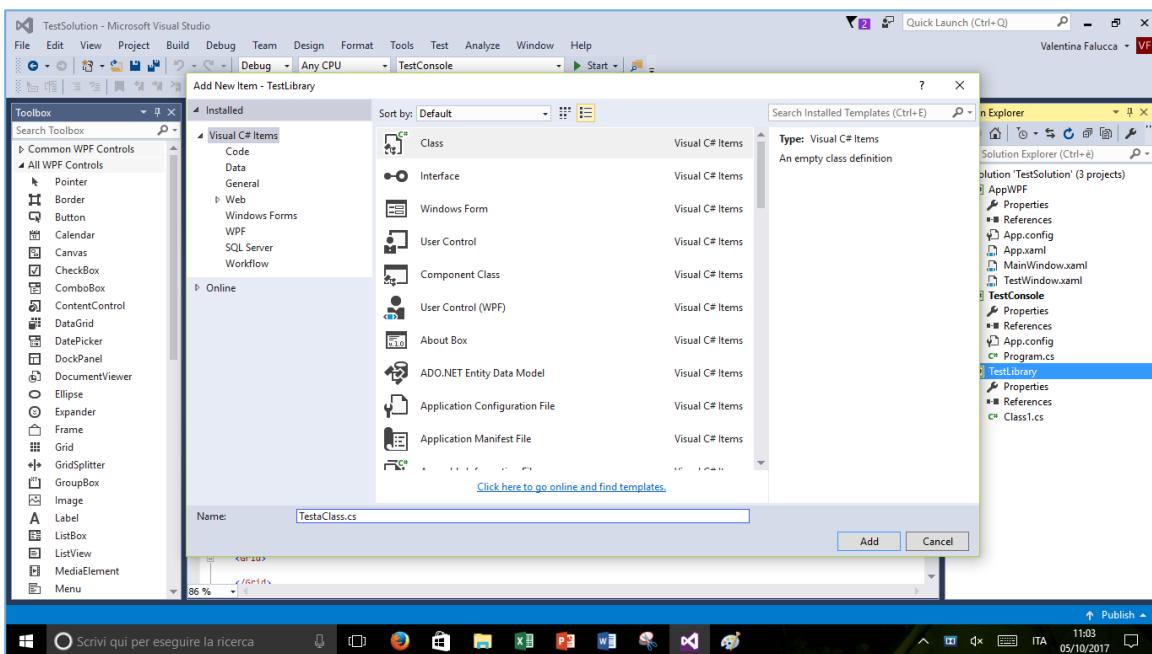
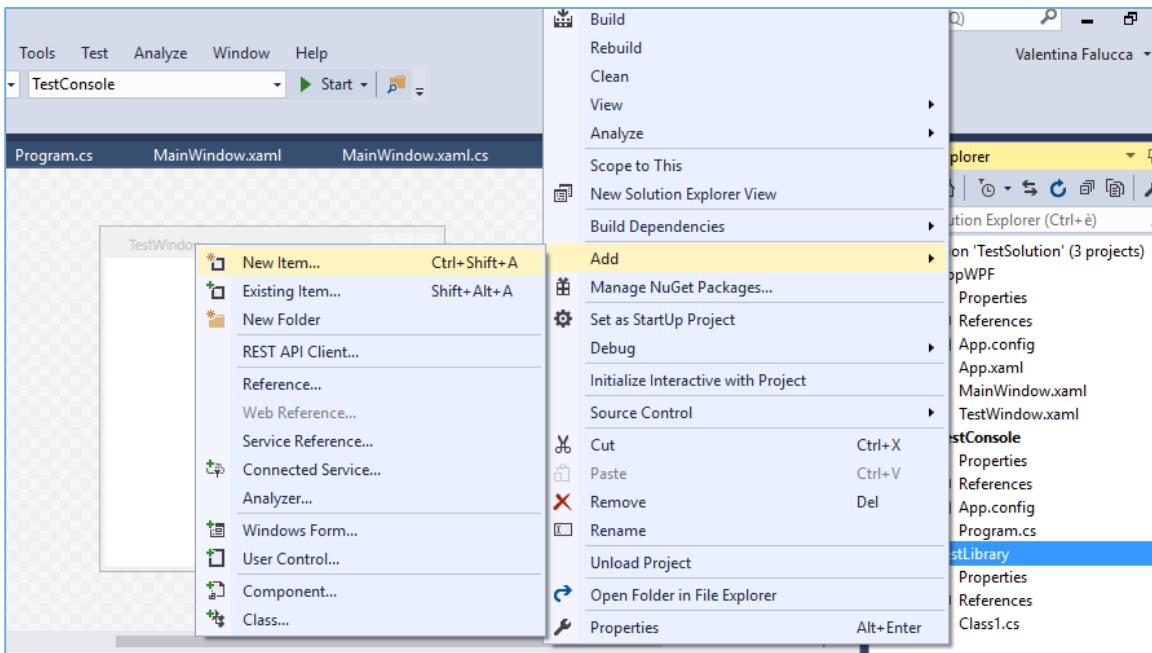
Per decidere quale progetto far partire selezionare con il tasto destro il prescelto e poi cliccare su "Set as StartUp Project".



In ogni progetto è possibile aggiungere item, nell'esempio proposto una Window (WPF).

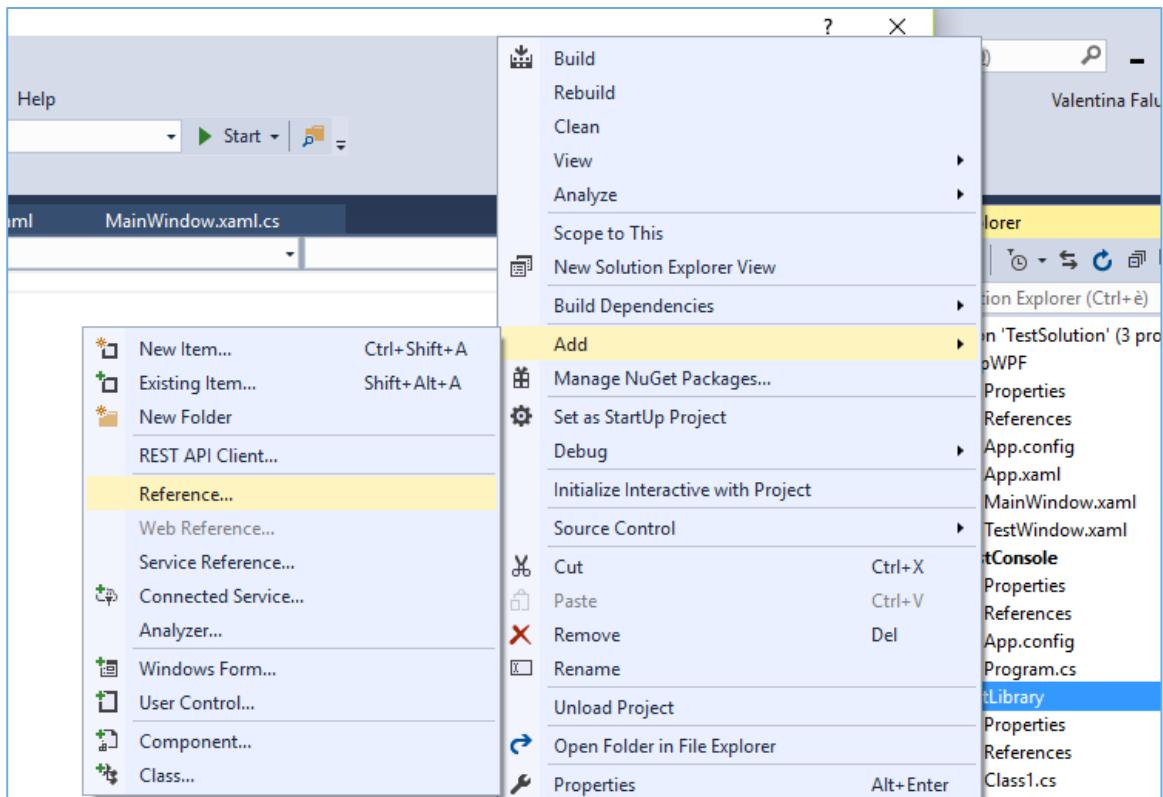


Vediamo come aggiungere un item alla Library, nell'esempio proposto si seleziona una classe.

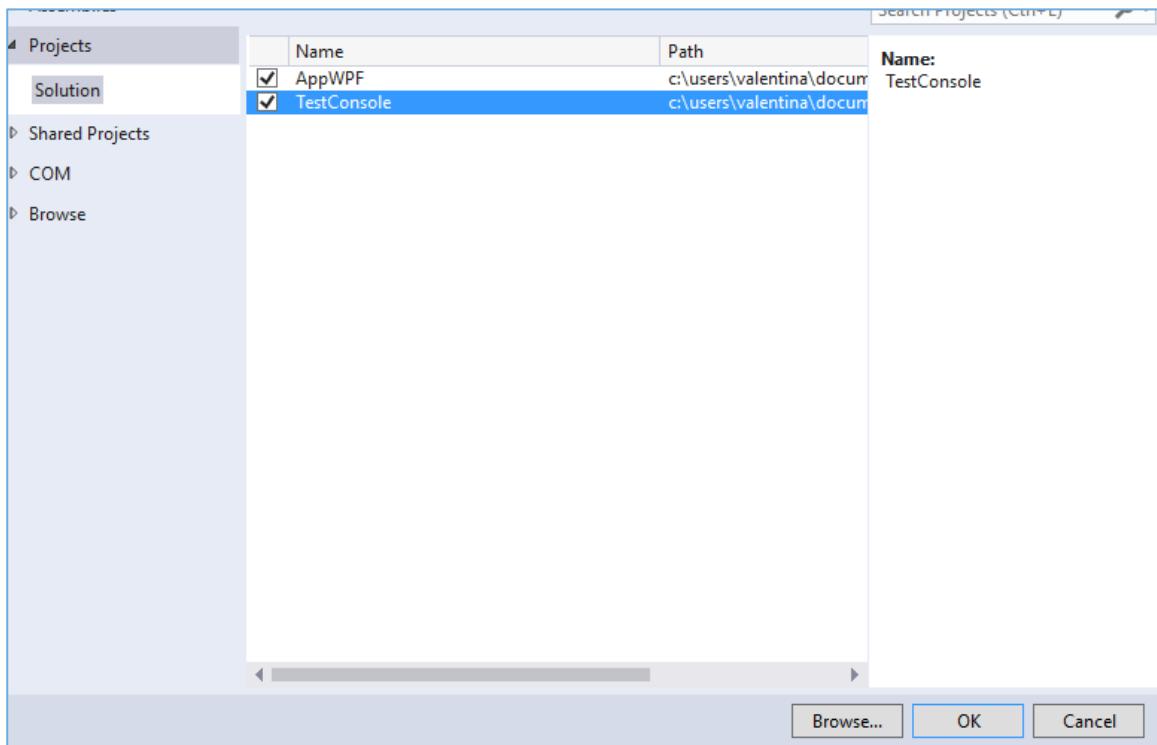


Quando si elimina un item ricordarsi che l'operazione è irreversibile.

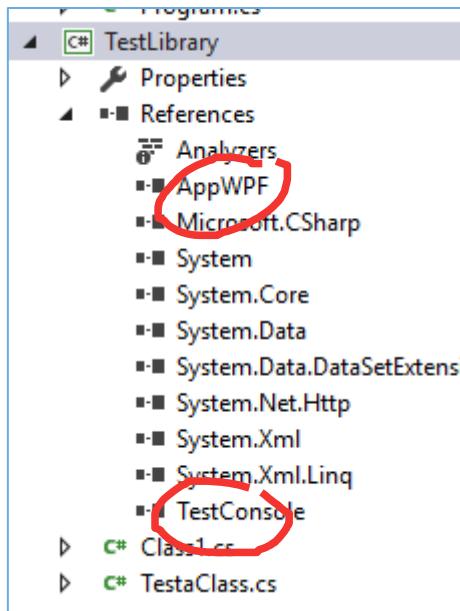
Vediamo come si inserisce un riferimento di un progetto ad un altro progetto. Nel nostro esempio si vuole aggiungere a TestLibrary il riferimento al progetto TestConsole ed al progetto AppWPF.



