Documentazione

[1 Introduzione 3](#_Toc71204340)

[1.1 Informazioni sul progetto 3](#_Toc71204341)

[1.2 Abstract 3](#_Toc71204342)

[1.3 Scopo 3](#_Toc71204343)

[2 Analisi 4](#_Toc71204344)

[2.1 Analisi del dominio 4](#_Toc71204345)

[2.2 Analisi e specifica dei requisiti 4](#_Toc71204346)

[2.3 Use case 8](#_Toc71204347)

[2.4 Pianificazione 9](#_Toc71204348)

[2.5 Analisi dei mezzi 10](#_Toc71204349)

[2.5.1 Software 10](#_Toc71204350)

[2.5.2 Hardware 11](#_Toc71204351)

[3 Progettazione 12](#_Toc71204352)

[3.1 Design dell’architettura del sistema 12](#_Toc71204353)

[3.2 Design dei dati e database 12](#_Toc71204354)

[3.3 Design delle interfacce 12](#_Toc71204355)

[3.4 Design procedurale 12](#_Toc71204356)

[4 Implementazione 13](#_Toc71204357)

[5 Test 14](#_Toc71204358)

[5.1 Protocollo di test 14](#_Toc71204359)

[5.1.1 Test funzionali 14](#_Toc71204360)

[5.1.2 Test non funzionali 17](#_Toc71204361)

[5.2 Risultati test 17](#_Toc71204362)

[5.3 Mancanze/limitazioni conosciute 18](#_Toc71204363)

[6 Consuntivo 19](#_Toc71204364)

[7 Conclusioni 20](#_Toc71204365)

[7.1 Sviluppi futuri 20](#_Toc71204366)

[7.2 Considerazioni personali 20](#_Toc71204367)

[8 Bibliografia 21](#_Toc71204368)

[8.1 Bibliografia per articoli di riviste: 21](#_Toc71204369)

[8.2 Bibliografia per libri 21](#_Toc71204370)

[8.3 Sitografia 21](#_Toc71204371)

[9 Allegati 22](#_Toc71204372)

# Introduzione

## Informazioni sul progetto

|  |  |
| --- | --- |
| **Responsabile progetto** | Geo Petrini |
| **Autori** | Thaisa De Torre, Viktorija Tilevska |
| **Inizio** | 14.01.2021 |
| **Consegna** | 20.05.2021 |

## 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.

Può contenere alcuni o tutti gli elementi seguenti:

* **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?

Esempio di abstract:

*As the size and complexity of today’s most modern computer chips increase, new techniques must be developed to effectively design and create Very Large Scale Integration chips quickly. For this project, a new type of hardware compiler is created. This hardware compiler will read a C++ program, and physically design a suitable microprocessor intended for running that specific program. With this new and powerful compiler, it is possible to design anything from a small adder, to a microprocessor with millions of transistors. Designing new computer chips, such as the Pentium 4, can require dozens of engineers and months of time. With the help of this compiler, a single person could design such a large-scale microprocessor in just weeks.*

## Scopo

Lo scopo del progetto è di creare un OCR Command Line Interface. L’OCR CLI è uno strumento a linea di comando in grado di estrare il testo dalle immagini e creare un file in formato mettendoci dentro il testo dell’immagine. Inoltre, è possibile vedere le statistiche per ogni scansione, come il tempo impiegato della scansione e il numero delle parole contenute nel file txt.

# Analisi

## Analisi del dominio

Questo applicativo è orientato verso persone che hanno bisogno di avere il contenuto di un’immagine in testo facile da modificare. Visto che l’applicativo si utilizza da linea di comando, gli utenti devono avere una conoscenza base di come funziona la linea di comando.

## Analisi e specifica dei requisiti

|  |  |
| --- | --- |
| **ID: REQ-001** | |
| **Nome** | Acquisizione di immagini in formato PNG o JPG/JPEG |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | L’OCR deve accettare solo immagini RNG e JPG/JPEG |

|  |  |
| --- | --- |
| **ID: REQ-002** | |
| **Nome** | Rilevamento del testo dall’immagine con un algoritmo OCR |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | L'OCR deve rilevare il testo correttamente dall'immagine |

|  |  |
| --- | --- |
| **ID: REQ-003** | |
| **Nome** | Output del testo come TXT |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Ci deve essere un file TXT che contiene il testo rilevato |

