by Javier Carrasco

Development environments
Block 1
Unit 8: Specific purpose environments

1.8.1. Development environments and IDEs

As you should know by now, a (*software*) **development environment** is a set of tools that let us automate some tasks, and give us support for analysis, design, coding, tests, validation and so on. There must be a high level of integration within these environments, so that these tasks can interoperate between them.

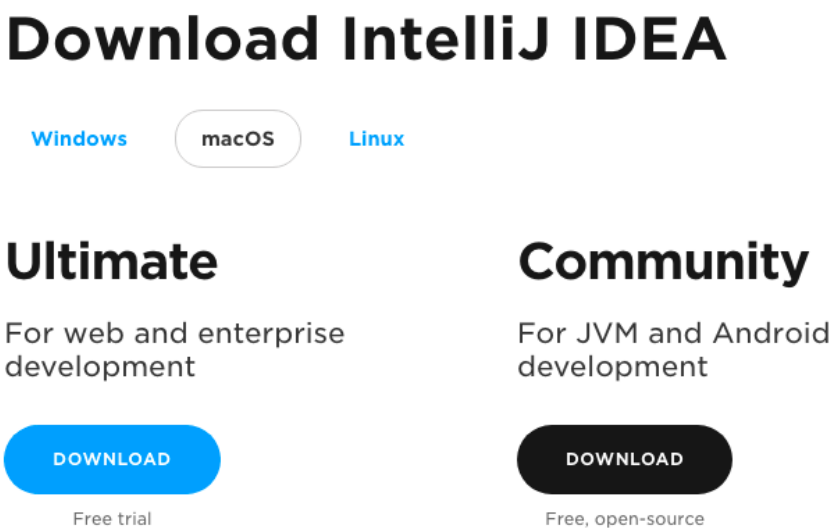
Among these development environments, we can find some general purpose environments, such as Geany or Visual Studio Code, that can cope with many different languages. Besides, there are other specific environments that are specially design for one language (or a reduced list of them). In this unit we are going to focus in this second group, since the first one has been explained in previous units.

To be more precise, we are going to focus on **IntelliJ IDEA**, since it is the IDE that we are going to use for most of the exercises of this module. Besides, we will see a quick overview of some other specific IDEs, such as Visual Studio.

1.8.2. IntelliJ IDEA

IntelliJ IDEA is a multi platform IDE developed by JetBrains. It was firstly released in 2001 as *IntelliJ*, and it was one of the first IDEs with advanced browsing and code refactoring.

It is available in two versions: *Community Edition* (free) and *Ultimate Edition* (commercial). The main difference between them can be found in the languages and version control systems supported. For instance, *Community* version does not allow PHP or Javascript. In our case, as we are going to work with Java Virtual Machine, we need to download the *Community* edition, which is free and open source. This is the download page.

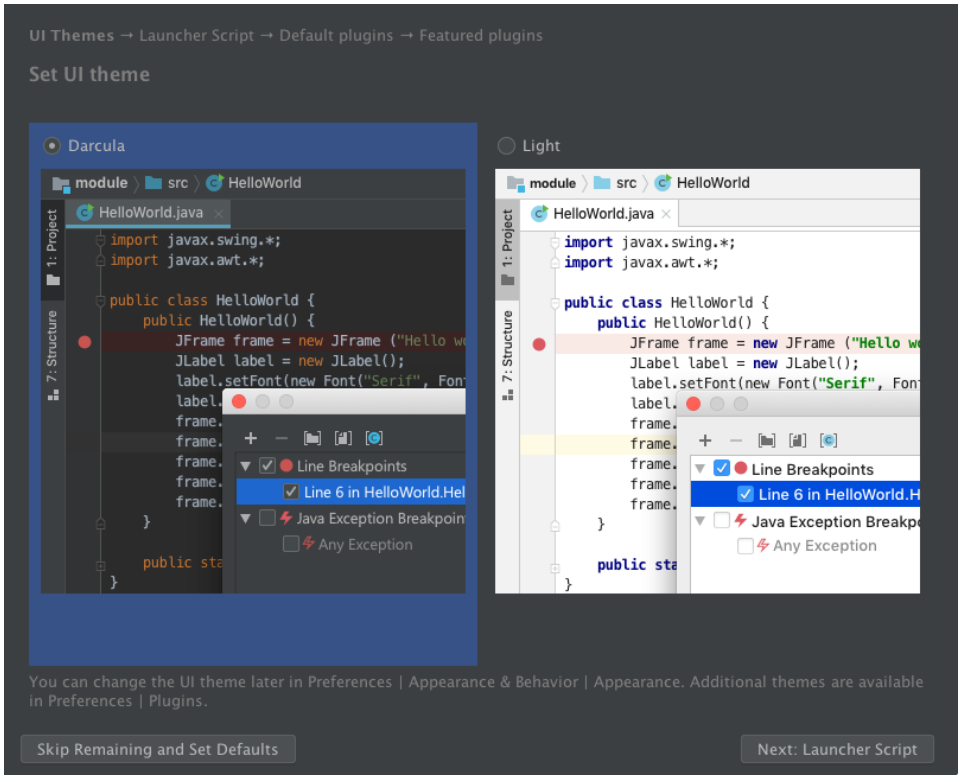


1.8.2.1. Installation and setup

Regarding the installation process:

- In **Windows**, we have a step-by-step wizard that guides us through all the installation process. We can choose the installation folder (or just leave the one set by default), and if we want to create a shortcut in the desktop.
- In **Mac OSX**, we have an installer that asks us to drag the application into the *Applications* folder.
- In **Linux**, we download a *tar.gz* file that we must unzip. Inside the main folder there is a *bin* subfolder. We must get into this folder from a terminal and run the command `./idea.sh` to launch the IDE. The first time we run it, it will create a shortcut somewhere in the applications menu, so we can launch IntelliJ using this shortcut from then on.

The first time that we run *IntelliJ*, it lets us import previous settings, if we had any previous version installed. If not, we can just choose "*Do not import settings*". Next, we can choose the UI theme...



