

Gestione Esercizi | Diario di lavoro - 03.09.2019

Gabriele Alessi

Canobbio, 03.09.2019

Lavori svolti

I docenti ci hanno esposto i vari progetti e ognuno ha scelto il proprio. Il mio responsabile è il docente Bernasconi e durante questo semestre lavorerò allo sviluppo di un'applicazione che gestisce gli esercizi delle prove formandone un documento.

Dopo che abbiamo scelto ci è stato consegnato il QdC e l'abbiamo analizzato per poi porre le domande (<https://github.com/gabrialessi/GestioneEsercizi/blob/master/Analisi/Domande.md>).

Infine ho preparato il repository e iniziato a creare le cartelle e i files iniziali.

Link al repository: <https://github.com/gabrialessi/GestioneEsercizi>

Orario	Lavori svolti
13:15 - 16:30	Analisi

Problemi riscontrati e soluzioni adottate

Nessun problema riscontrato.

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione.

Programma di massima per la prossima giornata di lavoro

Analisi del progetto.

Gestione Esercizi | Diario di lavoro - 05.09.2019

Gabriele Alessi

Canobbio, 05.09.2019

Lavori svolti

Per la maggior parte della giornata mi sono occupato di inserire le informazioni principali nella documentazione (capitolo 1) e nel frattempo ho riguardato il QdC analizzando il progetto.

Quindi mi sono venute in mente i seguenti quesiti, ai quali il docente mi ha risposto in giornata:

- Le impostazioni di base sono sempre fisse?
Non proprio, alla fine c'è dietro un DB quindi ogni docente può avere le sue impostazioni.
- Cosa bisogna poter inserire nei dettagli dell'esercizio? Titolo, testo, immagine?
Di base sì, ci dovrebbero essere dei moduli nel formato RFT che permette di inserire le informazioni dell'esercizio in modo semplice e veloce.
- Come funziona esattamente la creazione della prova? Campi fondamentali? Inserimento esercizi?
La prova esce sotto forma di documento stampabile alla fine dell'inserimento degli esercizi, i quali vanno memorizzati anch'essi, quindi si inseriscono i campi fondamentali (data, classe, modulo, ...).

In concreto questi sono i capitoli della documentazione completati: 1.1, 1.3, 1.4, 1.5

Questa è la prima versione dello scopo del progetto scritto nella documentazione:

Il progetto consiste nel sviluppare una piccola applicazione che gestisca e crei gli esercizi delle prove per poi prepararne il documento. Il programma deve funzionare in modo che si possano inserire delle informazioni di base: definizione di moduli, tematiche e classi.

In seguito si gestiscono i veri e propri esercizi, in cui si ha una visione generale delle impostazioni di base al fine di iniziare a inserire i dettagli dell'esercizio (titolo, testo, immagine, ...).

Infine si passa alla creazione del documento, dove vanno definiti i campi fondamentali e si selezionano gli esercizi da inserire nella prova.

Questa è il primo requisito di base che ho sviluppato:

ID: REQ-001

Nome	Realizzare un programma che gestisca gli esercizi
Priorità	1
Versione	1.0
Note	-

ID: REQ-001

Sotto requisiti

- | | |
|-----|---------------------------------------------------------------------|
| 001 | È necessario poter inserire delle informazioni di base (REQ-002) |
| 002 | Maschera di inserimento delle informazioni dell'esercizio (REQ-003) |
| 003 | Creazione delle prove (REQ-004) |

Intanto ho provato a scaricare MS Project così da poter iniziare la pianificazione sviluppando un Gantt preventivo.

Orario Lavori svolti

-
- | | |
|---------------|--------------------------|
| 13:15 - 16:30 | Analisi e Documentazione |
|---------------|--------------------------|

Problemi riscontrati e soluzioni adottate

Nessun problema riscontrato.

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione.

Programma di massima per la prossima giornata di lavoro

Analisi, documentazione, pianificazione.

Gestione Esercizi | Diario di lavoro - 06.09.2019

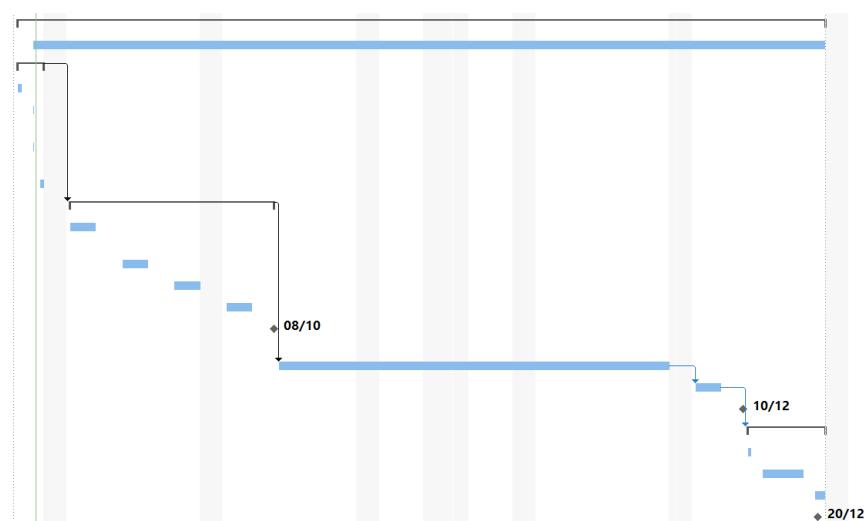
Gabriele Alessi

Canobbio, 06.09.2019

Lavori svolti

Durante questa giornata mi sono occupato inizialmente di sviluppare la pianificazione del progetto tramite la realizzazione di un diagramma di Gantt con MS Project.

Gestione Esercizi	180 hrs	Tue 03/09/19	Fri 20/12/19
Documentazione	176 hrs	Thu 05/09/19	Fri 20/12/19
Analisi	12 hrs	Tue 03/09/19	Fri 06/09/19
Analisi QdC	4 hrs	Tue 03/09/19	Tue 03/09/19
Analisi del dominio, Analisi dei mezzi, Scopo	2 hrs	Thu 05/09/19	Thu 05/09/19
Analisi e specifica dei requisiti	2 hrs	Thu 05/09/19	Thu 05/09/19
Pianificazione	4 hrs	Fri 06/09/19	Fri 06/09/19
Progettazione	48 hrs	Tue 10/09/19	Tue 08/10/19
Design dell'architettura del sistema	12 hrs	Tue 10/09/19	Fri 13/09/19
Design dei dati e database	12 hrs	Tue 17/09/19	Fri 20/09/19
Design delle interfacce	12 hrs	Tue 24/09/19	Fri 27/09/19
Design procedurale	12 hrs	Tue 01/10/19	Fri 04/10/19
Fine progettazione, inizio implementazione	0 hrs	Tue 08/10/19	Tue 08/10/19
Implementazione	84 hrs	Tue 08/10/19	Fri 29/11/19
Test	12 hrs	Tue 03/12/19	Fri 06/12/19
Inizio consegna del progetto	0 hrs	Tue 10/12/19	Tue 10/12/19
Conclusione progetto	24 hrs	Tue 10/12/19	Fri 20/12/19
Consuntivo	4 hrs	Tue 10/12/19	Tue 10/12/19
Presentazione	12 hrs	Thu 12/12/19	Tue 17/12/19
Conclusione e consegna	8 hrs	Thu 19/12/19	Fri 20/12/19
Consegna progetto	0 hrs	Fri 20/12/19	Fri 20/12/19



Dopo aver fatto ciò ho rifinito la documentazione in modo da concludere l'analisi del progetto ed essere pronto per iniziare la progettazione.

Infine ho reinstallato MS Visio così da realizzare lo schema dei casi d'uso, ma ho già l'idea di come realizzarlo (https://moodle.edu.ti.ch/cpt/pluginfile.php/7717/mod_folder/content/0/Esempi%20Use%20Case.pdf?forcedownload=1).

Orario Lavori svolti

13:15 - 16:30 Pianificazione, Analisi e Documentazione

Problemi riscontrati e soluzioni adottate

All'inizio ho avuto problemi con MS Project in quanto ho iniziato il progetto senza configurare le impostazioni corrette.

Dopo essermi consultato con qualche compagno e docente sono riuscito a impostare il file e ho completato il Gantt.

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione.

Programma di massima per la prossima giornata di lavoro

Progettazione (design dell'architettura del sistema), documentazione.

Gestione Esercizi | Diario di lavoro - 10.09.2019

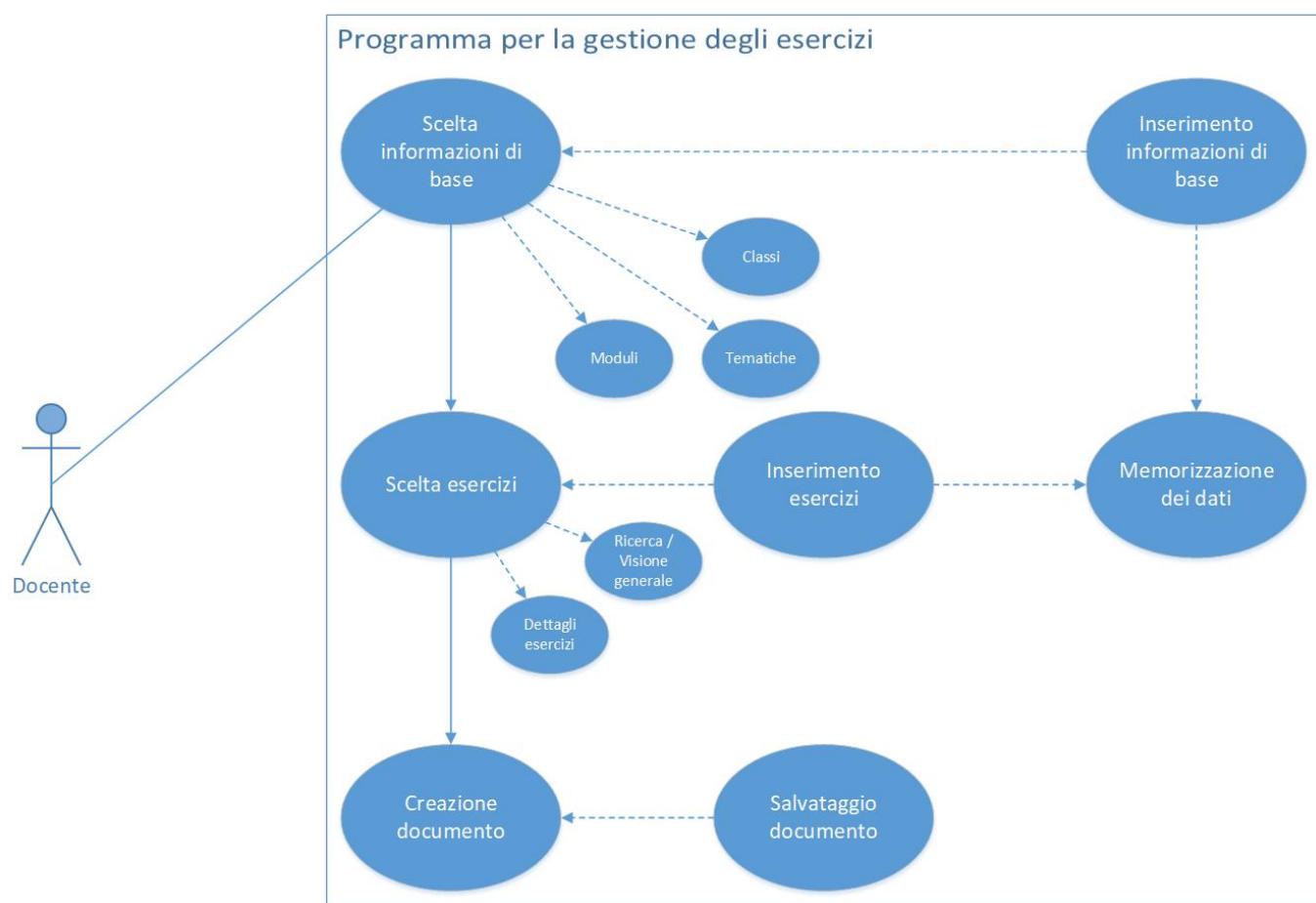
Gabriele Alessi

Canobbio, 10.09.2019

Lavori svolti

Durante questa giornata ho prima di tutto riguardato velocemente e poi concluso l'analisi del progetto (facendo anche lo use case), quindi ho cominciato a pensare al capitolo della progettazione.

Questo è lo schema per i casi d'uso che ho realizzato:



Per il resto della lezione ho lavorato al design dell'architettura del sistema. In particolare ho cominciato il diagramma delle classi del prodotto, mettendo giù gli elementi principali.

ImpostazioniBase
-moduli -tematiche -classi

Esercizio
-immagini -testo

Prova
-esercizi +Salva()

A fine giornata ho deciso di lasciare in sospeso il diagramma delle classi per fare lo schema del database. In questo modo sarà più facile vedere il progetto nel suo complesso e definire le classi.
In generale ora ho un'idea più chiara del sistema quindi durante le prossime occasioni dovrebbe iniziare a formarsi qualcosa di concreto.

Orario	Lavori svolti
13:15 - 16:30	Progettazione e Documentazione

Problemi riscontrati e soluzioni adottate

Nessun problema riscontrato.

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione.

Programma di massima per la prossima giornata di lavoro

Progettazione (database), documentazione.

Gestione Esercizi | Diario di lavoro - 12.09.2019

Gabriele Alessi

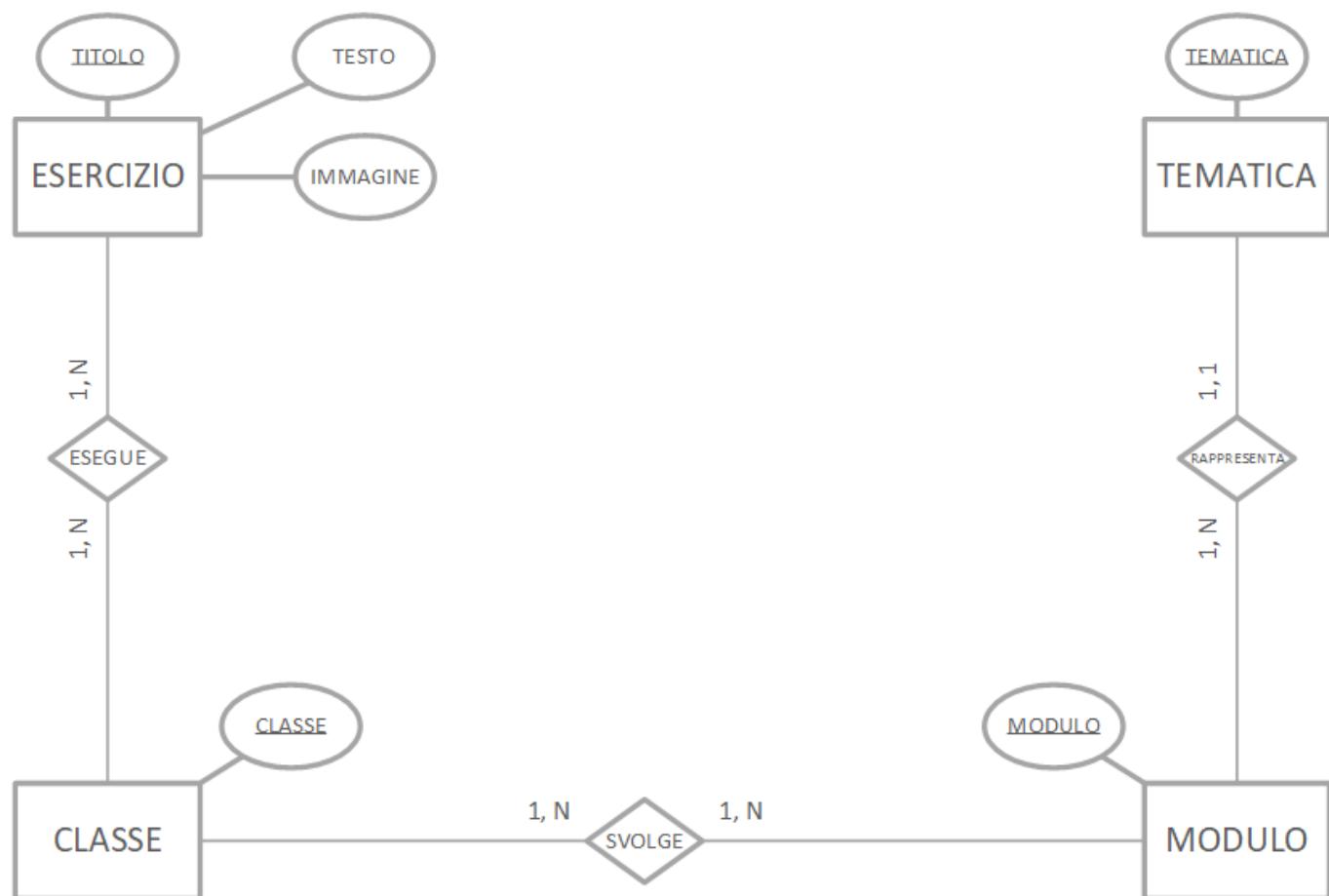
Canobbio, 12.09.2019

Lavori svolti

All'inizio di questa giornata mi sono subito preparato per progettare il database dell'applicazione. Per pensare a ciò ho riguardato il QdC e la documentazione che sviluppato finora. In sintesi questo sarà il modo in cui realizzerò lo schema:

