

Relazione Progetto PMO

Tommaso Pulici

Matricola:302063

2021/2022

Specifica del software

Il progetto consiste nel realizzare un gestionale di raccolta multe, con lo scopo di rendere più veloci molti degli aspetti burocratici che circondano questo ambiente.

Il progetto viene realizzato con Windows Form in linguaggio C#.

In questo progetto si avranno due tipi di utenti gli agenti di polizia e i cittadini.

Il primo è colui che procederà nella scrittura delle multe, mentre il secondo si troverà tutte le multe indicizzate con la possibilità di pagarle con un click.

Studio del problema

Punti critici:

I principali problemi che ho riscontrato nello sviluppo di questo progetto sono stati:

- Adattare il pattern builder, poiché esso non era congruo al 100% per questo progetto.
- L'utilizzo di un DataBase per la raccolta dei dati
- Rendere visivamente il progetto non un semplice gestionale, ma dargli quel senso di gradimento agli occhi e di minimalismo

Risoluzione:

Per il pattern builder c'è stata una rivisitazione per adattarlo al meglio al progetto e renderlo il più possibile sostenibile nel tempo

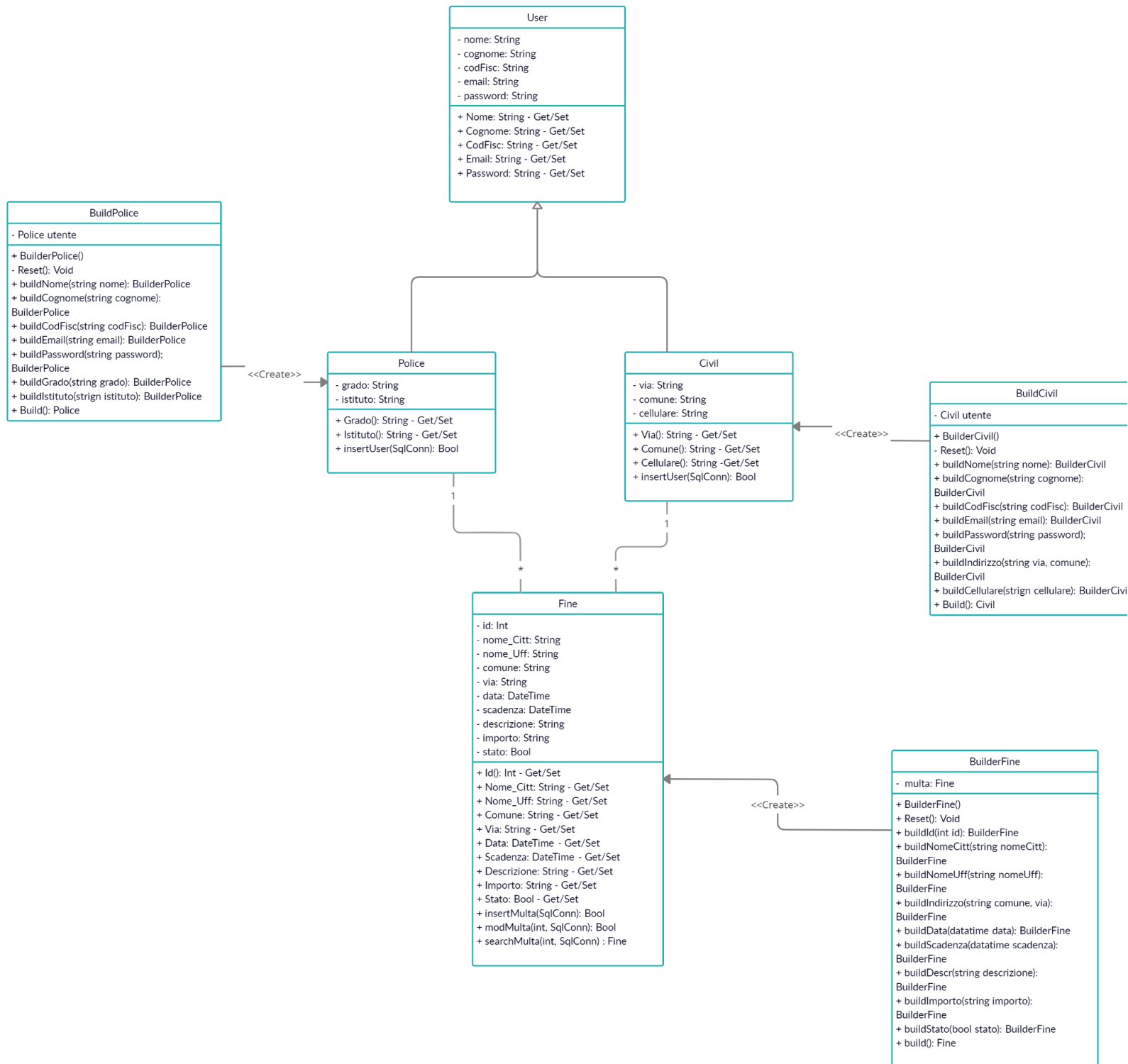
Per salvare tutti i dati delle multe e degli utenti ho utilizzato un DataBase che mi permette di prendere e scrivere su di esso quante volte voglio senza perdere ogni volta i dati quando il programma termina.