Next, it lets us define a script to launch programs from command line, but we can skip this step and move to the next one, in which we can choose to install some additional plugins. We can also leave this step with its default settings, and start using *IntelliJ*.

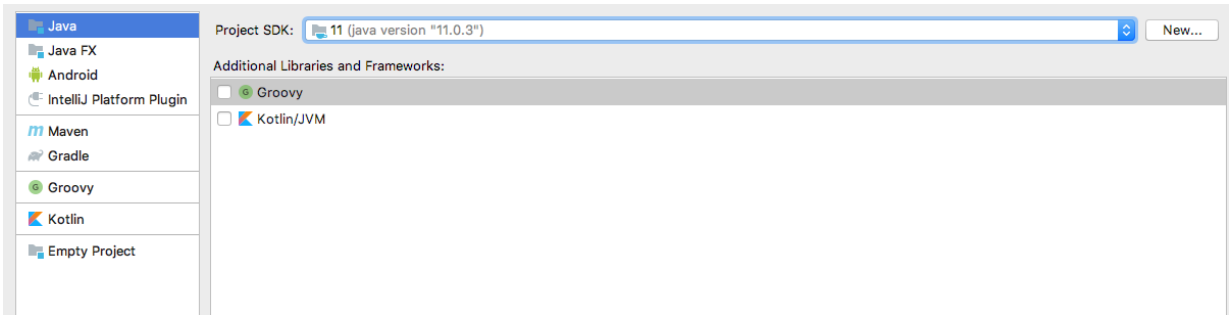
This is the welcome screen:

1.8.2.2. Creating Java projects

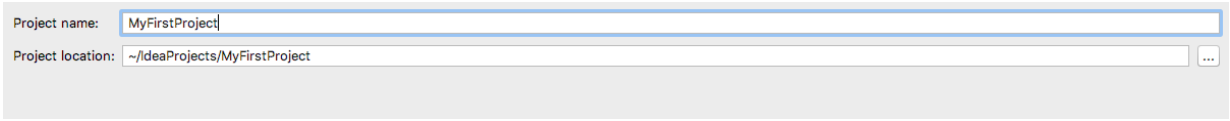
From the welcome screen, we can choose among:

- Creating a new project
- Importing an existing project (if we created it in another computer and want to use it in our current one).
- Open an existing project from our computer
- Check a project from a remote repository

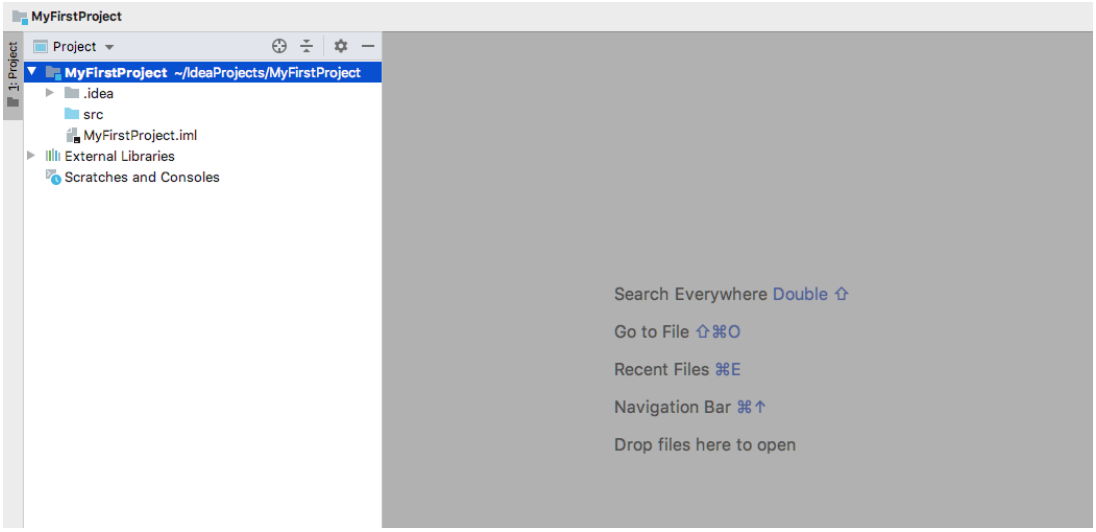
So, if we want to create a new project, we choose the first option *Create New Project*, and then specify that we want to create a Java project, from the left panel:



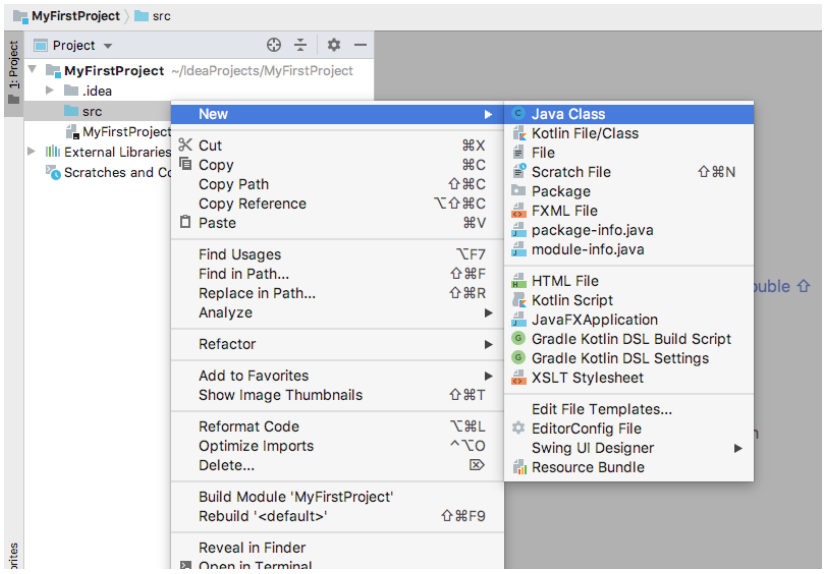
Next, we can choose a template for out project, but we can skip this step, and move to the next one, in which we must specify a project name and location (we can leave the default location if we want to):



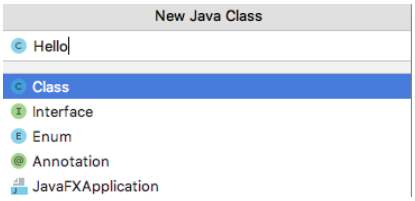
Then, we click on *Finish* and we will see our project. If we click on the project tab on the left, we can see the project folder structure, and create elements (source files) on it.



