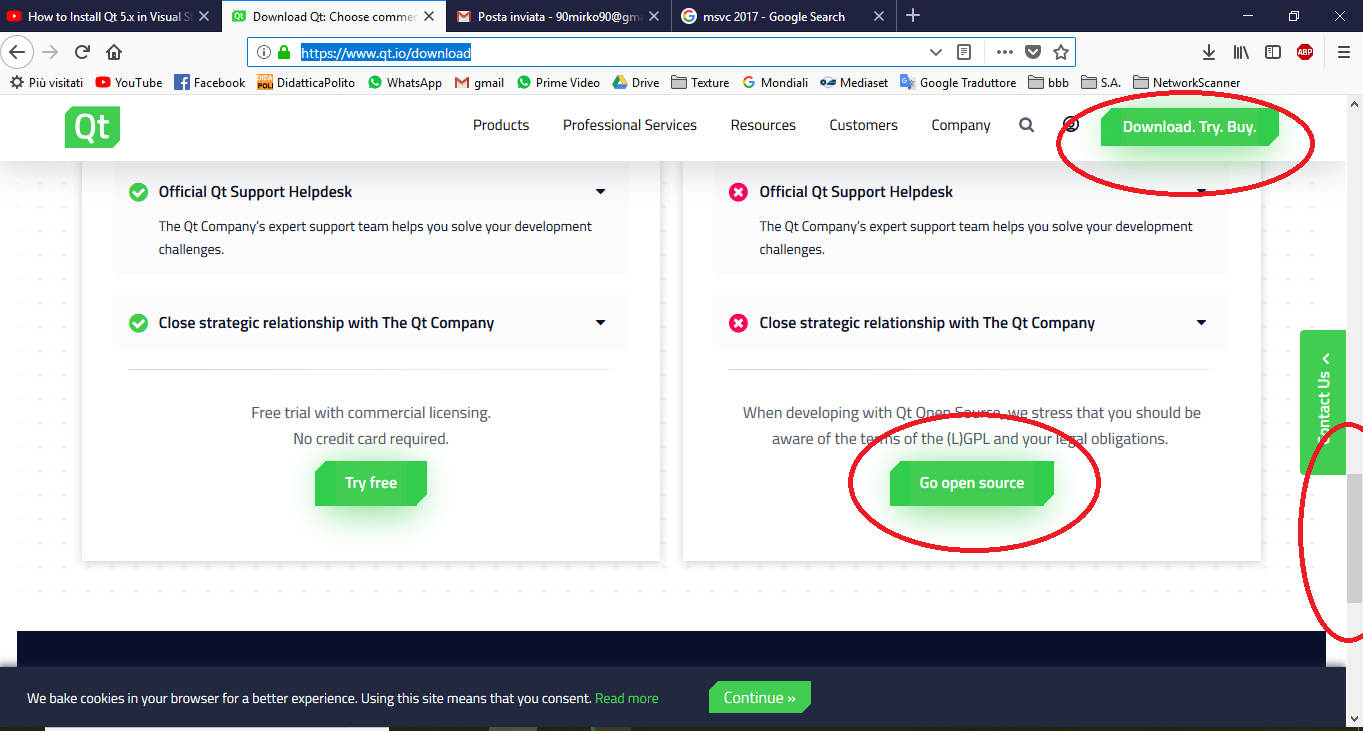
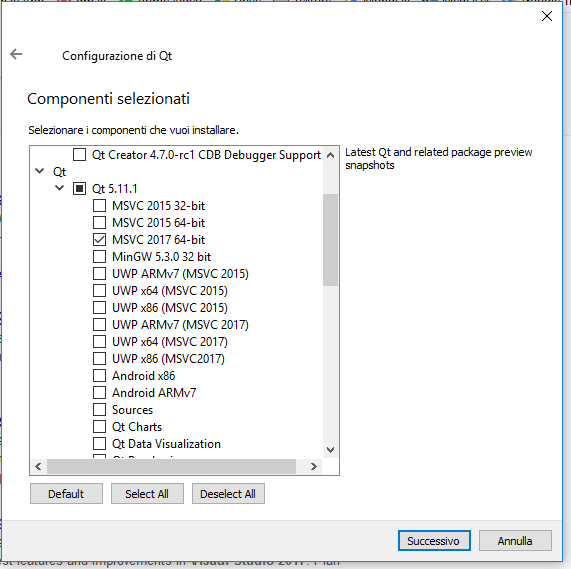
Installazione QT