Ho utilizzato il DB locale standard di visual studio.

Per rendere il progetto più appetibile visivamente ho utilizzato un font standard di google e utilizzato delle immagini che potessero raffigurare ognuna la sezione dedicata

Scelte architetturali

Diagrammi delle classi:



Descrizione dell'architettura:

L'architettura del software è basata su classi che determinano i vari comportamenti del programma.

Il programma si presenta con un form che ti dà la possibilità di scegliere quale tipo di utente sei



Attraverso la funzione Show() e Hide() possiamo aprire e chiudere tutte le form che ci interessano. In questo momento noi dovremo scegliere il tipo del nostro utente, quindi se agente di polizia oppure cittadino.

Una volta cliccato su uno dei due avremo la possibilità di fare il login del nostro utente.



Accedi



Codice fiscale

E-Mail

Password

Entra

[Non sei ancora registrato?](#)



Accedi



Codice fiscale

E-Mail

Password

Entra

[Non sei ancora registrato?](#)

In questi due form avremo la possibilità di accedere direttamente alla sezione di nostro interesse nel caso avessimo già un account creato.


Tramite il bottone entra, invocheremo una funzione, chiamata Login(), che ci permetterà di cercare all'interno del nostro DataBase se le credenziali inserite nel form sono esistenti.

Se il controllo andrà a buon fine ci si aprirà il form dove potremo iniziare ad utilizzare appieno il nostro programma, in caso contrario avremo la possibilità o di ritornare alla pagina precedente oppure di andare alla pagina della creazione del nostro utente.

Registrazione Agente

←

Registrati



Nome

Cognome

Codice fiscale

E-Mail

Password

Grado

☐ Dirigente

☐ Commissario

☐ Ispettore

Istituto


Registrati

[Sei già registrato?](#)

Registrazione Civile

←

Registrati



Nome

Cognome

Codice fiscale

E-Mail

Password

Cellulare

Comune

Via

Registrati

[Sei già registrato?](#)