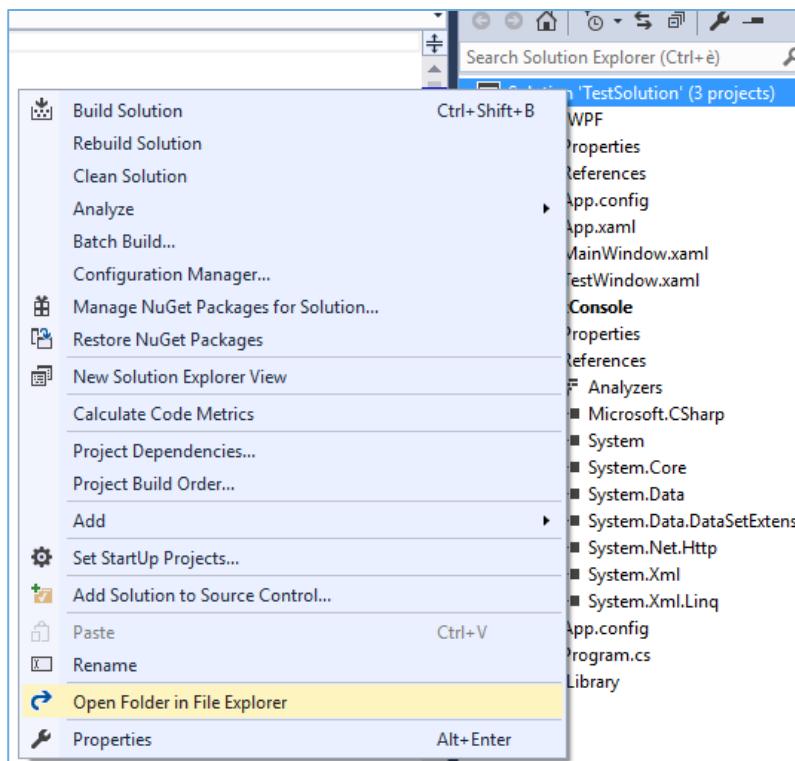
Si apre una nuova finestra che contiene i progetti già presenti.



E' possibile verificare l'esistenza della reference in Explore Solution.



Per eliminare un progetto è necessario accedere alla cartella dove è salvato il progetto (tasto destro sul nome del progetto da eliminare e cliccare su “Open Folder in File Explorer”).



Git and GitHub for Beginner

Per poter sviluppare in team sono necessari ambienti completi di sviluppo ed un sistema Git.

La parola chiave di un sistema Git è Version Control.

Un sistema a controllo di gestione tiene traccia di tutte le modifiche effettuate al codice e di tutta la storia del progetto (come fare una fotografia (Snapshots) per controllare le differenze tra una versione e l'altra). Si possono effettuare cambiamenti al codice senza collegamento ad internet. L'intero codice viene conservato in una macchina locale. Si può anche utilizzare un server remoto per condividere con il team il codice e tener traccia di tutte le modifiche.

Un sistema a controllo di versione consente:

- di tenere traccia dei cambiamenti.
- uno sviluppo collaborativo.
- di capire chi ha effettuato dei cambiamenti e quando.
- di tornare ai progetti precedenti.

GitHub è una particolare implementazione di Git con diverse funzionalità.

How developers work

Support your workflow with lightweight tools and features. Then work how you work best—we'll follow your lead.

New to GitHub? See how it works

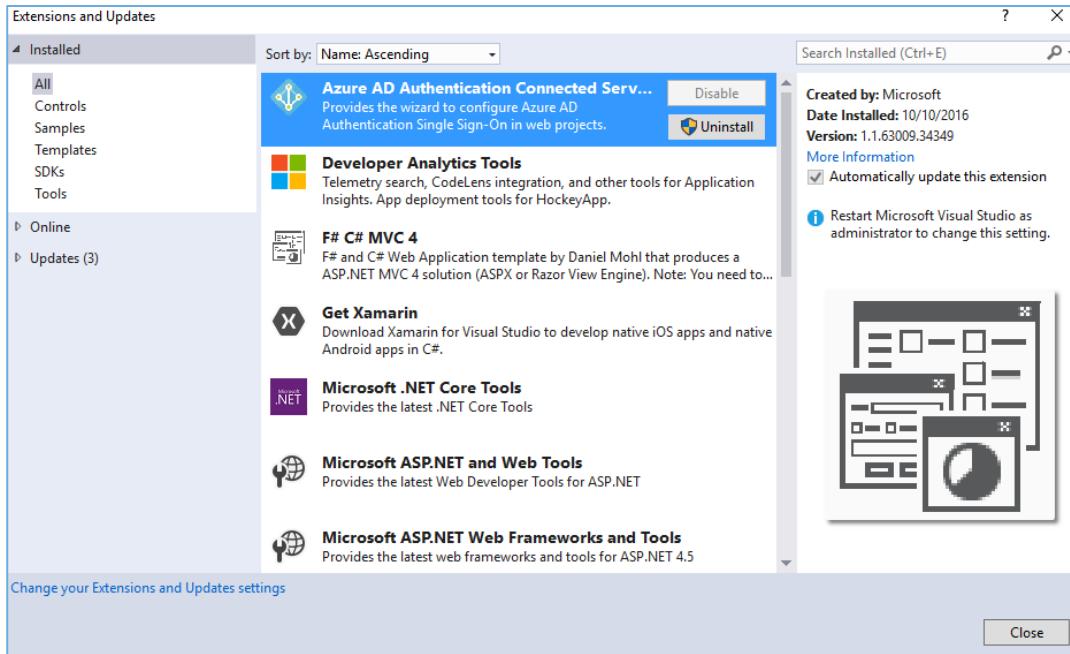
Code review Project management Integrations Team management Social coding Documentation Code hosting

I progetti in GitHub vengono chiamati Repository: Code Review, Project Management, Integrations, Team Management, Social Coding, Documentation, Code Hosting.

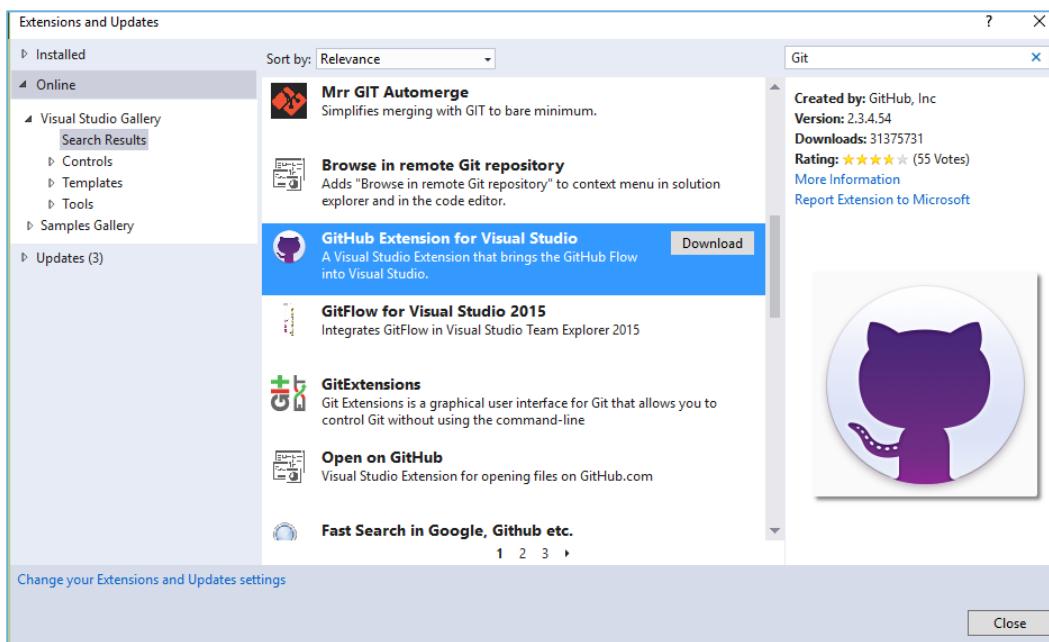
Attraverso git hub è possibile tracciare modifiche su un progetto, gestire più versioni (branch) e successivamente effettuare la fusione (merge).

Per gestire repository da Visual Studio è necessario scaricare un'estensione che consente di gestire una solution come un repository ed eseguire direttamente comandi dall'IDE.

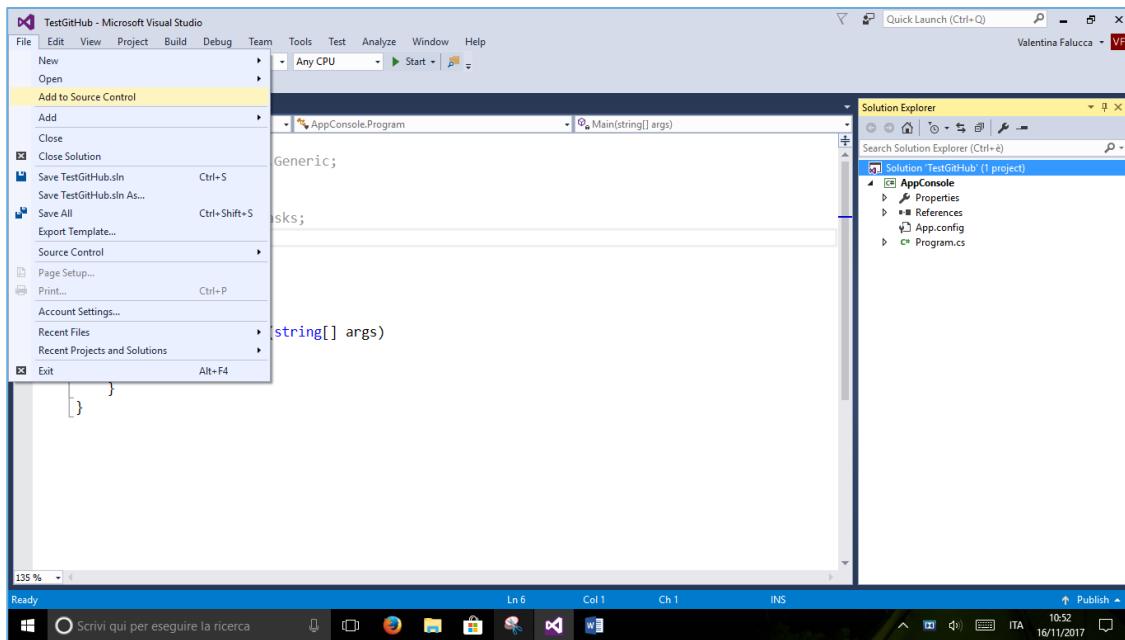
Dal menu Tools di Visual Studio cliccare “Extension and Updates”. Si aprirà una finestra in cui è possibile vedere tutte le estensioni installate.



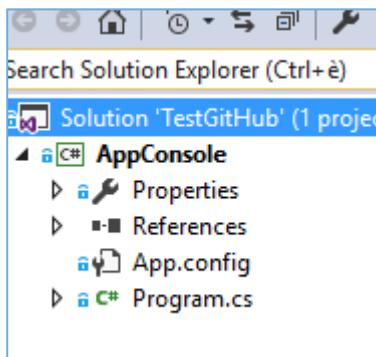
Dal menu “on line” digitare Git ed installare GitHub Extension for Visual Studio.



Dopo aver creato un Blank Solution chiamata TestGitHub ed un progetto Console chiamato AppConsole, seguiamo la procedura File-Add to Source Control e visualizzata nell’immagine che segue:



Notiamo che in Solution Explorer compare un lucchetto blu davanti ai nomi dei file.



Facciamo una modifica al file e salviamo.

Nonostante il salvataggio in Visual Studio, la versione non è cambiata, infatti vediamo una spunta rossa davanti al nome del file modificato.