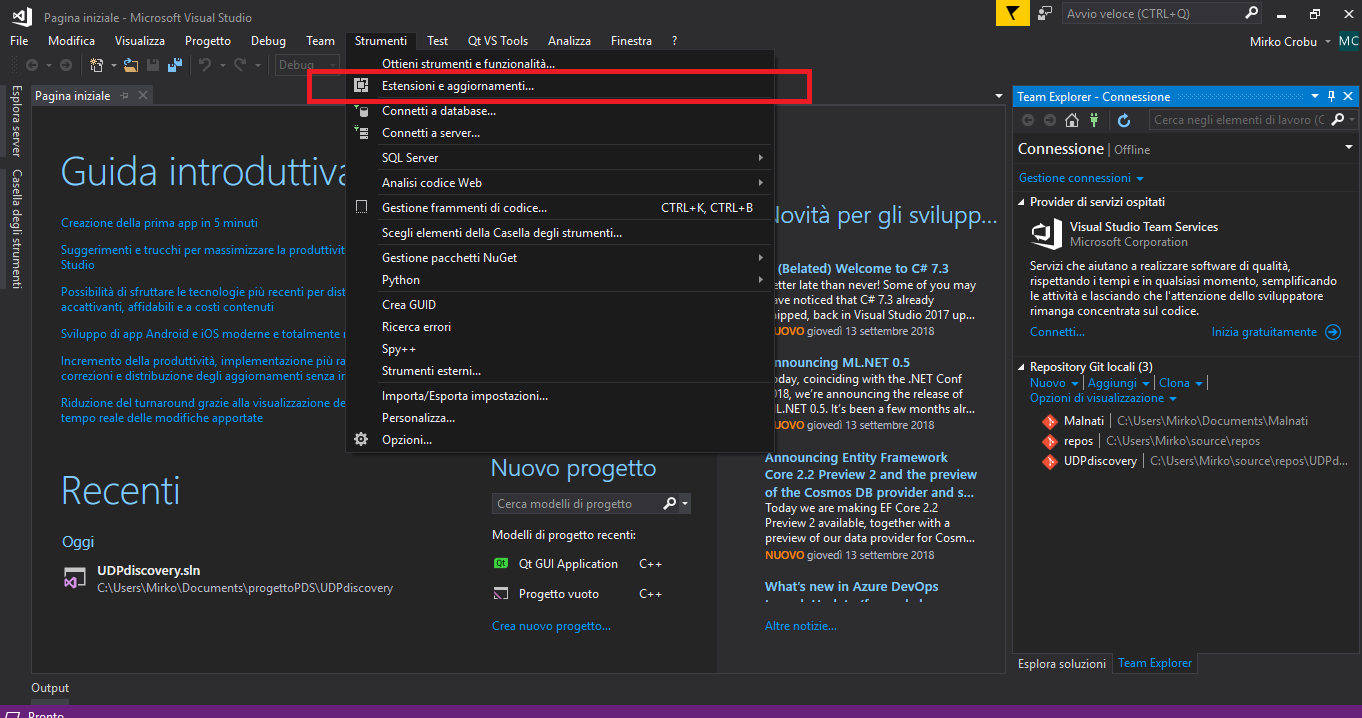
1. Download e installazione versione open source di QT <https://www.qt.io/download>.