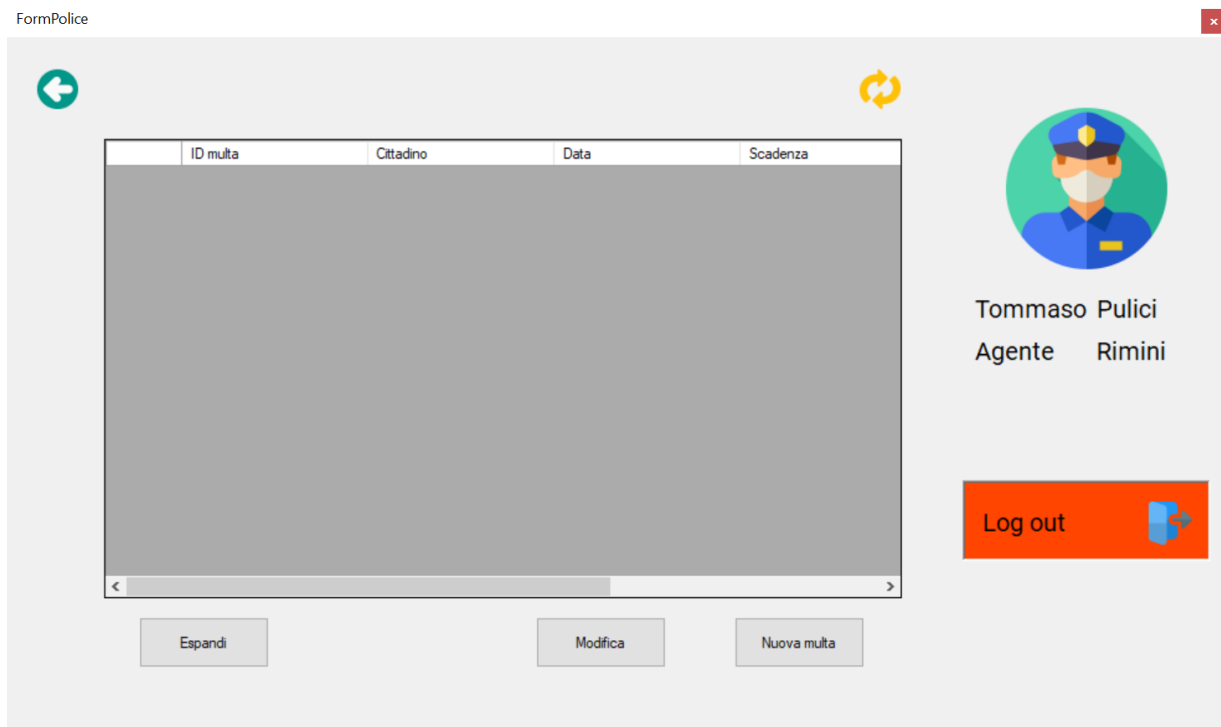
Con questi due form di registrazione avremo la possibilità di registrare il nostro utente.

Cliccando sul tasto registrati richiameremo una funzione `btRegister_Click()` che prenderà i dati presenti all'interno delle `TextBox` per creare una istanza di classe di tipo `Civil` o `Police` attraverso il design pattern `Builder`.

Queste classi al loro interno hanno un metodo che si chiama `insertUser()`, questo metodo permette di parlare direttamente con il `DataBase` e dare la possibilità, una volta passati i controlli, di inserire i dati all'interno delle tabelle (`Cittadino` o `Agente`).

Esso è un metodo che restituisce una variabile di tipo `bool`, quindi nel caso vengano inseriti correttamente i dati in tabella esso restituirà un valore di tipo `True` e ci permetterà di accedere direttamente al programma, in caso contrario restituirà un valore di tipo `False` con annesso errore tramite `MessageBox`.

Questa è la sezione che ci apparirà quando faremo l'accesso tramite l'agente di polizia:



In questa sezione possiamo trovare al centro una DataGridView che raccoglierà tutte le multe scritte dall'agente.

Sotto la DataGridView ci sono tre bottoni, che possono:


- Creare una nuova multa
- Modificare una multa già esistente
- Espandere una multa, così da vedere più dettagli di essa

Sulla destra invece troviamo il nome e cognome dell'agente loggato, il suo grado e l'istituto da cui proviene. Subito sotto i dati del profilo troviamo un bottone di log out che ci riporterà alla pagina iniziale.

Invece in alto a destra della DataGridView troviamo un bottone che ci permette di ricaricare la tabella, richiamando una funzione `UpdateDataGridView()`.

