Gestione degli esercizi delle prove svolte durante la professione di docente

1 Introduzione 3

1.1 Informazioni sul progetto 3

1.2 Abstract 3

1.3 Scopo 3

Analisi 4

1.4 Analisi del dominio 4

1.5 Analisi e specifica dei requisiti 4

1.6 Use case 6

1.7 Pianificazione 7

1.8 Analisi dei mezzi 8

1.8.1 Software 8

1.8.2 Hardware 8

2 Progettazione 9

2.1 Design dell’architettura del sistema 9

2.2 Design dei dati e database 11

2.3 Design delle interfacce 12

2.3.1 Schermata principale 12

2.3.2 Impostazioni di base 13

2.3.3 Esercizi 14

2.3.4 Creazione prova 15

2.4 Design procedurale 16

3 Implementazione 17

3.1 Creazione progetto 17

3.2 Libreria di Classi 18

3.2.1 Models 18

4 Test 19

4.1 Protocollo di test 19

4.2 Risultati test 19

4.3 Mancanze/limitazioni conosciute 19

5 Consuntivo 20

6 Conclusioni 21

6.1 Sviluppi futuri 21

6.2 Considerazioni personali 21

7 Bibliografia 21

7.1 Sitografia 21

8 Allegati 21

# Introduzione

## Informazioni sul progetto

* Allievo: Gabriele Alessi   
  Superiore professionale: Ugo Bernasconi
* Scuola d’Arti e Mestieri di Trevano, Sezione informatica, Classe 4, Progetti individuali
* Data inizio: 03.09.2019   
  Data fine: 20.12.2019

## 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.

## Scopo

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.

Il progetto ha anche uno scopo scolastico, cioè la preparazione al lavoro finale LPI che si svolge alla fine dell’anno per l’ottenimento dell’AFC.

## Analisi

## Analisi del dominio

Fino a ora i professori gestivano e creavano gli esercizi e le prove a mano, quindi l’obiettivo è quello di sviluppare un programma che agevoli appunto la gestione degli esercizi. Il prodotto è leggero e semplice e veloce da usare, quindi può funzionare senza problemi sui computer dei docenti della SAMT.   
Attualmente non sembra esistere una soluzione simile, dunque il sistema funzionerà in modo che sia user-friendly e che più docenti potranno usarlo.

## Analisi e specifica dei requisiti

|  |  |
| --- | --- |
| **ID: REQ-001** | |
| **Nome** | Realizzare un programma che gestisca gli esercizi |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | - |
| **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) |

|  |  |
| --- | --- |
| **ID: REQ-002** | |
| **Nome** | Impostazioni di base |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Il programma deve essere strutturato in modo che si possano definire delle impostazioni di base |
| **Sotto requisiti** | |
| **001** | Definizione dei moduli |
| **002** | Definizione delle tematiche (con una sequenza cronologica) |
| **003** | Definizione delle classi |

|  |  |
| --- | --- |
| **ID: REQ-003** | |
| **Nome** | Definizione degli esercizi |
| **Priorità** | 1 |
| **Versione** | 1.1 |
| **Note** | Si devono poter vedere le informazioni principali per poi creare gli esercizi |
| **Sotto requisiti** | |
| **001** | Impostazione informazioni di base |
| **002** | Testo esercizio |
| **003** | Immagine esercizio (Drag & Drop per semplificare l’UI) |

|  |  |
| --- | --- |
| **ID: REQ-004** | |
| **Nome** | Creazione prove |
| **Priorità** | 1 |
| **Versione** | 1.1 |
| **Note** | Deve essere preparato un documento con i vari esercizi |
| **Sotto requisiti** | |
| **001** | Definizione campi fondamentali |
| **002** | Definizione esercizi (Drag & Drop per semplificare l’UI) |
| **003** | Preparazione documento |

## Use case



## Pianificazione

La pianificazione del progetto è stata effettuata mediante la realizzazione di un diagramma di Gantt.

## Analisi dei mezzi

### Software

I software utilizzati per la realizzazione di questo progetto sono:

* Microsoft Word 2016   
  Realizzazione della documentazione.
* GitHub Desktop 2.1.3  
  Gestione del sistema di versioning del progetto.
* Visual Studio Code 1.37.1  
  Realizzazione dei diari giornalieri e gestione dei vari documenti di testo.
* Visual Studio 2019  
  Sviluppo del programma.
* Microsoft Project 2016  
  Realizzazione del diagramma di Gantt.
* Microsoft Visio 2016   
  Realizzazione di vari schemi e diagrammi.