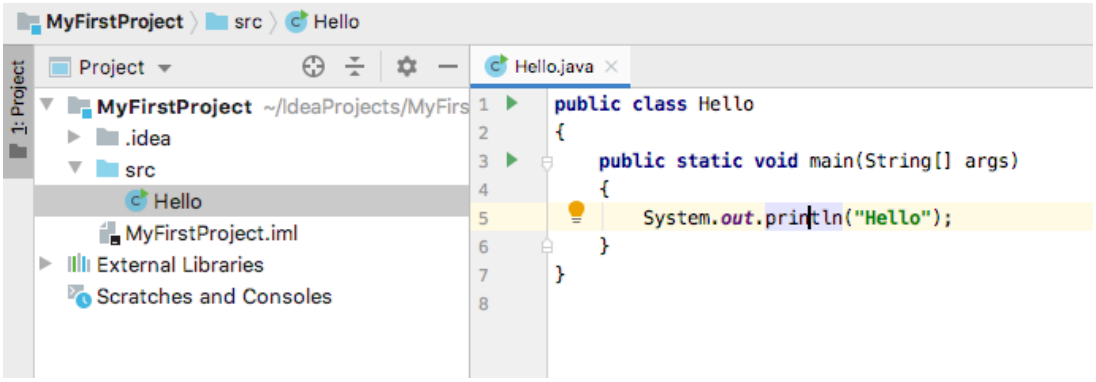
Now, let's create our first source file. Right click on the *src* folder and choose *New* > *Java Class*.



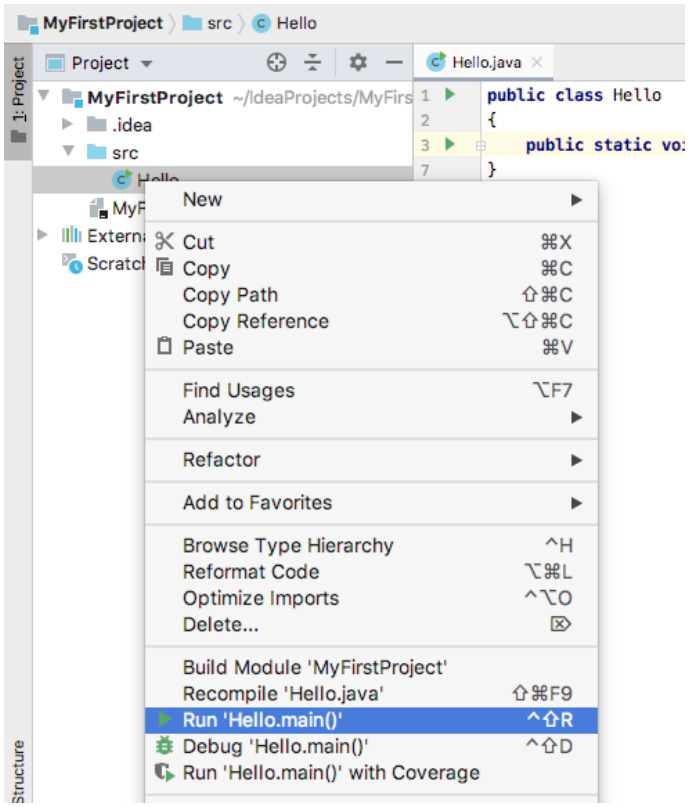
Then, we must specify the class name. For instance, `Hello`.



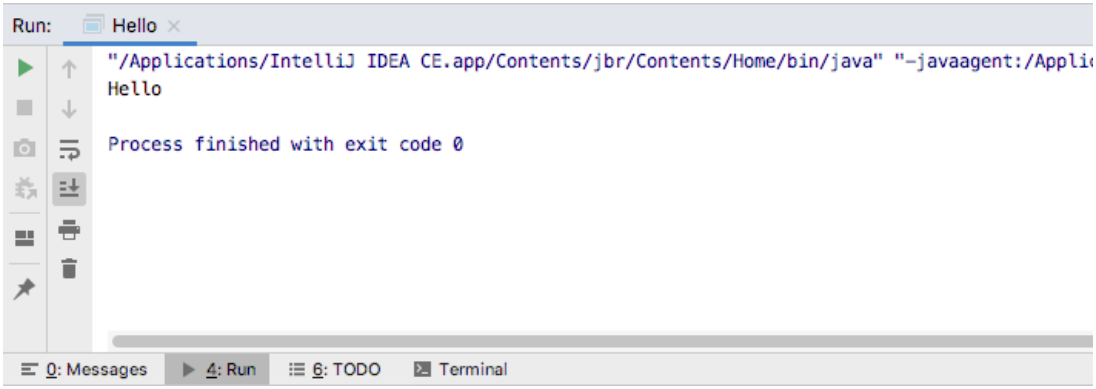
A new file will be created, and we can edit it in the main area. We can just leave a code like this one:



We can run the file by right clicking on it and choosing "*Run Hello.main()*" option:



Then, we can check the results in the embedded terminal at the bottom of the window:



1.8.2.3. Shortcuts

In the following table you can see some of the most common shortcuts available for *IntelliJ IDEA* under Windows systems.