|  |  |
| --- | --- |
| **ID: REQ-004** | |
| **Nome** | Dev’essere bilingue |
| **Priorità** | 2 |
| **Versione** | 1.0 |
| **Note** | Il contenuto può essere interpretato in 2 lingue: Italiano e Inglese |
| **ID: REQ-005** | |
| **Nome** | Upload file multipli con output singolo |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | L’elaborato di più file di input viene ritornato come output singolo.  file.png 🡪 file1.txt file.png file.jpg 🡪 file.txt  Anche con mask: \*.\* 🡪 file.txt |

|  |  |
| --- | --- |
| **ID: REQ-006** | |
| **Nome** | Statistiche scansioni |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Per ogni esecuzione dev’essere prodotta una statistica con: la quantità di parole rilevate e il tempo di elaborazione |

|  |  |
| --- | --- |
| **ID: REQ-007** | |
| **Nome** | Visualizzazione dati statistici tramite parametro opzionale |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | I dati statistici vengono visualizzati solamente se abilitati tramite parametro |

|  |  |
| --- | --- |
| **ID: REQ-008** | |
| **Nome** | Guida utilizzo |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Visualizzare una guida di utilizzo tramite parametro (--help). La guida viene visualizzata automaticamente in caso di parametri assenti |

|  |  |
| --- | --- |
| **ID: REQ-009** | |
| **Nome** | Utilità (applicazione) |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Il lavoro deve poter essere utilizzato in produzione nell’azienda |

|  |  |
| --- | --- |
| **ID: REQ-010** | |
| **Nome** | Gestione degli errori |
| **Priorità** | 2 |
| **Versione** | 1.0 |
| **Note** | eventuali errori sono identificati e gestiti tramite i mezzi adeguati |

|  |  |
| --- | --- |
| **ID: REQ-011** | |
| **Nome** | Registro eventi/Logging |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | L'elaborazione dei dati corretta o errata viene registrata in un file di registro. Al fine di permettere un’interpretazione corretta del registro, i dati pertinenti devono essere memorizzati nel formato appropriato e raggruppati in modo utile. |

|  |  |
| --- | --- |
| **ID: REQ-012** | |
| **Nome** | Tutte le dipendenze del software devono essere incluse |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Eventuali librerie esterne devono essere incluse e facilmente installabili |

|  |  |
| --- | --- |
| **ID: REQ-013** | |
| **Nome** | Gestione delle eccezioni (batch) |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Il programma deve essere in grado di funzionare senza supervisioni e di reagire alle potenziali situazioni secondo dei processi predefiniti. In caso d’errore, il programma effettua le azioni definite conformemente alla specifica, scrive i dati nel file di log in modo da permettere un’analisi della situazione che ha portato all’interruzione. |

|  |  |
| --- | --- |
| **ID: REQ-014** | |
| **Nome** | Attendibilità dei dati inseriti dall’utente |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | I campi di immissione sono contrassegnati in modo chiaro e vengono rivisti. Plausibilità: In caso di errore di digitazione, l'utente viene aiutato da indicazioni concrete e il campo corrispondente viene attivato/ evidenziato. |

|  |  |
| --- | --- |
| **ID: REQ-015** | |
| **Nome** | Organizzazione del programma |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Il programma è stato strutturato in maniera intelligente senza procedure e funzioni e sotto-procedure/funzioni ridondanti. |

|  |  |
| --- | --- |
| **ID: REQ-016** | |
| **Nome** | Utilizzo di diagrammi di flusso |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | I diagrammi di flusso rappresentano lo svolgimento completo del programma. È rappresentato in modo chiaro, facilmente leggibile e contiene anche dei commenti pertinenti. |

## Use case

I casi d’uso rappresentano l’interazione tra i vari attori e le funzionalità del prodotto.