Questa funzione inizialmente cancellerà le righe presenti, per poi ricercare nel DataBase tutte le multe in tabella (sempre e solo dell'agente di polizia corrente) e reinserirle nella DataGridView.

Ritornando ai bottoni troviamo quello della Nuova multa che, come si può dedurre dal nome, ci permette di creare una nuova multa. Cliccando su di esso ci aprirà una form dove dovremo inserire i dati della multa, la form ci si presenta in questo modo:



The screenshot shows a window titled "FormNewMulta" with a close button in the top right corner. Inside the window, there is a back arrow button in the top left. The main heading is "Inserisci nuova multa". Below the heading is an icon of a yellow document with three dollar signs (\$\$\$). The form contains the following fields:

- Cittadino**: A dropdown menu.
- Comune**: A text input field.
- Viale**: A text input field.
- Data**: A date picker showing "domenica 22 agosto".
- Importo**: A text input field.
- Descrizione**: A large text area.

At the bottom of the form is a button labeled "Conferma".

Tramite questo form possiamo inserire la nostra multa inserendo il codice fiscale del cittadino, il comune e il viale di dove è accaduta la trasgressione, la data, l'importo e la descrizione. Una volta inseriti tutti i parametri potremo confermare la multa tramite il bottone apposito che richiamerà la funzione `btConferma_Click()`, in questa funzione troviamo, oltre i consueti controlli per vedere se i campi sono stati inseriti correttamente, la costruzione della nostra multa che viene fatta attraverso il design pattern Builder e dopo averla inizializzata richiameremo il metodo della classe Fine "insertMulta()", che ci permetterà di inserire la multa all'interno del DataBase.

Questo metodo ha come valore di ritorno una variabile bool che nel caso il valore fosse True inserirà correttamente la multa, ma in caso contrario riporterà l'errore commesso tramite MessageBox.

Affianco al bottone della nuova multa c'è l'apposito tasto per modificarla, naturalmente prima di poter modificare una multa bisogna selezionarla, questo lo si fa semplicemente cliccando su una multa all'interno della DataGridView, una volta fatto ciò si potrà procedere a modificare la multa.

Il form si presenta in questo modo:



The screenshot shows a Windows form titled "FormModMulta" with a close button in the top right corner. The form has a light gray background and a title bar. In the top left corner, there is a green circular button with a white left-pointing arrow. The main title "Modifica Multa" is centered at the top in a bold black font. Below the title is a yellow icon of a document with three dollar signs (\$\$\$). The form contains several input fields and labels arranged vertically: "Cittadino" with a dropdown menu showing "Cod"; "Comune" with a text box containing "Riccione"; "Viale" with a text box containing "Portovenere 26"; "Data" with a date picker showing "sabato 25 settembre"; "Importo" with a text box containing "324"; and "Descrizione" with a text box containing "Multa per eccesso di velocità". At the bottom of the form, there are two buttons: "Elimina" and "Modifica".

In questa sezione potremo modificare o eliminare una multa.

Per modificarla basterà cambiare i parametri all'interno delle textbox per poi cliccare su conferma, facendo ciò richiameremo la funzione `btModifica_Click()` che si occuperà di cambiare i parametri della multa selezionata con quelli aggiornati del form, poi richiamerà il metodo `modMulta()` presente nella classe "Fine" per eseguire il comando che porterà alla modifica nel DataBase di quella stessa multa.

Invece se cliccassimo sul bottone Elimina richiameremo la funzione `btElimina_Click()` che semplicemente eseguirà la query di cancellazione della multa nel DataBase attraverso l'id.