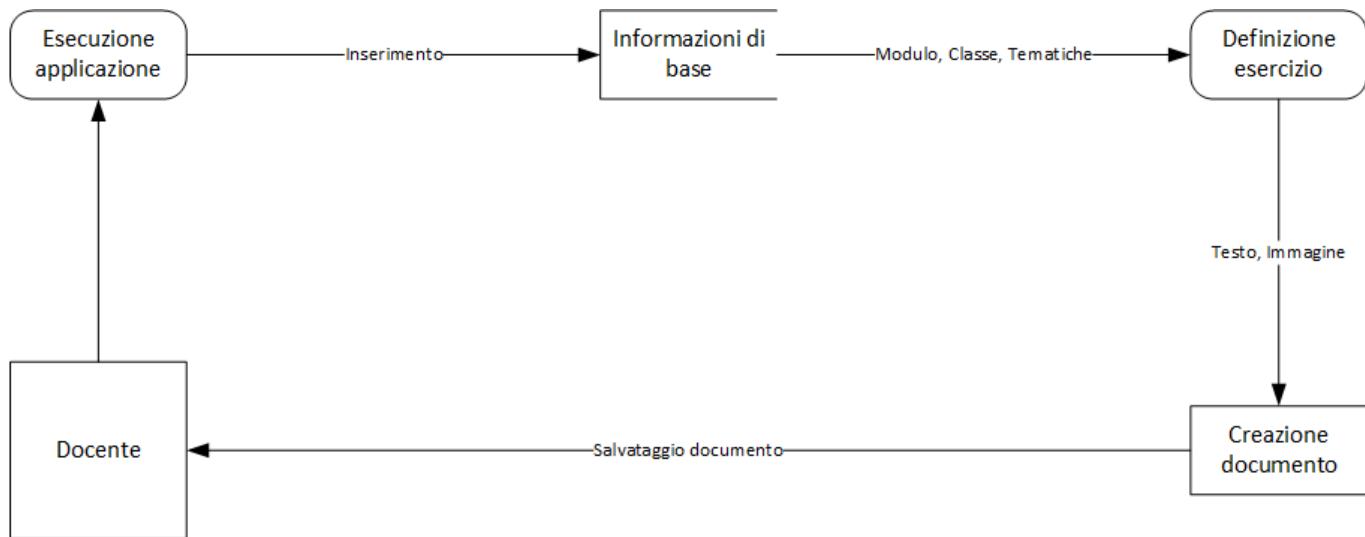
- Definizione delle entità
- Definizione degli attributi
- Definizione delle relazioni
- Completamento dello schema (cardinalità, primary key, ...)
- Schema logico

Questo è il diagramma progettato:



Prossimamente farò controllare lo schema al supervisore in modo da sapere che eventuali modifiche apportare.

Dopo aver fatto ciò ho iniziato il diagramma di flusso dei dati e il risultato è il seguente:



Questi schemi sono abbastanza grezzi quindi aspetto il parere del responsabile per sapere come muovermi in futuro.

Orario	Lavori svolti
---------------	----------------------

13:15 - 16:30 Progettazione e Documentazione

Problemi riscontrati e soluzioni adottate

La fase di progettazione per me è un po' lenta quindi ho avuto problemi a ideare gli schemi e il sistema in generale. Alla fine sono riuscito a concludere qualcosa ma spero di non ritrovarmi indietro con la tabella di marcia in futuro.

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione.

Programma di massima per la prossima giornata di lavoro

Progettazione, documentazione.

Gestione Esercizi | Diario di lavoro - 13.09.2019

Gabriele Alessi

Canobbio, 13.09.2019

Lavori svolti

Oggi ho continuato a lavorare principalmente sulla progettazione del progetto. Purtroppo il responsabile non è stato presente quindi non ho potuto consultarlo riguardo ai diagrammi sviluppati durante la giornata precedente. Quindi ho prima di tutto riguardato la documentazione per riprendere il punto della situazione e così ho deciso di cominciare a pensare al vero e proprio progetto di C# in modo da preparare il capitolo riguardante il design dell'architettura del sistema.

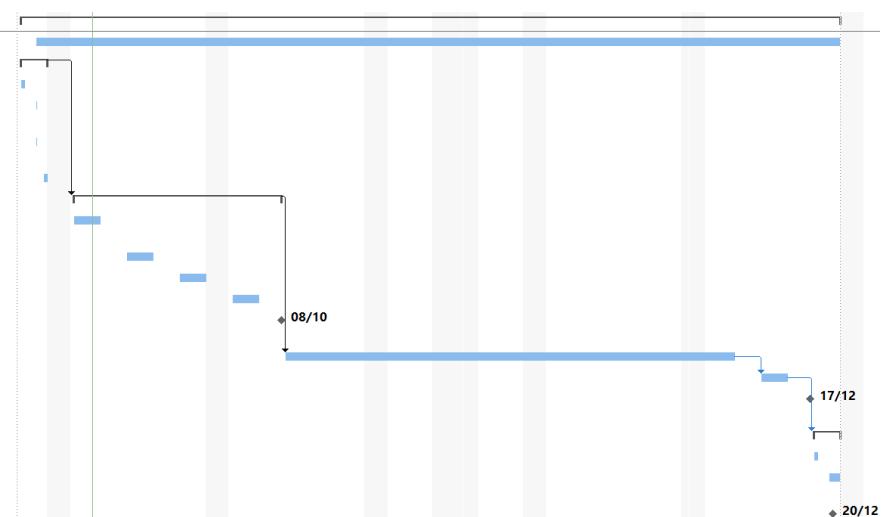
Qui di seguito un estratto di quest'ultimo:

Il programma è sviluppato in C# utilizzando il pattern MVVM, ma ci sono altri componenti per semplificare il lavoro che sono spiegati nel capitolo di implementazione. La struttura del progetto parte da una Soluzione Vuota che contiene un'app WPF dove vengono gestite le interfacce e una libreria di classi in cui vengono gestiti i dati.

Successivamente è intervenuto il docente Valsangiacomo per farci notare che nella pianificazione non andava inclusa la presentazione, dunque ho fatto delle piccole modifiche per togliere l'attività.

Questo è il risultato definitivo:

Gestione Esercizi		180 hrs	Tue 03/09/19	Fri 20/12/19
Documentazione	176 hrs		Thu 05/09/19	Fri 20/12/19
Analisi	12 hrs		Tue 03/09/19	Fri 06/09/19
Analisi QdC	4 hrs		Tue 03/09/19	Tue 03/09/19
Analisi del dominio, Analisi 2 hrs dei mezzi, Scopo			Thu 05/09/19	Thu 05/09/19
Analisi e specifica dei requisiti	2 hrs		Thu 05/09/19	Thu 05/09/19
Pianificazione	4 hrs		Fri 06/09/19	Fri 06/09/19
Progettazione	48 hrs		Tue 10/09/19	Tue 08/10/19
Design dell'architettura del sistema	12 hrs		Tue 10/09/19	Fri 13/09/19
Design dei dati e database	12 hrs		Tue 17/09/19	Fri 20/09/19
Design delle interfacce	12 hrs		Tue 24/09/19	Fri 27/09/19
Design procedurale	12 hrs		Tue 01/10/19	Fri 04/10/19
Fine progettazione, inizio implementazione	0 hrs		Tue 08/10/19	Tue 08/10/19
Implementazione	96 hrs		Tue 08/10/19	Fri 06/12/19
Test	12 hrs		Tue 10/12/19	Fri 13/12/19
Inizio conclusione del progetto	0 hrs		Tue 17/12/19	Tue 17/12/19
Conclusioni progetto	12 hrs		Tue 17/12/19	Fri 20/12/19
Consuntivo	4 hrs		Tue 17/12/19	Tue 17/12/19
Conclusione e revisione generale	8 hrs		Thu 19/12/19	Fri 20/12/19
Consegna progetto	0 hrs		Fri 20/12/19	Fri 20/12/19



Per il resto del tempo guardato un po' come fare per la creazione del progetto. Per quanto riguarda ciò ho avuto problemi visto che dovrei chiedere come impostare la base di progetto che ci è stata consegnato l'anno scorso in modo da avere la struttura già pronta.

Orario

Lavori svolti

13:15 - 16:30 Progettazione e Documentazione

Problemi riscontrati e soluzioni adottate

Nessun problema riscontrato

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione.

Programma di massima per la prossima giornata di lavoro

Progettazione, documentazione.

Gestione Esercizi | Diario di lavoro - 17.09.2019

Gabriele Alessi

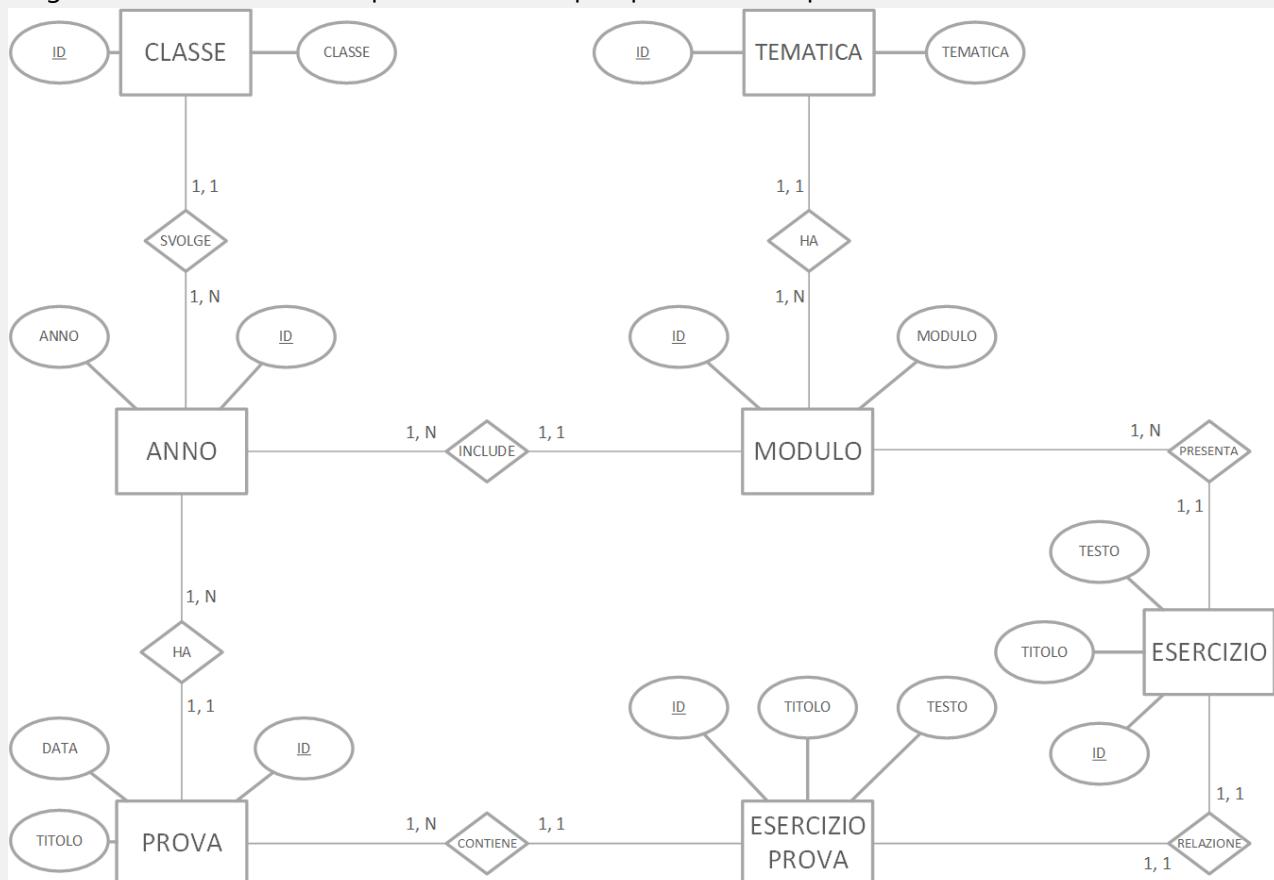
Canobbio, 17.09.2019

Lavori svolti

Durante questa giornata ho continuato principalmente con la progettazione e la documentazione del progetto. Inizialmente mi sono consultato con il supervisore per quanto riguarda gli schemi e diagrammi che ho realizzato finora. Quindi ho provveduto a riguardarli per metterli a posto e infine ho cominciato a pensare al design delle interfacce del prodotto.

Il seguente è il database aggiornato:

L'immagine seguente rappresenta la progettazione del database del sistema. In generale non è un diagramma molto articolato quindi da esso si può più o meno capire anche la struttura del sistema.



Le entità come tematica, classe, anno e modulo presenteranno bene o male sempre gli stessi valori in quanto rappresentano le impostazioni di base. Dopodiché ci sono le entità che simboleggiano gli esercizi e le prove, le quali sono relazionate tramite l'anno e il modulo. L'oggetto esercizio prova funge da collegamento tra prova e esercizio in modo che sia più facile gestire gli esercizi che si trovano nelle prove.

Ora che ho fatto il colloquio ho una visione molto più chiara del sistema del programma. Inoltre durante le prossime giornate mi verrà fornita la base del progetto Visual Studio in modo da sapere esattamente come disporre gli elementi del progetto.

In più ho anche rivisto lo schema del design procedurale e so come mettere a posto anche quello dopo aver rifatto il database.

Orario	Lavori svolti
13:15 - 16:30	Progettazione e Documentazione

Problemi riscontrati e soluzioni adottate

Nessun problema riscontrato.

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione.

Concluso 2.2 Design dei dati e database

Programma di massima per la prossima giornata di lavoro

Progettazione, documentazione.

Gestione Esercizi | Diario di lavoro - 19.09.2019

Gabriele Alessi

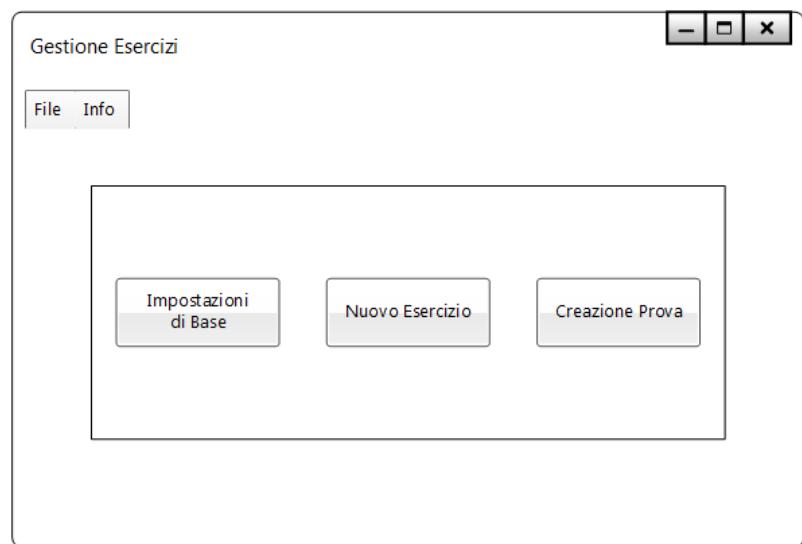
Canobbio, 19.09.2019

Lavori svolti

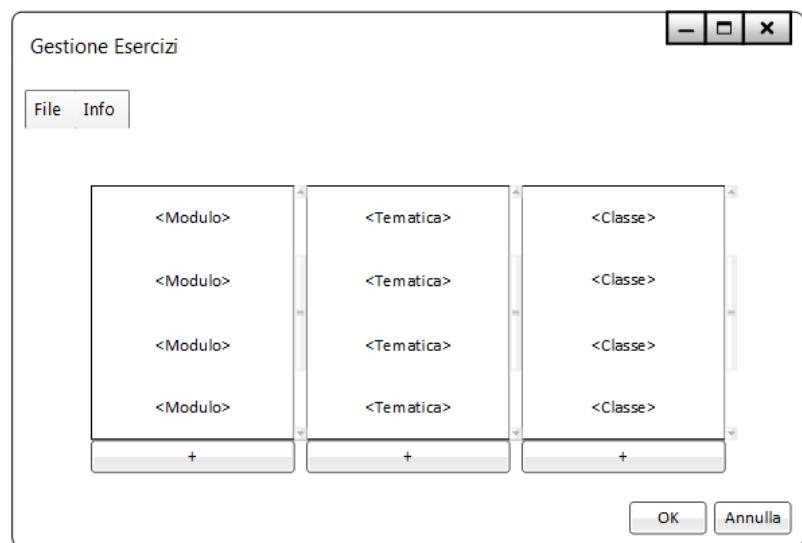
Durante questa giornata mi sono principalmente occupato di progettare il design delle interfacce del programma.

La prima che ho sviluppato è stata la schermata di base:

La prima schermata è molto semplice e presenta solamente tre pulsanti che rappresentano le tre azioni principali del programma.



Successivamente ho concluso anche l'interfaccia che gestisce le impostazioni di base:



Questa schermata è composta da tre menu in cui si può scorrere per vedere i dati inseriti (moduli, tematiche e classi) e i relativi pulsanti per aggiungere un nuovo elemento.

L'interfaccia sembra molto semplice, ma dietro ci sono anche alcune funzionalità che permettono di collegare queste tre entità tra loro e gestirle:

- Doppio click su una classe
Mostra i moduli che essa svolge e permette di gestirli.
- Doppio click su un modulo
Mostra le tematiche che esso comprende e permette di gestirli.
- Click + delete su un elemento lo elimina.

