Deduplicator

1 Introduzione 3

1.1 Informazioni sul progetto 3

1.2 Abstract 3

1.3 Scopo 3

2 Analisi 4

2.1.1 Analisi del dominio 4

2.1.2 Analisi e specifica dei requisiti 4

2.1.3 Use case 6

3 Pianificazione 6

3.1 Analisi dei mezzi 10

3.1.1 Software 10

3.1.2 Hardware 10

4 Progettazione 11

4.1 Design dell’architettura del sistema 11

4.2 Design delle interfacce 12

5 Implementazione 18

6 Test 18

6.1 Protocollo di test 18

6.2 Risultati test 19

6.3 Mancanze/limitazioni conosciute 19

7 Consuntivo 19

8 Conclusioni 19

8.1 Sviluppi futuri 19

8.2 Considerazioni personali 19

9 Bibliografia 19

9.1 Sitografia 19

10 Glossario 20

11 Allegati 20

# 1 Introduzione

Introduction

## Informazioni sul progetto

Questo progetto è estensione del progetto del primo semestre del anno scolastico 2019/2020 dove è stato creato un servizio per eseguire la ricerca e la gestione di duplicati di uno o più percorsi definiti dall’utente. Nel progetto del primo semestre è stata creata una GUI primitiva è incompleta.

Il progetto ha inizio in data 23.01.2020 e finisce il 06.04.2020, il docente responsabile è Geo Petrini.

## Abstract

The project that was terminated in the first semester of the 2019/2020 year consisted in the creation of a service to discover and manage duplicate files in one or more paths defined by the user, it had a primitive and incomplete GUI.   
This project has as the primary objective the creation of a modern GUI in order to use and manage all of the funcionalities of the service. The service has to have a modular approach, allowing additions to be implemented in the future. Furthermore there are some bug fixes to be made to the service.

## Scopo

Con questo progetto si vuole creare una interfaccia grafica, moderna, per l‘utente con il scopo di facilitare l’utilizzo e la gestione del servizio. L’implementazione deve avere un approccio modulare per permettere l’aggiunta di ulteriori moduli in futuro come la scansione di tracce audio o immagini.

# Analisi

### Analisi del dominio

Al momento il servizio creato nel primo progetto del semestre ha una GUI primitiva e incompleta, al mercato non esiste nessuna soluzione che potrebbe essere collegarsi e utilizzare le caratteristiche del servizio.

Il prodotto sarà sviluppato in modo da poterlo utilizzare tramite il browser e ogni servizio avrà una sua GUI che lavora in parallelo.

Gli utenti di questo prodotto saranno utenti singoli che useranno il programma per scopo personale ma si può utilizzare anche per lo scopo di piccole aziende con un paio di server.

L’interfaccia utente sarà utilizzata da utenti con conoscenze minime quindi dovrà eseguire tutti i controlli sui dati inseriti dall’utente.

### Analisi e specifica dei requisiti

I requisiti per questo progetto sono stati definiti dal docente responsabile Geo Petrini.

L’interfaccia utente dovrà avere un aspetto moderno ispirandosi al material design, un layout responsive e dovrà essere semplice e intuitiva per l’utilizzatore. Tramite l’interfaccia utente si dovranno inoltre poter configurare le impostazioni del servizio, quest’ultima parte è anche da aggiungere al servizio.

Il progetto deve essere strutturato in modo da permettere delle future espansioni di funzionalità.

|  |  |
| --- | --- |
| **ID: REQ-01** | |
| **Nome** | GUI moderna |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | L’interfaccia utente deve essere moderna e ispirata al material design |

|  |  |
| --- | --- |
| **ID: REQ-02** | |
| **Nome** | Layout responsive |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | L’interfaccia utente si deve adeguare a ogni grandezza di schermo del utente per permettere sempre una visuale utilizzabile. |
| **Sotto requisiti** | |
| **01** | Visuale desktop |
| **02** | Visuale tablet |
| **03** | Visuale mobile |

|  |  |
| --- | --- |
| **ID: REQ-03** | |
| **Nome** | GUI chiara e semplice |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | L’interfaccia utente deve essere chiara e semplice |

|  |  |
| --- | --- |
| **ID: REQ-04** | |
| **Nome** | Lo stato delle scansioni |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Nella GUI deve essere possibile visualizzare lo stato e una stima del tempo. |
| **Sotto requisiti** | |
| **01** | Modificare la parte del servizio per permettere di ricavare lo stato della scansione |
| **02** | Rappresentare lo stato con una barra di caricamento |
| **03** | Implementare una stima del tempo necessario al completamento della scansione |

|  |  |
| --- | --- |
| **ID: REQ-05** | |
| **Nome** | Panello per la gestione del servizio |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Nella GUI ci deve essere la possibilità di gestire completamente il servizio |

|  |  |
| --- | --- |
| **ID: REQ-06** | |
| **Nome** | Architettura modulare |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Il servizio deve essere sviluppato in modo modulare per permettere dei sviluppi e espansioni future. |

|  |  |
| --- | --- |
| **ID: REQ-07** | |
| **Nome** | Avvio semplice della GUI |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | La interfaccia utente deve essere semplice da avviare/eseguire. |

### Use case

Il prodotto avrà diversi casi d’uso, essi sono rappresentati con il seguente diagramma

Prima di fare qualsiasi operazione l’utente deve aver fatto l’accesso al server, questo non è rappresentato nello schema poiché farebbe lo schema molto complicato da leggere.