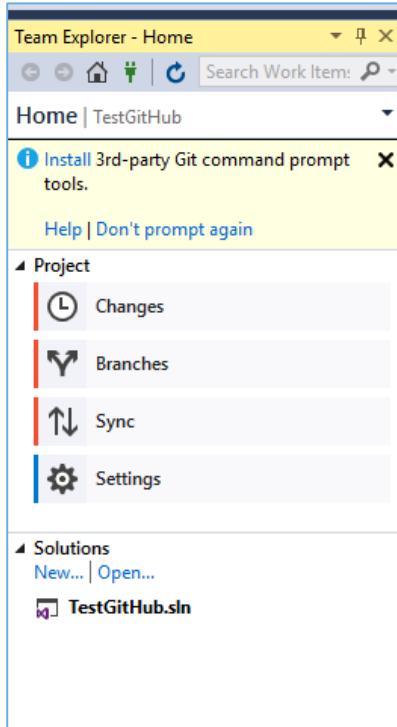
```

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

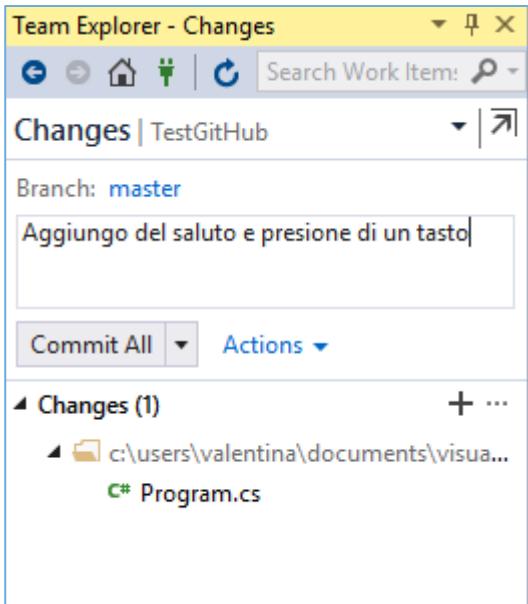
namespace AppConsole
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("ciao");
            Console.ReadLine();
        }
    }
}

```

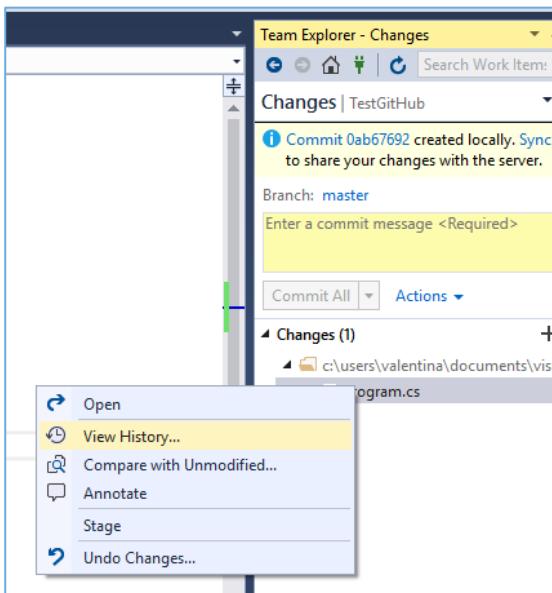
Apriamo la finestra Team Explorer (da View) e vediamo le azioni necessarie per salvare la versione.



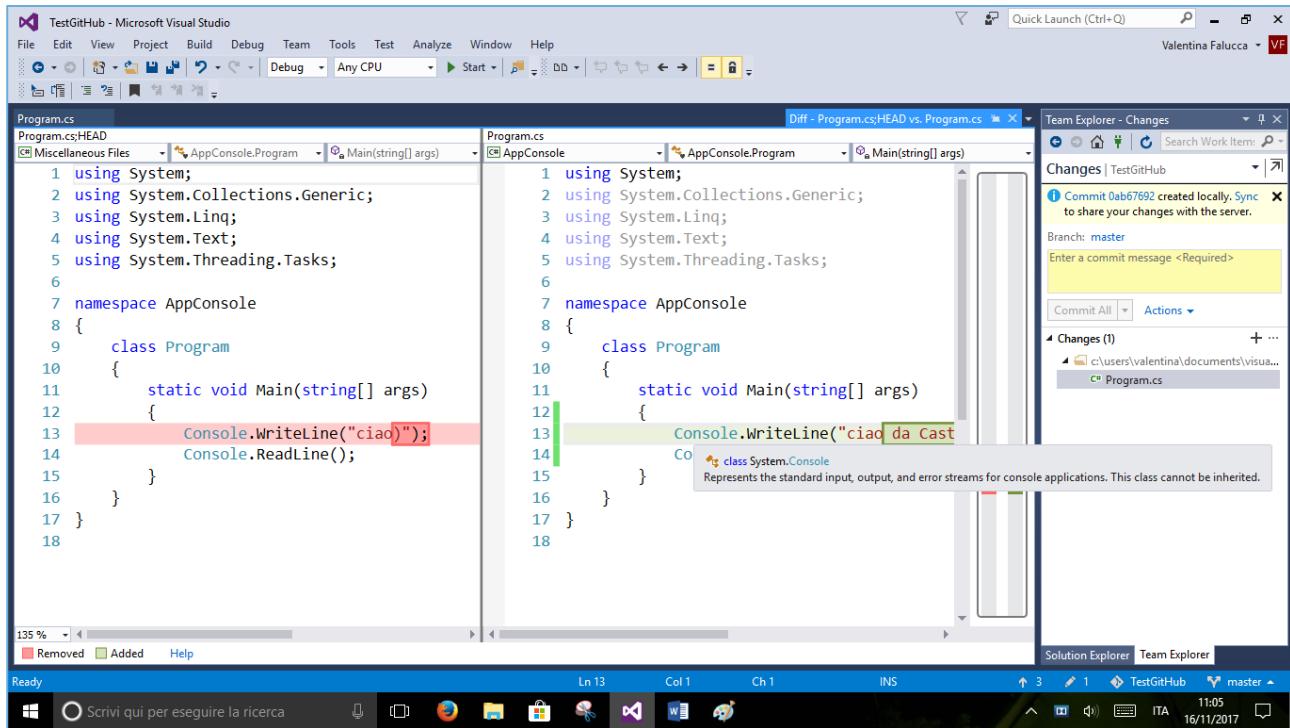
Cliccando su Changes si vedono le modifiche. Per salvare la commit localmente si inserisce una descrizione dettagliata della modifica fatta al codice e si clicca poi su Commit All. Tornando su solution si potrà osservare che tutti i lucchetti sono tornati visibili.



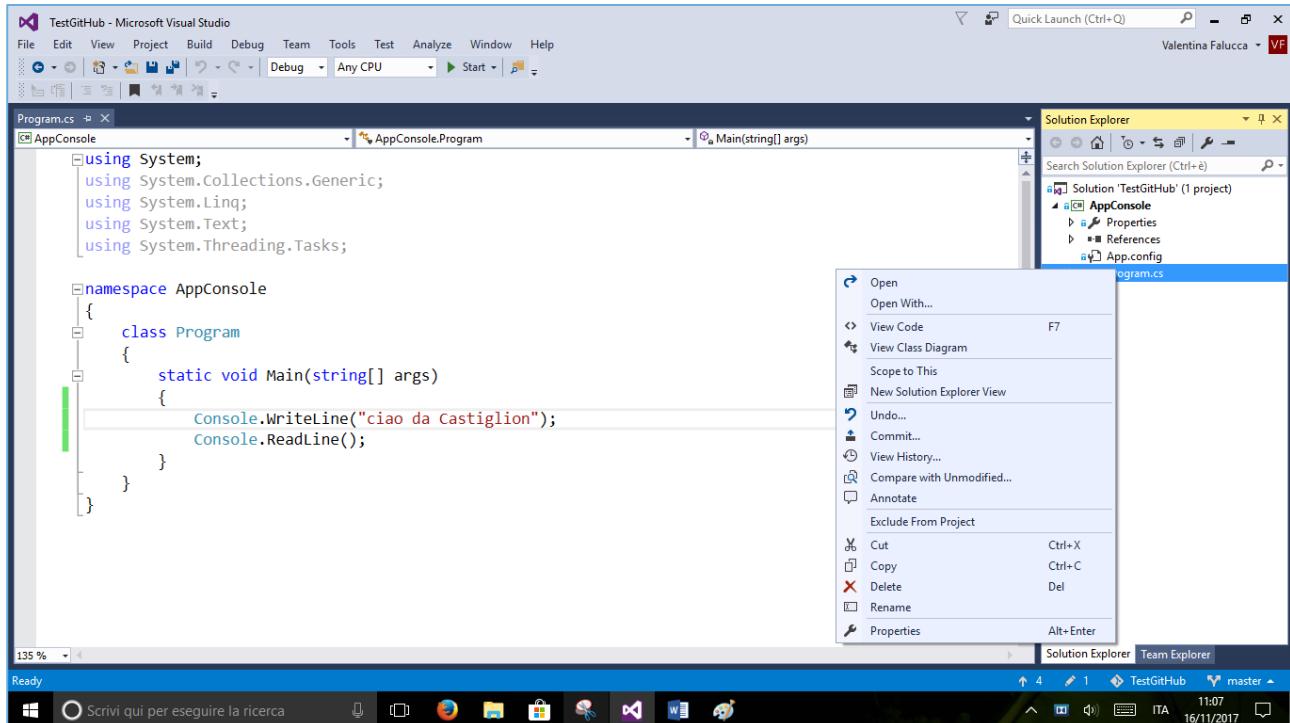
Se si effettua una seconda modifica al codice (salvando il file) e si torna alla finestra Team Explorer è possibile confrontare le due versioni, la nuova con la precedente cliccando con tasto destro sul nome del file modificato e poi selezionando su “Compare with Unmodified”.



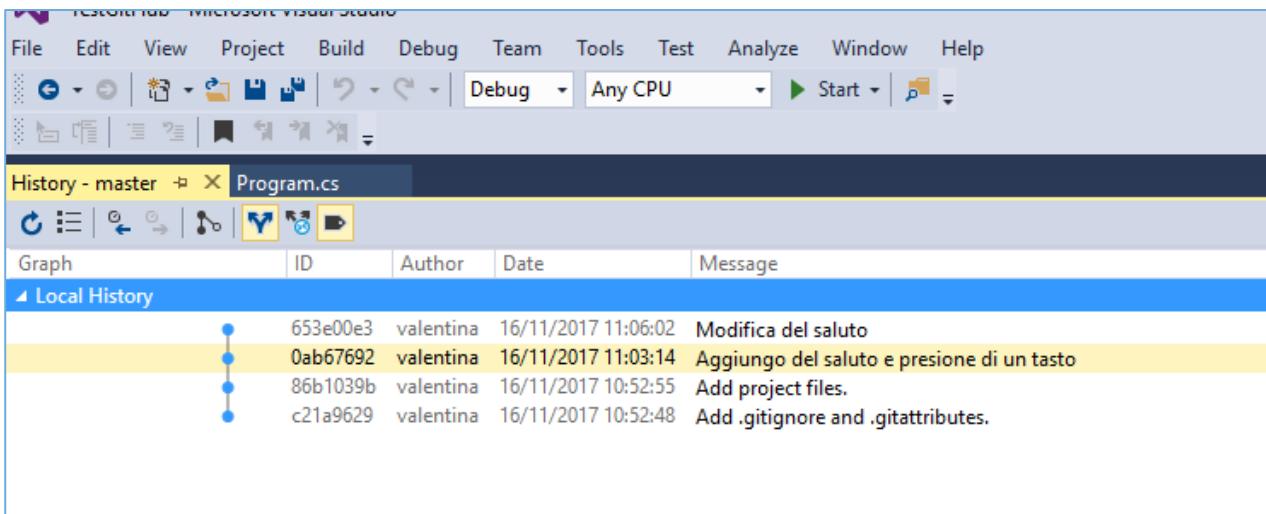
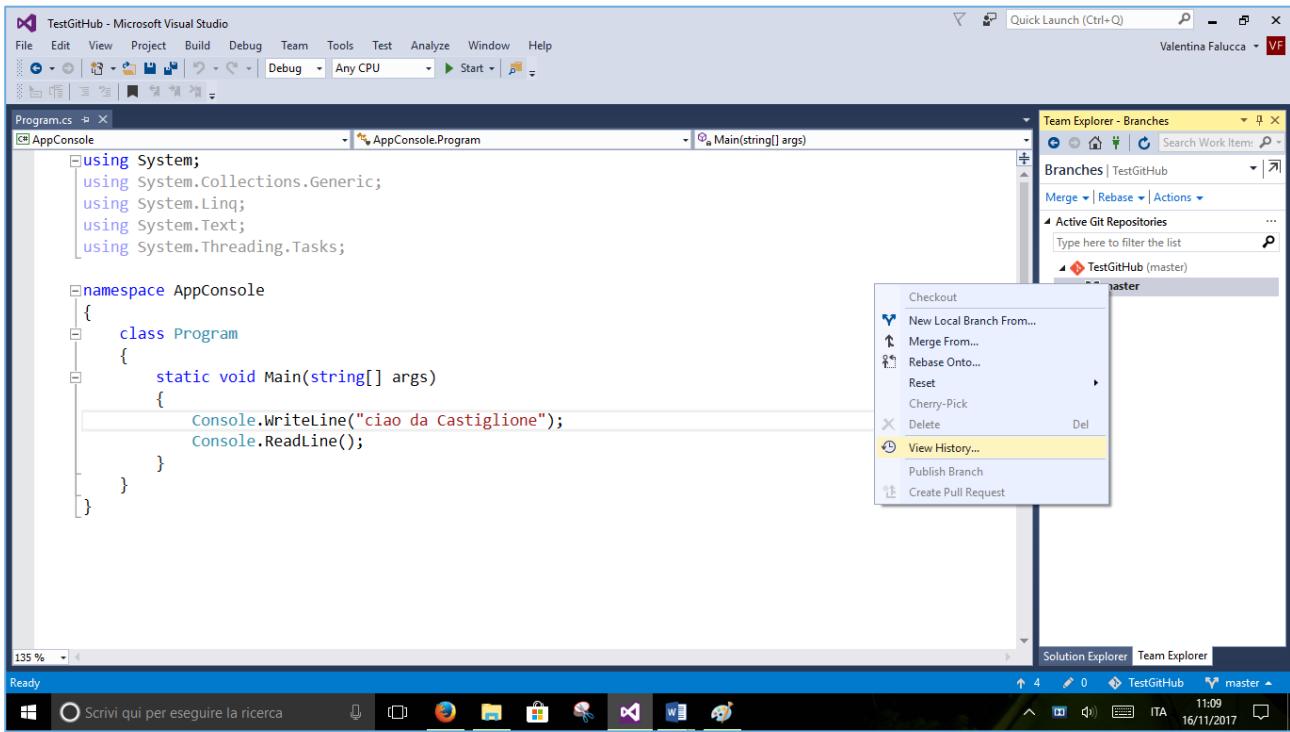
Ecco come vengono confrontate le due versioni:



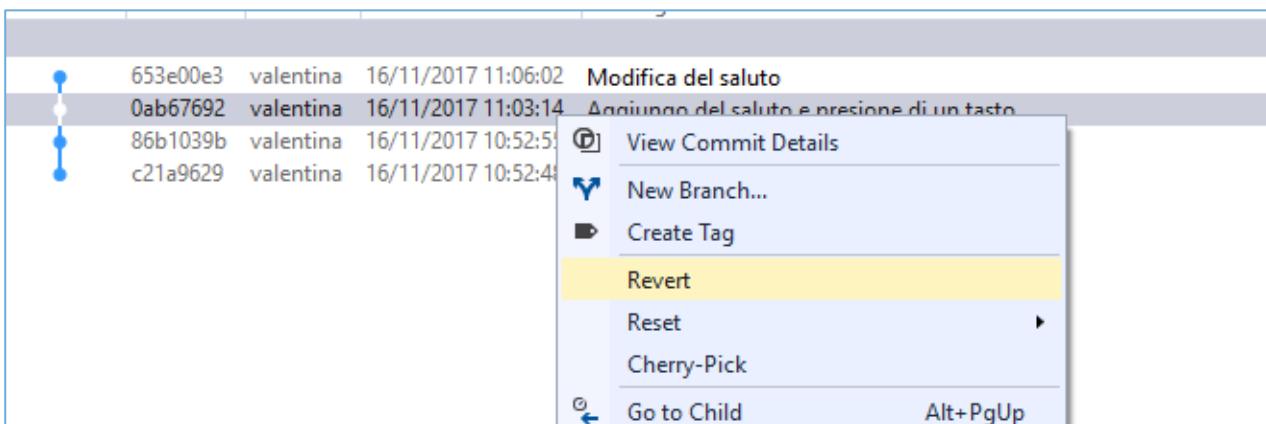
Si vada ad eseguire una commit alla seconda modifica e successivamente una terza modifica. Cliccando su Program.cs con tasto destro e poi su Undo si può tornare alla versione precedentemente salvata.



Adesso torniamo al menu Team Explorer e clicchiamo su Branches. Cliccando con tasto destro su "master" e poi su "View History" è possibile vedere tutta la storia delle versioni.



E' possibile tornare ad una versione precedente cliccando su "Revert".



Iscrizione GitHub.

Fino ad ora abbiamo effettuato il controllo di versione solo in locale. Per poter creare repository on line è necessario iscriversi a GitHub.

Dal sito <https://github.com>, in 3 semplici passi è possibile creare un account personale.

Join GitHub
The best way to design, build, and ship software.

Step 1: Create personal account	Step 2: Choose your plan
------------------------------------	-----------------------------

Create your personal account

Username

This will be your username. You can add the name of your organization later.

Email address

We'll occasionally send updates about your account to this inbox. We'll never share your email address with anyone.

Password

Use at least one lowercase letter, one numeral, and seven characters.

By clicking on "Create an account" below, you are agreeing to the [Terms of Service](#) and the [Privacy Policy](#).

Create an account

Terminata l'iscrizione è possibile iniziare a creare progetti!

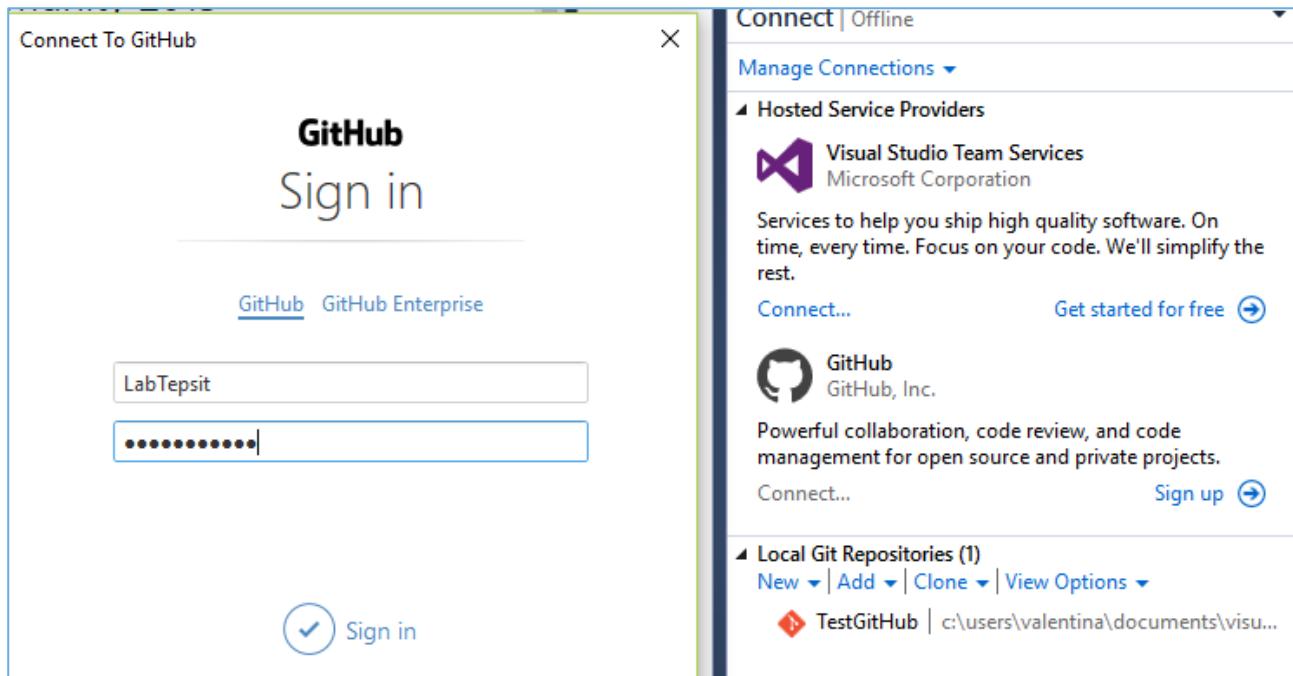
Learn Git and GitHub without any code!

Using the Hello World guide, you'll create a repository, start a branch, write comments, and open a pull request.

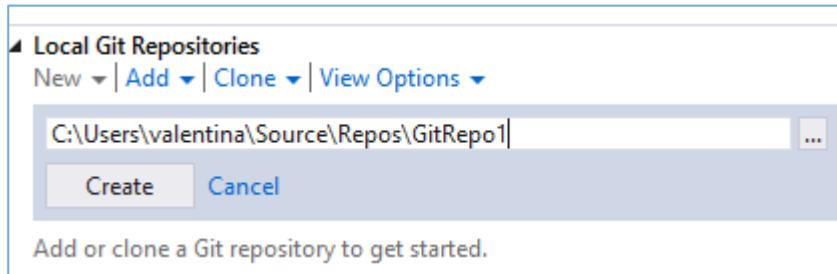
Read the guide **Start a project**

Ma prima di procedere vediamo come Visual Studio permette l'integrazione con sistema GitHub.

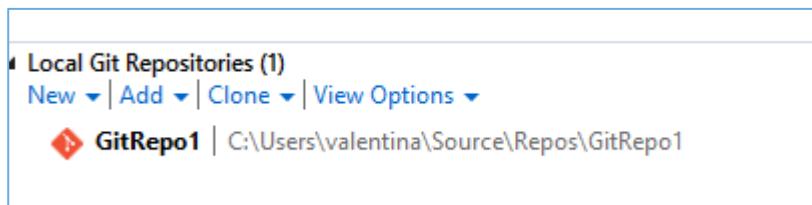
Da Team Explorer è possibile effettuare la conessione a GitHub con lo stesso account precedentemente creato. In Laboratorio, ogni volta che si entra in Visual Studio, è consigliabile cliccare Sign up e successivamente rieffettuare la connessione.



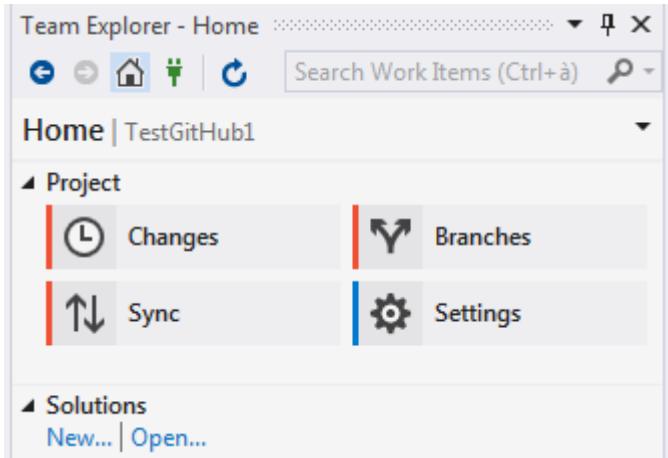
Cliccando su "New" creiamo in locale una nuova Repository denominata GitRepo1.



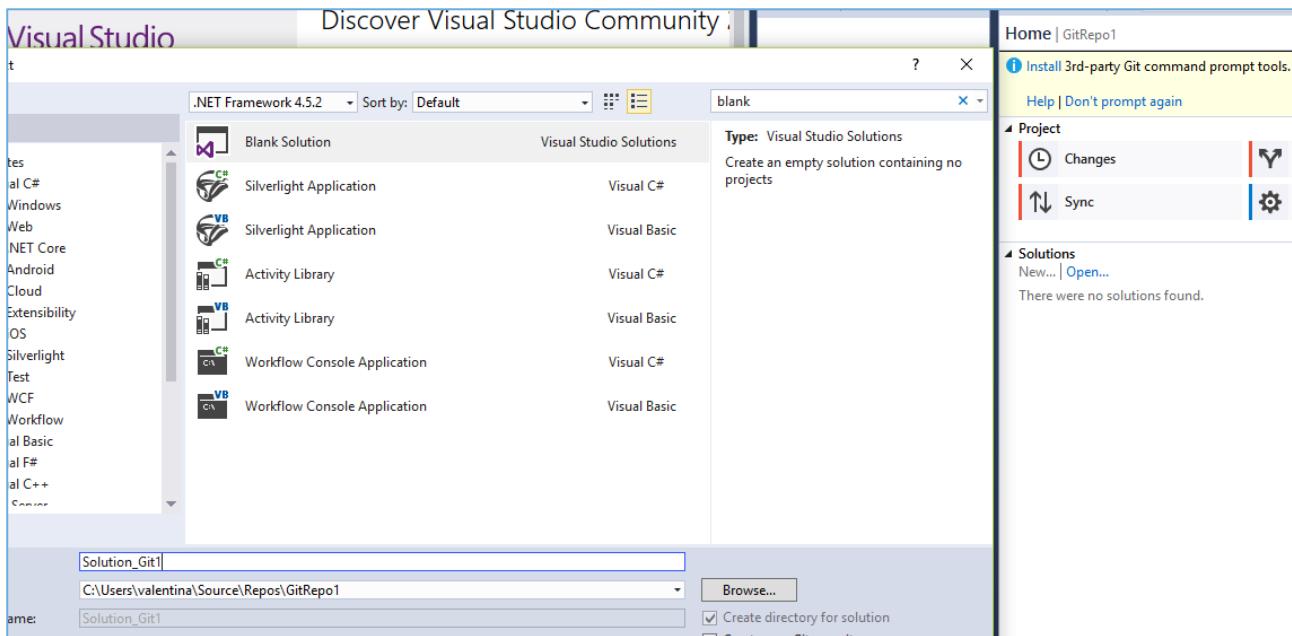
La schermata che si visualizzerà sarà la seguente:



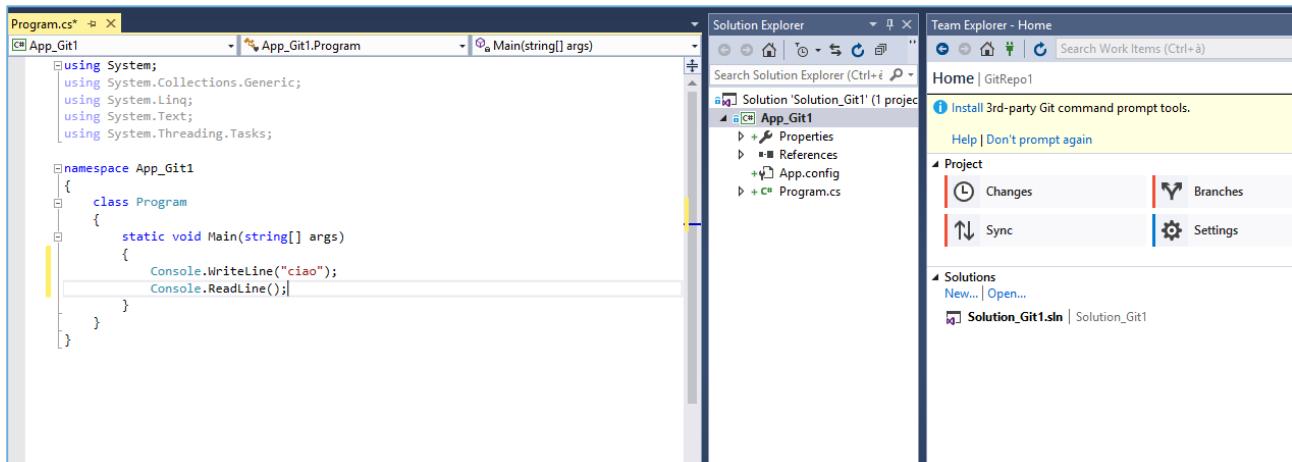
Con un doppio click su GitRepo1 si aprirà la seguente finestra:



Cliccare su “New” per creare una nuova Blank Solution (in questo esempio chiamata Solution_Git1).