Infine ci sono i soliti pulsanti che consentono di annullare o applicare le eventuali modifiche apportate.

Orario	Lavori svolti
13:15 - 16:30	Progettazione e Documentazione

Dovrei consultarmi con il responsabile per quanto riguarda il collegamento degli elementi nelle impostazioni di base (una classe che moduli segue e un modulo che tematiche presenta). In generale sto riuscendo a sviluppare qualcosa ma non sono sicuro su come progettare alcuni aspetti che si trovano nelle impostazioni di base.

In linea con la pianificazione.

Progettazione, documentazione.

Gestione Esercizi | Diario di lavoro - 20.09.2019

Gabriele Alessi

Canobbio, 20.09.2019

Lavori svolti

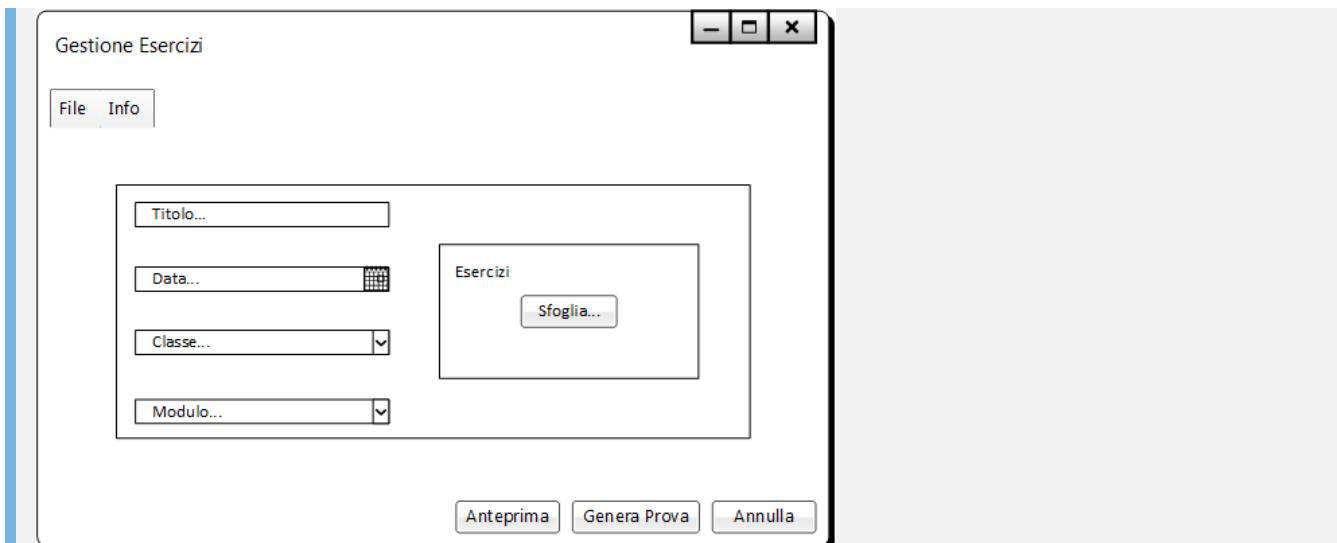
Durante questa giornata ho concluso la progettazione del design delle interfacce del programma. Questa è la schermata che crea gli esercizi:



La prima maschera consente di attribuire un titolo all'esercizio e di scegliere tramite un menu a tendina le informazioni di base stabilite nell'applicazione.

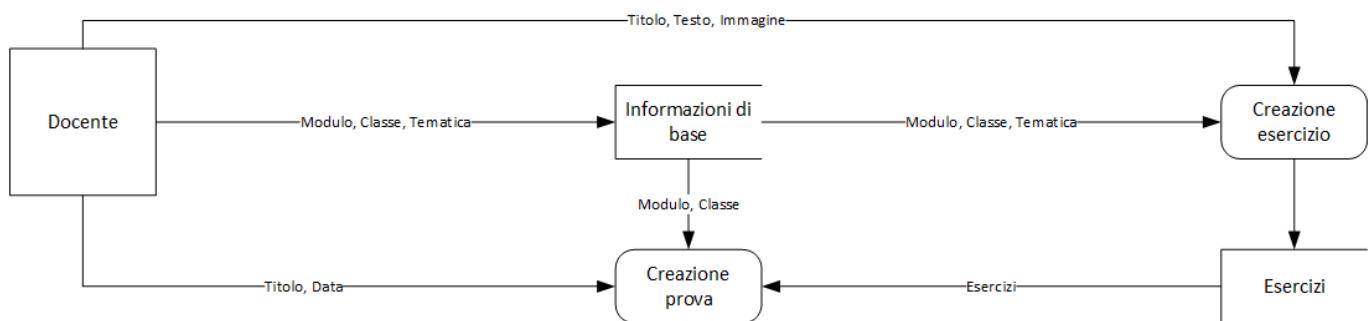
Nell'altra maschera è possibile comporre il testo dell'esercizio e inserire un'immagine attraverso Drag & Drop oppure sfogliando tra i file. Se si clicca "OK" il programma genererà un file RFT con il contenuto del testo e dell'immagine.

L'ultima interfaccia è quella che riguarda la creazione di una prova:



Nel primo campo si scrive il titolo della prova, nel secondo si inserisce la data (anche con l'aiuto del calendario), poi si inseriscono classe e modulo, che vengono estrapolati sempre dalle impostazioni di base. Finalmente si aggiungono gli esercizi tramite Drag & Drop o sfogliando tra i propri file. Per avere un'anteprima della prova esiste un apposito pulsante che mostrerà un'altra finestra con ciò che produrrà il programma se si dovesse generare il file.

A fine giornata ho avuto anche il tempo di rifare il diagramma di flusso dei dati:



Orario

Lavori svolti

13:15 - 16:30 Progettazione e Documentazione

Problemi riscontrati e soluzioni adottate

Nessun problema riscontrato.

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione.

Programma di massima per la prossima giornata di lavoro

Progettazione, documentazione.

Gestione Esercizi | Diario di lavoro - 24.09.2019

Gabriele Alessi

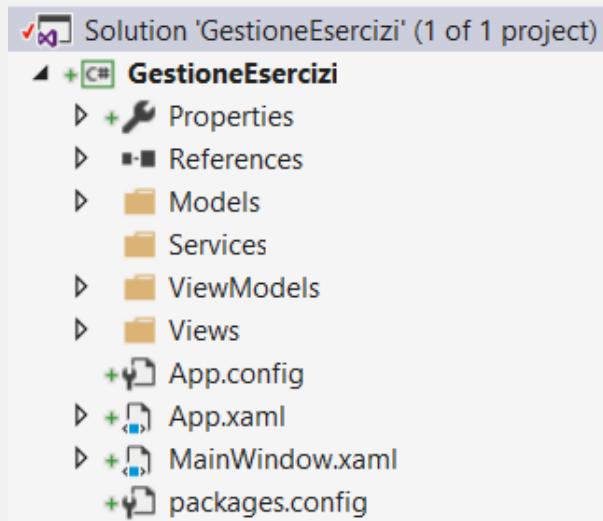
Canobbio, 24.09.2019

Lavori svolti

In questa giornata ho continuato con la progettazione. Ho principalmente visto il design dell'architettura del sistema in quanto ho ricevuto la base del progetto Visual Studio. Inoltre ho anche fatto valutare tutti gli schemi e diagrammi sviluppati finora e sono tutti a posto. In sintesi prossimamente concluderò il capitolo di progettazione.

Qui di seguito ci sono alcuni estratti di ciò che ho fatto concretamente oggi:

La struttura del progetto parte da una Soluzione Vuota che contiene un progetto App WPF il quale gestisce l'intero sistema di dati e interfacce.



Infine nella cartella Services vengono gestiti i DataRepository, cioè tutto ciò che riguarda la struttura dei dati.

In concreto ci saranno delle interfacce e delle classi che impostano le azioni che verranno sui dati (insert, delete, update).

```
public interface IDataRepository<T> where T : BaseEntity
{
    T Get(int id);
    IQueryable<T> Get();
    T Insert(T entity);
    void Update(T entity);
    void Delete(T entity);
}
```

Per il database verrà usato SQLite visto che funziona in modo facile e utilizza semplicemente un file per la memorizzazione dei dati.

Orario	Lavori svolti
---------------	----------------------

13:15 - 16:30	Progettazione e Documentazione
---------------	--------------------------------

Problemi riscontrati e soluzioni adottate

Nessun problema riscontrato.

Punto della situazione rispetto alla pianificazione

In anticipo di circa una settimana con la pianificazione.

Concluso i seguenti capitoli:

- 2.2 Design dei dati e database
- 2.4 Design procedurale

Mancano solo da concludere alcuni dettagli nel design delle interfacce (gestione dei collegamenti tra moduli, classi e tematiche) e manca poco anche alla conclusione del design del sistema.

Programma di massima per la prossima giornata di lavoro

Conclusione progettazione, documentazione.

Gestione Esercizi | Diario di lavoro - 26.09.2019

Gabriele Alessi

Canobbio, 26.09.2019

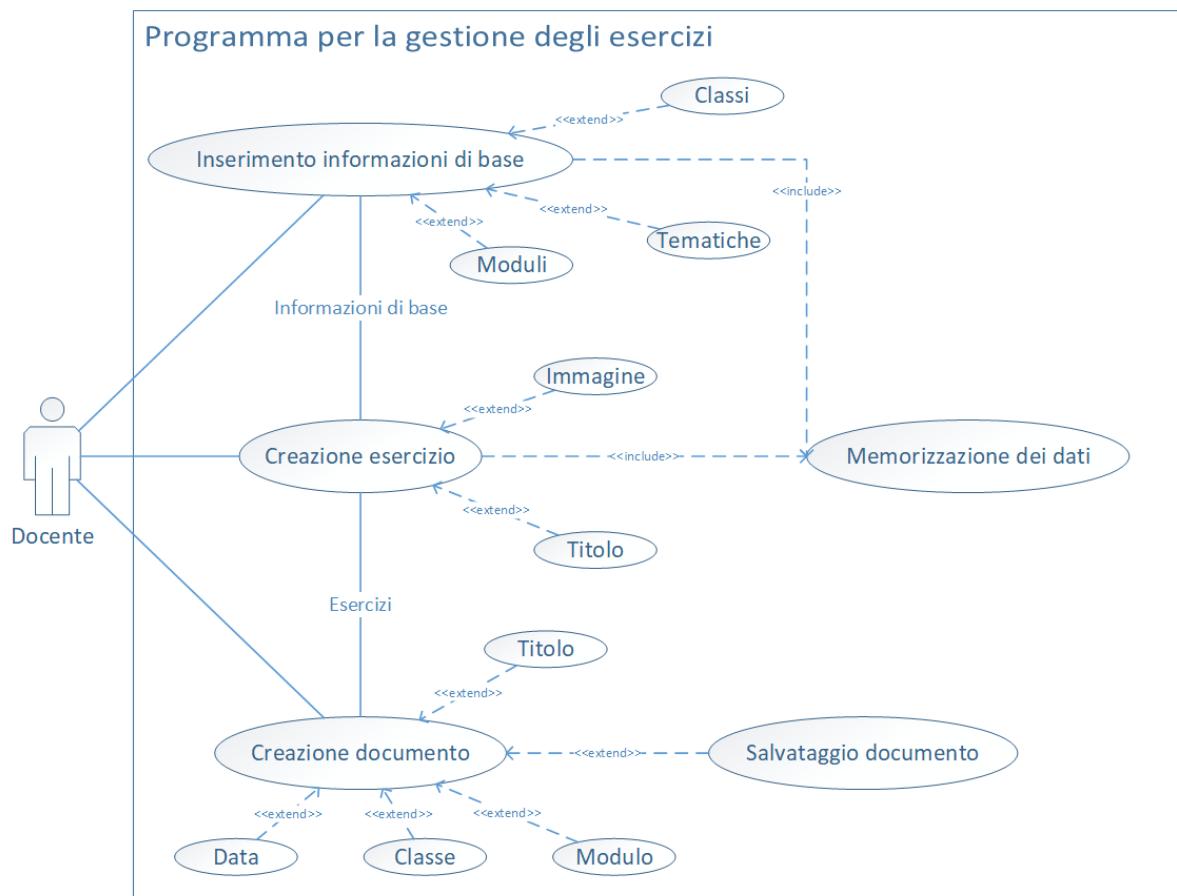
Lavori svolti

Durante questa giornata ho lavorato in modo da concludere la progettazione e il resto del progetto così da poter iniziare l'implementazione dell'applicazione.

Prima di tutto ho sviluppato l'abstract:

The teachers of the Scuola d'Arti e Mestieri di Trevano use to create the exercises and the tests without any type of help from a tool or system. This project consists of creating a computer program that allows you to create and manage the exercises and the tests quickly and easily. In detail, the program works so you can set up the basic info (subjects, classes and themes) to then use them to create the exercises. When you create an exercise, you have a global vision of the basic settings and then you can define the text and eventually insert an image. Finally, you can create the document by entering the fundamental fields (title, date, class and subject) and selecting the related exercises. The application is entirely developed in C# MVVM with Visual Studio 2019 and for the data storage, SQLite is used.

Inoltre dopo la progettazione ho avuto un'idea più chiara anche per quanto riguarda lo use case, quindi questa è la versione sistemata definitiva:



Infine ho approfittato per iniziare il progetto su Visual Studio e definire le basi dell'applicazione. In concreto ho creato una Soluzione Vuota con un App WPF che gestirà l'intero sistema e ho iniziato a creare le classi modello principali (classe, modulo, tematica). Prossimamente dovrò definire l'intero progetto possibilmente anche dopo aver consultato il supervisore così da iniziare la vera implementazione dei moduli.

Orario	Lavori svolti
13:15 - 16:30	Conclusione progettazione e Documentazione

Problemi riscontrati e soluzioni adottate

Nessun problema riscontrato.

Punto della situazione rispetto alla pianificazione

In anticipo di circa una settimana con la pianificazione.

Programma di massima per la prossima giornata di lavoro

Implementazione, documentazione.

Gestione Esercizi | Diario di lavoro - 27.09.2019

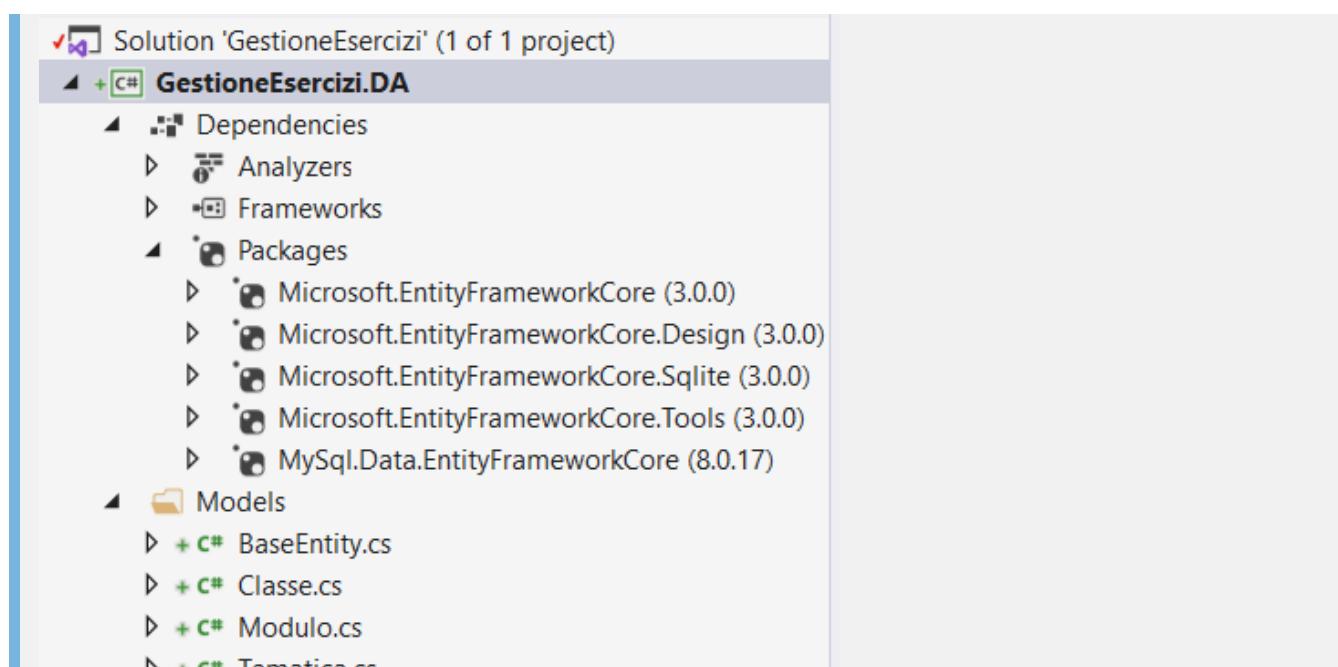
Gabriele Alessi

Canobbio, 27.09.2019

Lavori svolti

Durante questa giornata ho lavorato all'implementazione della base del progetto su Visual Studio. Fondamentalmente ho già deciso di cambiare da quanto progettato perché dopo aver visto i progetti dell'anno scorso ho optato per la creazione di una libreria di classi (GestioneEsercizi.DA) in cui inserire tutte le classi di base del database.