## Pianificazione

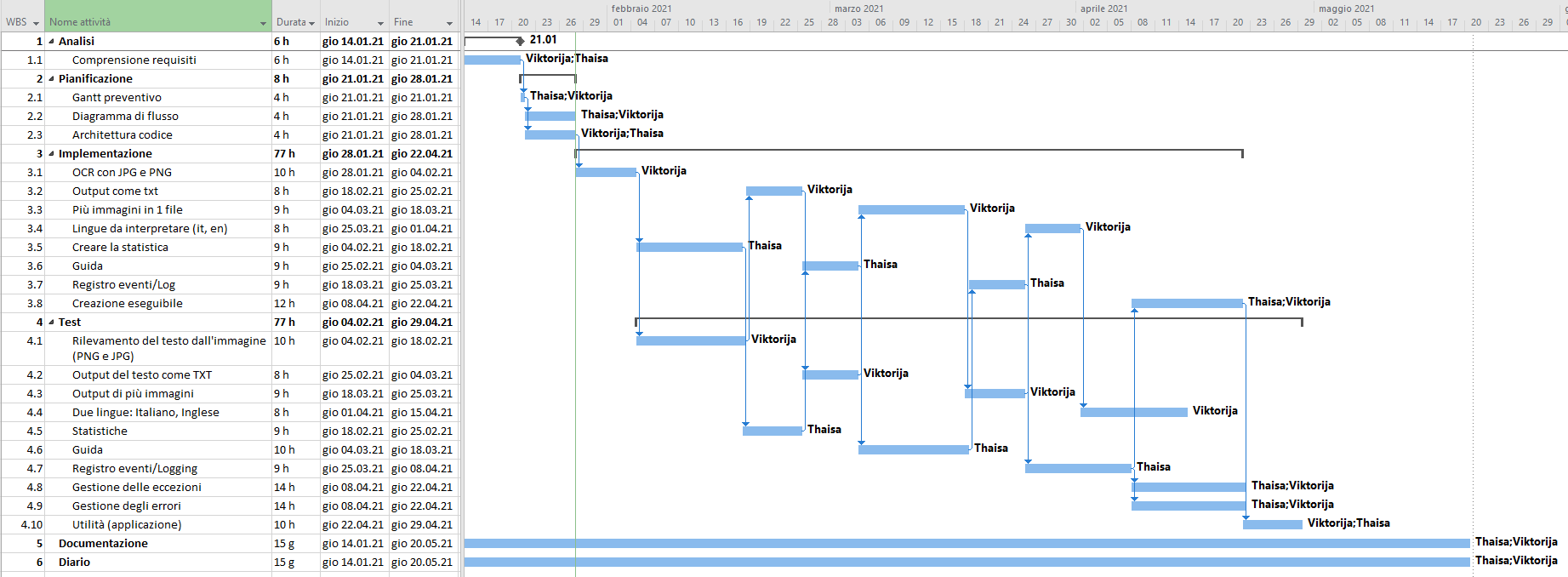
Nella foto sotto è rappresentato il nostro Gantt preventivo:

Figura 1 - Gantt preventivo

Il modello di sviluppo che abbiamo pensato di utilizzare era il modello a cascata (waterfall). C'è lo sviluppo a fasi sequenziali. Come si può vedere dal Gantt, la fine di una fase e l'inizio dell'altra fase.

Abbiamo scelto questo modello perché secondo noi era il modello più efficacie per il progetto. Abbiamo organizzato le fasi in modo che con la fine di una fase si può iniziare l'altra fase. Volevamo lavorare in un modo semplice dove dedichiamo tempo a ogni fase e dopo averla finita fare il test di solo quella fase e così via. In questo modo pensavamo di poter svolgere il lavoro in un modo organizzato senza avere tanti errori e senza dover tornare indietro e fare tutti i test di nuovo.

Per fare questo progetto avevamo a disposizione 5 mesi (dal 14 gennaio 2021 fino al 20 maggio 2021). Il lavoro l'abbiamo diviso nella fase di analisi, pianificazione, implementazione, test, e alla fine il tempo per la documentazione e il diario.

Abbiamo pensato di dedicare 1 settimana alle analisi, 1 settimana alla pianificazione, 12 settimane all'implementazione, altre 12 settimane ai test e alla fine un po’ di tempo ogni settimana per il diario e la documentazione.

Nella fase di analisi c'è l'attività di comprensione dei requisiti. Cresta attività consiste nel guardare e rileggere il quaderno e costruire i requisiti necessari per lo svolgimento del progetto. La seguente fase è la fase della pianificazione. In questa fase pensavamo di fare il Gantt preventivo, i diagrammi di flusso e di pianificare l'architettura del codice.

La prossima fase è quella dell'implementazione composta da tante attività necessarie per svolgere il progetto. Dopo l'implementazione c'è la fase dei test dove vengono fatti tutti i test. Nel nostro caso il fare d'implementazione e la fase di test vengono svolti contemporaneamente, nel senso che dopo un'attività d'implementazione viene eseguito il suo ripetitivo test. Oltre alle fasi principali abbiamo messo anche la documentazione e il diario come fasi perché abbiamo deciso che ogni settimana dedicheremmo del tempo che per scrivere il diario e la documentazione.