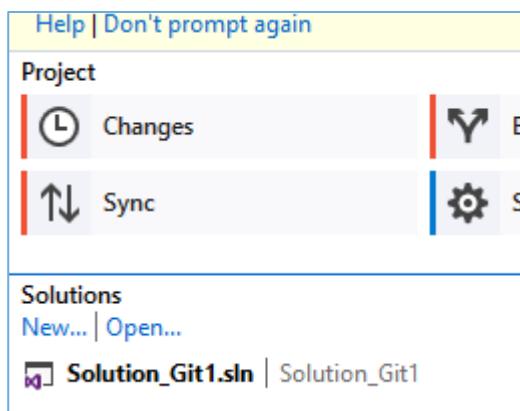
Si aggiunga poi alla Solution un progetto Console (chiamato App_Console1) ed nel file Program.cs scrivere il codice in figura:



A questo punto è possibile sincronizzare la repository locale al proprio profilo GitHub.

Tecnologie e progettazione di sistemi informatici e di telecomunicazioni
a.s. 2017-2018

Cliccare sull'icona che rappresenta una cassetta, cliccare su Sync e poi su Publish to GitHub.



Team Explorer - Synchronization

Search Work Items (Ctrl+à)

Publish | GitRepo1

Backup and share your code. Publish it to a Git service.

▲ Publish to Visual Studio Team Services

 Team Services
Microsoft Corporation
Free unlimited Git repos, code review, work items, build, and more. [Learn more](#)

▲ Publish to GitHub

 GitHub
GitHub, Inc
Powerful collaboration, code review, and code management for open source and private projects.

▲ Publish to Remote Repository

There is no remote configured for this local repository. Establish the remote by publishing to the URL of an existing empty repository.

E' necessario aggiungere una descrizione alla Repository che si vuole pubblicare e, così facendo, solo successivamente è possibile cliccare su Publish.

Backup and share your code. Publish it to a Git service.

▲ Publish to Visual Studio Team Services

 **Team Services**
Microsoft Corporation
Free unlimited Git repos, code review, work items, build, and more. [Learn more](#)

[Publish Git Repo](#)

▲ Publish to GitHub

 This repository does not have a remote. Fill out the form to publish it to GitHub.

GitHub

LabTepsit

GitRepo1

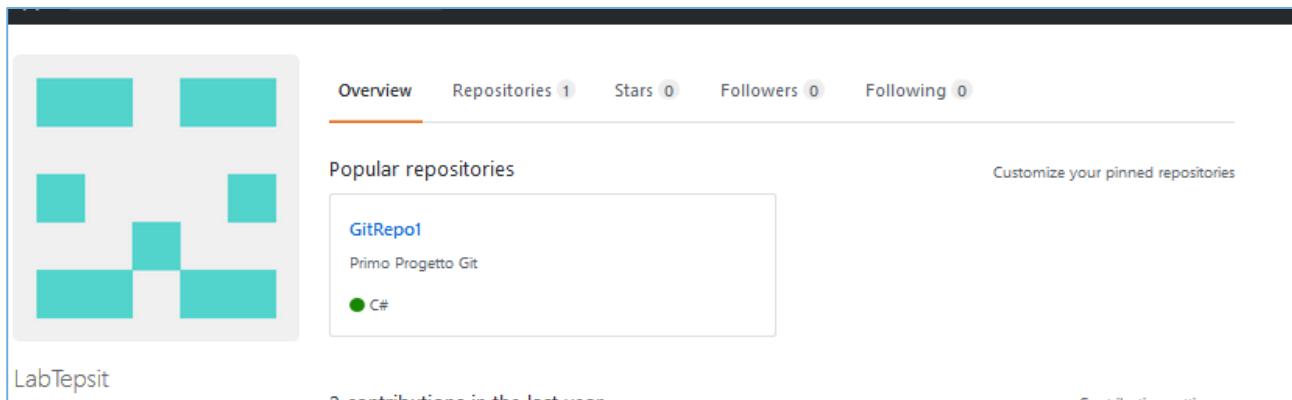
Primo progetto Git

Private Repository

[Publish](#)

▲ Publish to Remote Repository

Se tutto è andato a buon fine, accedendo dal sito GitHub al proprio profilo è possibile verificare l'effettiva sincronizzazione del progetto locale in remoto.



The screenshot shows a GitHub user profile. On the left, there is a placeholder image for a profile picture. Below it, the user's name "LabTepsit" is displayed. The main area is the "Overview" tab, which includes navigation links for "Overview", "Repositories 1", "Stars 0", "Followers 0", and "Following 0". Below these links, a section titled "Popular repositories" lists "GitRepo1" with the description "Primo Progetto Git" and the language "C#". To the right of this list, there is a link "Customize your pinned repositories".

Cliccando su **GitRepo1** è possibile ritrovare tutti file della Soluzione creata in locale.

LabTepsit / GitRepo1

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights Settings

Primo Progetto Git Edit

Add topics

2 commits 1 branch 0 releases 0 contributors

Branch: master New pull request Create new file Upload files Find file Clone or download

Valentina Falucca Commit1 Latest commit d53d609 7 minutes ago

Solution_Git1 Commit1 7 minutes ago

.gitattributes Prima commit 17 minutes ago

.gitignore Prima commit 17 minutes ago

Help people interested in this repository understand your project by adding a README. Add a README

Nel caso in cui la Solution non sia stata pubblicata, nella finestra Team Explorer, si visualizzerà il seguente messaggio:

Team Explorer - Synchronization

Synchronization | NewRepo

⚠️ Unable to push because the current branch does not track a remote branch. Publish the branch to push changes to the remote.

Branch: master

Sync | Fetch | Pull | Push | Actions

Per risolvere il problema è sufficiente cliccare su Publish nella sezione "Outgoing Commits".

Team Explorer - Synchronization

Synchronization | NewRepo

⚠️ Unable to push because the current branch does not track a remote branch. Publish the branch to push changes to the remote.

Branch: master

Sync | Fetch | Pull | Push | Actions

Incoming Commits

Fetch | Pull

The current branch does not track a remote branch.

Outgoing Commits

Push | Publish

The current branch does not track a remote branch. Publish the branch to push changes to the origin remote and set the upstream branch. [Learn more](#).

A questo punto torniamo in locale e effettuiamo una modifica al codice.

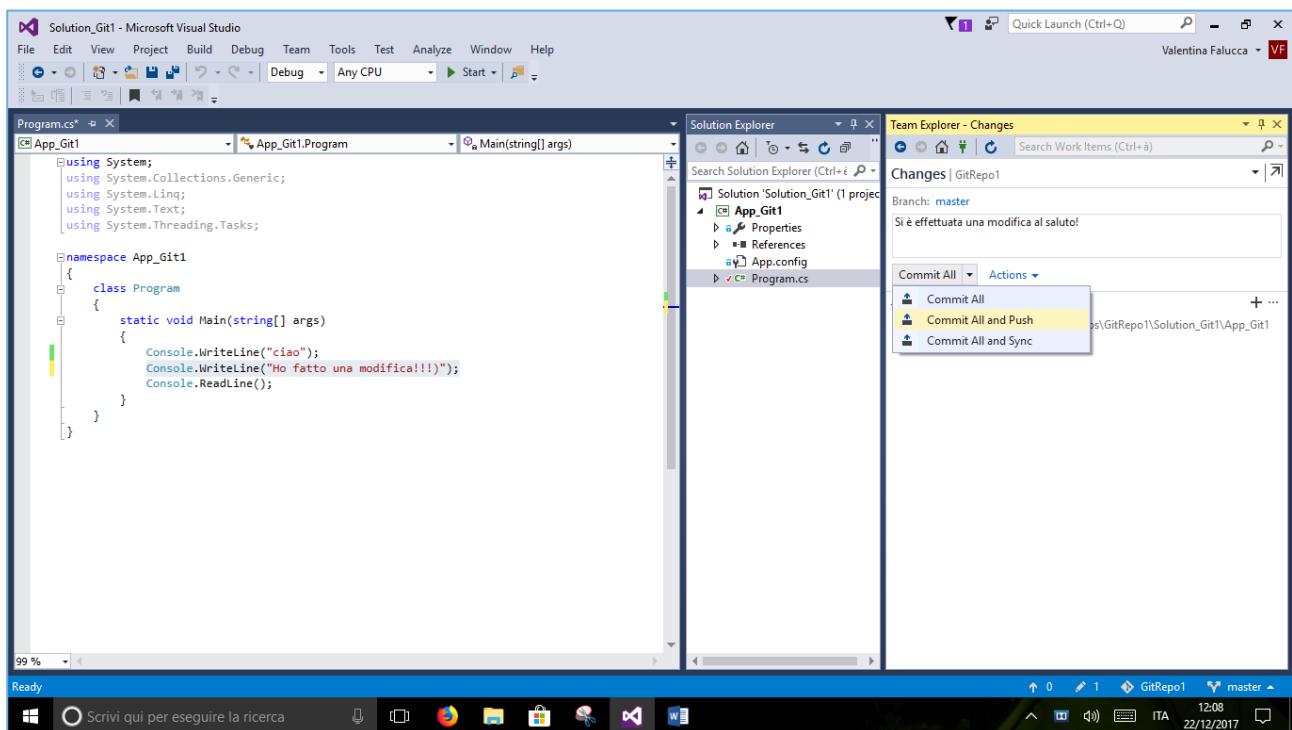
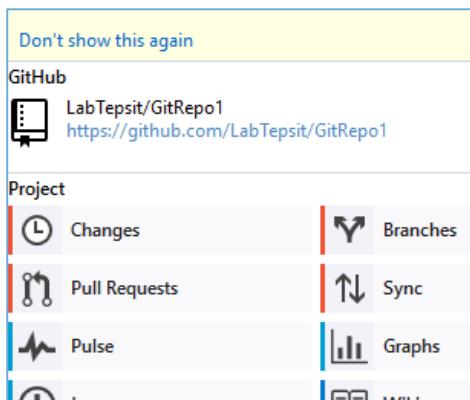
```

using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace App_Git1
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("ciao");
            Console.WriteLine("Ho fatto una modifica!!!");
            Console.ReadLine();
        }
    }
}

```

Cliccare su Changes, descrivere le modifiche effettuate e poi "Commit and Push".



On Line saranno immediatamente visualizzate le modifiche effettuate.

Primo Progetto Git

Add topics

5 commits 1 branch 0 releases 0 contributors

Branch: master ▾ New pull request Create new file Upload files Find file Clone or download ▾

Valentina Falucca Aggiungo Console.WriteLine("Ho fatto una modifica!!!!"); Latest commit e4170b a minute ago

Solution_Git1 Aggiungo Console.WriteLine("Ho fatto una modifica!!!!"); 43 seconds ago

.gitattributes Prima commit 33 minutes ago

.gitignore Prima commit 33 minutes ago

Help people interested in this repository understand your project by adding a README.

Add a README

Clonare Repository in locale.