Un altro grosso cambiamento è il fatto di usare .NET Core, in quanto dopo aver visto l'articolo "[Differenze .NET Core vs .NET Framework](#)" e i vecchi progetti scolastici ho pensato che sarebbe più semplice.



Per iniziare si crea una Soluzione Vuota che fungerà da contenitore generale, quindi si inseriscono i progetti del sistema .NET Core, cioè una App WPF e una Libreria di Classi. Il prossimo passo è l'implementazione dei pacchetti necessari per la gestione dei dati tramite il gestore NuGet.

Questa è la nuova struttura di base, dove ci sono le classi di base e dove sono presenti tutti i pacchetti necessari per la gestione dei dati.

Inoltre ho iniziato a sviluppare le classi riguardanti il database (DataRepository e DbContext) usando come punto di riferimento i progetti dell'anno scorso del modulo 223.

```
public interface IDataRepository<T> where T : BaseEntity
{
    1 reference
    T Get(int id);
    2 references
    IQueryable<T> Get();
    1 reference
    T Insert(T entity);
    1 reference
    void Update(T entity);
    1 reference
    void Delete(T entity);
}

public class AppDbContext : DbContext
{
    0 references
    public DbSet<Classe> Classi { get; set; }

    0 references
    public DbSet<Modulo> Moduli { get; set; }

    0 references
    public DbSet<Tematica> Tematiche { get; set; }
}
```

Infine ho rifinito la documentazione inserendo le nuove informazioni nel capitolo di implementazione e definendo i nuovi sottocapitoli.

Per un'ora non ho potuto lavorare perché sono stato impegnato con una riunione riguardo lo stage all'estero.

Orario	Lavori svolti
13:15 - 14:45 / 15:45 - 16:30	Implementazione e Documentazione

Problemi riscontrati e soluzioni adottate

Nessun problema riscontrato.

Punto della situazione rispetto alla pianificazione

In anticipo di circa una settimana con la pianificazione.

Programma di massima per la prossima giornata di lavoro

Implementazione, documentazione.

Gestione Esercizi | Diario di lavoro - 01.10.2019

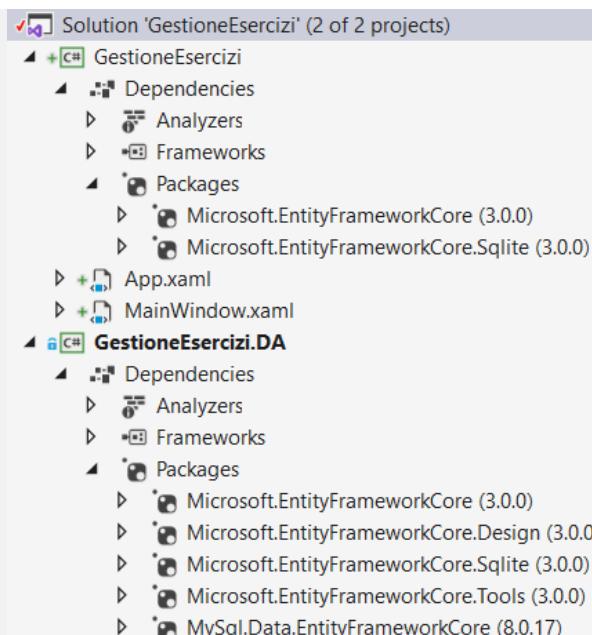
Gabriele Alessi

Canobbio, 01.10.2019

Lavori svolti

Durante questa giornata mi sono principalmente concentrato sullo sviluppo del programma.

Inizialmente mi sono consultato con il docente per fare il punto della situazione e guardare un po' il lavoro fatto finora (il fatto di usare .NET Core e la struttura usata l'anno scorso). A quanto pare è tutto a posto e lo sviluppo procede in maniera soddisfacente, quindi ho continuato implementando gli ultimi modelli di dati (Anno, Prova, Esercizio e EsercizioProva). Ho lavorato anche sulla documentazione, aggiornando il capitolo di creazione progetto e concludendo il capitolo dei Models.



Dall'immagine si può vedere che i pacchetti utilizzati sono quelli inerenti a EntityFrameworkCore, infatti questi sono utili per l'accesso ai dati e la gestione del database (in questo caso SQLite).

La Libreria di Classi (.NET Core) è uno dei progetti della soluzione Visual Studio e contiene tutto ciò che ha a che fare con la gestione dei dati e la definizione del database.

La cartella Models contiene le entità di base del sistema che rappresentano i dati, quindi per fare ciò si fa riferimento al diagramma del database.

BaseEntity è la superclasse principale ed è la base di ogni altra entità, infatti essa contiene solo l'Id, che è un campo comune a ogni oggetto.

```
public class BaseEntity
{
    public int Id { get; set; }
}
```

La classe Modulo presenta i campi che descrivono il nome (Modulo x), l'anno, le tematiche e gli esercizi.

```
public class Modulo : BaseEntity
{
    public string Nome { get; set; }
    public Anno Anno { get; set; }
    public ICollection<Tematica> Tematiche { get; set; }
    public ICollection<Esercizio> Esercizi { get; set; }
}
```

La classe Anno è principalmente utilizzata per collegare una classe e un modulo con una prova. Fondamentalmente la classe include l'annata e gli insiemi di classi, moduli e prove.

```
public class Anno : BaseEntity
{
    public string Annata { get; set; }
    public ICollection<Classe> Classi { get; set; }
    public ICollection<Modulo> Moduli { get; set; }
    public ICollection<Prova> Prove { get; set; }
}
```

Per la prossima giornata cercherò di completare anche il capitolo riguardo i Services, quindi le interfacce che interagiscono con il database.

Orario	Lavori svolti
13:15 - 16:30	Implementazione e Documentazione

Problemi riscontrati e soluzioni adottate

Nessun problema riscontrato.

Punto della situazione rispetto alla pianificazione

Abbastanza in anticipo rispetto alla pianificazione.

Programma di massima per la prossima giornata di lavoro

Implementazione, documentazione.

Gestione Esercizi | Diario di lavoro - 03.10.2019

Gabriele Alessi

Canobbio, 03.10.2019

Lavori svolti

Oggi ho lavorato principalmente alla revisione del codice sviluppato durante questi giorni e poi ho documentato il tutto. Inizialmente ho voluto rifare il codice implementando i commenti ([Documentation comments](#)) e capire meglio il funzionamento del sistema **Services**.

Questo è un esempio di una classe di modello di dati documentata:

```
/// <summary>
/// Modello di dati del modulo.
/// </summary>
public class Modulo : BaseEntity
{
    /// <summary>
    /// Nome del modulo (esempio: "Modulo 151").
    /// </summary>
    public string Nome { get; set; }

    /// <summary>
    /// Anno del modulo.
    /// </summary>
    public Anno Anno { get; set; }

    /// <summary>
    /// Insieme delle tematiche del modulo.
    /// </summary>
    public ICollection<Tematica> Tematiche { get; set; }

    /// <summary>
    /// Insieme degli esercizi del modulo.
    /// </summary>
    public ICollection<Esercizio> Esercizi { get; set; }
}
```

Poi ho proseguito sviluppando le classi relative all'interfacciamento con i dati e il database (Services) e per la prossima giornata dovrei concluderlo.

```
/// <summary>
/// Interfaccia di base che implementa i metodi relativi alle
/// operazioni sui dati nel database.
/// </summary>
/// <typeparam name="T">Modello di dati di riferimento.</typeparam>
public interface IDataRepository<T> where T : BaseEntity
{
    T Get(int id);

    IQueryable<T> Get();

    T Insert(T entity);

    void Update(T entity);

    void Delete(T entity);
}
```

Orario	Lavori svolti
13:15 - 16:30	Implementazione e Documentazione

Problemi riscontrati e soluzioni adottate

Nessun problema riscontrato.

Punto della situazione rispetto alla pianificazione

Abbastanza in anticipo rispetto alla pianificazione.

Programma di massima per la prossima giornata di lavoro

Implementazione, documentazione.

Gestione Esercizi | Diario di lavoro - 04.10.2019

Gabriele Alessi

Canobbio, 04.10.2019

Lavori svolti

Durante questa giornata ho continuato con l'implementazione del progetto completando in contemporanea anche la documentazione riguardo al lavoro sviluppato.

Penso di aver concluso la parte della libreria di classi, quindi tutto ciò che ha a che fare con i dati e il database, ma prima di cominciare con l'App WPF vorrei consultarmi con il docente per verificare quanto svolto finora.

AppDbContext è la classe che praticamente mette insieme i modelli di dati (Models) e le interfacce con il database (Services) per impostare la base di dati configurando le raccolte delle entità e il percorso di memorizzazione SQLite.

```
public class AppDbContext : DbContext
{
    public DbSet<Classe> Classi { get; set; }
    public DbSet<Modulo> Moduli { get; set; }
    public DbSet<Tematica> Tematiche { get; set; }
    public DbSet<Anno> Anno { get; set; }
    public DbSet<Esercizio> Esercizi { get; set; }
    public DbSet<EsercizioProva> EserciziProva { get; set; }
    public DbSet<Prova> Prove { get; set; }

    public AppDbContext() : base()
    {
    }

    public AppDbContext(DbContextOptions<AppDbContext> options) : base(options)
    {
    }

    protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
    {
        if (!optionsBuilder.IsConfigured)
        {
            optionsBuilder.UseSqlite("Data Source=D:\\Temp\\\\GestioneEsercizi.db");
        }
    }
}
```

DbDataRepository è la classe che implementa l'interfaccia IRepository e viene usata principalmente come base per i repository degli altri modelli di dati implementando i metodi relativi al database.

```
protected C context;

protected DbDataRepository(C ctx)
{
    context = ctx;
}

public T Get(int id)
{
    return Get().SingleOrDefault(be => be.Id == id);
}

public virtual IQueryable<T> Get()
{
    return context.Set<T>();
}

public virtual T Insert(T entity)
{
    context.Set<T>().Add(entity);
    context.SaveChanges();
    return entity;
}

public virtual void Update(T entity)
{
    context.Entry(entity).State = EntityState.Modified;
    context.SaveChanges();
}

public virtual void Delete(T entity)
{
    context.Set<T>().Remove(entity);
    context.SaveChanges();
}

public virtual IQueryable<T> Where(Expression<Func<T, bool>> predicate)
{
    return context.Set<T>().Where(predicate);
}
```

Riguardo questa classe dovrei informarmi dell'utilità del metodo Where() in quanto guardando i progetti del modulo dell'anno scorso risulta un po' ambiguo.

I EsercizioRepository è un'interfaccia figlia di IDataRepository relativa al modello di dati dell'esercizio.

```
/// <summary>
/// Interfaccia che implementa i metodi relativi al database
/// sul modello di dati dell'esercizio.
/// </summary>
public interface IEsercizioRepository : IDataRepository<Esercizio>
{
}
```

Orario	Lavori svolti
13:15 - 16:30	Implementazione e Documentazione

Problemi riscontrati e soluzioni adottate

Nessun problema riscontrato.

Punto della situazione rispetto alla pianificazione

Abbastanza in anticipo rispetto alla pianificazione.

Programma di massima per la prossima giornata di lavoro

Implementazione, documentazione.

Gestione Esercizi | Diario di lavoro - 08.10.2019

Gabriele Alessi

Canobbio, 08.10.2019

Lavori svolti

Oggi ho iniziato con l'implementazione dell'App WPF del progetto su Visual Studio. Dopo aver concluso la libreria delle classi (cioè le classi che gestiscono i dati e il database) mi sono consultato con il docente responsabile per controllare ciò e per vedere come cominciare con il resto del progetto.

Come prima cosa ho capito l'utilità del metodo `Where()`:

```
/// <summary>
/// Metodo utile per filtrare dei dati.
/// </summary>
/// <param name="predicate">Espressione filtrante.</param>
/// <returns>I dati filtrati.</returns>
public virtual IQueryable<T> Where(Expression<Func<T, bool>> predicate)
{
    return context.Set<T>().Where(predicate);
}
```

Quindi ho cominciato con lo sviluppo dell'App WPF partendo dai ViewModels. Prima di tutto ho reperito le basi Helper di MVVM dai vecchi progetti. Ad esempio `BindableBase` è la superclasse di ogni altro ViewModel in quanto contiene i metodi base sulla gestione dei dati e il collegamento con le View.

```
public class ClasseListViewModel : BindableBase
{
    /// <summary>
    /// Insieme dinamico dei dati delle classi.
    /// </summary>
    public ObservableCollection<Classe> Classi { get; set; }

    /// <summary>
    /// Metodo costruttore del ViewModel.
    /// </summary>
    public ClasseListViewModel()
    {
        ClasseDbRepository repo = new ClasseDbRepository(new AppDbContext());
        Classi = new ObservableCollection<Classe>(repo.Get());
    }
}
```

```

public abstract class BindableBase : INotifyPropertyChanged
{
    public event PropertyChangedEventHandler PropertyChanged = delegate { };

    protected void OnPropertyChanged(string propertyName)
    {
        PropertyChangedEventHandler handler = PropertyChanged;
        if (handler != null)
        {
            handler(this, new PropertyChangedEventArgs(propertyName));
        }
    }

    protected virtual void SetProperty<T>(ref T member, T value,
[CallerMemberName] string propertyName = null)
    {
        if (object.Equals(member, value))
            return;
        member = value;
        OnPropertyChanged(propertyName);
    }

    protected void OnPropertyChanged<T>(Expression<Func<T>> propertyExpression)
    {
        var body = propertyExpression.Body as MemberExpression;
        var expression = body.Expression as ConstantExpression;
        PropertyChanged(expression.Value, new
PropertyChangedEventArgs(body.Member.Name));
    }
}

```

Ora che ho capito il sistema dei ViewModel dovrei riuscire a concludere questo capitolo per questa settimana.

Orario	Lavori svolti
13:15 - 16:30	Implementazione e Documentazione

Problemi riscontrati e soluzioni adottate

Nessun problema riscontrato.

Punto della situazione rispetto alla pianificazione

Abbastanza in anticipo rispetto alla pianificazione.

Programma di massima per la prossima giornata di lavoro

Implementazione, documentazione.

Gestione Esercizi | Diario di lavoro - 10.10.2019

Gabriele Alessi

Canobbio, 10.10.2019

Lavori svolti

Durante questa giornata mi sono principalmente occupato dell'implementazione delle classi ViewModel sul progetto WPF di Visual Studio. Prima di tutto ho rinominato le classi in quanto prima contenevano la parola "List" ([ClasseListViewModel](#)) perché erano progettate per mostrare solo la lista di dati, mentre invece possono avere altre funzioni.

```
public class ProvaViewModel : BindableBase
{
    /// <summary>
    /// Insieme dinamico dei dati delle prove.
    /// </summary>
    public ObservableCollection<Prova> Prove { get; set; }
    /// <summary>
    /// Metodo costruttore del ViewModel.
    /// </summary>
    public ProvaViewModel()
    {
        ProvaDbRepository repo = new ProvaDbRepository(new AppDbContext());
        Prove = new ObservableCollection<Prova>(repo.Get());
    }
}
```

Per ora ho quasi concluso il capitolo ViewModels, manca solo da capire come gestire il MainViewModel e infine completare le classi dopo aver progettato la struttura delle Views.

Orario	Lavori svolti
13:15 - 16:30	Implementazione e Documentazione

Problemi riscontrati e soluzioni adottate

Nessun problema riscontrato.

Punto della situazione rispetto alla pianificazione

Abbastanza in anticipo rispetto alla pianificazione.

Programma di massima per la prossima giornata di lavoro

Implementazione, documentazione.

Gestione Esercizi | Diario di lavoro - 11.10.2019

Gabriele Alessi

Canobbio, 11.10.2019

Lavori svolti

Oggi durante le prime due ore c'è la stata la presentazione del docente Valsangiacomo riguardo agli esami LPI.

Nelle ore successive ho riguardato il progetto (soprattutto la documentazione) in tenendo conto delle specifiche spiegate e ho approfittato del tempo rimanente per finire un lavoro di un altro modulo.

Quindi riassumendo questa è stata una giornata abbastanza tranquilla dove non ho avuto l'occasione di lavorare molto sull'implementazione.

Per la prossima settimana l'obiettivo sarebbe di concludere i ViewModels e iniziare le Views dopo aver consultato il docente.

Orario	Lavori svolti
13:15 - 14:45	Presentazione esami di diploma
15:00 - 16:30	Documentazione

Problemi riscontrati e soluzioni adottate

Nessun problema riscontrato.

Punto della situazione rispetto alla pianificazione

Abbastanza in anticipo rispetto alla pianificazione.

Programma di massima per la prossima giornata di lavoro

Implementazione, documentazione.