### Hardware

Personal Computer:

* HP Envy Notebook
* Intel© Core™ i7-6500U @ 2.50GHz
* 16GB
* Windows 10 Home 64bit
* Intel® HD Graphics 520

# Progettazione

## Design dell’architettura del sistema

In questo capitolo viene spiegata la struttura generale del sistema e in particolare l’architettura del progetto. Il programma è sviluppato in C# utilizzando il pattern MVVM, ma ci sono altri componenti utili per semplificare il lavoro che sono spiegati anche nel capitolo di implementazione.

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.



Per creare questa struttura mi sono valso dell’aiuto del superiore professionale che mi ha consegnato un’estensione che crea automaticamente l’oggetto Model, View, e ViewModel definito.



Per fare ciò ho anche usato il pacchetto DevExpressMvvm che è un insieme di strumenti che agevola il lavoro se si utilizza appunto il pattern MVVM. Inoltre esso lavora insieme all’estensione SAMT, quindi è molto semplice creare la struttura del progetto e gestire il sistema.



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).



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

## Design dei dati e database

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.

## Design delle interfacce

In questa sezione vengono mostrate e descritte le interfacce dell’applicazione con cui l’utente interagisce.

### Schermata principale

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



Il menu ha due opzioni e si estendono in questo modo:

* File
  + Esercizi  
    Mostra una schermata con lista degli esercizi salvati nel database.
  + Prove  
    Mostra una schermata con lista delle prove salvati nel database.
  + Opzioni  
    Permette di configurare diverse opzioni come la posizione dei salvataggi.
  + Esci  
    Chiusura dell’applicazione.
* Info
  + Guida  
    Mostra una schermata con una breve guida sul prodotto.
  + About  
    Mostra i diritti d’autore e altre informazioni riguardanti il prodotto (versione, licenza, …).

I tre pulsanti apriranno una nuova schermata in base alla relativa scelta e non sarà possibile uscire da quella schermata senza l’uso dei pulsanti “OK” o “Annulla”.

### Impostazioni di base

Le impostazioni di base sono una parte molto importante per il corretto funzionamento e uso del sistema. Infatti definendole sarà possibile creare gli esercizi per le specifiche classi che svolgono i moduli e seguono delle precise tematiche.

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.

### Esercizi

L’interfaccia che gestisce gli esercizi permette innanzitutto di scegliere tra le informazioni di base, per poi definire i dettagli dell’esercizio, cioè il testo e un eventuale immagine. Gli esercizi salvati verranno memorizzati per poi essere usati durante la realizzazione di una prova.



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.

### Creazione prova

L’opzione finale dell’applicazione è la creazione delle prove. In questa schermata vengono dapprima definiti i campi fondamentali (più o meno come negli esercizi) e infine vengono selezionati gli esercizi da inserire.



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.

## Design procedurale

In questo capitolo viene spiegato il ciclo di vita del prodotto e il suo comportamento in conseguenza di determinate azioni. Ciò è stato schematizzato in un diagramma di flusso dei dati.



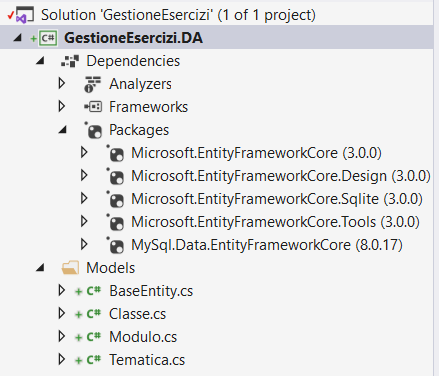
In questo caso il docente è l’unico oggetto che effettua delle azioni di inserimento dati. Le informazioni che quest’ultimo inserisce vengono memorizzate nelle informazioni di base o possono servire per creare un esercizio o una prova.

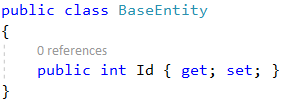
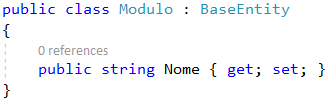
# Implementazione

Nel capitolo di implementazione viene spiegato in dettaglio come il prodotto finale è stato sviluppato.

