Diario di lavoro

Luogo	SAM Trevano
Data	03.10.2019

Lavori svolti

Prima di descrivere i passaggi svolti oggi aggiungo un passaggio che mi sono dimenticato di scrivere nello scorsa diario ma che merita di essere notato in quanto se non l'avessi fatto il progetto non avrebbe compilato. Quando ho sposato le classi dal Model del progetto WPF al Model del nuovo progetto .DA ho dovuto cambiare la definizione dei metodi di supporto Get(), Update() e Delete() da internal void a public. Questo perché nel ViewModel tali metodi non erano più accessibili dal momento che ho spostato la classe in un altro progetto.

Oggi ho modificato la view per le informazioni generali (Generale.xaml) aggiungendo tutti i campi che mi servono (come definito nella progettazione). Ho aggiunto gli stessi campi nella classe Generale per permettere il binding e la scrittura sul database. Dopodiché ho eliminato il database creato l'ultima lezione per evitare problemi con la creazione del nuovo. Quindi ho eseguito le migrazioni sul progetto QDCeValutazioni.DA (dalla console dei pacchetti NuGet, ho scelto questo come progetto predefinito). Dopo aver ricreato il database ho notato che alcuni dei campi erano Consenti Null (le stringhe) mentre altri erano Not Null (DateTime e Int). Siccome sono ancora in fase di test mi servono tutti i campi con l'opzione Null abilitata. Ho controllato nel file generato automaticamente dalle migrazioni **(QDCeValutazioni.DA – Migrations – Id_InitialCreate.cs**) e effettivamente i campi DateTime e Int avevano settata l'opzione **nullable = false**. Togliendo quell'opzione ed eseguendo i comandi Update-Database o Update-Migrations il risultato non cambiava. Grazie all'aiuto del docente ho scoperto che si può settare un campo come nullable anche dalla classe, usando il "?" dopo il formato:

```
public DateTime? DataInizio { get; set; }
```

Inoltre ho settato a true l'attributo AutomaticMigrationsEnabled (**QDCeValutazioni.DA – Migrations – Configurations.cs**) che di default all'esecuzione delle migrazioni e settato a false. Questo campo come spiegatomi dal docente, viene solitamente tenuto a false in fase di pubblicazione perché rende più sicuro il database (non vengono eliminati i dati, non si possono eseguire modifiche contemporaneamente, il database è sempre sincronizzato,…). In fase di test invece è più conveniente settarlo a true, per fare in modo che al comando Update-Migrations qualsiasi modifica venga applicata, con il rischio però di perdere alcuni dati:

```
public Configuration()
{
    AutomaticMigrationsEnabled = true;
}
```

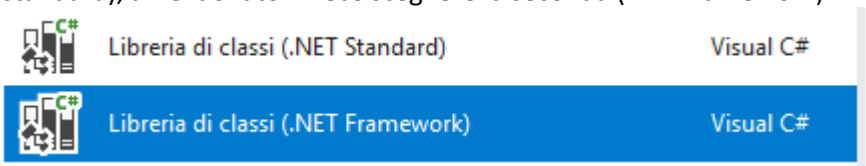
Dopo aver finito queste operazioni ho creato la view per l'inserimento di una descrizione. Questa view è composta da una label "descrizione", da un TextBox per l'inserimento della descrizione e da un bottone per salvare sul database. L'idea iniziale era quella di usare una TextArea per permettere di inserire testi lunghi e di andare a capo, però ho visto che nella mia versione di Visual Studio non esiste, quindi ho dovuto ripiegare su una text box. Per permettere di andare a capo ho dovuto inserire alcuni attributi particolari:

```
<TextBox
    TextWrapping="Wrap"
    AcceptsReturn="True"
    HorizontalScrollBarVisibility="Disabled"
    VerticalScrollBarVisibility="Auto"
    Grid.Row="1" Grid.Column="0"
    Margin="20,8,20,8"
    x:Name="DescrizionePeritoTextBox" />
```

Grazie all'inserimento degli attributi TextWrapping e AcceptsReturn è possibile andare a capo nel TextBox. Gli attributi VerticalScrollBarVisibility e HorizontalScrollBarVisibility rendono possibile scrollare all'interno del TextBox nel caso lo spazio non fosse abbastanza.
Per quanto riguarda questa view non ho usato il tool Menu Samt per creare anche il viewmodel e il model perché non mi servono. Infatti DescrizioneView si riferisce al model Generale.cs e al viewmodel MainViewModel.cs, perché l'attributo descrizione è definito nella tabella Generale, quindi altre classi non sono necessarie.
Ho creato invece nel DA la classe Requisito.cs e i relativi Services. In questa classe ho inserito i campi dei requisiti come definito nello schema ER. Non ho ancora eseguito nessun test su questa classe e non ho imlementato né il viewmodel né la view.
Per ultima cosa ho aggiornato la documentazione con quanto svolto oggi.

Problemi riscontrati e soluzioni adottate

Facendo le migrazioni mi generava un errore ***object reference not set to an instance of an object c#.*** Cercando su StackOverflow qualche possibile soluzione ho capito di aver sbagliato a creare il progetto QDCeValutazioni.DA. Quando ho creato il progetto ho scelto la prima Library Class disponibile (NET standard), avrei dovuto invece scegliere la seconda (NET Framework).



Ho quindi dovuto cambiare il progetto (ne ho creato uno nuovo, ho copiato i files e ho eliminato quello sbagliato).

Punto della situazione rispetto alla pianificazione

Sono leggermente in ritardo anche a causa di un assenza la scorsa lezione (27.09.2019). Probabilmente riuscirò a recuperare senza dover recuperare ore extra-scolastiche, nel caso fosse necessario comunque il martedì mattina ho due ore liberi a disposizione. Al momento comunque guardando il gantt il ritardo non è per nulla grave.

Programma di massima per la prossima giornata di lavoro

Creare la view per i requisiti. Fare tutti i test necessari sul database dalle varie view. Per quanto riguarda l'integrazione tra le view ho deciso di non farlo ora in quanto non è una priorità. Aggiornre la documentazione e il gantt consuntivo.