Gestione Esercizi | Diario di lavoro - 15.10.2019

Gabriele Alessi

Canobbio, 15.10.2019

Lavori svolti

Oggi ho cominciato a sviluppare le Views del progetto su Visual Studio. Ho deciso di lasciare un attimo in sospeso i ViewModels perché lavorando sulle Views capirei meglio le implementazioni da completare. Come prima cosa ho creato la MainView e BenvenutoView, cioè la schermata principale che contiene i tre pulsanti per le opzioni principali.

```
<Button  
    VerticalAlignment="Center"  
    HorizontalAlignment="Center"  
    Content="Impostazioni di Base"  
    Padding="10" >  
</Button>  
<Button  
    Grid.Column="1"  
    VerticalAlignment="Center"  
    HorizontalAlignment="Center"  
    Content="Nuovo Esercizio"  
    Padding="10" >  
</Button>  
<Button  
    Grid.Column="2"  
    VerticalAlignment="Center"  
    HorizontalAlignment="Center"  
    Content="Creazione Prova"  
    Padding="10" >  
</Button>
```

Inoltre ho iniziato a implementare il menu che si trova nella MainWindow.

```
<DockPanel>  
    <Menu DockPanel.Dock="Top">  
        <MenuItem Header="_File">  
            <MenuItem Header="_Esercizi"/>  
            <MenuItem Header="_Prove"/>  
            <MenuItem Header="_Opzioni"/>  
            <MenuItem x:Name="miEsci" Header="_Esci" Click="miEsci_Click"/>  
        </MenuItem>  
    </Menu>  
</DockPanel>
```

Infine ho iniziato i implementare la finestra **About** che si trova sotto il menu **Info** e presenta informazioni come l'autore, il riferimento a GitHub e la versione.

```
<TextBlock
    Grid.Column="1"
    Grid.Row="1"
    HorizontalAlignment="Center"
    VerticalAlignment="Center">
    Gestione Esercizi
    <LineBreak/>
    <Hyperlink
        NavigateUri="https://github.com/gabrialessi/GestioneEsercizi">
        GitHub
    </Hyperlink>
</TextBlock>
<Button
    Grid.Column="2"
    Grid.Row="2"
    x:Name="bChiudi"
    HorizontalAlignment="Right"
    VerticalAlignment="Bottom"
    Margin="15"
    Width="80"
    Content="Chiudi" Click="bChiudi_Click">
</Button>
```

Dovrei consultare il docente che oggi non sono riuscito a incontrare per riguardare un po' la struttura generale in quanto nel mio progetto sono presenti diversi elementi e devo capire meglio come metterli insieme.

Orario	Lavori svolti
13:15 - 16:30	Implementazione

Problemi riscontrati e soluzioni adottate

Nessun problema riscontrato.

Punto della situazione rispetto alla pianificazione

Abbastanza in anticipo rispetto alla pianificazione.

Programma di massima per la prossima giornata di lavoro

Implementazione, documentazione.

Gestione Esercizi | Diario di lavoro - 17.10.2019

Gabriele Alessi

Canobbio, 17.10.2019

Lavori svolti

Durante questa giornata ho continuato con lo sviluppo delle interfacce grafiche del progetto su Visual Studio. Ho implementato diverse modifiche rispetto a ciò che avevo in mente fino alla scorsa settimana. Infatti dal sistema che ho sviluppato risulta necessario implementare la gestione dei ViewModels e delle Views nella MainWindow, e non nella MainView.

```
<Window.DataContext>
    <viewmodel:MainViewModel/>
</Window.DataContext>
<Window.Resources>
    <DataTemplate DataType="{x:Type viewmodel:AboutViewModel}">
        <view:AboutView/>
    </DataTemplate>
    <DataTemplate DataType="{x:Type viewmodel:BenvenutoViewModel}">
        <view:BenvenutoView/>
    </DataTemplate>
</Window.Resources>
<Grid>
    <DockPanel>
        <Menu DockPanel.Dock="Top">
            <MenuItem Header="_File">
                <MenuItem Header="_Esercizi"/>
                <MenuItem Header="_Prove"/>
                <MenuItem Header="_Opzioni"/>
                <MenuItem x:Name="miEsci" Header="_Esci" Click="miEsci_Click"/>
            </MenuItem>
            <MenuItem Header="_Info">
                <MenuItem Header="_Guida"/>
                <MenuItem Header="_About" Command="{Binding Path=AboutCommand}"/>
            </MenuItem>
        </Menu>
        <StackPanel></StackPanel>
    </DockPanel>
    <ContentControl Content="{Binding Path=CurrentViewModel}"/>
</Grid>
```

Il fatto è che utilizzo il menu (`<DockPanel>`) in cui dovrei chiamare diverse Views, quindi la soluzione è stata quella di includere tutto nel MainWindow.

Di conseguenza ho dovuto cancellare la AboutWindow e ho creato una AboutView, AboutViewModel e BenvenutoViewModel.

```
public class AboutViewModel : BindableBase
{
    public AboutViewModel()
    {
    }
}

public class BenvenutoViewModel : BindableBase
{
    public BenvenutoViewModel()
    {
    }
}
```

Neanche oggi il docente si è presentato e ho bisogno abbastanza urgente di verificare quanto fatto finora e fare alcune domande sul sistema da adottare per quanto riguarda gli altri ViewModels e la gestione delle Views e dei dati.

Orario	Lavori svolti
13:15 - 16:30	Implementazione

Problemi riscontrati e soluzioni adottate

Nessun problema riscontrato.

Punto della situazione rispetto alla pianificazione

Ambientata in anticipo rispetto alla pianificazione.

Programma di massima per la prossima giornata di lavoro

Implementazione, documentazione.

Gestione Esercizi | Diario di lavoro - 18.10.2019

Gabriele Alessi

Canobbio, 18.10.2019

Lavori svolti

Oggi ho continuato a lavorare alle interfacce grafiche del progetto Visual Studio. Inizialmente ho creato un pulsante nella schermata [About](#) per tornare indietro alla schermata di benvenuto (che per ora non funziona anche a causa dei problemi riscontrati).

```
<Button x:Name="bChiudi" Grid.Column="2" Grid.Row="2"
        Content="Indietro" VerticalAlignment="Bottom" Margin="20"
        Command="{Binding Path=BenvenutoCommand}" />
```

Poi ho sviluppato la View delle impostazioni di base, che è quella che mi ha creato problemi dopo l'implementazione di [ImpostazioniDiBaseViewModel](#).

```
<UserControl.DataContext>
    <viewmodel:ImpostazioniBaseViewModel/>
</UserControl.DataContext>
<Grid Margin="50">
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="*" />
        <ColumnDefinition Width="*" />
        <ColumnDefinition Width="*" />
    </Grid.ColumnDefinitions>
    <Grid.RowDefinitions>
        <RowDefinition Height="*" />
        <RowDefinition Height="auto" />
    </Grid.RowDefinitions>
    <DataGrid Grid.Column="0" Grid.Row="0" ItemsSource="{Binding Path=Classi}"/>
    <DataGrid Grid.Column="1" Grid.Row="0" ItemsSource="{Binding Path=Moduli}"/>
    <DataGrid Grid.Column="2" Grid.Row="0" ItemsSource="{Binding
Path=Tematiche}"/>
    <Button Grid.Column="0" Grid.Row="1" Content="+" FontSize="18" />
    <Button Grid.Column="1" Grid.Row="1" Content="+" FontSize="18" />
    <Button Grid.Column="2" Grid.Row="1" Content="+" FontSize="18" />
</Grid>
```

```
public ImpostazioniBaseViewModel()
{
    AppDbContext ctx = new AppDbContext();
    ClasseDbRepository repoClasse = new ClasseDbRepository(ctx);
    ModuloDbRepository repoModulo = new ModuloDbRepository(ctx);
    TematicaDbRepository repoTematica = new TematicaDbRepository(ctx);
    Classi = new ObservableCollection<Classe>(repoClasse.Get());
    Moduli = new ObservableCollection<Modulo>(repoModulo.Get());
    Tematiche = new ObservableCollection<Tematica>(repoTematica.Get());
}
```

Orario	Lavori svolti
13:15 - 16:30	Implementazione e Documentazione

Problemi riscontrati e soluzioni adottate

Ho riscontrato un problema che riguarda la relazione uno a uno tra **Esercizio** e **EsercizioProva**:

The child/dependent side could not be determined for the one-to-one relationship between 'Esercizio.EsercizioProva' and 'EsercizioProva.Esercizio'. To identify the child/dependent side of the relationship, configure the foreign key property. If these navigations should not be part of the same relationship configure them without specifying the inverse.

La possibile soluzione si trova in [questo link](#) ma comunque non sono riuscito a risolvere il problema.

```
protected override void OnModelCreating(ModelBuilder modelBuilder)
{
    modelBuilder.Entity<Esercizio>()
        .HasOne(p => p.EsercizioProva)
        .WithOne(i => i.Esercizio)
        .HasForeignKey<EsercizioProva>(b => b.Esercizio);
}
```

Ho pensato di cambiare la struttura e creare una dipendenza ma dovrei riguardare con il docente questa relazione così da trovare una soluzione.

Punto della situazione rispetto alla pianificazione

Abbastanza in anticipo rispetto alla pianificazione.

Programma di massima per la prossima giornata di lavoro

Implementazione, documentazione.

Gestione Esercizi | Diario di lavoro - 22.10.2019

Gabriele Alessi

Canobbio, 22.10.2019

Lavori svolti

In questa giornata ho inizialmente riguardato l'intero progetto per poi consultare il docente responsabile e risolvere il problema riscontrato con la relazione tra Esercizio e EsercizioProva. Quindi siamo giunti alla conclusione che è necessario cambiare la relazione da uno a uno, a uno a molti.

```
/// <summary>
/// Modello di dati dell'esercizio.
/// </summary>
public class Esercizio : BaseEntity
{
    /// <summary>
    /// Titolo dell'esercizio.
    /// </summary>
    public string Titolo { get; set; }
    /// <summary>
    /// Testo dell'esercizio (include l'immagine).
    /// </summary>
    public string Testo { get; set; }
    /// <summary>
    /// Modulo relativo all'esercizio.
    /// </summary>
    public Modulo Modulo { get; set; }
    /// <summary>
    /// Esercizi della prova relativi all'esercizio.
    /// </summary>
    public ICollection<EsercizioProva> EserciziProva { get; set; }
}
```

Dunque ho continuato creando le Migrations e il Database tramite la console di gestione dei pacchetti NuGet.

Add-Migration Initial

Update-Database -Verbose

Poi ho anche trasferito il sistema del MainWindow nella MainView, anche se credevo ciò non fosse possibile a causa dei ViewModels gestiti in diverse Views.

Alla fine il docente mi ha spiegato un metodo per gestire la comunicazione tra più ViewModels ([Messenger](#)).

Orario	Lavori svolti
13:15 - 16:30	Implementazione e Documentazione

Problemi riscontrati e soluzioni adottate

Nessun problema riscontrato.

Punto della situazione rispetto alla pianificazione

In anticipo rispetto alla pianificazione.

Programma di massima per la prossima giornata di lavoro

Implementazione, documentazione.

Gestione Esercizi | Diario di lavoro - 24.10.2019

Gabriele Alessi

Canobbio, 24.10.2019

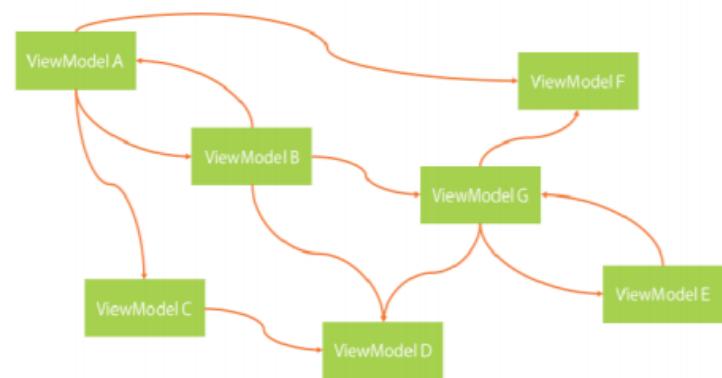
Lavori svolti

Durante questa giornata ho continuato a lavorare sullo sviluppo delle interfacce del programma su Visual Studio. In particolare mi sto concentrando sulla comunicazione tra i ViewModels in modo da consentire il passaggio dei dati tra essi. Per fare ciò ho ricevuto del materiale da parte del docente responsabile di progetto:

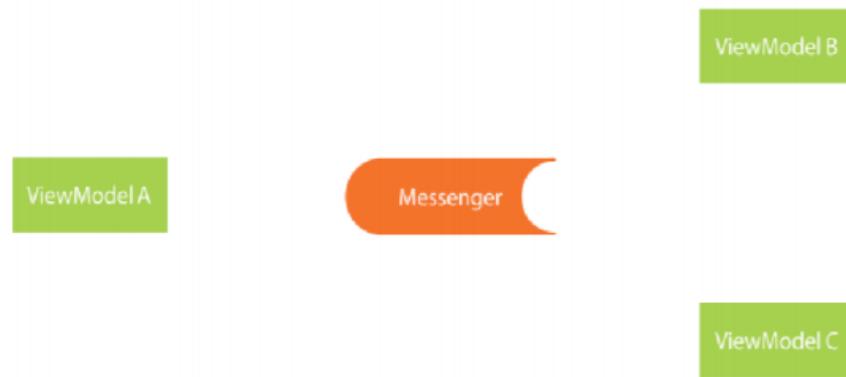
Comunicazione tra ViewModel

Un aspetto importante è la comunicazione tra i vari ViewModel per passare informazioni tra le View senza avere alcun HardLink.

Per risolvere questo problema ci si affida a un Messenger (o Mediator). Si trova in mezzo nella comunicazione tra più ViewModels che tutti conoscono e comunicano attraverso lui.



Per esempio si registra ViewModelA con il Messenger e si definisce che potrebbe inviare un messaggio di un certo tipo. ViewModelB e ViewModelC sono interessati a quello che "dice" ViewModelA. Questi "informano" il Messenger che sono interessati ad un certo tipo di messaggio e non appena riceve un messaggio del tipo coinvolto, questi lo invierà agli interessati.



Quindi ho implementato questo sistema e ho concluso [AboutViewModel](#) e [BenvenutoViewModel](#).

```
public class AboutViewModel : BindableBase
{
    private BenvenutoViewModel benvenutoViewModel;

    public IDDelegateCommand BenvenutoCommand { get; set; }

    public AboutViewModel() { RegisterCommands(); }

    private void RegisterCommands()
    {
        benvenutoViewModel = new BenvenutoViewModel();
        BenvenutoCommand = new DelegateCommand(OnBenvenuto, CanBenvenuto);
    }

    private void OnBenvenuto(object obj)
    {
        Messenger.Default.Send<BindableBase>(benvenutoViewModel);
    }
    private bool CanBenvenuto(object arg) { return true; }
}
```

```
private void OnImpostazioniBase(object obj)
{
    Messenger.Default.Send<BindableBase>(impostazionibaseViewModel);
}
private bool CanImpostazioniBase(object arg) { return true; }
private void OnEsercizio(object obj)
{
    Messenger.Default.Send<BindableBase>(esercizioViewModel);
}
private bool CanEsercizio(object arg) { return true; }
private void OnProva(object obj)
{
    Messenger.Default.Send<BindableBase>(provaViewModel);
}
private bool CanProva(object arg) { return true; }
```

Per la prossima giornata dovrei procedere con i ViewModels in quanto ho risolto i problemi riscontrati e ho trovato i metodi corretti per implementare il sistema.

Orario	Lavori svolti
13:15 - 16:30	Implementazione e Documentazione

Problemi riscontrati e soluzioni adottate

Ho riscontrato problemi durante l'implementazione della comunicazione tra ViewModels tramite il **Messenger**. Questo perché non sono sicuro di come debbano essere implementati i metodi **Send** e **Register** (in che classi e che parametri passare e come gestire il tutto nelle Views).

Alla fine ho risolto correggendo i Commands grazie all'aiuto del docente:

```
private void OnBenvenuto(object obj)
{
    Messenger.Default.Send<BindableBase>(benvenutoViewModel);
}
```

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione.

Programma di massima per la prossima giornata di lavoro

Implementazione, documentazione.

Gestione Esercizi | Diario di lavoro - 25.10.2019

Gabriele Alessi

Canobbio, 25.10.2019

Lavori svolti

Oggi ho proseguito con lo sviluppo delle Views e dei ViewModels del progetto su Visual Studio. Nel dettaglio mi sono dedicato all'implementazione dei ViewModels tenendo conto del database, quindi ho modificato alcuni parametri della connessione e ho inserito alcuni dati. A proposito di questo ho riscontrato dei problemi (descritti sotto), però poi ho continuato sviluppando la View e il ViewModel della lista di prove memorizzate.

```
public class ProvaListViewModel : BindableBase
{
    private BenvenutoViewModel benvenutoViewModel;
    public IDelegateCommand BenvenutoCommand { get; set; }
    public ObservableCollection<Prova> Prove { get; set; }

    public ProvaListViewModel()
    {
        RegisterCommands();
        ProvaDbRepository repo = new ProvaDbRepository(new AppDbContext());
        Prove = new ObservableCollection<Prova>(repo.Get());
    }
    [...]
```

```
<UserControl.DataContext>
    <viewmodel:ProvaListViewModel/>
</UserControl.DataContext>
<Grid Margin="10">
    <Grid.RowDefinitions>
        <RowDefinition Height="*"/>
        <RowDefinition Height="auto"/>
    </Grid.RowDefinitions>
    <DataGrid Grid.Row="0" ItemsSource="{Binding Path=Prove}" />
    <Button
        Grid.Row="1" Content="Indietro" Margin="10" Width="100"
        VerticalAlignment="Bottom" HorizontalAlignment="Right"
        Command="{Binding Path=BenvenutoCommand}" />
</Grid>
```

Infine ho provato a nascondere certe colonne nelle ListView mostrando solo quelle interessate (Titolo, Data, ...). Purtroppo anche a questo non ho trovato una soluzione funzionante per ora.

```
public EsercizioListView()
{
    InitializeComponent();
    this.dgEsercizi.Columns[1].Visibility = Visibility.Collapsed;
    this.dgEsercizi.Columns[3].Visibility = Visibility.Collapsed;
    this.dgEsercizi.Columns[4].Visibility = Visibility.Collapsed;
}
```

Orario

Lavori svolti

13:15 - 16:30 Implementazione e Documentazione

Problemi riscontrati e soluzioni adottate

Ho riscontrato problemi che riguardano i pacchetti NuGet. Infatti, dopo aver cambiato il percorso della connessione al database SQLite e aver inserito alcuni dati, non vengono identificate le tabelle.

```
optionsBuilder.UseSqlite("Data Source="
    + System.AppDomain.CurrentDomain.BaseDirectory
    + "GestioneEserciziDB.sqlite");
```

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione.