Shortcut	Action
`Ctrl` + `Shift` + `N`	Open new file.
`Ctrl` + `N`	Open any class.
`Ctrl` + `Spacebar`	Complete code.
`Ctrl` + `Shift` + `Spacebar`	Smart code completion.
`Ctrl` + `S`	Save file.
`Ctrl` + `O`	Overwrite methods.
`Ctrl` + `I`	Implement all.
`Ctrl` + `/`	Comment / Uncomment line.
`Ctrl` + `D`	Duplicate line.
`Ctrl` + `Z`	Undo last action.
`Ctrl` + `Shift` + `Z`	Redo last undone action.
`Ctrl` + `F`	Show search dialog.
`Ctrl` + `R`	Show replace dialog.
`Ctrl` + `F9`	Compile project.
`Shift` + `F10`	Run project.
`Shift` + `F9`	Debug.
`F7`	Step into function (in *debug* mode).
`F8`	Next line (in *debug* mode).
`F9`	Stop debug.
`Ctrl` + `F8`	Create breakpoint.
`Ctrl` + `Shift` + `F12`	Maximize editor panel.

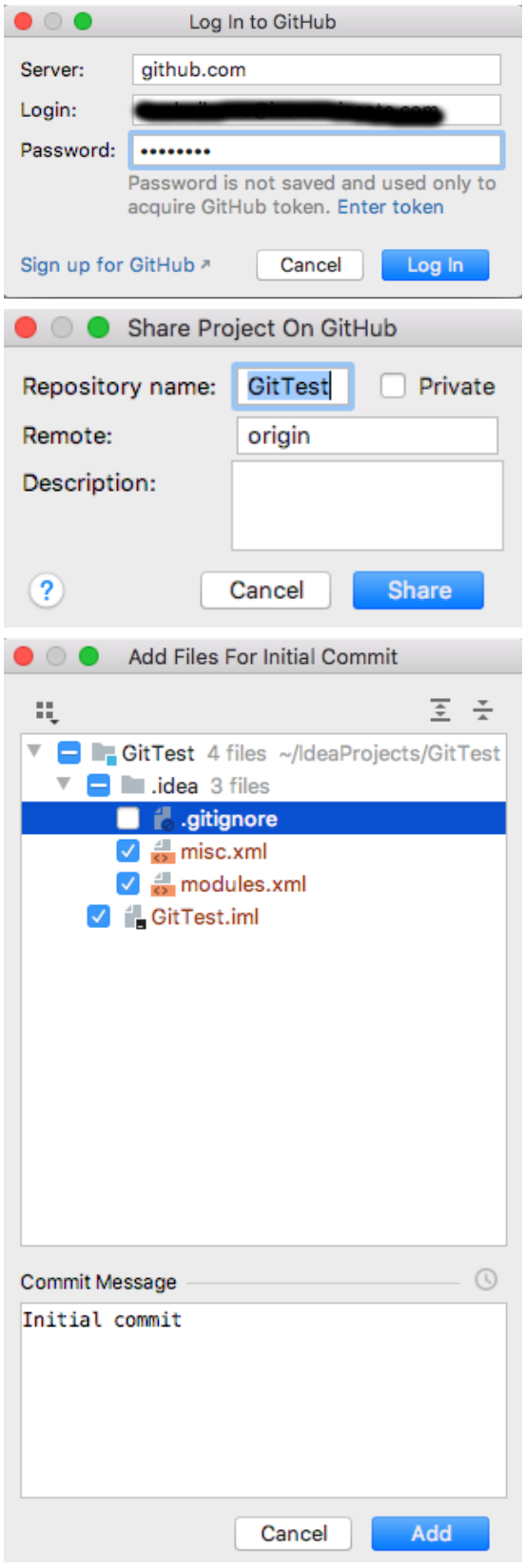
Most of these shortcuts are also available under Linux. Regarding MacOSX systems, you must replace `Ctrl` key with `Cmd` key. You can find more shortcuts here.

1.8.2.4. Integration with Git

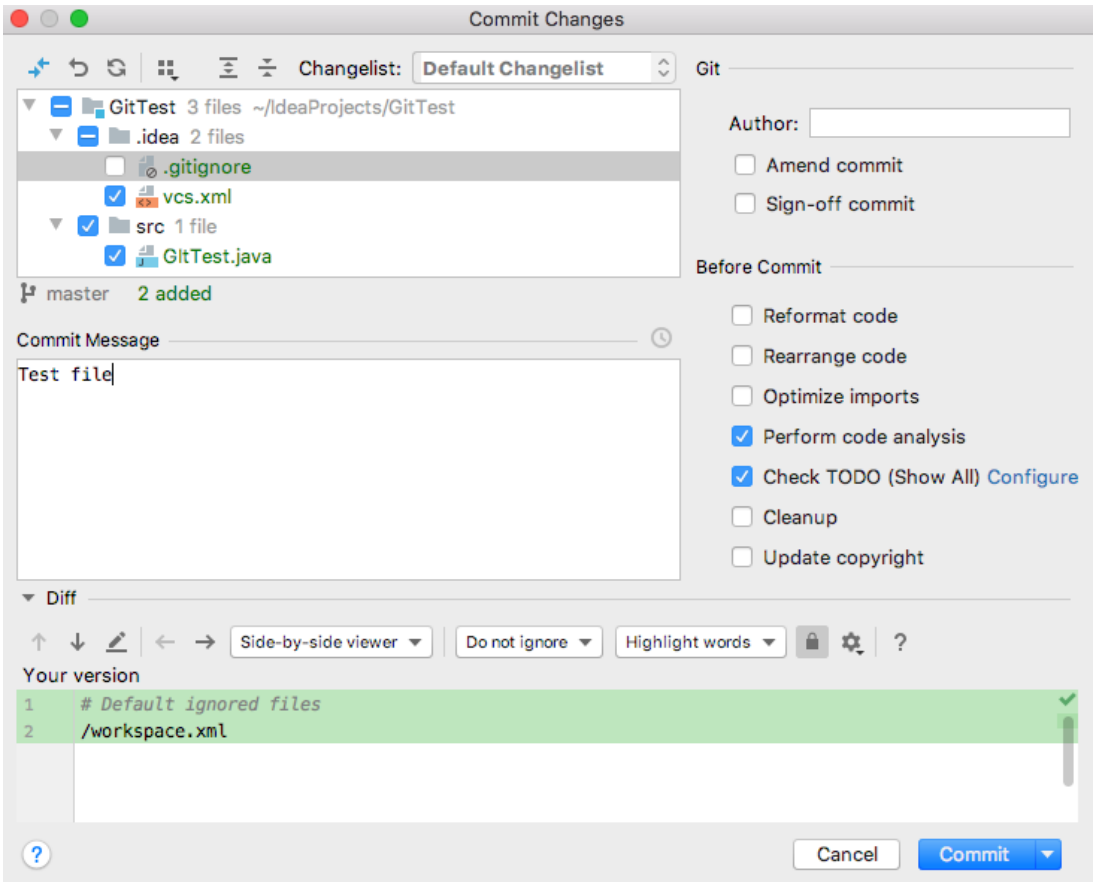
The integration of *IntelliJ* with Git repositories is really easy:

- If we want to **clone or download a remote repository** with an existing IntelliJ project to start working on it, we just go to *VCS* > *Checkout from Version Control* > *Git* menu. Then, we specify the URL of the repository, and it will be downloaded to our specified location (in our local machine).

- If we want to **create a new, remote repository**, we must create the *IntelliJ* project first, and then choose *VCS* > *Import into Version Control* > *Share project on GitHub*. Then, we will be asked to type our GitHub username and password, and the repository name (by default, IntelliJ suggests our project's name). Finally, we can choose the files for the initial commit.



- Whenever we want to **commit** new changes, we choose *VCS* > *Commit...* menu, and then choose which changes we want to commit, and the comment associated to this commit.



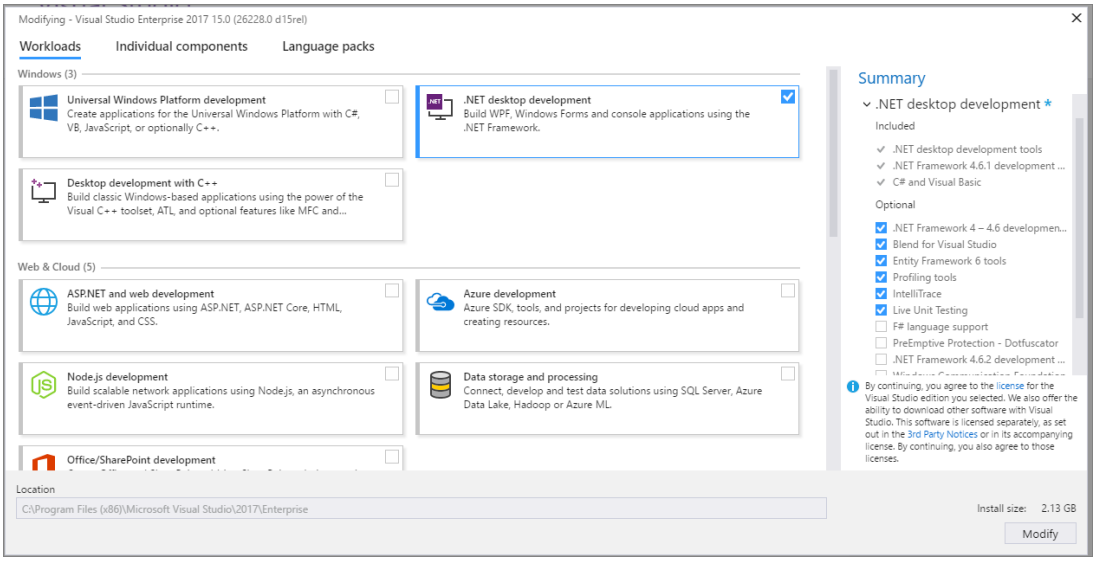
- If we want to **push** a commit, or **pull** the changes from the remote repository, we can find these options inside *VCS* > *Git* submenu.

> **Proposed exercises**
>
> **1.8.2.1.** Create a new local Java project called *IntelliJGit*, and then share this project in your GitHub account. Then, add a new class called `Test`, and commit and push these changes.

1.8.3. Visual Studio

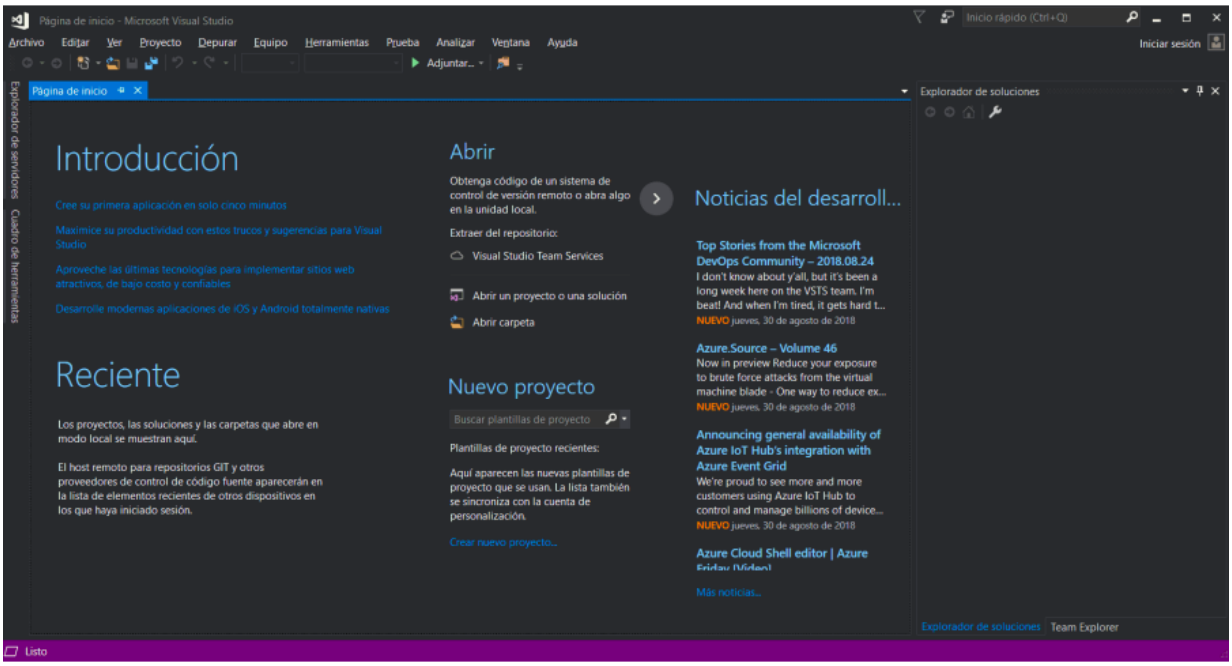
Visual Studio was an IDE created in 1997. It was initially created to develop C++ and Visual Basic applications under Windows, but it has evolved and now it includes some other languages, such as C#, ASP.NET... In general, you can develop any kind of application supported by the .NET platform, but there are also some other languages that are also supported, such as Java or Python, among others