Una repository può essere creata on line e clonata in Visual Studio. Per far ciò, dalla propria pagina Web, nella sezione Repository, è necessario cliccare sul tasto New.

GitRepo2OnLine Pull requests Issues Marketplace Explore + New

Overview Repositories 1 Stars 0 Followers 0 Following 0

Search repositories... Type: All Language: All New

GitRepo1
Primo Progetto Git
C# Updated 2 hours ago

LabTepsit Add a bio Edit profile

Al momento della creazione della Repository è possibile scegliere anche il tipo di Licenza e creare una file ReadMe che conterrà la descrizione e le indicazioni sul funzionamento del progetto.

Create a new repository

A repository contains all the files for your project, including the revision history.

Owner **Repository name**

 LabTepsit ▾ / GitRepo2 ✓

Great repository names are short and memorable. Need inspiration? How about [glowing-disco](#).

Description (optional)

 **Public**
Anyone can see this repository. You choose who can commit.

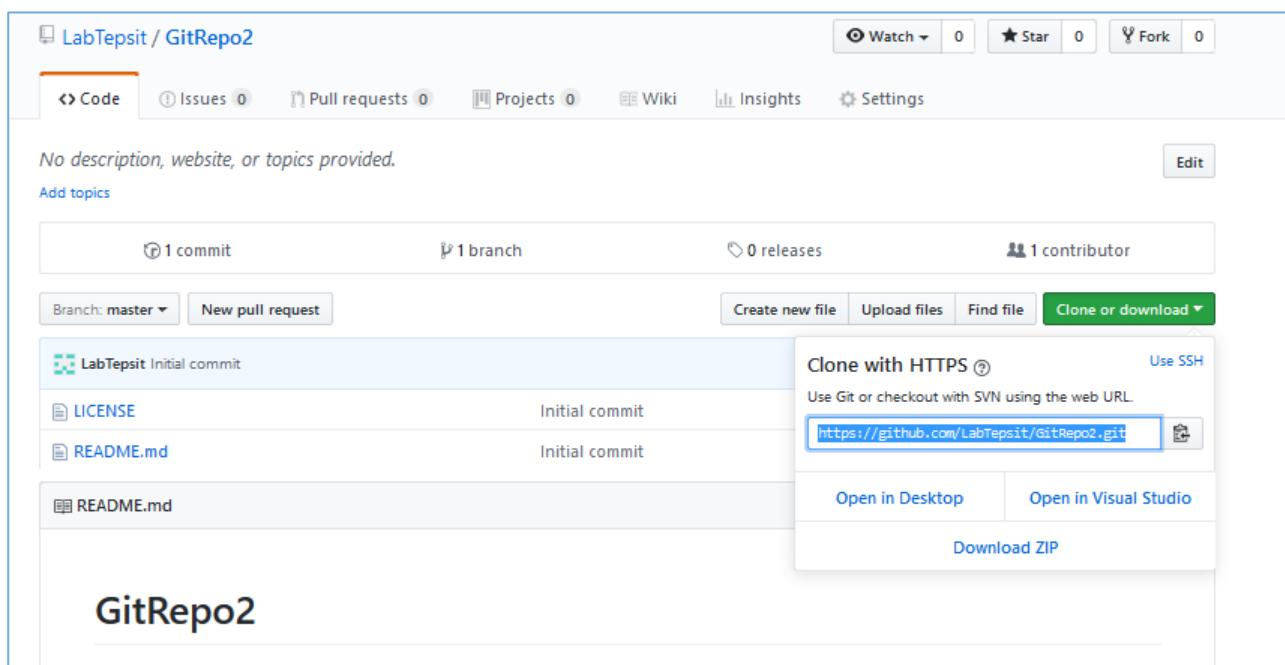
 **Private**
You choose who can see and commit to this repository.

Initialize this repository with a README
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: **None** ▾ | Add a license: **GNU General Public License v3.0** ▾ 

Create repository

A questo punto è sufficiente cliccare sul tasto Cone and Download e copiare il link mostrato.



LabTepsit / GitRepo2

Watch 0 ★ Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights Settings

No description, website, or topics provided. Edit

Add topics

1 commit 1 branch 0 releases 1 contributor

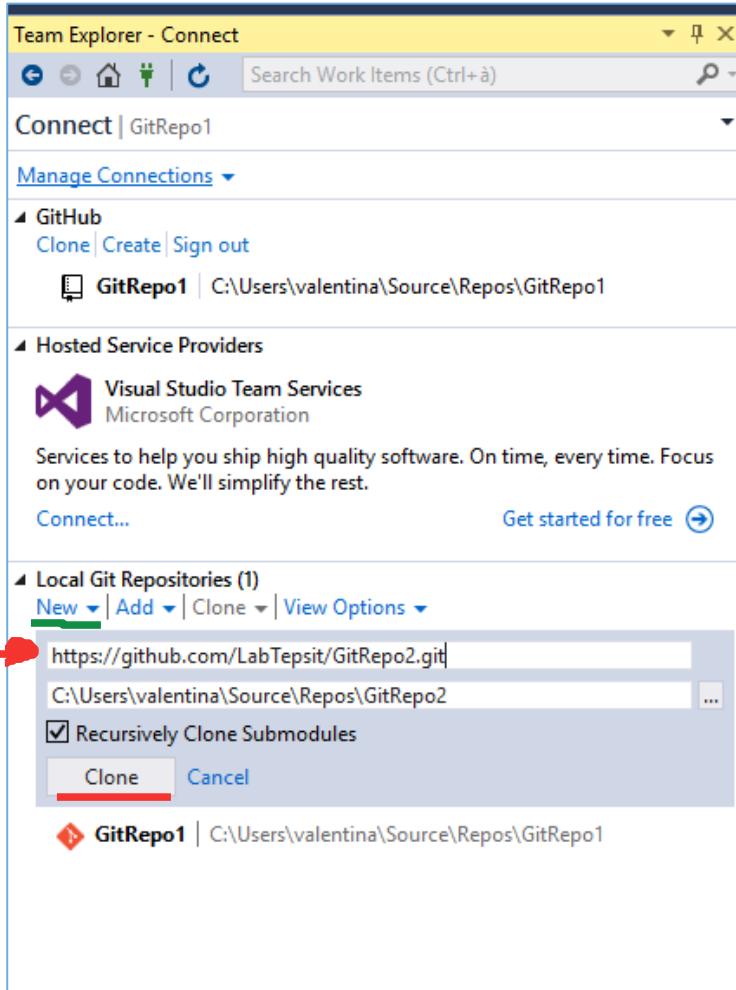
Branch: master ▾ New pull request Create new file Upload files Find file Clone or download ▾

LabTepsit Initial commit
LICENSE Initial commit
README.md Initial commit
README.md

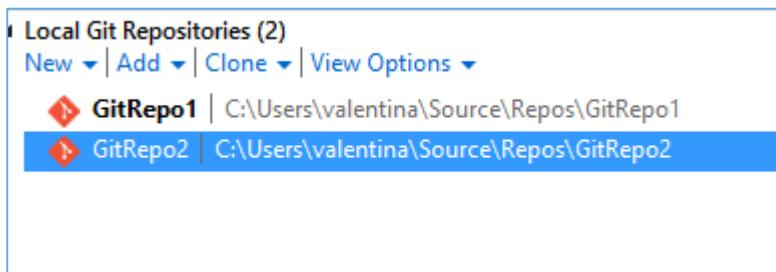
Clone with HTTPS ⓘ Use SSH
Use Git or checkout with SVN using the web URL.
<https://github.com/LabTepsit/GitRepo2.git> ⌂
Open in Desktop Open in Visual Studio
Download ZIP

GitRepo2

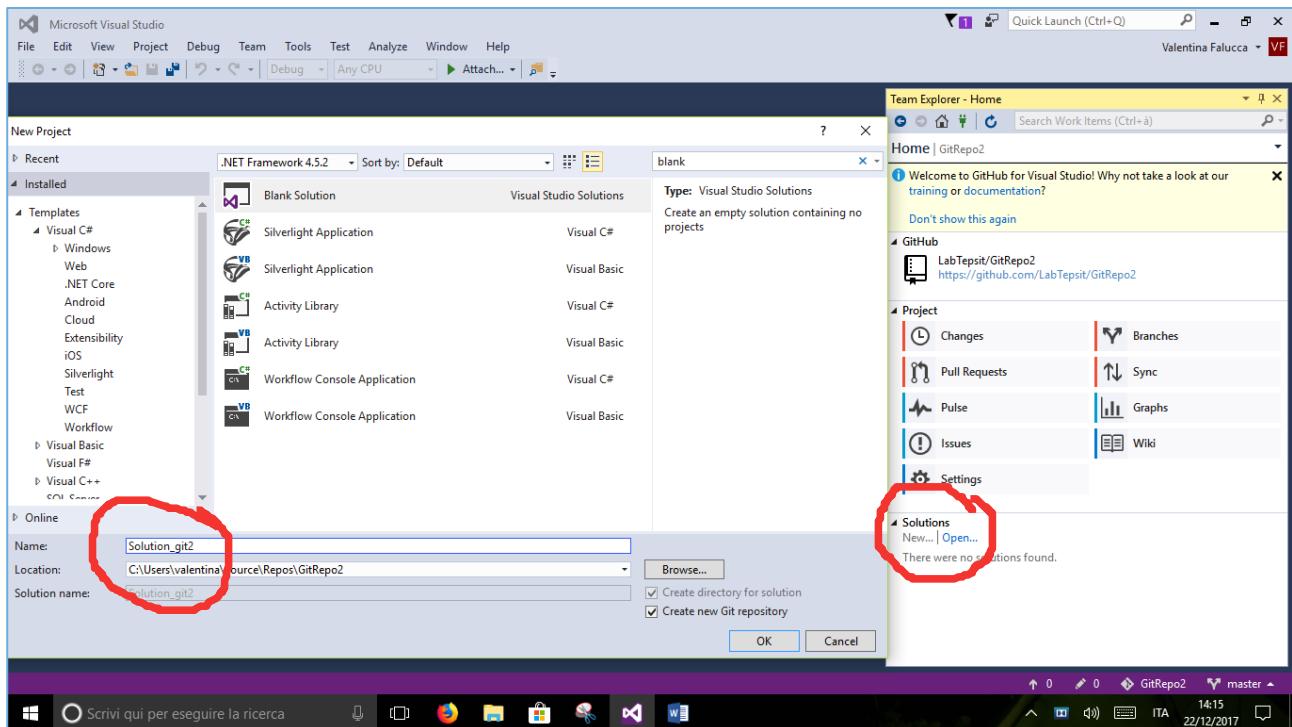
Torniamo a Visual Studio, dalla finestra Team Explorer, nella sezione Local Git Repositories, cliccare su Clone ed incollare il link copiato.



Ci fermata l'azione col il tasto Clone, anche in Visual studio si avrà la stessa Repository creata nella pagina personale di GitHub.



Con un doppio Click su GitRepo2 si potrà procedere con al creazione della Solution con l'aggiunta di progetti e librerie.



Aggiungiamo alla Solution una Libreria ed un progetto Console e successivamente cliccando su Changes effettuare il Commit e Push.

Le modifiche saranno subito visibili OnLine

The screenshot shows the GitHub repository page for 'LabTepsit / GitRepo2'. At the top, there are navigation links for Code, Issues (0), Pull requests (0), Projects (0), Wiki, Insights, and Settings. Below this, a summary bar shows 2 commits, 1 branch, 0 releases, 1 contributor, and the license as GPL-3.0. A dropdown menu shows the branch is set to 'master'. There are buttons for New pull request, Create new file, Upload files, Find file, and Clone or download. The main area displays the commit history:

- Valentina Falucca Aggiunta di Libreria e progetto console (Latest commit d502b2e 2 minutes ago)
- Solution_git2 Aggiunta di Libreria e progetto console (2 minutes ago)
- .gitattributes Aggiunta di Libreria e progetto console (2 minutes ago)
- .gitignore Aggiunta di Libreria e progetto console (2 minutes ago)
- LICENSE Initial commit (27 minutes ago)
- README.md Initial commit (27 minutes ago)
- README.md (empty commit) (27 minutes ago)

Branches

Qualsiasi software, una volta effettuata una Release Stabile, ha due vite parallele:

1. Ha bisogno di essere corretto (MANUTENZIONE).
2. Ha necessità che vi siano aggiunte funzionalità (SVILUPPO).

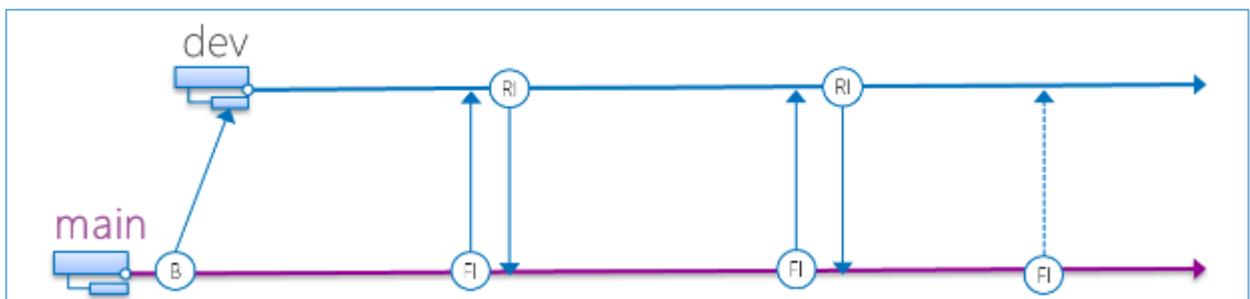
Main Only Strategy

Una possibile strategia da intraprendere per la gestione della manutenzione e dello sviluppo è quella che consiste nel fare entrambe le azioni nello stesso progetto, in gergo “nello stesso ramo” (Main Only Strategy). Tale strategia è però sconsigliata perché aggiungere funzionalità e contemporaneamente correggere vecchi errori potrebbe causare ulteriori problemi.