L'ultimo bottone è Espandi, esso ci servirà per vedere tutti quei dettagli di cui abbiamo bisogno ma che non sono presenti all'interno della DataGridView. Il modulo si presenta in questo modo:

Descrizione_multa ✕

 **Multa id: 70°**

Nome: Tommaso
Cognome: Pulici
Ufficiale: Tommaso, Pulici
Data: 25/09/2021
Scadenza: 25/12/2021
Comune: Riccione Via: Portovenere 26
Importo: 324.00 €
Descrizione: Multa per eccesso di velocità

Stato pagamento: Non pagato

Qui dentro troveremo i dettagli più importanti della multa: dal nome e cognome del trasgressore, il nome e cognome dell'agente di polizia, la data e la scadenza della multa, l'indirizzo di dove è accaduta la trasgressione, l'importo, la descrizione e infine lo stato di pagamento (cioè se è stata pagata o meno la multa). Non si può fare nulla all'interno di questa sezione poiché serve solo per consultare i dettagli della multa.

Una volta finito di spiegare tutto all'interno della sezione dedicata all'agente di polizia ci possiamo spostare sull'altro fronte, quello dedicato al cittadino.

FormUser

| | ID multa | Data | Scadenza | Importo |
|---|----------|------------|------------|------------|
| ▶ | 24 | 26/08/2021 | 26/11/2021 | 300.00 € |
| | 44 | 22/08/2021 | 22/11/2021 | 432.00 € |
| | 55 | 22/08/2021 | 22/11/2021 | 30000.00 € |
| | 66 | 22/08/2021 | 22/11/2021 | 123.00 € |
| | 67 | 31/08/2021 | 30/11/2021 | 300.00 € |
| | 69 | 22/08/2021 | 22/11/2021 | 123.00 € |
| | 70 | 25/09/2021 | 25/12/2021 | 324.00 € |

Tommaso Pulici

Log out

Espandi Paga

Questo form non si differenzia molto da quello dell'agente, tranne per il fatto che lo scopo del cittadino non è quello di inserire multe, ma di pagarle.

A tal proposito troveremo i bottoni sotto alla DataGridView differenziati dal form precedente, dove spicca un bottone Paga che serve, appunto, per pagare le multe che riceveremo.

Il bottone Paga richiamerà una funzione `btPaga_Click()` dove all'interno troveremo dei controlli per vedere se l'id della multa selezionata è presente nel DataBase, una volta passati i controlli ci sarà una semplice query che aggiornerà lo stato iniziale della multa da False a True.

| | | | |
|----|------------|------------|------------|
| 55 | 22/08/2021 | 22/11/2021 | 30000.00 € |
| 66 | 22/08/2021 | 22/11/2021 | 123.00 € |
| 67 | 31/08/2021 | 30/11/2021 | 300.00 € |
| 69 | 22/08/2021 | 22/11/2021 | 123.00 € |
| 70 | 25/09/2021 | 25/12/2021 | 324.00 € |

Tommaso Pulici

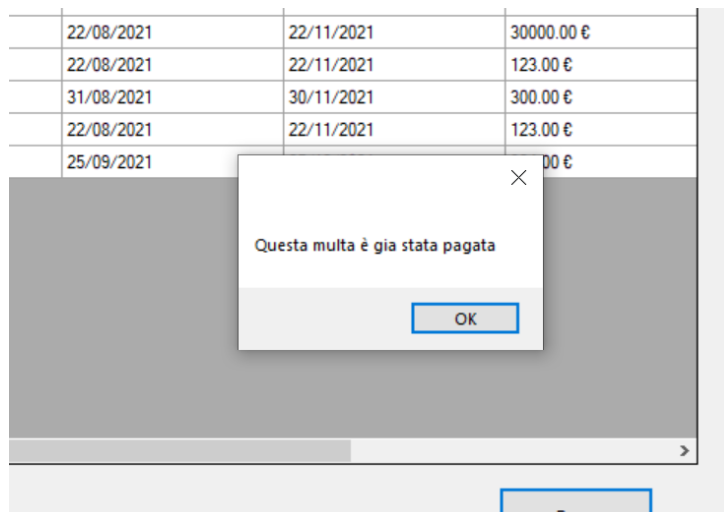
Log out

Espandi Paga

Hai pagato la multa

OK

Nel caso in cui la multa fosse già stata pagata avremo un messaggio che ci avvertirà del fatto che essa è già stata pagata in precedenza



Il resto delle funzioni come il bottone Espandi e il tasto Log out rimangono le medesime della sezione precedente dell'agente di polizia.