## Creazione progetto

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.



Come base si implementa una classe BaseEntity che conterrà solamente la proprietà Id ed essa sarà superclasse di ogni altra entità in quanto l’Id è comune in tutto il sistema. Infine si aggiungono le classi che rappresentano le singole entità delle impostazioni di base (classe, modulo, tematica).

## Libreria di Classi

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.

### Models

La cartella Models contiene le entità di base del sistema, quindi classi, moduli, tematiche, anni, esercizi, prove e esercizi delle prove.

# Test

## Protocollo di test

Definire in modo accurato tutti i test che devono essere realizzati per garantire l’adempimento delle richieste formulate nei requisiti. I test fungono da garanzia di qualità del prodotto. Ogni test deve essere ripetibile alle stesse condizioni.

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-001  REQ-012 | **Nome:** | Import a card with KIC, KID and KIK keys, but not shown with the GUI |
| **Descrizione:** | Import a card with KIC, KID and KIK keys with no obfuscation, but not shown with the GUI | | |
| **Prerequisiti:** | Store on local PC: Profile\_1.2.001.xml (appendix n\_n) and Cards\_1.2.001.txt (appendix n\_n).  PIN (OTA\_VIEW\_PIN\_PUK\_KEY) and ADM (OTA\_VIEW\_ADM\_KEY) user right not set. | | |
| **Procedura:** | 1. Go to “Cards manager” menu,  in main page click “Import Profiles” link, Select the “1.2.001.xml” file, Import the Profile 2. Go to “Cards manager” menu,  in main page click “Import Cards” link, Select the “1.2.001.txt” file, Delete the cards,  Select the “1.2.001.txt” file, Import the cards 3. Research the “41795924770” Card, Click the imsi card link Check the card details 4. Execute the SQL: SELECT imsi, dir, keyset, cntr, rawtohex(kickey), rawtohex(kidkey), rawtohex(kikkey), rawtohex(chv), rawtohex(dap)FROM otacardkey a where imsi='340041795924770' ORDER BY keyset; | | |
| **Risultati attesi:** | Keys visible in the DB (OtaCardKey) but not visible in the GUI (Card details) | | |

## Risultati test

Tabella riassuntiva in cui si inseriscono i test riusciti e non del prodotto finale. Se un test non riesce e viene corretto l’errore, questo dovrà risultare nel documento finale come riuscito (la procedura della correzione apparirà nel diario), altrimenti dovrà essere descritto l’errore con eventuali ipotesi di correzione.

## Mancanze/limitazioni conosciute

Descrizione con motivazione di eventuali elementi mancanti o non completamente implementati, al di fuori dei test case. Non devono essere riportati gli errori e i problemi riscontrati e poi risolti durante il progetto.

# Consuntivo

Consuntivo del tempo di lavoro effettivo e considerazioni riguardo le differenze rispetto alla pianificazione (cap 1.7) (ad esempio Gannt consuntivo).

# Conclusioni

Quali sono le implicazioni della mia soluzione? Che impatto avrà? Cambierà il mondo? È un successo importante? È solo un’aggiunta marginale o è semplicemente servita per scoprire che questo percorso è stato una perdita di tempo? I risultati ottenuti sono generali, facilmente generalizzabili o sono specifici di un caso particolare? ecc

## Sviluppi futuri

Migliorie o estensioni che possono essere sviluppate sul prodotto.

## Considerazioni personali

Cosa ho imparato in questo progetto? ecc

# Bibliografia

## Sitografia

* <https://fmoralesdev.com/2019/05/16/generate-class-diagram-vs2019-net-core/>, Generate a class diagram in VS2019, 10-09-2019
* <https://www.c-sharpcorner.com/article/difference-between-net-framework-and-net-core/>, Difference Between .NET Framework and .NET Core, 27-09-2019

# Allegati

Elenco degli allegati, esempio:

* Diari di lavoro
* Codici sorgente/documentazione macchine virtuali
* Istruzioni di installazione del prodotto (con credenziali di accesso) e/o di eventuali prodotti terzi
* Documentazione di prodotti di terzi
* Eventuali guide utente / Manuali di utilizzo
* Mandato e/o Qdc
* Prodotto
* …