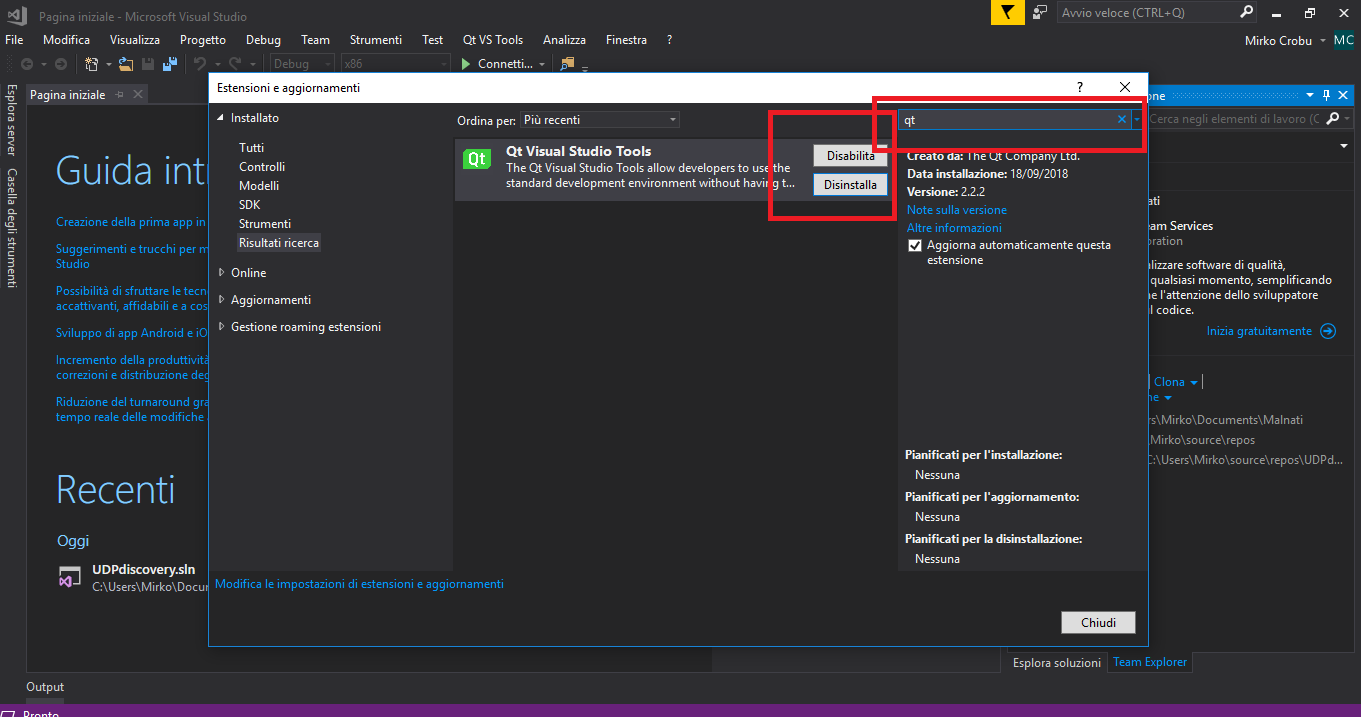
1. Scaricare e avviare l’ exe. Seguire la procedura guidata e creare un account QT. Andare avanti fino ad arrivare alla schermata in cui bisogna scegliere le librerie precompilate per il nostro ambiente di sviluppo preferito. MSVC 2017 si riferisce a Visual Studio 2017.

La versione installata da me è la 5.11.1 come da immagine.