## Analisi dei mezzi

### Software

|  |  |  |  |
| --- | --- | --- | --- |
| **Nome del software** | **Versione** | **Utilizzo** **nel** **progetto** | **Collegamento** |
| Oracle VM VirtualBox | 6.1.16 | Software usato per creare le macchine virtuali | https://www.virtualbox.org/ |
| GitHub | 2.17.1 | Piattaforma che abbiamo utilizzato per gestire il progetto e sincronizzare il lavoro | https://github.com/ |
| GitHub Desktop | 2.6.3 | Interfaccia grafica che semplifica la gestione dei repository di GitHub | https://desktop.github.com/ |
| VisualStudio Code [extention Python] | 1.52.1 | Editor di testo utilizzato per sviluppare il programma in Python | https://code.visualstudio.com/ |
| Python | 3.9.0 | Linguaggio di programmazione | https://www.python.org/ |
| Pytesseract | 0.3.7 | È uno strumento OCR per Python | https://pypi.org/project/pytesseract/ |
| Tesseract OCR | 4.1.1 | Lo strumento OCR usato nel progetto | https://tesseract-ocr.github.io/ |
| Pillow | 8.1.0 | Libreria di Python per l’apertura, la manipolazione e il salvataggio immagini. | https://pillow.readthedocs.io/en/stable/ |
| Draw.io | 14.4.9 | Software per creare dei diagrammi di flusso. | https://app.diagrams.net/ |
| Microsoft Project Professional 2016 | 2016 | Software per creare il diagramma di Gantt. | https://www.microsoft.com/en-us/microsoft-365/project/project-management-software |
| PowerPoint 2016 | 2016 | Software per creare la presentazione. | https://www.microsoft.com/en-us/microsoft-365/powerpoint?ms.officeurl=powerpoint&rtc=1 |

### Hardware

Questo progetto è stato fatto sui computer forniti dalla scuola.

# Progettazione

## Design dell’architettura del sistema

Per sviluppare il programma abbiamo pensato di scrivere 4 classi diverse. Una classe sarà la classe principale che conterrà il metodo main. Un'altra classe si occuperà della lettura delle immagini e della scrittura del file txt. La terza classe servirà a fare le statistiche e l'ultima, a creare i file di log e fare logging.

L'utente eseguirà la classe ocr aggiungendo i parametri e la classe ocr a sua volta eseguirà le altre classi. Il gran parte del lavoro verrà fatto nella classe di lettura e scrittura. Li ci saranno metodi per controllare i dati inseriti dall'utente, controllare se i file di input esistono, sono accessibili e se sono nel formato giusto. Ci saranno metodi per leggere il testo di un'immagine e convertirlo in una stringa. Poi ci saranno metodi per creare il file di output e scrivere la stringa con il testo dell'immagine nel file.

Oltre alla classe reader ci sarà la classe logger che si occuperà di creare i file di log e scrivere dentro messaggi di log che serviranno all'utente in caso di problemi.

Poi ci sarà la classe stats che si occuperà di calcolare il tempo impiegato dal programma e di calcolare il numero di parole contenute nel file di output. Questo output verrà stampato soltanto se il parametro per visualizzare le statistiche sarà true.

Questa è la struttura generali che abbiamo pianificato per sviluppare il programma.

## Design dei dati e database

Non abbiamo utilizzato dei database per questo progetto.

## Design delle interfacce

Questo progetto non ha un'interfaccia grafica e funziona da linea di comando. L'utente chiama il programma ocr.py inserendo i parametri e riceve un file contenente tutte le scannerizzazioni di tutte le immagini.

Il primo parametro, il percorso del file da scannerizzare, è obbligatorio e se non viene inserito quello il programma termina e viene stampata la guida d'utilizzo.

Se l'utente non specifica gli altri parametri (la lingua, la destinazione e il nome del file) il programma prende i valori di default.

Se il parametro stats è true al termine del programma vengono stampati anche le statistiche del file di output, altrimenti non viene stampato niente.

## Design procedurale

<foto> Qui sopra c'è il diagramma di flusso del progetto. Questo è come abbiamo pensato di sviluppare il programma.