Development Isolation

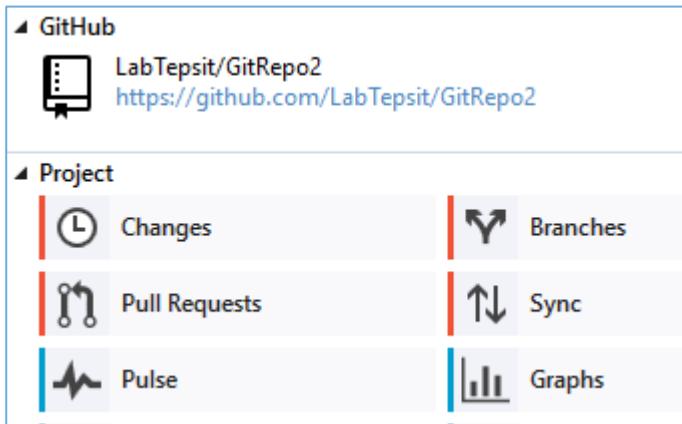
Si procede quindi con un altro tipo di strategia: duplicare il progetto in due rami (branches); nella copia si aggiungono funzionalità, nel progetto principale si effettuano le correzioni di errori. Si lavora pertanto in isolamento.



Si parla di Forward Integration (FI) quando le correzioni effettuate nel ramo principale vengono riporotate nel ramo di sviluppo, mentre si parla di Reverse Integration (RI) quando dal ramo di sviluppo si riportano le nuove funzionalità al ramo principale.

Vediamo come avviene la creazione di due rami in Visual studio.

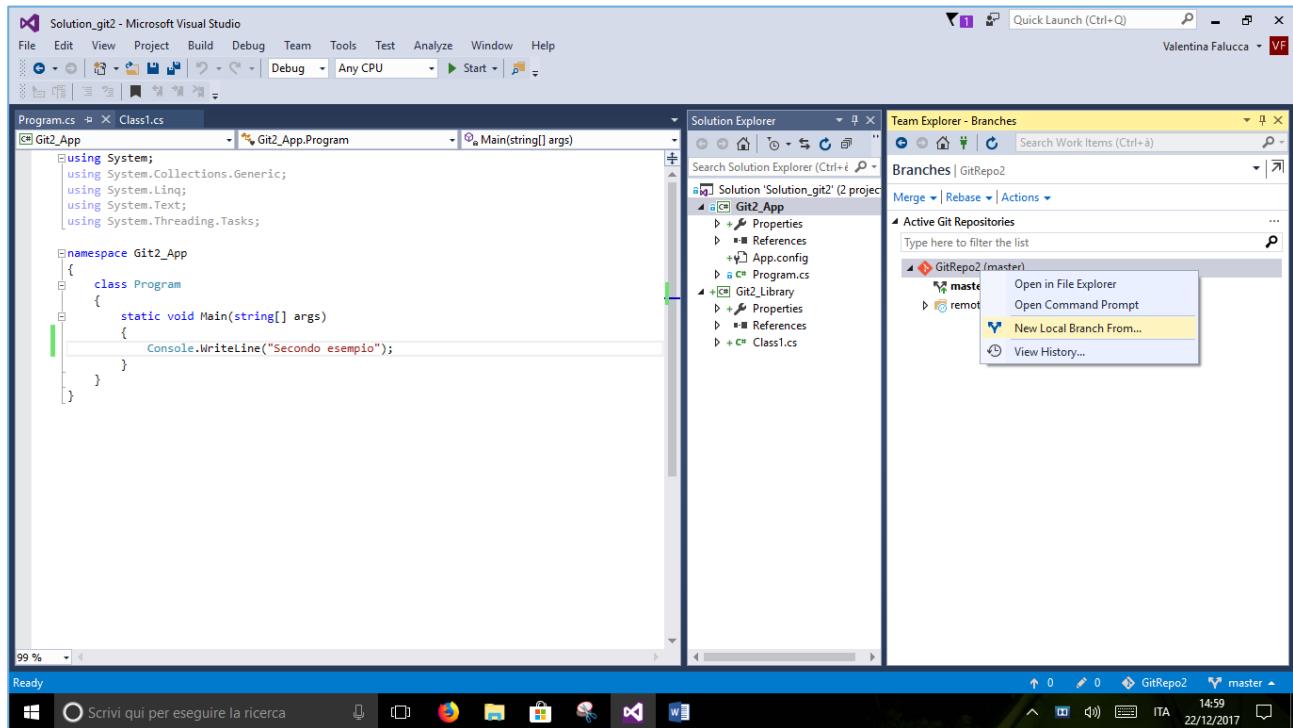
Dal menu selezionare Branches



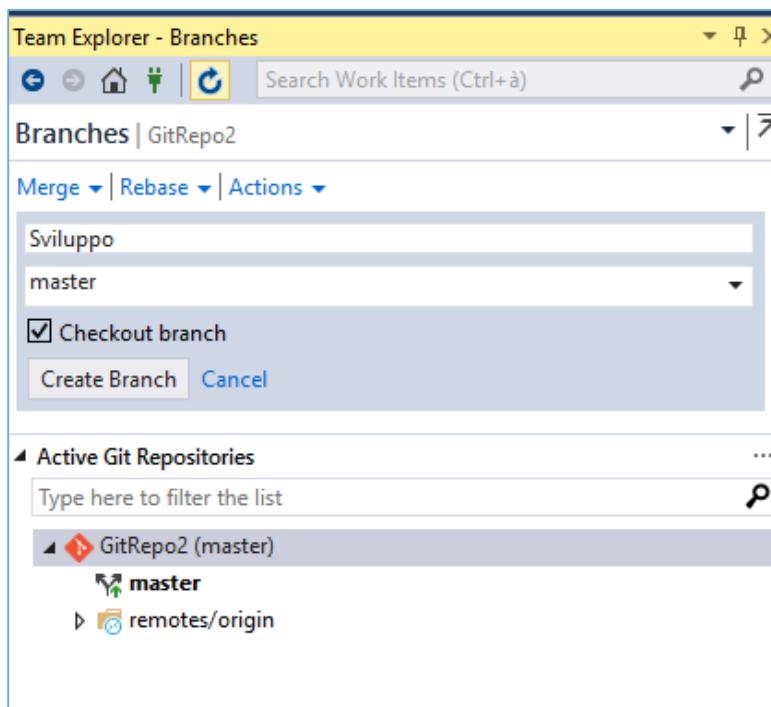
E' possibile effettuare diverse azioni dalla finestra branches:

1. Creare un nuovo ramo (New Branch)
2. Unire due rami in uno solo (Merge)
3. Integrare un ramo con le correzioni (Rebase)

Procediamo con la suddivisione del progetto principale in due branch cliccando con il tasto destro sul progetto su New Local Branch from...



Il nuovo ramo viene denominato Sviluppo.



Cliccando su Create Branch comparirà Sviluppo. Selezionando con il testo desto Sviluppo è possibile pubblicare il nuovo ramo.

Le modifiche saranno da subito visualizzate on line.

Selezionando il Ramo Sviluppo è possibile adesso aggiungere funzionalità e procedere con azioni di Commit ed Push.

Unit Test

Lo unit testing è una porzione di codice che serve a testare una singola funzionalità, cioè una porzione di codice (metodi, funzione, classi) che ha una singola responsabilità.

Il test viene svolto anche per le User Interface per validare l'interazione con l'utente, in tal caso lo unit test prende il nome di UI testing. La metodologia utilizzata è chiamata Test Driven Development che si basa su 3 fasi ripetute: Red-Green-Refactor.

Fase Red: Si scrive il test. Regola del TDD è “mai scrivere codice di produzione senza aver scritto test rosso”

Fase Green: Si scrive il minimo codice possibile che faccia funzionare i test.

Fase Refactor: si ottimizza il codice.

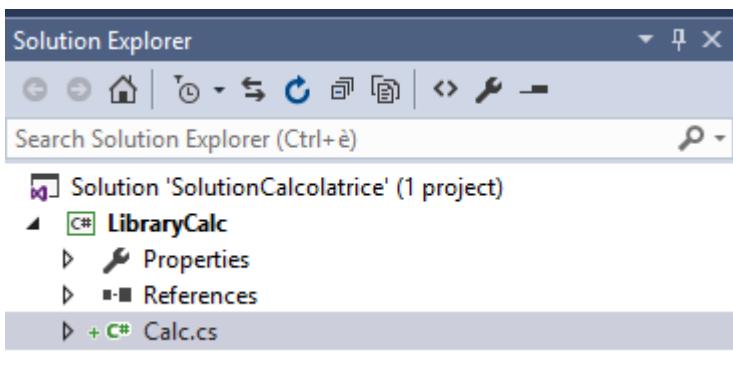
Unit Test in Visual Studio

Andiamo a creare un Blank Solution con una Library ed una classe.

Scopo dell'esercizio sarà quello di testare in metodo che calcola la somma tra due numeri.

Si sottolinea l'importanza di assegnare nomi significativi a qualsiasi elemento del progetto creato, dalla solution alle variabili o ai metodi.

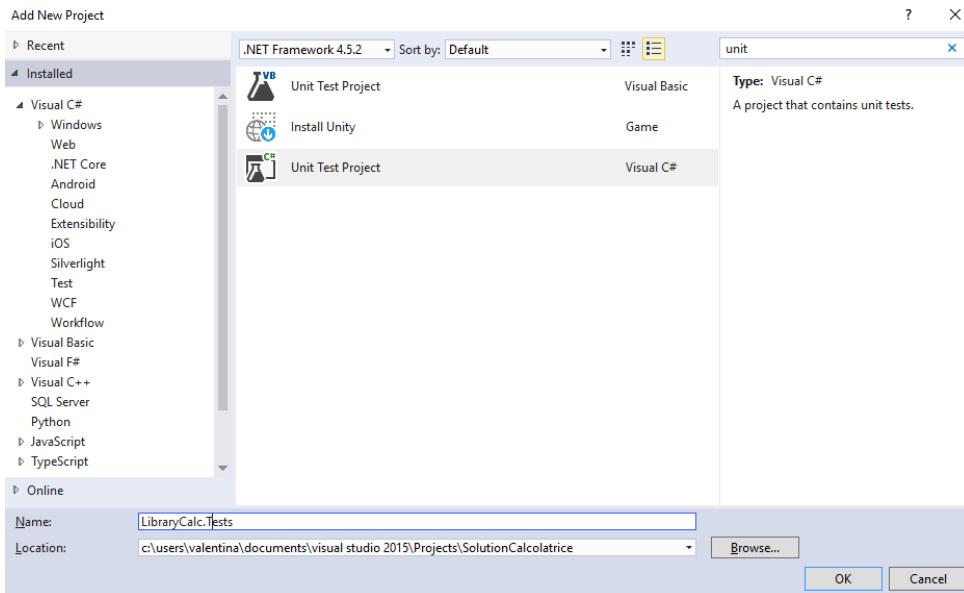
Ecco un esempio di come deve essere strutturata la solution.