Figura 1 Diagramma use case

# Pianificazione

Nella figura 2 si può vedere la pianificazione iniziale, la durata del progetto ammonta a circa   
118 ore.

Il progetto verrà sviluppato seguendo un modello a cascata con l’eccezione che i test verranno fatti man mano che vado avanti con le parti d’implementazione. Il progetto essendo modulare non avrà bisogno di test che controllino l’integrità dell’intero sistema unico, poiché ogni modulo è indipendente.



Figura 2 Gantt iniziale

Nella figura 3 si può vedere le macro sezioni Introduzione e Analisi dove ho pianificato di spendere 6 ore in totale durante i primi giorni del progetto. Nella fase di analisi ho analizzato i requisiti che si trovano nel QdC e gli ho inseriti nella documentazione nella sezione 1.4.2 Analisi e specifica dei requisiti.



Figura 3 Sezione analisi del gantt

Nella figura 4 si può vedere in che ordine sarà la durata della creazione dei mockup delle schermate che verranno implementate, in totale ho stimato che mi serviranno 12 ore per la creazione dei mockup delle interfacce e della struttura del programma e altre 2 per progettare le modifiche da fare sul progetto del primo progetto.



Figura 4 Sezione Progettazione del gantt

Nella figura 5 si può vedere la macro sezione dell’implementazione che è stimata di durare 78 ore, quindi la maggior parte del progetto. All’interno di essa le prime 12 ore ho pianificato che servono a effettuare delle modifiche sul progetto del primo semestre. Queste modifiche riguardano la correzione del orario dello scheduler e il modo in cui vengono fatte partire le thread per la scansione dei percorsi, inoltre le modifiche che andrò a fare per migliorare la scansione mi permetteranno anche di poter ricavare lo stato e il progresso della scansione.

L’implementazione vera e propria inizierà solo dopo che le modifiche del progetto del primo semestre saranno finite.

Visto che il progetto dovrà essere modulare anche i test verranno eseguiti man mano che vengono implementate le diversi parti del progetto, per questo motivo non c’è nessuna macro sezione di test alla fine della progettazione.

Le ultime due attività descrivono il tempo necessario che penso di impegnare per creare uno script di installazione e una guida, che è di 6 ore in totale.



Figura 5 Sezione implementazione gantt

Nella figura 6 si può vedere l’ultima parte del progetto dove dedico le ultime 20 ore del progetti alla stesura della documentazione, questo include la stampa e la rilegatura.



Figura 6 Sezione finale gantt

## Analisi dei mezzi

Per la creazione di questo progetto ho a disposizione tutti i programmi che sono messi a disposizione dalla scuola e 1 accesso presso l'hosting interno nel caso che sia necessario caricare il progetto sull’FTP della scuola.

### Software

* Vaadin 14
* VSCode 1.41.1
* MySQL 8.0
* Java 11
* Apache Maven 3.6.1
* Postman 7.8.0

TODO:mettere tool e versioni complete

### Hardware

Per lo sviluppo verrà utilizzato il mio portatile personale che ha le seguenti specifiche:

HP Pavilion 15-0800nz

CPU: i7-8550U

RAM: 16 GB DDR4

OS: Pop!\_OS 19.10 / Kernel: 5.3.0-7625-generic

# Progettazione

Per lo sviluppo della GUI utilizzerò il framework vaadin che permette di creare webapp con una interfaccia moderna utilizzando codice java, le interfacce create sono responsive e e funzionano completamente su la maggior parte dei browser

## Design dell’architettura del sistema



Figura 7 Schema dell'architettura del sistema

## Modifiche da apportare al progetto Deduplicator

Ci sono alcune modifiche che sono da eseguire sul vecchio progetto per correggere il funzionamento di esso. Queste modifiche sul progetto del primo semestre includono:

* La correzione del modo in qui viene fatta la scansione
* La possibilità di ricavare lo stato della scansione
* La correzione del tempo nello scheduler poiché adesso l’orario e la data d’esecuzione delle scansioni pianificate è sfasato.