All'avvio del programma, viene dichiarata una variabile che contiene il tempo attuale e ci servirà per calcolare il tempo che il programma ha impiegato a fare tutto. Dopo di questo vengono controllati i parametri inseriti dall'utente. Se i parametri non sono accettabili viene stampata la guida d'utilizzo e il programma termina, mentre se i parametri sono accettabili si controlla il percorso. Viene controllato se il percorso è una directory o un file. Se si tratta di una directory, tutti i file nella cartella vengono estratti e aggiunti alla lista di file, mentre se è un file, lo stesso viene aggiunto alla lista dei file. Da questo punto tutti i file nella lista vengono controllati se sono dei file validi ovvero se esistono, sono accessibili in lettura e sono nel formato accettato. I file validi vengono inseriti in un'altra lista mentre tutti gli altri file vengono ignorati. L'utente viene avvisato che ci sono stati dei file ignorati.

Tutti i file nella lista dei file validi vengono passati a un metodo che utilizza il pytesseract per leggere il contenuto dell'immagine e ritornare una stringa nella lingua specificata dal utente, contenente il testo rilevato dall'immagine. Tutte le stringhe ritornate vengono salvate in un'altra lista.

Se la lista dei file validi è vuota viene stampato un messaggio d'errore e il programma termina.

Quando abbiamo la lista con tutti gli output delle immagini possiamo mettere tutto in un file txt. Il nome del file txt è specificato dall'utente e viene creato nella destinazione specificata dall'utente. Se l'utente non specifica la lingua, la destinazione e il nome del file, vengono utilizzati i valori di default.

Dopo che abbiamo il file di output calcoliamo il tempo impiegato dal programma. Se all'inizio l'utente ha messo il parametro stats, vengono stampate anche le statistiche del file txt altrimenti il programma termina.

# Implementazione

Dopo aver fatto un po' di ricerche abbiamo deciso di lavorare con Python. Python è comodo per lavorare con le CLI e ci sono molti algoritmi OCR buoni. Come OCR abbiamo scelto il Tesseract. Quindi abbiamo dovuto installare Python sulle nostre macchine virtuali.

Per lavorare con il Tesseract in Python abbiamo dovuto installare le librerie pytesseract, Pillow.  
Pytesseract è lo strumento OCR per Python, mente Pillow è una libreria per l'apertura, la modifica e il salvataggio delle immagini.

Durante l'installazione del pytesseract ci sono stati vari problemi con il proxy. Con l'auto del mandante abbiamo deciso di sfruttare l'ambiente virtuale di Python e abbiamo risolto il problema dell'installazione. Dopo aver installato tutto ciò che ci serviva abbiamo fatto un po' di ricerca per capire meglio come lavorare in Python visto che era la nostra prima volta a lavorare in Python.

Quando era tutto pronto abbiamo deciso che Thaisa farà la parte di logging, statistiche e la classe con il metodo main, mentre Viktorija scriverà la classe che si occupa di leggere l’immagine e stamparne l’output.

Viktorija

Abbiamo creato le 4 classi che abbiamo pianificato:

* Reader.py
* Stats.py
* Ocr.py
* Log\_handler.py

Reader.py è la classe che si occupa di leggere l'immagine e stamparne il contenuto in un file. Oltre a questo controlla anche se i file inseriti sono validi e controlla se l'utente ha il permesso di leggere il file che vuole scannerizzare. La classe reader.py è composta da tanti metodi e usa tante librerie.

Le librerie che usa sono:

* argparse
* pytesseract
* logging
* sys
* os.path
* glob
* from pathlib import Path
* from os import path
* from PIL import Image

I metodi presenti nella classe sono:

* scan(source, lang)
* validate\_source(source)
* has\_valid\_ext(src)
* check\_permission(path)
* get\_dir\_content(path)
* img\_to\_text(img, lang)
* write\_output(text, path)
* output(output, dest, prefix)
* merge\_output(output)
* reate\_dir(dir)
* validate\_dest(dest, prefix)

Il programma funziona che quando l’utente esegue la classe ocr.py, tramite la classe viene chiamata la classe reader.py che contiene il metodo scan (source, lang).

**scan(source, lang)**

Questo metodo fa lo scan di tutti i file validi.  
Il parametro source è la sorgente di files da scannerizzare mentre il parametro lang è la lingua del file.  
Alla fine il metodo ritorna una lista che contiene i dizionari di tutti gli elementi dell'array.