There are different distributions for Visual Studio: Community, Professional or Enterprise. The first one is free, and you can download it here. It is available for Windows and MacOSX systems. Once you run the installer, you need to choose your workload. For instance, if you are planning to develop desktop or console applications with C# and Windows Forms, you can just choose *.NET desktop development*.

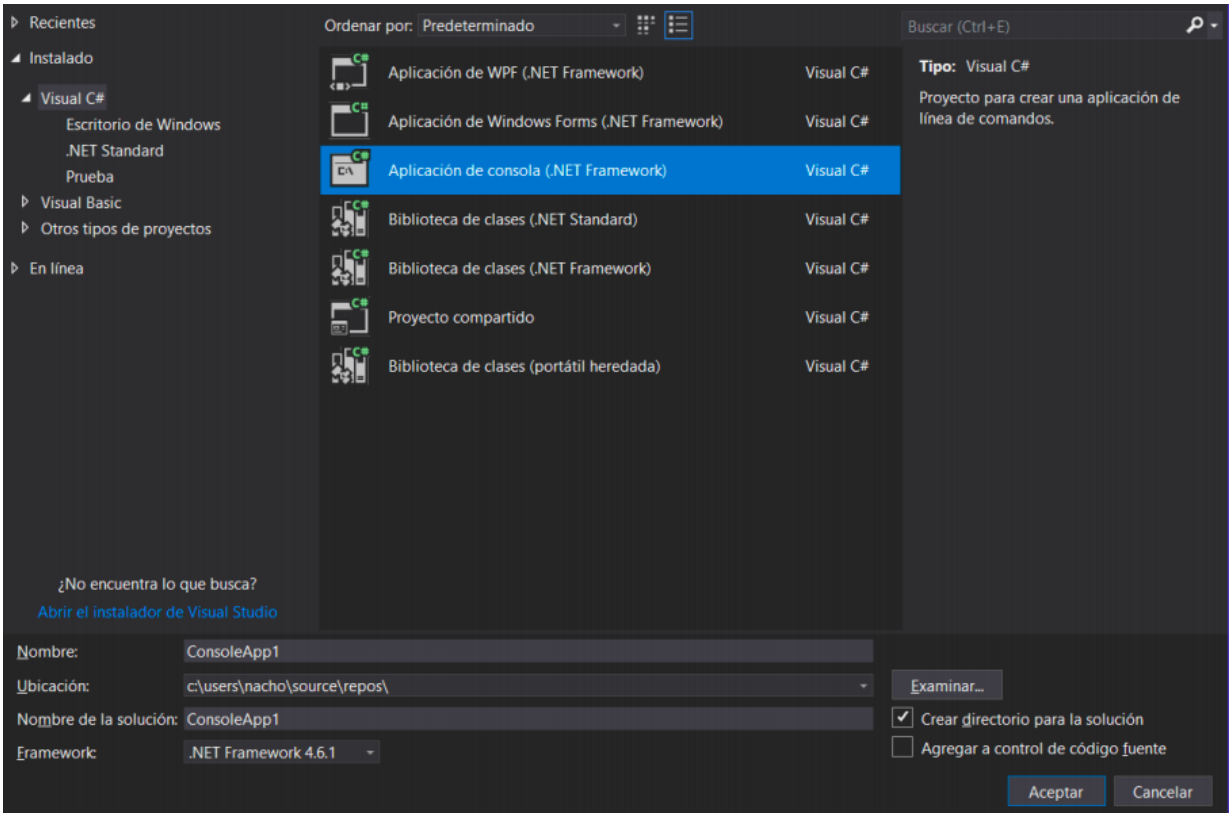


1.8.3.1. Creating projects

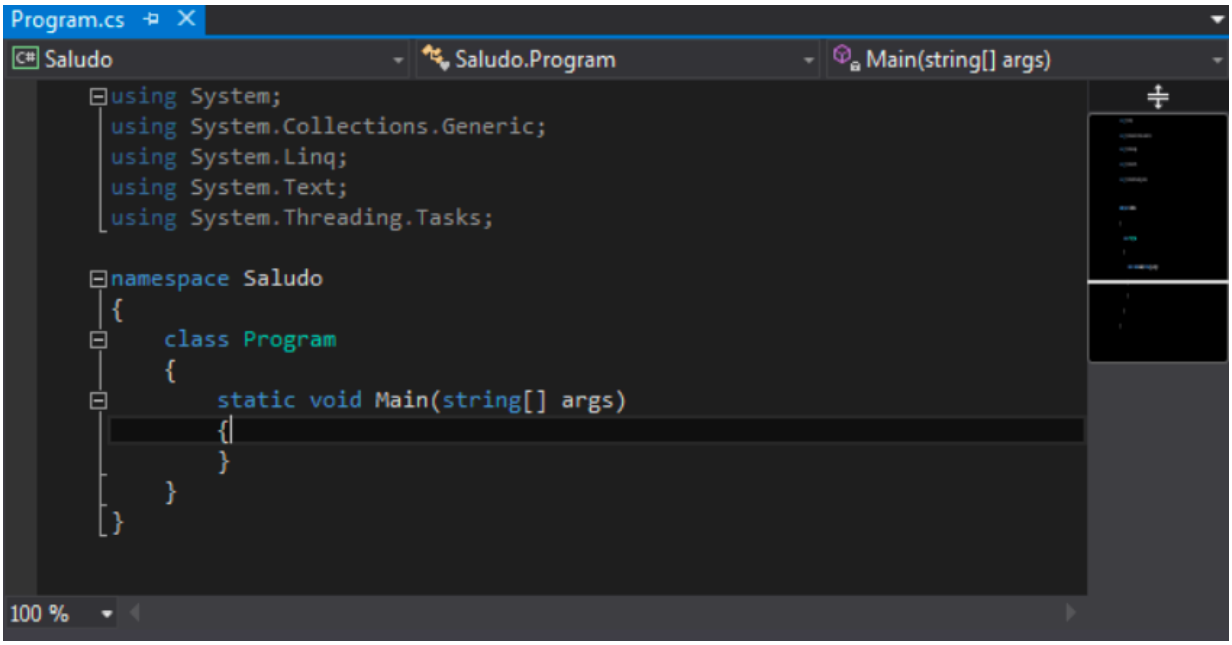
This is the welcome screen for Visual Studio:



If we want to create a new project, we go to *File* > *New project* menu. Then, we usually choose a C# console application:



In the bottom form, we must specify the project name and location (we can just leave the default location). Then, a new project will be shown, with a default, initial source file called `Program.cs`, with some default code already written in it:



In order to run the program, we just click on the *Start* button at the toolbar, or press `F5`, or `Ctrl + F5` if you want the program to stop after finishing, before closing the terminal.