Descrizione dei pattern utilizzati:

All'interno del progetto sono riuscito ad implementare soltanto una tipologia di pattern, quello creazione più specificatamente il pattern Builder.

Builder

Questo pattern l'ho utilizzato per creare tutte le istanze di classi presenti nel progetto (tranne per la classe Condb, che mi serviva solo per dare la sorgente del DataBase), infatti per le classi Fine, Civil e Police esiste il corrispettivo Builder che si occupa del raccoglimento dati per queste classi.

Per implementare questo Builder ho dovuto fare delle rivisitazioni, poichè non era propriamente supportato dalla mia specifica.

Comunque mi è stato molto utile per rendere il codice più leggibile e fluente, dato che non ho dovuto usare un costruttore dove inserire semplicemente dati che potrebbero risultare illeggibili per un altro programmatore che mette mano al codice.

Il Builder ci dà un vantaggio riguardo alla riusabilità del codice e alla sua manutenzione.

```
//Inizializzazione dell'utente civile
user = builderC.buildCodFisc(reader.GetString(0))
                .buildNome(reader.GetString(1))
                .buildCognome(reader.GetString(2))
                .buildEmail(reader.GetString(3))
                .buildPassword(reader.GetString(4))
                .buildIndirizzo(reader.GetString(5), reader.GetString(6))
                .buildCellulare(reader.GetString(7))
                .Build();
```

Documentazione sull'utilizzo

Il programma in questione necessita di Sql server express per poter collegarsi al DataBase locale.

Una volta installato SqlServer bisogna recarsi nel file Condb.cs che si trova dentro "Progetto_PMO\Progetto_PMO" e cambiare il valore della variabile string "con" che si trova nel costruttore della classe Condb con il percorso adeguato

Quello attuale:

```
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\\Users\\pulic\\source\\repos\\Progetto_PMO\\Progetto_PMO\\Database.mdf;Integrated Security=True
```

Come bisogna cambiarlo:

```
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename= "Inserisci percorso DataBase" Database.mdf;Integrated Security=True
```

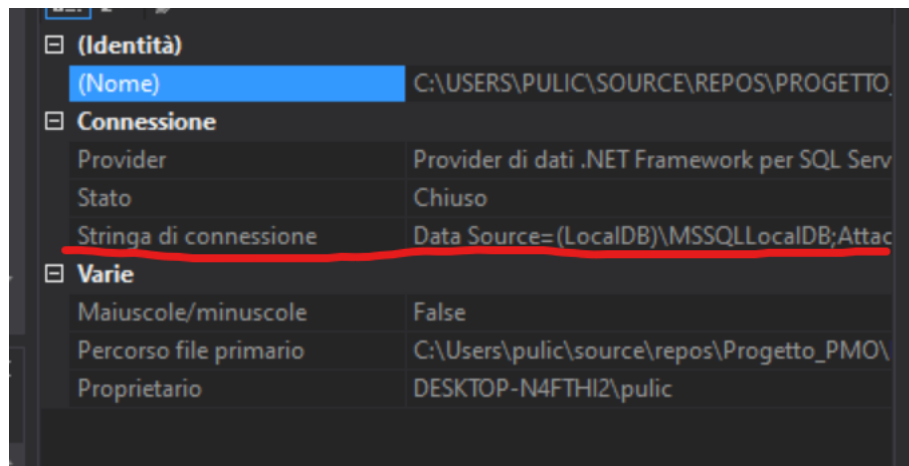
ricordandosi di mettere sempre il doppio backslash.