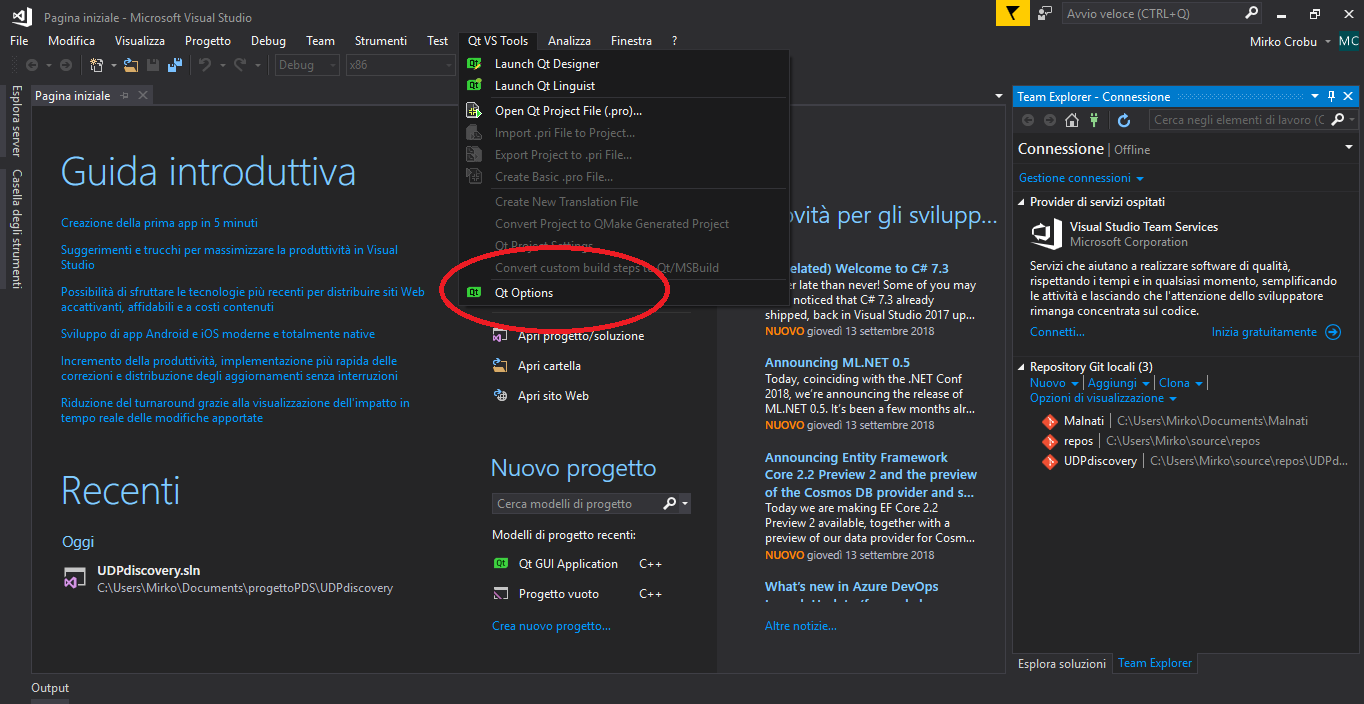
1. Aprire Visual Studio, Strumenti>Estensioni e Aggiornamenti.



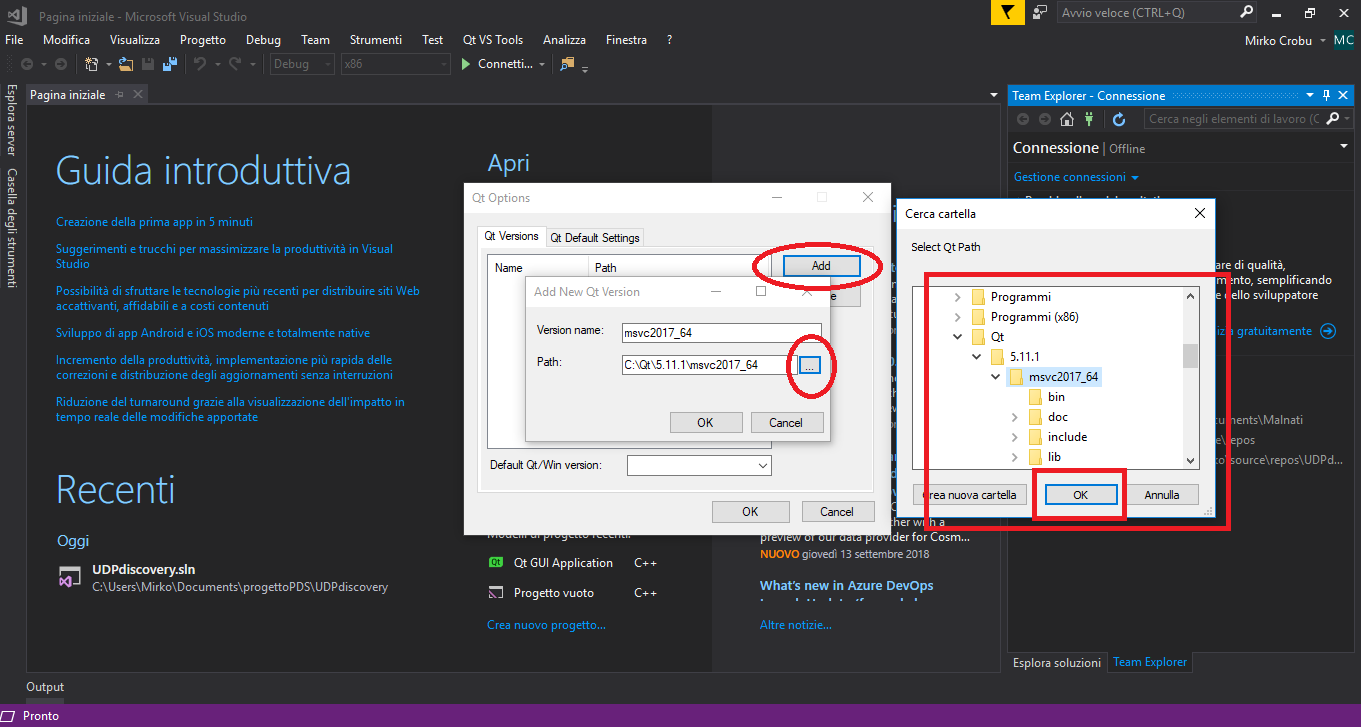
1. Nella casella di ricerca scrivere “QT” e installare il pacchetto trovato.