Programma di massima per la prossima giornata di lavoro

Implementazione, documentazione.

Gestione Esercizi | Diario di lavoro - 05.11.2019

Gabriele Alessi

Canobbio, 05.11.2019

Lavori svolti

Durante questa giornata ho continuato con lo sviluppo delle Views e dei ViewModels del progetto. Prima di cominciare ho riguardato il punto della situazione e quindi ho chiesto aiuto al docente riguardo agli errori riscontrati prima delle vacanze sul database e sui pacchetti NuGet. Ho risolto ricostruendo il database e la Migration dopo aver reimpostato il percorso assoluto nel [AppDbContext](#).

Oltre a ciò ho ripreso con lo sviluppo della View e del ViewModel delle impostazioni di base. Dopo aver corretto il database ho implementato i metodi [ToString\(\)](#) utili per stampare le informazioni nei [DataGrid](#) correttamente.

```
public class Tematica : BaseEntity
{
    public string Nome { get; set; }
    public Modulo Modulo { get; set; }
    public override string ToString()
    {
        return Nome;
    }
}
```

Quindi ora che ho il controllo del database e dei ViewModels dovrei riuscire a concludere il capitolo delle impostazioni di base per questa settimana.

Orario	Lavori svolti
13:15 - 16:30	Implementazione e Documentazione

Problemi riscontrati e soluzioni adottate

Nessun problema riscontrato.

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione.

Programma di massima per la prossima giornata di lavoro

Implementazione, documentazione.

Gestione Esercizi | Diario di lavoro - 07.11.2019

Gabriele Alessi

Canobbio, 07.11.2019

Lavori svolti

Oggi ho continuato con l'implementazione lavorando principalmente su EsercizioView e EsercizioViewModel. Prima però ho riguardato la View che riguarda le impostazioni di base in quanto ci sono problemi nel mostrare alcuni dati delle entità.

Concretamente sono riuscito a sviluppare la base della schermata per creare un nuovo esercizio.

```
<!-- Titolo esercizio -->
<TextBox x:Name="tbTitolo" Grid.Column="0" Grid.Row="0" Margin="5" />
<TextBlock IsHitTestVisible="False" Text="Titolo esercizio..." VerticalAlignment="Center" Foreground="DarkGray" Margin="10 0 0 0">
    <TextBlock.Style>
        <Style TargetType="{x:Type TextBlock}">
            <Setter Property="Visibility" Value="Collapsed"/>
            <Style.Triggers>
                <DataTrigger Binding="{Binding Text, ElementName=tbTitolo}" Value="">
                    <Setter Property="Visibility" Value="Visible"/>
                </DataTrigger>
            </Style.Triggers>
        </Style>
    </TextBlock.Style>
</TextBlock>
<!-- Menu - Modulo, Tematica, Classe -->
<ComboBox Grid.Column="1" Grid.Row="0" SelectedIndex="0" ItemsSource="{Binding Path=Moduli}" Margin="5" />
<ComboBox Grid.Column="2" Grid.Row="0" SelectedIndex="0" ItemsSource="{Binding Path=Tematiche}" Margin="5" />
<ComboBox Grid.Column="3" Grid.Row="0" SelectedIndex="0" ItemsSource="{Binding Path=Classi}" Margin="5" />
<!-- Immagine -->
<Grid Grid.Column="2" Grid.ColumnSpan="2" Grid.Row="1" Margin="50" AllowDrop="True" Drop="Grid_Drop">
    <Label x:Name="lImmagine" Content="Immagine" />
    <Button x:Name="bSfoglia" VerticalAlignment="Center" HorizontalAlignment="Center" Content="Sfoglia..." Click="bSfoglia_Click" />
</Grid>
```

Ho trovato anche il modo di inserire le immagini sfogliando con l'explorer o tramite drag&drop.

```
private void bSfoglia_Click(object sender, RoutedEventArgs e)
{
    // Create OpenFileDialog
    Microsoft.Win32.OpenFileDialog dlg = new Microsoft.Win32.OpenFileDialog();

    // Set filter for file extension and default file extension
    dlg.Filter = "JPEG Files (*.jpeg)|*.jpeg|PNG Files (*.png)|*.png|JPG Files (*.jpg)|*.jpg|GIF Files (*.gif)|*.gif";

    // Display OpenFileDialog by calling ShowDialog method
    Nullable<bool> result = dlg.ShowDialog();

    // Get the selected file name and display in a TextBox
    if (result == true) lImmagine.Content = dlg.FileName;
}

private void Grid_Drop(object sender, DragEventArgs e)
{
    if (e.Data.GetDataPresent(DataFormats.FileDrop))
    {
        // Note that you can have more than one file.
        string[] files = (string[])e.Data.GetData(DataFormats.FileDrop);
        lImmagine.Content = files[0].ToString();
    }
}
```

Riferimenti:

[Drag and drop WPF C# getting file path](#)

[Open file dialog and select a file using WPF controls and C#](#)

Orario	Lavori svolti
--------	---------------

13:15 - 16:30	Implementazione e Documentazione
---------------	----------------------------------

Problemi riscontrati e soluzioni adottate

Alcuni dati delle entità nei **DataGrid** non vengono mostrati.

```
public ImpostazioniBaseViewModel()
{
    AppDbContext ctx = new AppDbContext();
    ClasseDbRepository repoClasse = new ClasseDbRepository(ctx);
    ModuloDbRepository repoModulo = new ModuloDbRepository(ctx);
    TematicaDbRepository repoTematica = new TematicaDbRepository(ctx);
    Classi = new ObservableCollection<Classe>(repoClasse.Get());
    Moduli = new ObservableCollection<Modulo>(repoModulo.Get());
    Tematiche = new ObservableCollection<Tematica>(repoTematica.Get());
}
```

```
<DataGrid ItemsSource="{Binding Path=Classi}" />
<DataGrid ItemsSource="{Binding Path=Moduli}" />
<DataGrid ItemsSource="{Binding Path=Tematiche}" />
```

Riguardo a ciò ho chiesto aiuto al docente e pare che avrà una soluzione nei prossimi giorni.

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione.

Programma di massima per la prossima giornata di lavoro

Implementazione, documentazione.

Gestione Esercizi | Diario di lavoro - 08.11.2019

Gabriele Alessi

Canobbio, 08.11.2019

Lavori svolti

Ho dedicato questa giornata interamente alla documentazione. In generale ho rivisto velocemente i primi capitoli per avere il punto della situazione e poi ho pulito il codice della libreria di classi semplificando anche la documentazione.

```
public abstract class DbDataRepository<C, T> : IDataRepository<T> where T : BaseEntity
{
    protected C context;
    protected DbDataRepository(C ctx) => context = ctx;

    public T Get(int id) => Get().SingleOrDefault(be => be.Id == id);

    public virtual IQueryable<T> Get() => context.Set<T>();

    public virtual T Insert(T entity)
    {
        context.Set<T>().Add(entity);
        context.SaveChanges();
        return entity;
    }

    public virtual void Update(T entity)
    {
        context.Entry(entity).State = EntityState.Modified;
        context.SaveChanges();
    }

    public virtual void Delete(T entity)
    {
        context.Set<T>().Remove(entity);
        context.SaveChanges();
    }
}
```

```
public class AppDbContext : DbContext
{
    public DbSet<Classe> Classi { get; set; }
    public DbSet<Modulo> Moduli { get; set; }
    public DbSet<Tematica> Tematiche { get; set; }
    public DbSet<Anno> Anni { get; set; }
    public DbSet<Esercizio> Esercizi { get; set; }
    public DbSet<EsercizioProva> EserciziProva { get; set; }
    public DbSet<Prova> Prove { get; set; }

    public AppDbContext() : base() { }
    public AppDbContext(DbContextOptions<AppDbContext> options) : base(options) {
    }

    protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
    {
        if (!optionsBuilder.IsConfigured)
        {
            string dbPath =
                "D:\\Desktop\\Scuola\\4SAMT\\Progetti\\GestioneEsercizi\\Project"
                +
                "\\GestioneEsercizi\\GestioneEsercizi.DA\\db\\GestioneEsercizi.sqlite";
            optionsBuilder.UseSqlite("Data Source=" + dbPath);
        }
    }
}
```

Successivamente è necessario creare il database tramite la console di gestione dei pacchetti. La prima cosa da fare è la creazione delle migrazioni, le quali definiscono i parametri del database tramite il DbContext, per poi aggiornare la base di dati.

Orario	Lavori svolti
13:15 - 16:30	Implementazione e Documentazione

Problemi riscontrati e soluzioni adottate

Nessun problema riscontrato.

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione.

Programma di massima per la prossima giornata di lavoro

Implementazione, documentazione.

Gestione Esercizi | Diario di lavoro - 12.11.2019

Gabriele Alessi

Canobbio, 12.11.2019

Lavori svolti

Durante questa giornata ho continuato a lavorare principalmente sui ViewModels del progetto su Visual Studio. Inizialmente ho provato la soluzione trovata dal docente responsabile riguardo agli output nelle Views dei dati provenienti dai Models. Per fare ciò ho installato il pacchetto `Microsoft.EntityFrameworkCore.Proxies` e ho implementato il metodo nell'`AppDbContext` modificando i campi interessati tramite la keyword `virtual` ([Lazy Loading](#)).

```
protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
{
    if (!optionsBuilder.IsConfigured)
    {
        string dbPath =
            "D:\\Desktop\\Scuola\\4SAMT\\Progetti\\GestioneEsercizi\\Project"
            +
        "\\GestioneEsercizi\\GestioneEsercizi.DA\\db\\GestioneEsercizi.sqlite";
        optionsBuilder.UseLazyLoadingProxies().UseSqlite("Data Source=" + dbPath);
    }
}
```

```
public class Classe : BaseEntity
{
    public string Nome { get; set; }
    public virtual Anno Anno { get; set; }
    public override string ToString() => Nome;
}
```

Per il resto oltre al problema riscontrato ho proseguito pulendo il codice semplificandolo e ho fatto un po' di documentazione concludendo il capitolo del [MainViewModel](#).

Orario	Lavori svolti
13:15 - 16:30	Implementazione e Documentazione

Problemi riscontrati e soluzioni adottate

Nonostante la soluzione del Lazy Loading trovata alcuni dati non vengono mostrati. Questo vale per i Models che presentano campi **ICollection** come il **Modulo**.

```
public class Modulo : BaseEntity
{
    public string Nome { get; set; }
    public virtual Anno Anno { get; set; }
    public virtual ICollection<Tematica> Tematiche { get; set; }
    public virtual ICollection<Esercizio> Esercizi { get; set; }
    public override string ToString() => Nome;
}
```

Ne ho parlato con il docente e ha detto che cercherà una soluzione per questo.

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione.

Programma di massima per la prossima giornata di lavoro

Implementazione, documentazione.

Gestione Esercizi | Diario di lavoro - 14.11.2019

Gabriele Alessi

Canobbio, 14.11.2019

Lavori svolti

Oggi ho maggiormente lavorato sulla documentazione delle Views e dei ViewModels del progetto. La prima cosa che ho fatto però è l'implementazione del drag&drop nella schermata di creazione di un esercizio. Riassumendo ho semplificato un po' il codice provvisorio di qualche giorno fa correggendo il filtro dei files e migliorando la notifica del file inserito.

```
<!-- Immagine -->
<Border Grid.Column="2" Grid.ColumnSpan="2" Grid.Row="1"
        BorderBrush="Black" BorderThickness=".5" Margin="50"
        AllowDrop="True" Drop="Image_Drop">
    <Grid>
        <Label x:Name="lImmagine" Content="Trascina immagine" />
        <Button x:Name="bSfoglia" VerticalAlignment="Center"
                HorizontalAlignment="Center" Content="Sfoglia..." Click="bSfoglia_Click"
            />
    </Grid>
</Border>
```

```
private void bSfoglia_Click(object sender, RoutedEventArgs e)
{
    // Creazione File Explorer
    Microsoft.Win32.OpenFileDialog dlg = new Microsoft.Win32.OpenFileDialog();
    // Impostazione estensioni files
    dlg.Filter = "Image Files | *.jpg; *.jpeg; *.png; *.gif";
    // Mostrare il file selezionato nella label
    if (dlg.ShowDialog() == true) SetLabelContent(dlg.FileName);
}

private void Image_Drop(object sender, DragEventArgs e)
{
    if (e.Data.GetDataPresent(DataFormats.FileDrop))
    {
        string[] files = (string[])e.Data.GetData(DataFormats.FileDrop);
        SetLabelContent(files[0]);
    }
}

private void SetLabelContent(string filename)
{
    lImmagine.Content = Path.GetFileName(filename);
}
```

Successivamente ho velocemente creato una bozza del ViewModel e della View per la guida del prodotto, che dovrebbe essere l'ultimo elemento del sistema:

- BenvenutoViewModel
- AboutViewModel
- GuidaViewModel
- EsercizioListViewModel
- ProvaListViewModel
- ImpostazioniBaseViewModel
- EsercizioViewModel
- ProvaViewModel

Per la documentazione ho concluso i capitoli sulla MainView e la BenvenutoView.

La MainView è l'interfaccia principale e ha il compito di rappresentare il MainViewModel, quindi viene selezionata la View da mostrare attraverso i Commands e il campo CurrentViewModel. Sono inoltre presenti i riferimenti di tutti gli altri ViewModels e le rispettive Views nella sezione delle risorse. Questa View sarà l'unico contenuto presente nella finestra principale del progetto, cioè MainWindow.

Orario	Lavori svolti
13:15 - 16:30	Implementazione e Documentazione

Problemi riscontrati e soluzioni adottate

Ho problemi nell'implementare dei link a delle pagine web nella schermata di informazioni, ho cercato diversi modi di fare ciò ma non trovo soluzioni.

```
<Label Grid.Column="1" Grid.Row="1"  
       VerticalAlignment="Center" HorizontalAlignment="Center">  
<Hyperlink  
      NavigateUri="https://github.com/gabrialessi/GestioneEsercizi">  
      GestioneEsercizi  
</Hyperlink>
```

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione.

Programma di massima per la prossima giornata di lavoro

Implementazione, documentazione.

Gestione Esercizi | Diario di lavoro - 15.11.2019

Gabriele Alessi

Canobbio, 15.11.2019

Lavori svolti

In questa giornata ho lavorato principalmente allo sviluppo delle Views del progetto su Visual Studio. Ho riguardato un po' in generale il sistema pensando a come continuare nei prossimi giorni. Ho pensato che la cosa più complicata sarà l'inserimento di una nuova impostazione di base, visto che è necessario farlo attraverso un sistema che permette di inserire anche i dati relativi all'entità interessata (ad esempio se inserisco una classe devo specificare un'annata che sia presente nel database).

```
<!-- Tabelle con i dati -->
<DataGrid x:Name="classiDataGrid" Grid.Column="0" Grid.Row="1"
           ItemsSource="{Binding Path=Classi}" />
<DataGrid x:Name="moduliDataGrid" Grid.Column="1" Grid.Row="1"
           ItemsSource="{Binding Path=Moduli}" />
<DataGrid x:Name="tematicheDataGrid" Grid.Column="2" Grid.Row="1"
           ItemsSource="{Binding Path=Tematiche}" />
<!-- Pulsanti per aggiunta dati -->
<Button Grid.Column="0" Grid.Row="2" Content="Nuova classe"/>
<Button Grid.Column="1" Grid.Row="2" Content="Nuovo modulo" />
<Button Grid.Column="2" Grid.Row="2" Content="Nuova tematica" />
```

Nel frattempo ho documentato tutto ciò che ho concluso, quindi la MainView, BenvenutoView, AboutView, EsercizioListView e ProvaListView.

Orario	Lavori svolti
---------------	----------------------

13:15 - 16:30 Implementazione e Documentazione

Problemi riscontrati e soluzioni adottate

Continuo ad avere problemi nell'inserire i link alle pagine web nella AboutView. Ho già spiegato questo problema (viene sollevata un'eccezione) e vorrei risolvere nei prossimi giorni chiedendo aiuto al docente ([Using Hyperlink in WPF](#)).

Poi ho anche problemi nel nascondere una colonna di una **GridView** visto che durante il debugging l'array **Columns** risulta vuoto.

```
eserciziDataGrid.Columns[0].Visibility = Visibility.Collapsed;
```

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione.

Programma di massima per la prossima giornata di lavoro

Implementazione, documentazione.

Gestione Esercizi | Diario di lavoro - 19.11.2019

Gabriele Alessi

Canobbio, 19.11.2019

Lavori svolti

Oggi ho continuato con l'implementazione delle Views completando anche la documentazione di alcuni ViewModels.