1.8.3.2. Shortcuts

In the following tables you can see some of the most common Visual Studio shortcuts (under Windows).

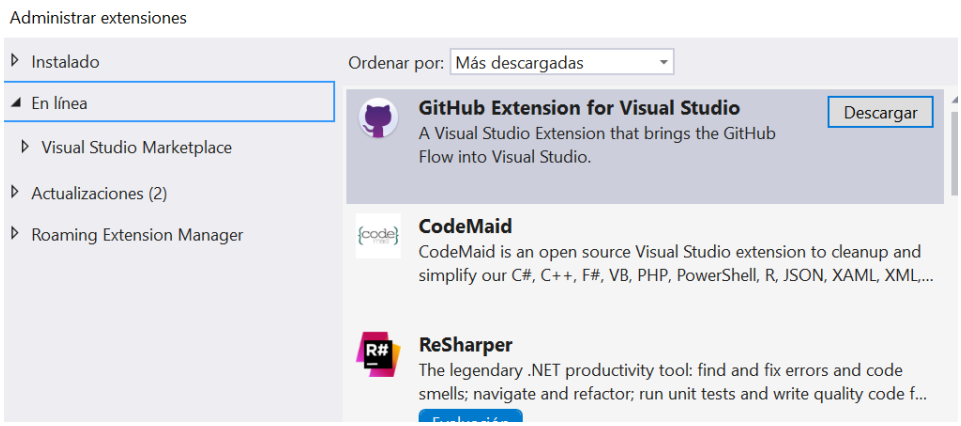
Shortcut	Action
`Ctrl` + `Shift` + `N`	Create a new project.
`Ctrl` + `N`	Create a new file.
`Ctrl` + `S`	Save current file.
`Ctrl` + `Shift` + `S`	Save every open file.
`Ctrl` + `C` / `V` / `X`	Copy, paste and cut text.
`Ctrl` + `Z`	Undo last action.
`Ctrl` + `Y`	Redo last undone action.
`Ctrl` + `F`	Show search dialog.
`Ctrl` + `H`	Show replace dialog.

Shortcut	Action
`Ctrl` + `L`	Remove line.
`Ctrl` + `R`	Rename selected element.
`F5` / `Ctrl` + `F5`	Run application with/without debugging.
`Ctrl` + `Shift` + `B`	Rebuild the project.
`F11` / `F10`	Step into function / next step (in *debug* mode).
`Shift` + `F11`	Exit function (in *debug* mode).

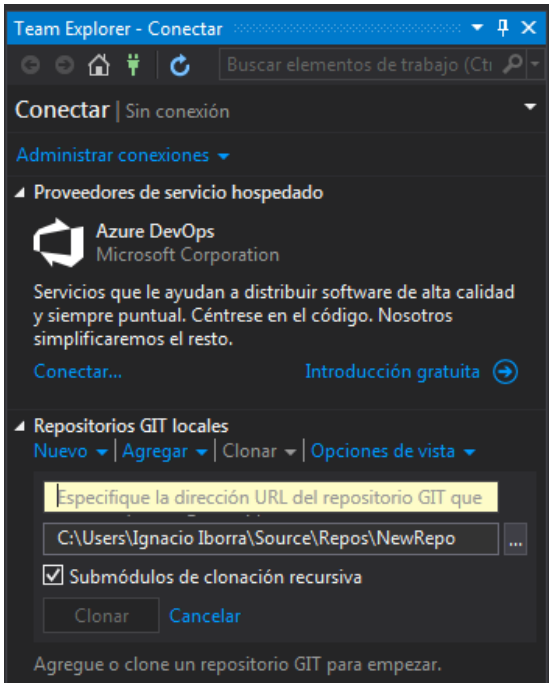
Regarding MacOSX, you must replace `Ctrl` key with `Cmd` key. You can find more shortcuts here