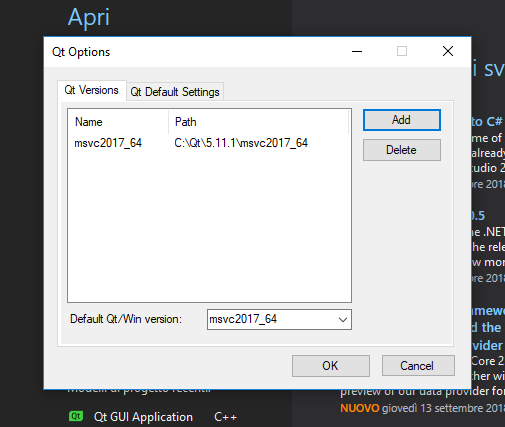
1. Al termine, riavviare VS e andare sul nuovo menù QT VS Tools e andare su opzioni.



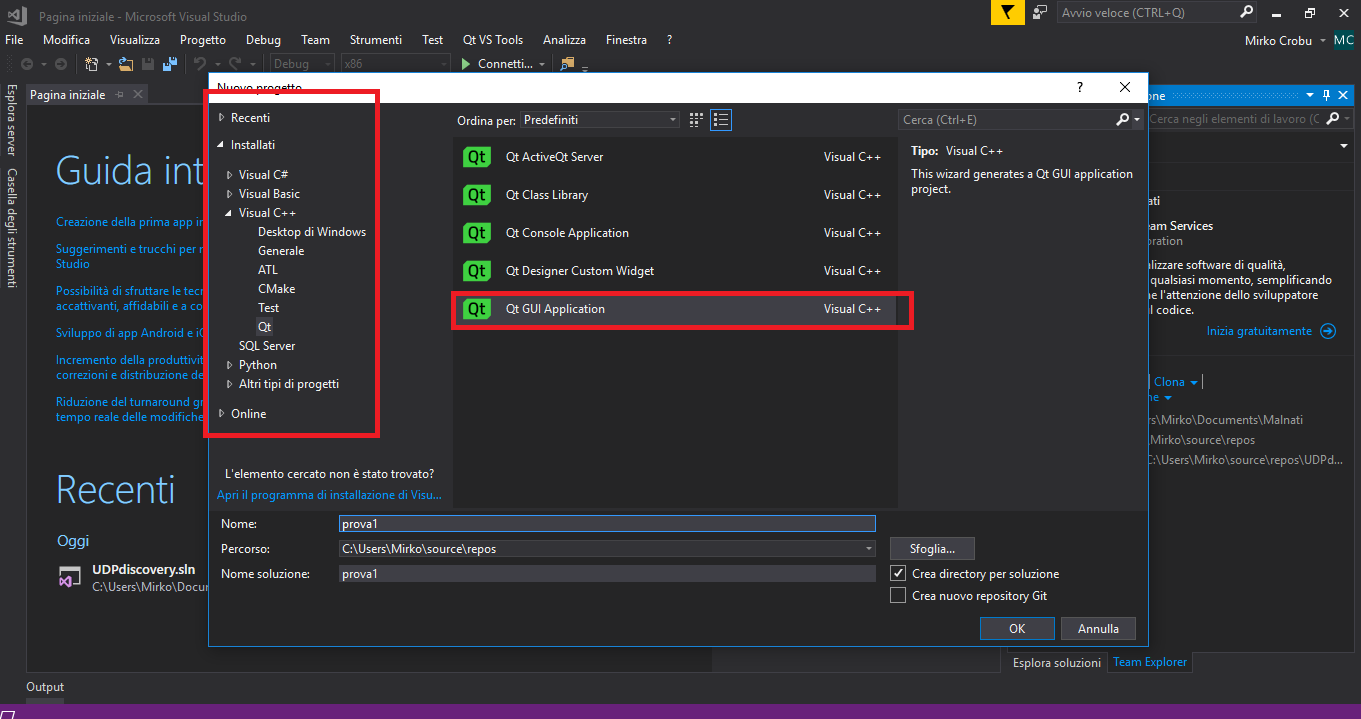
1. Aggiungere il percorso delle librerie QT per il nostro ambiente di sviluppo. Nell’esempio QT per Visual studio 2017.