Gli elementi presenti in questa View permettono di creare un nuovo esercizio. Vengono definite le impostazioni di base, il titolo, il testo e un'eventuale immagine. La seguente parte di XML è utile per creare del testo di suggerimento che viene mostrato quando il campo è vuoto e viene usato per il titolo e per il testo.

```
<!-- Titolo esercizio -->
<TextBox x:Name="titoloTextBox" Grid.Column="0" Grid.Row="0" />
<!-- Placeholder -->
<TextBlock IsHitTestVisible="False" Text="Titolo esercizio..." VerticalAlignment="Center" Foreground="DarkGray" Margin="10 0 0 0">
    <TextBlock.Style>
        <Style TargetType="{x:Type TextBlock}">
            <Setter Property="Visibility" Value="Collapsed"/>
            <Style.Triggers>
                <DataTrigger Binding="{Binding Text, ElementName=titoloTextBox}" Value="">
                    <Setter Property="Visibility" Value="Visible"/>
                </DataTrigger>
            </Style.Triggers>
        </Style>
    </TextBlock.Style>
</TextBlock>
```

Poi grazie alle indicazioni del docente sono riuscito a mostrare le liste di tematiche e esercizi nella sezione dei moduli nell'ImpostazioniBaseViewModel. Infatti ho dovuto aggiungere una proprietà non mappata nel Model del modulo in cui ritorno la lista sotto forma di stringa.

```
public virtual ICollection<Tematica> Tematiche { get; set; }  
[NotMapped]  
public string TematicheList => string.Join(", ", Tematiche);  
public virtual ICollection<Esercizio> Esercizi { get; set; }  
[NotMapped]  
public string EserciziList => string.Join(", ", Esercizi);
```

Il prossimo passo sarà cercare di mostrare solo delle colonne specifiche nel DataGrid (anche per questo ho un'idea della soluzione).

Orario	Lavori svolti
13:15 - 16:30	Implementazione e Documentazione

Problemi riscontrati e soluzioni adottate

Nessun problema riscontrato.

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione.

Programma di massima per la prossima giornata di lavoro

Implementazione, documentazione.

Gestione Esercizi | Diario di lavoro - 21.11.2019

Gabriele Alessi

Canobbio, 21.11.2019

Lavori svolti

Durante questa giornata ho continuato lavorando sulle Views e sui ViewModels del progetto su Visual Studio. Inizialmente ho risolto il problema delle colonne del **DataGrid** mostrando solo quelle selezionate.

```
<!-- Classi -->
<DataGrid x:Name="classiDataGrid" Grid.Column="0" Grid.Row="1" IsReadOnly="True"
          ItemsSource="{Binding Path=Classi}" AutoGenerateColumns="False">
    <DataGrid.Columns>
        <DataGridTextColumn Header="Classe" Binding="{Binding Path=Nome}" />
        <DataGridTextColumn Header="Anno" Binding="{Binding Path=Anno}" />
    </DataGrid.Columns>
</DataGrid>
```

Poi ho continuato implementando il salvataggio durante la creazione di un esercizio. Per fare ciò ho prima dovuto modificare un paio di Models per far funzionare il sistema e poter creare una nuova entità **Esercizio**.

```
public Esercizio(string titolo, string testo, Modulo modulo)
{
    Titolo = titolo;
    Testo = testo;
    Modulo = modulo;
}

public Esercizio() { }
```

Quindi ho sviluppato l'EsercizioViewModel e l'EsercizioView per implementare il salvataggio delle informazioni inserite e fare l'Insert del nuovo esercizio.

```
public string Titolo { get; set; }
public string Testo { get; set; }
public Modulo Modulo { get; set; }

public EsercizioViewModel()
{
    SalvaCommand = new DelegateCommand(OnSalva, CanSalva);
}

private void OnSalva(object obj)
{
    EsercizioDbRepository repoEsercizio = new EsercizioDbRepository(new
DbContext());
    // Aggiungo l'esercizio
    repoEsercizio.Insert(new Esercizio(Titolo, Testo, Modulo));
    OnBenvenuto(obj);
}

private bool CanSalva(object arg) => true;
```

Non ho avuto molto tempo di testare questo codice e ho rilevato un problema nel metodo `Insert` della classe `DbDataRepository` in quanto non viene trovato il dato da mettere sotto la colonna `Discriminator` (dovrei chiedere al docente perché si crea questa colonna).

Orario	Lavori svolti
13:15 - 16:30	Implementazione e Documentazione

Problemi riscontrati e soluzioni adottate

Nessun problema riscontrato.

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione.

Programma di massima per la prossima giornata di lavoro

Implementazione, documentazione.

Gestione Esercizi | Diario di lavoro - 22.11.2019

Gabriele Alessi

Canobbio, 22.11.2019

Lavori svolti

Oggi durante le prime due ore ho prima di tutto implementato i metodi costruttori in tutti i Models del progetto su Visual Studio. Questo perché ci sono problemi con i DBrepository che hanno bisogno di lavorare con i modelli e i metodi degli stessi repository per funzionare correttamente. In contemporanea a ciò ho anche aggiornato la documentazione sperando di non modificare più questi capitoli base.

```
public Prova(string titolo, DateTime data, Anno anno, ICollection<EsercizioProva>
esercizi)
{
    Titolo = titolo;
    Data = data;
    Anno = anno;
    EserciziProva = esercizi;
}
```

Poi ho cominciato a sviluppare la View e il ViewModel della creazione della prova. Prima ho dovuto creare il repository per il modello dell'anno.

```
public AnnoDbRepository(AppDbContext ctx) : base(ctx) { }
```

Successivamente ho implementato il ViewModel (simile a quello per l'esercizio) e la View.

```
<!-- Data della prova -->
<DatePicker x:Name="dataDatePicker" SelectedDate="{Binding Path=Data}" />
<!-- Anno della prova -->
<ComboBox ItemsSource="{Binding Path=Anni}" Text="{Binding Path=Anno}" />
<!-- Griglia con i dati degli esercizi -->
<DataGrid x:Name="eserciziDataGrid" ItemsSource="{Binding Path=Esercizi}"
AutoGenerateColumns="False">
    <DataGrid.Columns>
        <DataGridTextColumn Header="Titolo" Binding="{Binding Path=Titolo}" />
        <DataGridTextColumn Header="Modulo" Binding="{Binding Path=Modulo}" />
        <DataGridCheckBoxColumn Header="Inserire nella prova"
            Selector.IsSelected="{Binding Path=Esercizio}" />
    </DataGrid.Columns>
</DataGrid>
```

Orario	Lavori svolti
13:15 - 16:30	Implementazione e Documentazione

Problemi riscontrati e soluzioni adottate

Mentre stavo creando il necessario per far funzionare il ViewModel della prova mi sono accorto che ci sono dei problemi di base che riguardano la struttura del database. Infatti non mi è chiaro come funziona l'entità [EsercizioProva](#) e come devo utilizzarla nel discorso della creazione della prova. L'altra entità che collega il tutto è [Anno](#), quindi devo chiarire con il docente come utilizzare questi oggetti per avere un sistema coerente.

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione.

Programma di massima per la prossima giornata di lavoro

Implementazione, documentazione.

Gestione Esercizi | Diario di lavoro - 26.11.2019

Gabriele Alessi

Canobbio, 26.11.2019

Lavori svolti

In questa giornata ho continuato con lo sviluppo delle Views lavorando soprattutto sulla parte della creazione di un esercizio. Prima però ho voluto eliminare il database per poi ricrearlo in modo da avere una versione definitiva in quanto i **Models** e l'**AppDbContext** dovrebbero essere conclusi dopo le ultime modifiche apportate. Ho impostato il percorso del database in modo che venga salvato nella cartella **AppData**, così facendo il sistema sarà uguale per qualunque utente.

```
protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
{
    if (!optionsBuilder.IsConfigured)
    {
        string dbPath =
            Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData)
            + "\\GestioneEsercizi";
        Directory.CreateDirectory(dbPath);
        optionsBuilder.UseLazyLoadingProxies().UseSqlite(
            "Data Source=" + dbPath + "\\GestioneEsercizi.sqlite");
    }
}
```

Dopodiché ho deciso di eliminare il campo **Tematiche** dall'**EsercizioViewModel** perché mi sono reso conto che non è utile per la creazione di un nuovo esercizio in quanto nel database non viene considerato.

```
<ComboBox Grid.Column="1" Grid.Row="0" SelectedIndex="0"
          ItemsSource="{Binding Path=Moduli}" SelectedItem="{Binding
Path=Modulo}"/>
```

```
public EsercizioViewModel()
{
    BenvenutoCommand = new DelegateCommand(OnBenvenuto, CanBenvenuto);
    SalvaCommand = new DelegateCommand(OnSalva, CanSalva);
    ModuloDbRepository repoModulo = new ModuloDbRepository(new AppDbContext());
    Moduli = new ObservableCollection<Modulo>(repoModulo.Get());
}
```

Orario	Lavori svolti
13:15 - 16:30	Implementazione e Documentazione

Problemi riscontrati e soluzioni adottate

Mi manca poco per concludere la creazione dell'esercizio, però c'è un problema quando bisogna salvarlo nel database. Infatti non viene salvato il modulo relativo all'esercizio quando si esegue l'`Insert` tramite l'`EsercizioDbRepository`. Ho provato a cambiare la proprietà che salva il modulo dalla View da `Text={Binding Path=Modulo}` a `SelectedItem={Binding Path=Modulo}`, ma in questo modo viene dato un errore nel repository.

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione.

Programma di massima per la prossima giornata di lavoro

Implementazione, documentazione.

Gestione Esercizi | Diario di lavoro - 28.11.2019

Gabriele Alessi

Canobbio, 28.11.2019

Lavori svolti

Durante questa giornata ho lavorato all'implementazione del salvataggio di una prova. Per fare ciò ho dovuto eliminare da costruttore del modello di dati il parametro che rappresenta gli esercizi e poi ho potuto sviluppare il metodo di salvataggio nel ViewModel. Inoltre ho anche dovuto creare i servizi dell'**EsercizioProva** per poter inserire i dati dei nuovi esercizi nel database.

```
ProvaDbRepository repoProva = new ProvaDbRepository(new DbContext());
EsercizioProvaDbRepository repo = new EsercizioProvaDbRepository(new
DbContext());
// Aggiungo la prova
Prova prova = new Prova(Titolo, Data, Anno);
repoProva.Insert(prova);
// Aggiungo gli esercizi della prova
foreach (Esercizio esercizio in new List<Esercizio>(Esercizi))
{
    repo.Insert(new EsercizioProva(esercizio, prova));
}
```

Per il resto del tempo ho documentato il lavoro svolto, cioè i ViewModels di Esercizio e Prova.

Orario	Lavori svolti
---------------	----------------------

13:15 - 16:30	Implementazione e Documentazione
---------------	----------------------------------

Problemi riscontrati e soluzioni adottate

Ho un problema che riguarda il salvataggio della prova. Infatti ho bisogno di definire gli esercizi da inserire però non riesco a trovare un modo su come reperirli dalla View. Per ora utilizzo una **DataGridCheckBoxColumn** pensando di poter ottenere gli esercizi che vengono visti, ma questo sistema sembra non funzionare. Dovrei consultare il docente per cercare una soluzione.

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione.

Programma di massima per la prossima giornata di lavoro

Implementazione, documentazione.

Gestione Esercizi | Diario di lavoro - 29.11.2019

Gabriele Alessi

Canobbio, 29.11.2019

Lavori svolti

Oggi ho continuato lavorando principalmente sulla documentazione del progetto. Sto cercando di concludere tutto ciò che non riguarda lo sviluppo dell'applicazione così da concentrarmi di più sulle ultime cose da implementare. Quindi ho riguardato un po' la documentazione inserendo alcune immagini e scrivendo la [ProvaView](#). Infine ho fatto la [GuidaView](#) e la [GuidaViewModel](#), le quali non contengono parti di codice particolari ma solo parti di testo al fine di informare l'utente sull'utilizzo del prodotto.

```
<!-- Titolo -->
<Label FontSize="25" FontWeight="Bold" Content="Guida - Gestione Esercizi"/>
<!-- Introduzione -->
<Label FontSize="20" Content="Introduzione"/>
<TextBlock x:Name="introduzione" Style="{StaticResource styleTextBlock}"/>
<!-- Impostazioni base -->
<Label FontSize="20" Content="Impostazioni di Base"/>
<TextBlock x:Name="impostazioniBase" Style="{StaticResource styleTextBlock}"/>
<!-- Nuovo Esercizio -->
<Label FontSize="20" Content="Nuovo Esercizio"/>
<TextBlock x:Name="esercizio" Style="{StaticResource styleTextBlock}"/>
<!-- Creazione Prova -->
<Label FontSize="20" Content="Creazione Prova"/>
<TextBlock x:Name="prova" Style="{StaticResource styleTextBlock}"/>
```

Poi ho anche creato i ViewModels utili per la creazione di nuove impostazioni di base. Ho dovuto fare ciò perché quando si vuole inserire una nuova entità sarebbe necessario usare il sistema dei Models e dei Repository quindi bisogna creare anche una View per ogni elemento che si vuole aggiungere.

```
private void OnSalva(object obj)
{
    ClasseDbRepository repo = new ClasseDbRepository(new AppDbContext());
    // Aggiungo la classe
    repo.Insert(new Classe(Nome, Anno));
    OnBenvenuto(obj);
}
```

Orario	Lavori svolti
13:15 - 16:30	Implementazione e Documentazione

Problemi riscontrati e soluzioni adottate

Nessun problema riscontrato.

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione.

Programma di massima per la prossima giornata di lavoro

Implementazione, documentazione.

Gestione Esercizi | Diario di lavoro - 03.12.2019

Gabriele Alessi

Canobbio, 03.12.2019

Lavori svolti

Durante questa giornata ho inizialmente avuto l'occasione di consultarmi con il docente per risolvere i problemi relativi all'inserimento di nuovi dati nel database. Mentre osservavamo il database è venuto fuori che ci sono un paio di incongruenze per quanto riguarda la relazione Anno-Classe. Quindi ho dovuto ristrutturare lo schema dei dati fondamentalmente scambiando le due entità.

```
public class Classe
{
    public string Nome { get; set; }
    public virtual Anno Anno { get; set; }
    public virtual ICollection<Modulo> Moduli { get; set; }
    public virtual ICollection<Prova> Prove { get; set; }
}

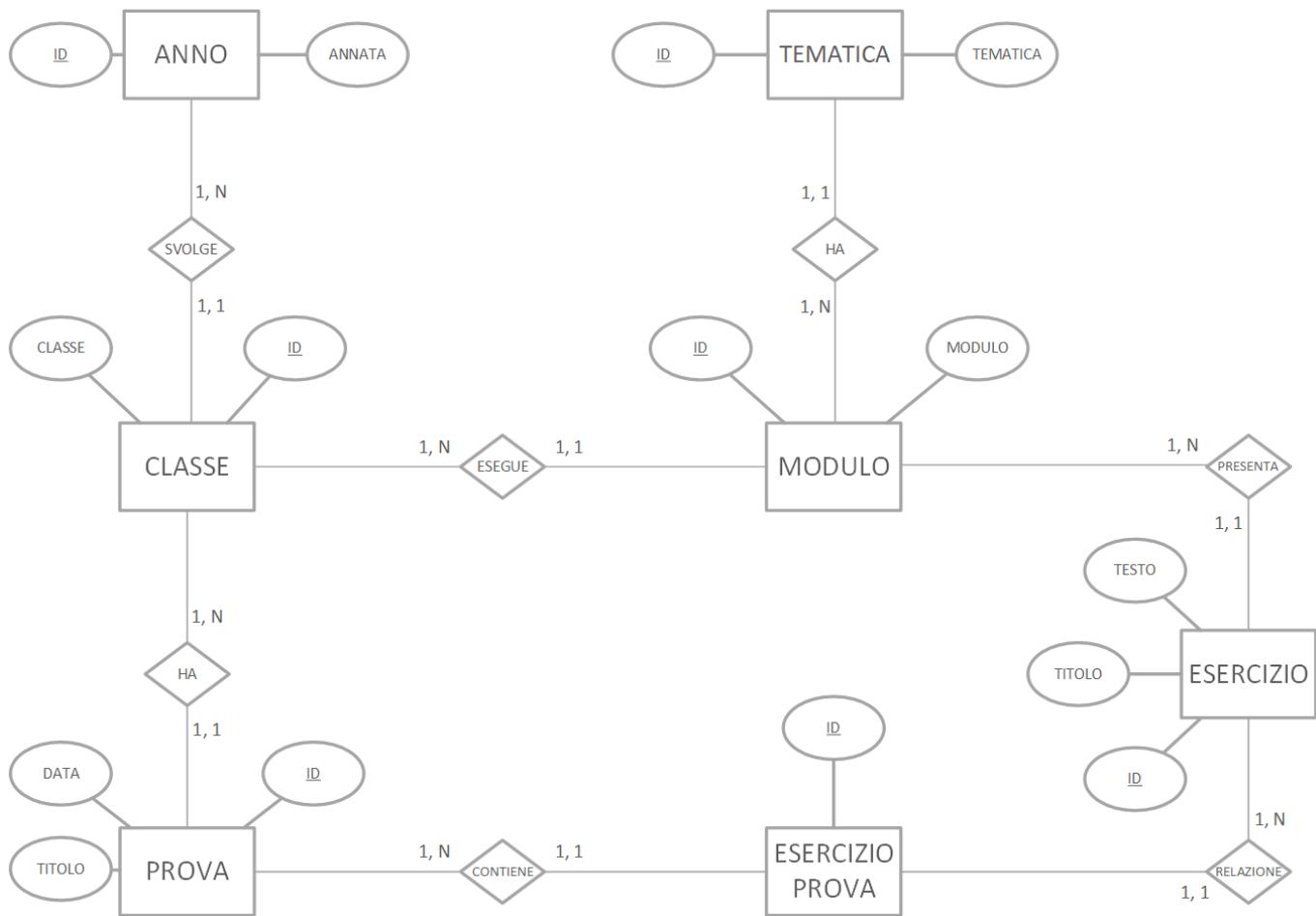
public class Anno
{
    public string Annata { get; set; }
    public virtual ICollection<Classe> Classi { get; set; }
}
```

Per risolvere il problema dell'inserimento nel database è necessario implementare una proprietà relativa all'ID dell'entità esterna in modo che funzioni come una foreign key.

```
public string Nome { get; set; }
public virtual Anno Anno { get; set; }
public int AnnoId { get; set; }

public Classe(string nome, Anno anno)
{
    Nome = nome;
    AnnoId = anno.Id;
}
```

Nel frattempo ho provveduto anche aggiornando la documentazione perfezionando anche i capitoli relativi alle interfacce e ai ViewModels e creando il nuovo schema del database.



