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 10

2.3 Design delle interfacce 11

2.4 Design procedurale 12

3 Implementazione 13

4 Test 14

4.1 Protocollo di test 14

4.2 Risultati test 14

4.3 Mancanze/limitazioni conosciute 14

5 Consuntivo 15

6 Conclusioni 16

6.1 Sviluppi futuri 16

6.2 Considerazioni personali 16

7 Bibliografia 16

7.1 Sitografia 16

8 Allegati 16

# 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

E’ una breve e accurata rappresentazione dei contenuti di un documento, senza notazioni critiche o valutazioni. Lo scopo di un abstract efficace dovrebbe essere quello di far conoscere all’utente il contenuto di base di un documento e metterlo nella condizione di decidere se risponde ai suoi interessi e se è opportuno il ricorso al documento originale.

* **Background/Situazione iniziale**
* **Descrizione del problema e motivazione**: Che problema ho cercato di risolvere? Questa sezione dovrebbe includere l'importanza del vostro lavoro, la difficoltà dell'area e l'effetto che potrebbe avere se portato a termine con successo.
* **Approccio/Metodi**: Come ho ottenuto dei progressi? Come ho risolto il problema (tecniche…)? Quale è stata l’entità del mio lavoro? Che fattori importanti controllo, ignoro o misuro?
* **Risultati**: Quale è la risposta? Quali sono i risultati? Quanto è più veloce, più sicuro, più economico o in qualche altro aspetto migliore di altri prodotti/soluzioni?

## 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 di testi |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Si devono poter vedere le informazioni principali per poi formare gli esercizi |
| **Sotto requisiti** | |
| **001** | Ricerca / Visione generale |
| **002** | Dettagli esercizi |

|  |  |
| --- | --- |
| **ID: REQ-004** | |
| **Nome** | Creazione prove |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Deve essere preparato un documento con i vari esercizi |
| **Sotto requisiti** | |
| **001** | Definizione campi fondamentali |
| **002** | Definizione esercizi |
| **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.
* StarUML 3.1.0   
  Realizzazione diagramma delle classi.

### 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, in particolare l’architettura del progetto e il diagramma delle classi.

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.

## 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, …).

### Impostazioni di base

## 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.

# Implementazione

In questo capitolo dovrà essere mostrato come è stato realizzato il lavoro. Questa parte può differenziarsi dalla progettazione in quanto il risultato ottenuto non per forza può essere come era stato progettato.

Sulla base di queste informazioni il lavoro svolto dovrà essere riproducibile.

In questa parte è richiesto l’inserimento di codice sorgente/print screen di maschere solamente per quei passaggi particolarmente significativi e/o critici.

Inoltre dovranno essere descritte eventuali varianti di soluzione o scelte di prodotti con motivazione delle scelte.

Non deve apparire nessuna forma di guida d’uso di librerie o di componenti utilizzati. Eventualmente questa va allegata.

Per eventuali dettagli si possono inserire riferimenti ai diari.

# 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

# 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
* …