### Modifiche thread di scansione

Attualmente il servizio deduplicator esegue la scansione di file nei percorsi impostati grazie a 10 thread di scansione. Queste thread dovrebbero eseguire la scansione dei percorsi in parallelo una con l’altra, ma questo non è il caso. Le thread aspettano che la thread che è stata eseguita prima di essa sia finita, rendendo la scansione sequenziale e non parallela.

Bisogna fare in modo che ci sia una thread che cerca i file nei percorsi specificati dall’utente e gli aggiunge ad una coda e poi ci saranno 10 thread che lavoreranno a svuotare quella coda prendendo quelli file e ricavano tutte le informazioni necessari (hash, data di modifica, grandezza...), inoltre le 10 thread salveranno i file e le loro informazioni nel database.

### Ricavare lo stato della scansione

Questa modifica è legata a quella menzionata sopra. Si tratta di ricavare lo stato della scansione grazie alla thread di scansione che verrà implementata. La thread di scansione mi darà l’informazione su quanti file ci sono e grazie al numero di file che ci sono nel database si può ricavare quanti file sono stati scansionati e quanti ne rimangono ancora da scansionare.

### Correzione del tempo di esecuzione dello scheduler

Al momento nel file **ScheduleChecker.java** sulle righe 79 e 80 ci sono due istruzioni che modificano il tempo di esecuzione de impostare alla thread di scansione per le scansioni pianificate:

79 startCalendar.add(Calendar.MONTH, 1); // Correzione del Calendario

80 startCalendar.add(Calendar.HOUR\_OF\_DAY, 2); // Correzione dell'ora

Durante l’implementazione del vecchio programma ho aggiunto queste due righe per correggere il mese di un mese in avanti l’ora di due ore in avanti, visto che l’implementazione di questa parte del programma è avvenuta in dicembre dell’anno scorso la correzione del mese era inserita per correggere il mese dal valore di 11 a 12, dopo ho scoperto che il range del mese nel tipo Calendar di java va da 0 a 11. Il motivo dietro la correzione del orario è una differenza nel fuso orario e il cambiamento da ora legale a ora solare che non ho previsto.

## Design delle interfacce

Nella figura 8 si può vedere come verrà strutturata la schermata di login, per accedere all’interfaccia utente.

// TODO controllare grandezza e posizione immagini



Figura 8 Schermata login

Nella figura 9 si può vedere la schermata dell’inserimento dei percorsi, che diventa accessibile insieme a tutte le altre schermate dopo che l’utente effettua il login. L’utente ha la possibilità di inserire un percorso che può essere impostato da scansionare o da ignorare durante una scansione. Cliccando l’icona della cartella si aprirà una visuale modale, che conterrà un file browser del filesystem del server, rappresentata nella Figura 10. Nella metà inferiore della finestra ci sarà una lista di percorsi già salvati sul server, essi si possono eliminare o modificare.



Figura 9 Schermata dell'inserimento dei percorsi



Figura 10 Schermata del file browser per la selezione del percorso

Nella figura 11 è rappresentata la schermata della gestione delle scansioni. L’utente può avviare, fermare o mettere in pausa una scansione.



Figura 11 Schermata della gestione delle scansioni

Nella figura 12 si può vedere la schermata dello scheduler, qua si può impostare la data di una futura scansione oppure la data sulla quale verrà eseguita settimanalmente o mensilmente una scansione.



Figura 12 Schermata della gestione dello scheduler

Nella figura 13 si possono vedere i rapporti delle scansioni fatte, cliccando il bottone “i”, blue, accanto al dropdown menu si apre una visuale con le informazioni riguardo la scansione scelta.

Nella tabella sotto il dropdown menu si possono vedere i duplicati trovati in quella tabella. Un duplicato è rappresentato dalla barra grigia con informazioni riguardo a quel duplicato (numero di file, grandezza di ogni file e hash di contenuti dei file), sotto la barra grigia si possono vedere i percorsi dei file duplicati con la data della loro ultima modifica e le azioni che si possono fare su quel file (ignorare, eliminare o muovere). Inoltre c’è la possibilità di ignorare tutti i file di quel duplicato.



Figura 13 Schermata della gestione dei rapporti e duplicati

Nella figura 14 si può vedere la pagina di gestione del servizio e della GUI. In questa schermata l’utente ha la possibilità di cambiare la password, nome utente, le credenziali per il server mysql del servizio e la posizione del file di log. Inoltre si può impostare l’intervallo di tempo nel quale verrà aggiornato lo stato della scansione rappresentato tramite la barra e i dati sulla parte sinistra della schermata nella figura 11.



Figura 14 Schermata della gestione del servizio e della GUI

# Implementazione

## Correzione errori primo progetto

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://gist.github.com/warmwaffles/8534618>, *PausableThreadPoolExecutor.java*, 03-02-2020.

# Glossario

// TODO fare glossario

# 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
* …