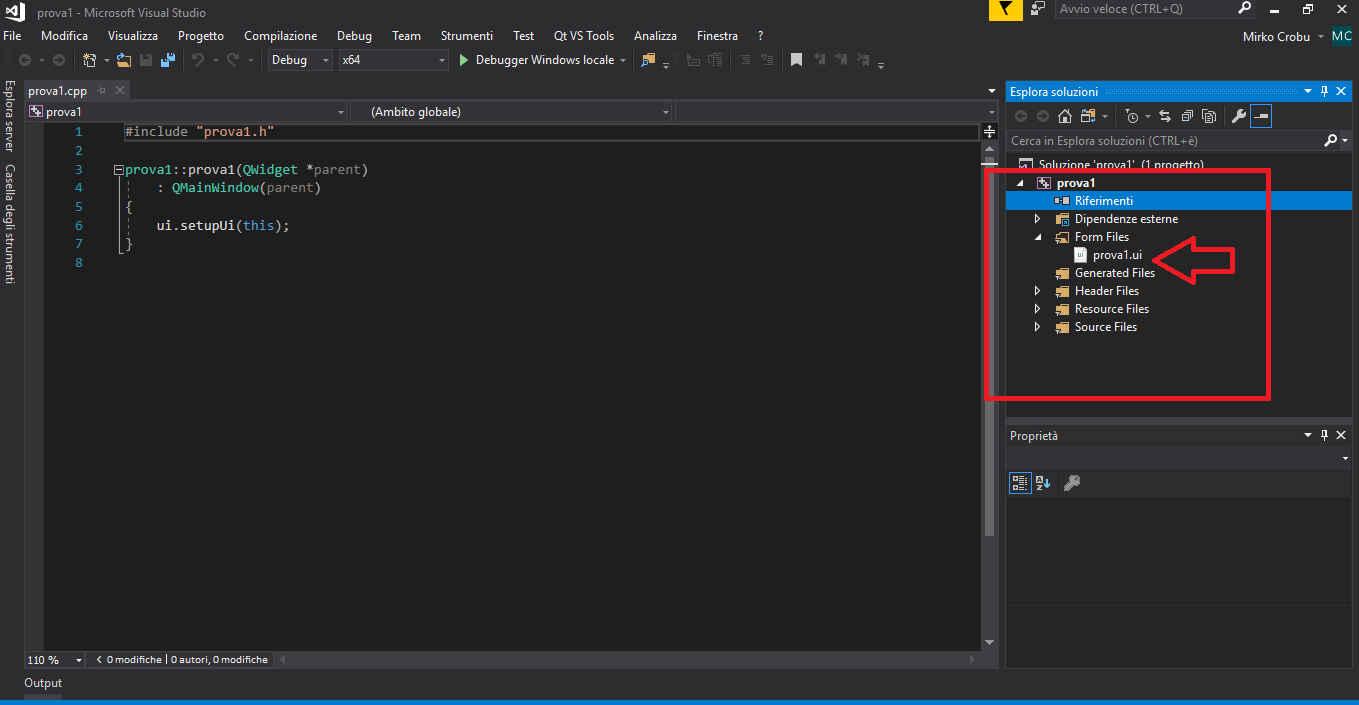
1. Cliccare su ok.