Una volta fatto ciò si potrà avviare il programma tramite il file eseguibile Progetto_PMO.exe che si trova in questo percorso
"Progetto_PMO\Progetto_PMO\bin\Debug"

Con visual studio:

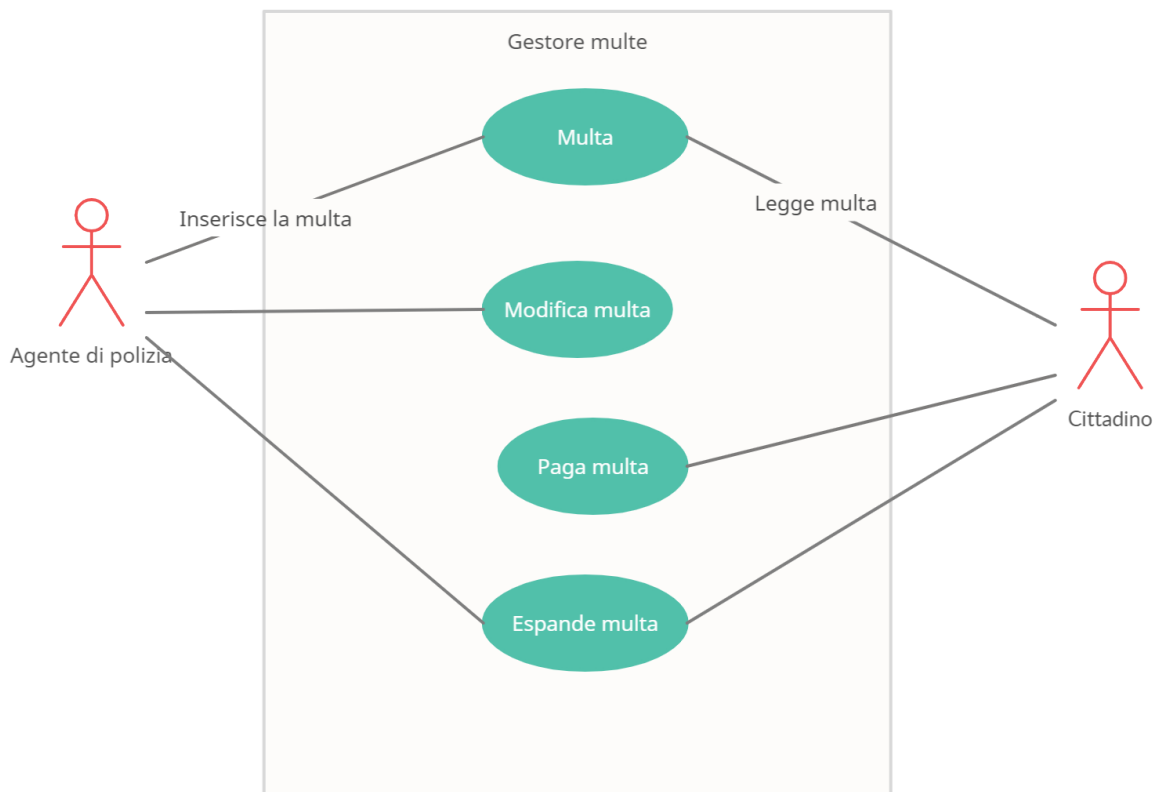
Se si utilizza visual studio c'è la possibilità di avere già installato il pacchetto SqlServer.

Bisogna comunque cambiare la stringa di connessione, a quel punto senza fare troppe ricerche si può copia incollare la stringa cliccando sul database nella sezione Esplora server e poi trovare questa:



A questo punto basterà far partire il debug e il programma sarà pronto all'utilizzo

Use case con relativo UML



Il diagramma in questione ci mostra come in questo gestionale ci sono due tipi di utenti:

- L'agente di polizia
- Il cittadino

Queste due tipologie di utenze hanno delle funzionalità.

L'agente deve inserire le multe e il cittadino le deve pagare