1.8.3.3. Integration with Git

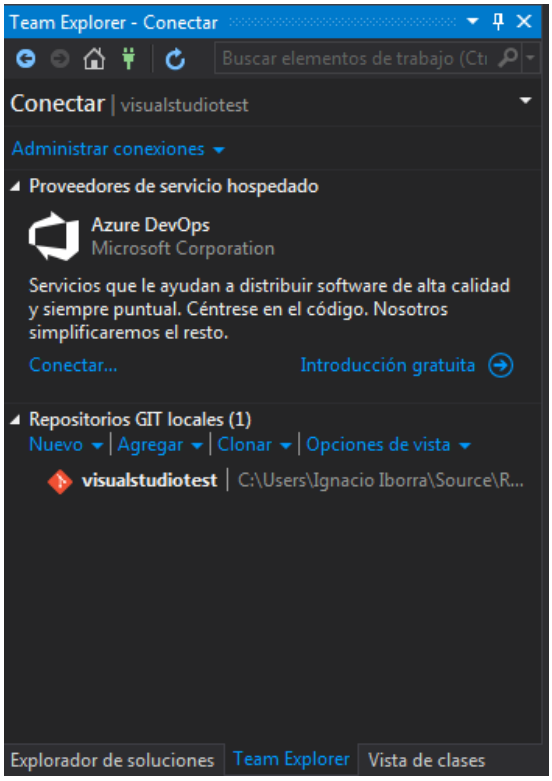
Visual Studio has an option to deal with Git repositories. First of all, we may need to install GitHub extension for Visual Studio, from *Extensions* > *Manage extensions* menu:



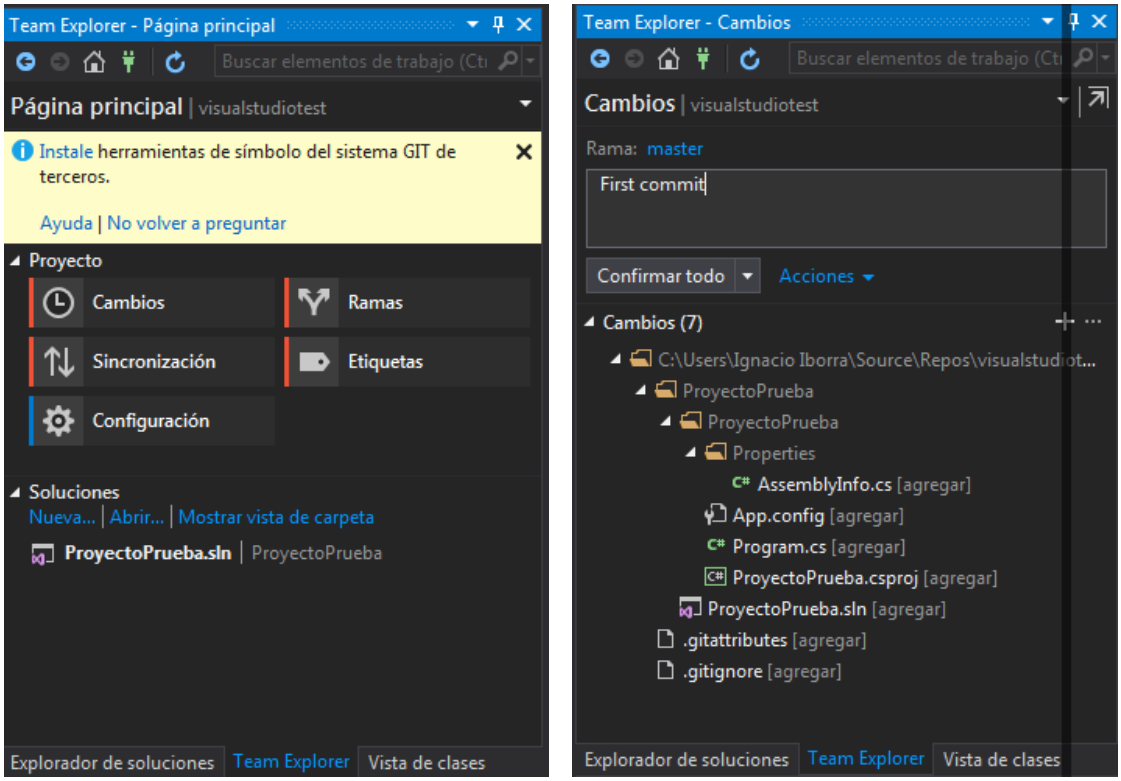
Then, if we want to communicate with a remote repository, we need to go to *Team* > *Manage connections* menu. The first time we choose this option, we will need to log in to GitHub to give some permissions to Visual Studio. Then, we can either create a local Git repository, or clone a remote one.



If we choose the *Clone* option, we need to specify the remote URL to be cloned, and the local folder to download it. Then, we can add project(s) to this repository, and make changes. Everytime we want to upload these changes, we need to doubleclick on the repository:



Then, we can click on the *Changes* option to see all the changes that are pending to be uploaded. We can add a comment for all of them and confirm the commit:



Finally we need to choose the *Synchronize* option to upload (*push*) the changes to the remote repository. We may be asked to enter our credentials to let us push the contents.

We can also use this option at the beginning of our session, to download (*pull*) the updated contents from the remote repository, so that we can make our changes and upload them from a previously updated version.

> **Proposed exercises**
>
> **1.8.3.1.** Create a new remote repository in your GitHub or Bitbucket account called *VisualStudioTest*. Clone it in Visual Studio, add a new project on it and upload the changes from Visual Studio to the remote repository.

1.8.3.4. Other features

There are other settings that can be set up from *Tools* > *Options* menu. For instance, we can show/hide the line numbers from the *Text editor* subsection, for each specific language (C#, Basic...), or the indentation vertical lines to easily see the boundaries of an `if` or `for` statement...

1.8.4. To learn more...

You can learn more about all these IDEs in the following links

- IntelliJ IDEA
- IntelliJ IDEA (Wikipedia)
- Visual Studio
- Download Visual Studio

- shortcutworld.com (general database with shortcuts for many IDEs)