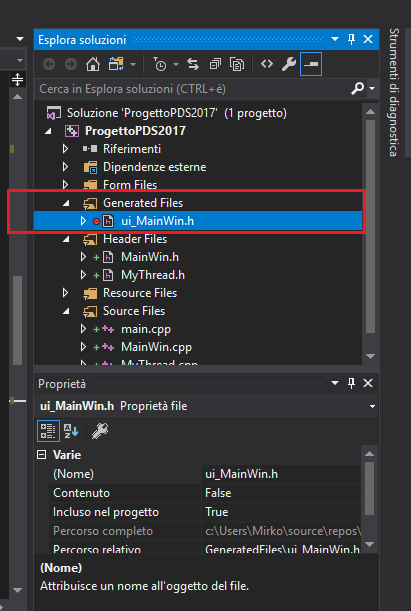
1. Creare un nuovo progetto e selezionare le scelte come da immagine seguente.



1. Cliccare il file con estensione .UI per accedere al “tools di disegno”.



Notare che QT Designer(programma che si apre con il doppio click sul file nell’immagine qui sopra) ha un aspetto quasi uguale a QT Creator e possono essere quindi confusi:

**- QT Designer** permette solo di modellare la GUI dal punto di visto grafico, inserendo Widget e dando la possibilità di cambiare nome, misure e aspetto.

- **QT Creator** invece, è lo strumento presente in seguito all’installazione di QT, e che lavora con differente IDE, aggiunge la possibilità di collegare i Widget alle relative funzioni e fornire quindi uno scheletro in c++ dove poter aggiungere il comportamento al click, oppure agli altri eventi possibili per ogni widget.

Con Visual Studio è possibile aprire usare solo il QT Designer.

QT Designer creerà un file.h in chi vengono creati e inizializzati tutti gli oggetti grafici. Questo file, che si trova nella cartella “Generated Files”, non è accessibile direttamente da Visual Studio. Esso va aggiunto manualmente al progetto se lo vogliamo visualizzare nell’explore soluzione come da immagine qui a destra.

Possiamo controllare gli oggetti creati col QT Designer attraverso questo file.

QT Designer crea un file. h(**ui\_MainWin.h**) contenente una classe in cui dichiara come attributi pubblici tutti gli oggetti che compongono la GUI. Creando un oggetto di tale classe è possibile accedere a tutti gli elementi della GUI.

2) Struttura di un’applicazione QT

**ui\_MainWin.h :** è il file auto - generato con QT Designer in cui sono dichiarati tutti gli oggetti presenti nella GUI e i metodi per la loro visualizzazione.

**MainWin.h e MainWin.cpp** : contengono la classe con cui andiamo a controllare gli oggetti della GUI. Essa ha come attributo un oggetto della classe ui\_MainWin appena vista. Attraverso tale oggetto chiamato “ui\_MainWin ui;” è possibile accedere a qualsiasi componente grafico e controllarlo attraverso i vari metodi.

**Main.cpp :** classico punto di partenza di un programma in C++. Qui creiamo un oggetto di tipo MainWin e attraverso il metodo Show() la mostriamo a video.

**QT Ragiona in termini di connessioni, eventi, segnali, slot:**

**I segnali**: sono emessi al verificarsi di determinati eventi. Se per esempio ci riferiamo ad un pulsante, pigiandoci sopra, verrà scatenato l’**evento** “cliked()”. Tale evento emetterà un segnale in grado di risvegliare una determinata funzione. Ogni segnale verrà connesso ad uno **slot:** cioè alla funzione che vorremmo fosse svolta dal pulsante ogni volta che viene premuto. Questa connessione si effettua trame la funzione connect().Con questo meccanismo possiamo quindi dire:

1)del pulsante “bottone1” connetti il segnale “clicked()” allo slot “invia messaggio()”;

2)del pulsante “bottone1” connetti il segnale “bottone\_rilasciato()” allo slot “play()”;

3)del pulsante “bottone2” connetti il segnale “clicked()” allo slot “avviaThred()”;

e così via …..

Quindi possiamo dire che un evento lancia un segnale che attiva uno slot.

**eventi->segnali->slot**

possiamo creare tutti gli slot che vogliamo, e fare tutte le connessioni con i vari segnali che vogliamo.

Vedere il meccanismo implementato per la classe MyThread per vedere un funzionamento più dettagliato del sistema **eventi->segnali->slot**.