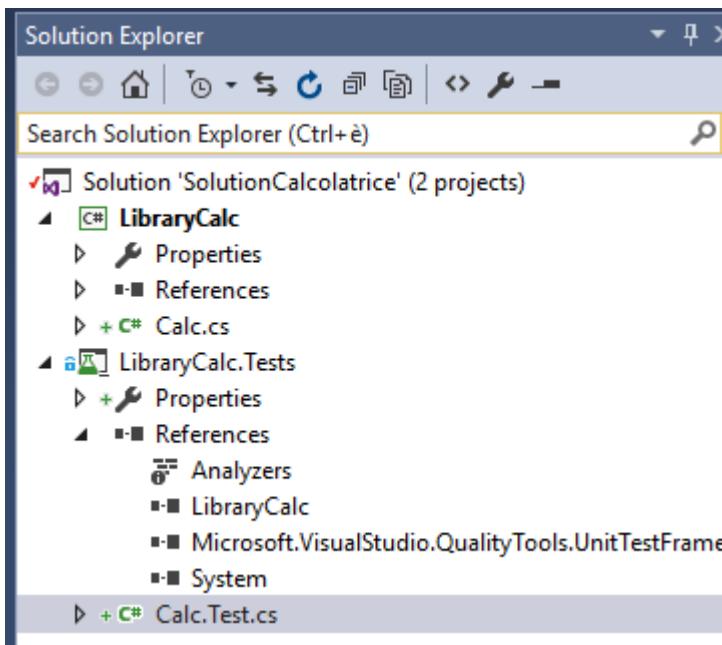
Per fare i test occorre creare un nuovo tipo di Progetto alla Solution (Add-NewProject-UnitTest).

Ogni classe dell'applicazione avrà una corrispondente classe nel TestProject.

E' buona pratica assegnare al progetto UnitTest il nome della Library da testare con l'aggiunta della parola Tests.



La soluzione sarà così strutturata:



Nella classe Calc andiamo a scrivere la firma del Metodo:

```
namespace LibraryCalc
{
    public static class Calc
    {

        public static double Somma (double a, double b)
        {
            double ris = 0;

            return ris;
        }
    }
}
```

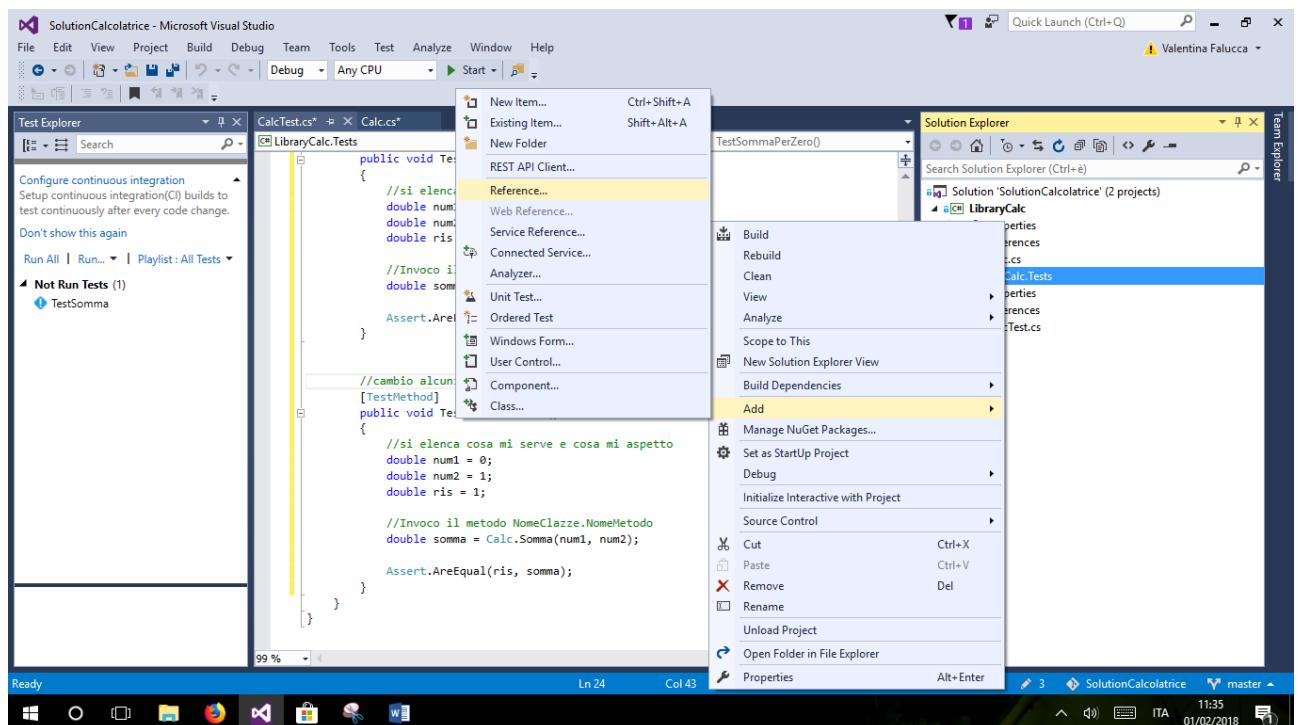
Prima di scrivere il metodo si va a studiare i possibili risultati che deve restituire il metodo.

a, b	SOMMA	DIVISIONE
0,0	0	IMPOSSIBILE
5,0	5	IMPOSSIBILE
7,2	9	3,5
0,1	1	0

Passiamo a scrivere i test.

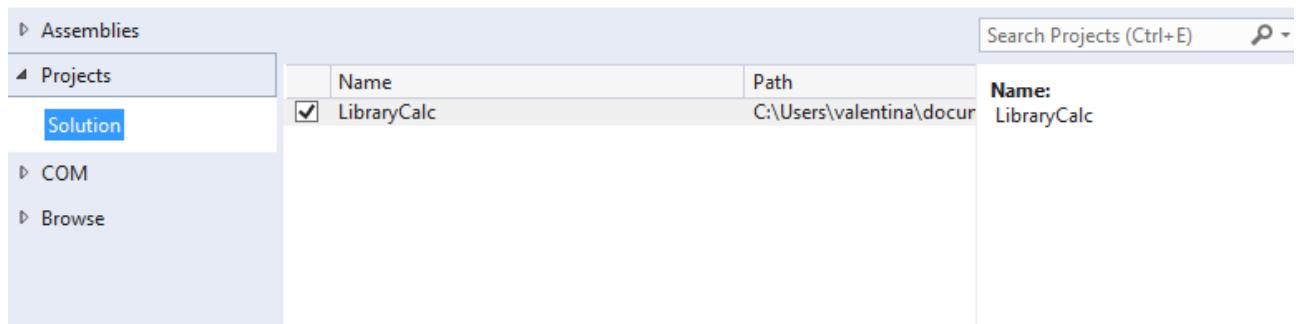
Prima di tutto è necessario aggiungere la reference della LibraryCalc alla LibraryCalc.Tests seguendo la seguente procedura: tasto desto sulla libreria, add Reference

1.



2. Spuntare la Libreria a cui si vuole aggiungere il riferimento:

Reference Manager - LibraryCalc.Tests



3. Aggiungere la using che fa riferimento alla Library

```

CalcTest.cs*  X  Calc.cs*
C# LibraryCalc.Tests  LibraryCalc.Tests.CalcTest

using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using LibraryCalc;

namespace LibraryCalc.Tests
{
    [TestClass]
    public class CalcTest
    {
        [TestMethod]
        public void TestSomma()
        {
            //si elenca cosa mi serve e cosa mi aspett

```

A questo punto si passa a scrivere il codice all'interno della classe.

Un metodo si testa seguendo i seguenti passi:

1. Si elenca cosa serve per il metodo e cosa mi aspetto che restituisca.
2. Si invoca il metodo usando la seguente sintassi: NomeClasse.nomeMetodo e passando i giusti paramtri.
3. Si controlla se quanto ci si aspetta è uguale a quanto restituisce il metodo.
4. Si ripetono i punti 1-2-3 scrivendo un altro Test ma cambiando i valori iniziali.

```

namespace LibraryCalc.Tests
{
    [TestClass]
    public class CalcTest
    {
        [TestMethod]
        public void TestSomma()
        {
            //si elenca cosa mi serve e cosa mi aspetto
            double num1 = 7;
            double num2 = 2;
            double ris = 9;

            //Invoco il metodo NomeClasse.NomeMetodo
            double somma= Calc.Somma(num1, num2);

            Assert.AreEqual(ris, somma);
        }

        //cambio alcuni valori di partenza
        [TestMethod]
        public void TestSommaPerZero()
        {
            //si elenca cosa mi serve e cosa mi aspetto
            double num1 = 0;
            double num2 = 1;
            double ris = 1;

            //Invoco il metodo NomeClasse.NomeMetodo
            double somma = Calc.Somma(num1, num2);
        }
    }
}

```

```
        Assert.AreEqual(ris, somma);  
    }  
}
```

Esercizio: Completare i casi discussi per testare il metodo Somma e effettuare la Commit del progetto su GitHub.

Provediamo con i test per la divisione.

```
[TestMethod]
public void TestDivisione()
{
    //si elenca cosa mi serve e cosa mi aspetto
    double num1 = 7;
    double num2 = 2;
    double ris = 3.5;

    //Invoco il metodo NomeClasse.NomeMetodo
    double divisione = Calc.Divisione(num1, num2);

    Assert.AreEqual(ris, divisione);
}
```

E' necessario far attenzione alla divisione per zero!

In tal caso esiste una parola chiave `Nan` che permette di dire che il risultato non è un numero.

```
[TestMethod]
public void TestDivisionePerZero()
{
    //si elenca cosa mi serve e cosa mi aspetto
    double num1 = 1;
    double num2 = 0;
    double ris = double.NaN;

    //Invoco il metodo NomeClasse.NomeMetodo
    double divisione = Calc.Divisione(num1, num2);

    Assert.AreEqual(ris, divisione);
}
```

```
[TestMethod]
public void TestDivisione2Zeri()
{
    //si elenca cosa mi serve e cosa mi aspetto
    double num1 = 0;
    double num2 = 0;

    //0 diviso 0 lo rappresentiamo con un "not a Number".
    //Tutto quello che "non si può fare" non è un numero.
    double ris = double.NaN;

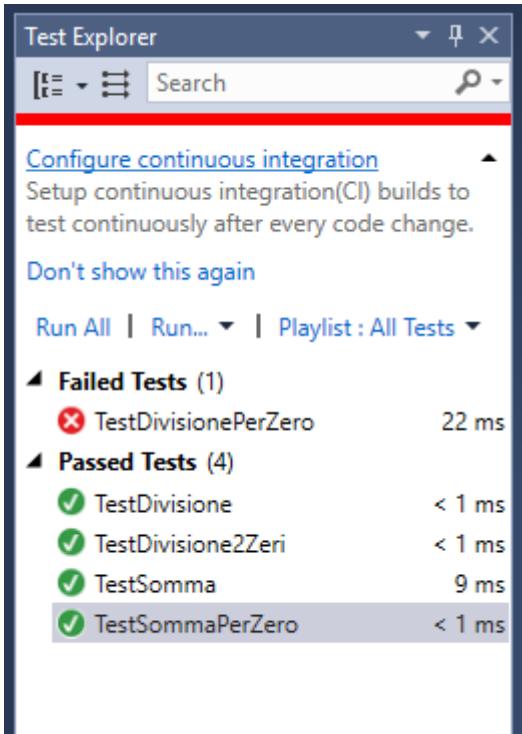
    //Invoco il metodo NomeClasse.NomeMetodo
    double divisione = Calc.Divisione(num1, num2);

    Assert.AreEqual(ris, divisione);
}
```

Scriviamo il metodo Divisione

```
public static double Divisione(double a, double b)
{
    double ris = 0;
    ris = a / b;
    return ris;
}
```

Eseguendo i test ci accorgiamo che un test non passa.



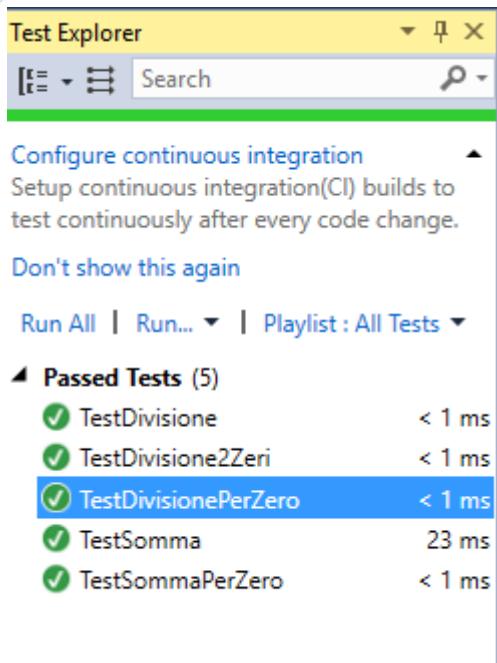
Il test è utile per modificare il metodo Divisione.

```
public static double Divisione(double a, double b)
{
    double ris = 0;
    if (b != 0)

        ris = a / b;
    else
        ris = double.NaN;
    return ris;
}
```

A questo punto passano tutti i test!

Procediamo per la divisione.



Esercizio: Trovare il massimo tra due numeri.

a, b	Max
5,2	5
3,4	4
7,7	7