Orario

Lavori svolti

13:15 - 16:30 Implementazione e Documentazione

Problemi riscontrati e soluzioni adottate

Nessun problema riscontrato.

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione.

Programma di massima per la prossima giornata di lavoro

Implementazione, documentazione.

Gestione Esercizi | Diario di lavoro - 06.12.2019

Gabriele Alessi

Canobbio, 06.12.2019

Lavori svolti

Oggi ho lavorato all'implementazione del progetto su Visual Studio e nel frattempo ho cercato di portarmi avanti con la documentazione. In generale sono a buon punto per quanto riguarda lo sviluppo dell'applicazione, manca solo qualche modulo qua e là per iniziare a concludere la cosa. Come prima cosa ho sviluppato l'algoritmo che aggiunge l'entità **Anno** in base alla data corrente quando si avvia l'applicazione.

```
AnnoDbRepository repo = new AnnoDbRepository(new AppDbContext());
// Anno corrente
string anno = DateTime.Today.Year.ToString()
    + "/" + (DateTime.Today.Year + 1).ToString();
// Inserisco l'anno se non è presente
try
{
    if (repo.Get().FirstOrDefault().Annata != anno) repo.Insert(new Anno(anno));
}
catch (NullReferenceException)
{
    repo.Insert(new Anno(anno));
}
```

Dopodiché ho concluso i Models e il database implementando le ultime proprietà per la gestione delle FK.

```
public class EsercizioProva : BaseEntity
{
    public virtual Esercizio Esercizio { get; set; }

    public int EsercizioId { get; set; }

    public virtual Prova Prova { get; set; }

    public int ProvaId { get; set; }

    public EsercizioProva(Esercizio esercizio, Prova prova)
    {
        EsercizioId = esercizio.Id;
        ProvaId = prova.Id;
    }
}
```

Quindi ho continuato sistemando tutto ciò che riguarda Views e ViewModels delle impostazioni di base concludendo questo capitolo (anche nella documentazione).

File Info		Classi				Moduli				Tematiche			
Classe	Anno	Nome	Classe	Tematiche	Esempio	Nome	Modulo						
I4AA	2019/2020	Modulo 151	I4AA	PDO		PDO	Modulo 151						
I3AA	2019/2020	Modulo 223	I3AA	MVVM, SQLite, MVC		MVVM	Modulo 223						
I4AC	2019/2020	Modulo 300	I4AC	Linux NAS, Client MacOS		SQLite	Modulo 223						
						MVC	Modulo 223						
						Linux NAS	Modulo 300						
						Client MacOS	Modulo 300						
Nuova classe		Nuovo modulo				Nuova tematica							
Indietro													

Orario

Lavori svolti

13:15 - 16:30 Implementazione e Documentazione

Problemi riscontrati e soluzioni adottate

Nessun problema riscontrato.

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione.

Programma di massima per la prossima giornata di lavoro

Implementazione, documentazione.

Gestione Esercizi | Diario di lavoro - 10.12.2019

Gabriele Alessi

Canobbio, 10.12.2019

Lavori svolti

Durante questa giornata ho continuato con l'implementazione delle Views proseguendo anche con la rispettiva documentazione. Posso dire di aver concluso tutto tranne le parti dell'esercizio e della prova. Inizialmente ho documentato le impostazioni di base e le Views relative alla loro aggiunta ([ClasseView](#), [ModuloView](#), [TematicaView](#)).

Questa View serve ad aggiungere una nuova classe nel sistema definendo il nome della classe e l'anno.

```
<TextBox Text="{Binding Path=Nome}" />
<ComboBox ItemsSource="{Binding Path=Anni}" SelectedItem="{Binding Path=Anno}" />
```

Poi mi sono concentrato cercando di concludere l'[EsercizioView](#) implementando l'inserimento dell'immagine nel testo dell'esercizio. Prima di tutto ho cambiato la struttura della View sviluppando un contenitore fatto apposta per vedere l'anteprima dell'immagine, uno spazio per fare il drag&drop e infine il pulsante per sfogliare tra i file.

```
<Image x:Name="immagine" Source="{Binding Path=Immagine}" />
<Button Content="Sfoglia..." Click="Sfoglia" />
<Label Content="Trascina l'immagine qui" AllowDrop="True" Drop="ImageDrop"/>
```

Quindi nel ViewModel ho dovuto aggiungere la proprietà per ottenere tramite il Binding l'immagine inserita:

```
public BitmapImage Immagine { get; set; }
```

Infine nella parte di codice della View ho sviluppato un metodo in modo che la sorgente dell'immagine fosse implementata correttamente.

```
private void SetImage(string filename)
=> immagine.Source = new BitmapImage(new Uri(filename, UriKind.Absolute));
```

Orario	Lavori svolti
13:15 - 16:30	Implementazione e Documentazione

Problemi riscontrati e soluzioni adottate

Ho riscontrato dei problemi per quanto riguarda l'ottenimento dell'immagine nel ViewModel tramite il Binding. Infatti la proprietà rimane a null nonostante vari tentativi di debug. La proprietà è di tipo `BitmapImage` in quanto la Source dell'immagine ha bisogno di questo tipo ma comunque non funziona.

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione.

Programma di massima per la prossima giornata di lavoro

Implementazione, documentazione.

Gestione Esercizi | Diario di lavoro - 12.12.2019

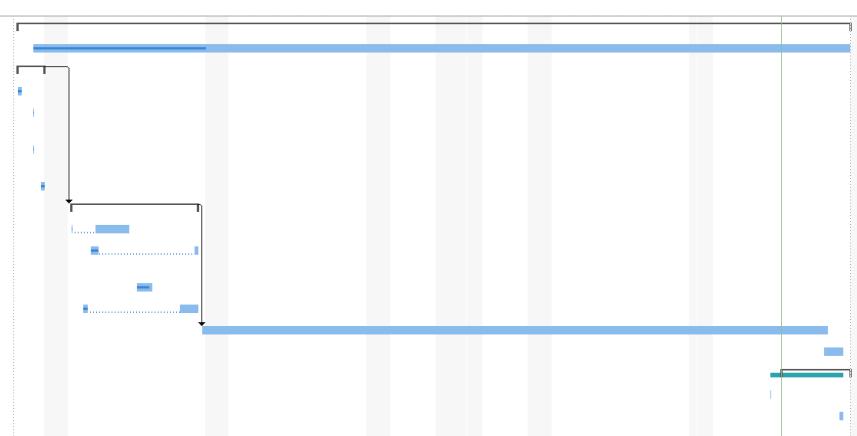
Gabriele Alessi

Canobbio, 12.12.2019

Lavori svolti

Oggi mi sono concentrato a lavorare su tutto ciò che non mi avrebbe creato problemi in modo da poter concludere le ultime cose del progetto con più concentrazione. Quindi mi sono messo a buon punto con la documentazione concludendo gran parte del capitolo delle Views e dei ViewModels lasciando solo un attimo in sospeso la creazione dell'esercizio e della prova. Inoltre ho anche cominciato con i test di sistema e ho fatto il Gantt consuntivo.

Gestione Esercizi	180 hrs	Tue 03/09/19	Fri 20/12/19
Documentazione	176 hrs	Thu 05/09/19	Fri 20/12/19
Analisi	12 hrs	Tue 03/09/19	Fri 06/09/19
Analisi QdC	4 hrs	Tue 03/09/19	Tue 03/09/19
Analisi del dominio, Analisi 2 hrs dei mezzi, Scopo		Thu 05/09/19	Thu 05/09/19
Analisi e specifica dei requisiti	2 hrs	Thu 05/09/19	Thu 05/09/19
Pianificazione	4 hrs	Fri 06/09/19	Fri 06/09/19
Progettazione	32 hrs	Tue 10/09/19	Thu 26/09/19
Design database	8 hrs	Tue 10/09/19	Tue 17/09/19
Design della struttura del progetto	8 hrs	Fri 13/09/19	Thu 26/09/19
Design delle interfacce	8 hrs	Thu 19/09/19	Fri 20/09/19
Design procedurale	8 hrs	Thu 12/09/19	Thu 26/09/19
Implementazione	128 hrs	Fri 27/09/19	Tue 17/12/19
Test	8 hrs	Tue 17/12/19	Thu 19/12/19
Conclusione	25 hrs	Thu 12/12/19	Fri 20/12/19
Consuntivo	2 hrs	Tue 10/12/19	Tue 10/12/19
Conclusione e revisione generale	4 hrs	Thu 19/12/19	Thu 19/12/19



Orario

Lavori svolti

13:15 - 16:30 Implementazione e Documentazione

Problemi riscontrati e soluzioni adottate

I problemi in sospeso con l'implementazione sono i seguenti:

- Ottenimento dell'immagine dalla View al ViewModel dell'esercizio tramite il Binding.
- Salvataggio dell'immagine in bytes per poterla salvare sotto forma di testo nel database e includerla nel campo **Testo** dell'entità **Esercizio**.
- Inserimento degli esercizi nella prova tramite la ProvaView, creazione e salvataggio della prova.

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione.

Programma di massima per la prossima giornata di lavoro

Conclusione implementazione, documentazione.

Gestione Esercizi | Diario di lavoro - 13.12.2019

Gabriele Alessi

Canobbio, 13.12.2019

Lavori svolti

Durante questa giornata ho principalmente lavorato alla documentazione e in generale alla conclusione del progetto.

Il seguente è un esempio di test case:

Test Case:	TC-006	Nome:	Inserimento tematica
Riferimento:	REQ-002		
Descrizione:	Si deve poter inserire una nuova tematica nel sistema.		
Prerequisiti:	Nel sistema devono essere presenti dei moduli per poter definire una nuova tematica.		
Procedura:	<ul style="list-style-type: none">Aprire l'applicazione e scegliere l'opzione "Impostazioni di Base".Cliccare sul pulsante "Nuova tematica".Digitare il nome della tematica e scegliere il modulo dal menu.Cliccare "Salva". Se i campi sono vuoti il salvataggio non viene effettuato.		
Risultati attesi:	La tematica appena inserita sarà visibile nella griglia.		

Inoltre ho concluso il capitolo 6 (Conclusioni) e sono già a buon punto per quanto riguarda la consegna della documentazione.

L'unica cosa che manca è l'implementazione dei problemi riscontrati come scritto nei diari dei giorni scorsi. Per questo ho inviato una mail al docente per chiedere aiuto a causa della sua assenza. Quindi spero di finire almeno l'EsercizioViewModel risolvendo quel problema.

Orario	Lavori svolti
13:15 - 16:30	Documentazione

Problemi riscontrati e soluzioni adottate

Nessun problema riscontrato.

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione.

Programma di massima per la prossima giornata di lavoro

Conclusione implementazione, documentazione.

Gestione Esercizi | Diario di lavoro - 17.12.2019

Gabriele Alessi

Canobbio, 17.12.2019

Lavori svolti

Oggi ho continuato lavorando alla documentazione del progetto. Prima di tutto ho riguardato tutti i capitoli precedenti a quelli dell'implementazione così da essere sicuro che non ci siano errori. Poi ho proseguito concludendo i vari Models, ViewModels e Views.

Per la prossima lezione dovrei riuscire a concludere tutto e prepararmi per consegnare.

Orario	Lavori svolti
13:15 - 16:30	Documentazione

Problemi riscontrati e soluzioni adottate

Nessun problema riscontrato.

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione.

Programma di massima per la prossima giornata di lavoro

Conclusione progetto.

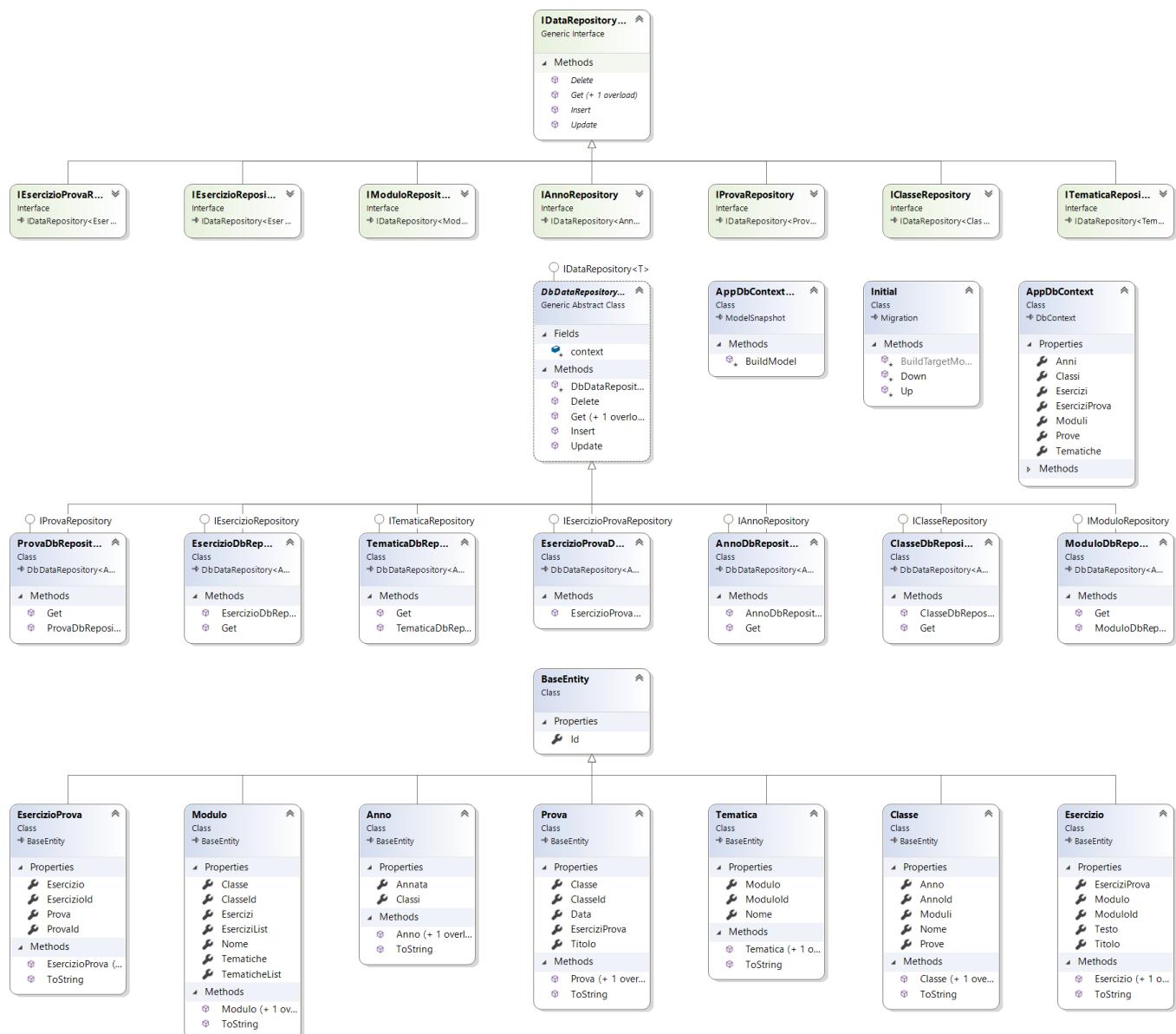
Gestione Esercizi | Diario di lavoro - 19.12.2019

Gabriele Alessi

Canobbio, 19.12.2019

Lavori svolti

Oggi ho concluso la documentazione e di conseguenza ho stampato tutto concludendo il progetto. Per finire ciò ho riguardato tutto e generato i diagrammi delle classi.



Orario	Lavori svolti
---------------	----------------------

13:15 - 16:30	Conclusione Documentazione
---------------	----------------------------

Problemi riscontrati e soluzioni adottate

Nessun problema riscontrato.

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione.

Programma di massima per la prossima giornata di lavoro

Consegna progetto.

Gestione Esercizi | Diario di lavoro - 20.12.2019

Gabriele Alessi

Canobbio, 20.12.2019

Lavori svolti

Durante questa giornata ho consegnato il progetto.