La sorgente viene salvata in un array e poi viene controllata la dimensione dell’array.   
Se l'array non è vuoto per ogni elemento (immagine) nell'array viene rilevato il testo. Per fare questo viene usato il metodo img\_to\_text(img, lang) nella lingua specificata con il parametro lang. Poi viene creato un dizionario che contiene l'id dell'elemento (immagine) e il testo rilevato.   
Invece se l'array è vuoto il programma termina.

**validate\_source(source)**

Questo metodo fa tutti i controlli e i cambiamenti in modo da avere una lista con solamente i file validi da scannerizzare.   
Il parametro source è una lista con i percorsi sorgente.   
Alla fine il metodo ritorna una lista con tutti i percorsi validi per l'OCR.

In una variabile viene controllata la dimensione della sorgente e viene salvato il numero di immagini che l'utente ha inserito. Per ogni percorso inserito viene controllato se è una mask. Se è una mask, il percorso viene esteso e diventa più corsi derivati dalla mask.   
Adesso che abbiamo risolto le mask e abbiamo tutti i percorsi bisogna controllare quali di quelli sono delle cartelle e quali sono dei file.

Per ogni percorso passato si controlla se l'utente ha il permesso di leggere/aprire il percorso con il metodo check\_permission(img). Dopo che abbiamo controllato i permessi bisogna controllare se il percorso è un file usando la funzione isFile() della libreria Path. Dopo di questo si controlla se i file passati sono accettabili dall'OCR richiamando il metodo has\_valid\_ext(src). Se i file sono accettabili vengono inseriti nella lista contenente tutti i file validi.

Per controllare se il percorso è una cartella bisogna usare la funzione isdir() della libreria Path. Se il percorso è una cartella, si ricava il contenuto della cartella usando il metodo get\_dir\_content() e il contenuto viene salvato in una variabile. Dopo di questo si controlla l'estensione anche di questi file e se l'estensione è accettabile il file viene salvato nella lista contenente tutti i file validi. La lista, alla fine, viene ritornata.

**has\_valid\_ext(src)**

Questo metodo controlla se il formato del file al percorso src è accettato dall'OCR. Src è il percorso del file. Alla fine il metodo ritorna true se il formato è accettato, altrimenti ritorna false.

In un array abbiamo tutte le estensioni accettate dall'OCR. In una variabile viene salvata solo l'estensione del file passato usando la funzione splitext() della libreria os.path e scegliendo l'ultimo elemento (-1). Se l'estensione del file è nell'array allora il metodo ritorna true, altrimenti ritorna false.

**check\_permission(path)**

Questo metodo controlla se il file è accessibile in lettura. Path è il percorso del file. Il metodo ritorna true se il file è accessibile, altrimenti ritorna false.

Viene creata la variabile valid che di default è false dopo di che viene aperto il percorso. Se il file, effettivamente, si apre allora il valore della variabile cambia in true, altrimenti il valore rimane false e nel file di log viene scritto che il percorso non è accessibile in lettura. Alla fine il metodo ritorna la variabile valid.

**get\_dir\_content(path)**

Questo metodo prende e ritona il contenuto di una cartella. Path è il percorso della cartella. Il metodo ritona una lista con tutti i file della cartella.

Viene controllata la dimensione della cartella e se la dimensione non è 0 il metodo usa la funzione listdir della libreria os per ritornare la lista di tutti i file. Se la dimensione è 0 non viene ritornato niente.

img\_to\_text(img, lang)

Questo metodo passa un'immagine all'OCR che la legge e ne ritorna il testo. Img è il percorso del file, lang é la lingua. Il metodo ritorna una stringa con il contenuto dell'immagine.

In una variabile viene salvato il contenuto dell immagine che viene ricavato usando la funzione image\_to\_string() della libreria pytesseract. Se il ricavimento dell testo ha successo allora viene ritornata la variabile altrimenti non viene ritornato niente.

write\_output(text, path)

Questo metodo scrive il contenuto text nel file al percorso path in utf-8. Text è il testo da scrivere nel file, path è il percorso del file.

In una variabile path viene salvato il percorso normalizzato del parametro path. Il percorso nella variabile path viene aprerto in lettura in utf-8 e dentro il file viene scritto il testo contenuto nel parametro text.

output(output, dest, prefix)

Questo metodo gestisce tutta la parte di output, controlla che la destinazione esista e che si possa scrivere. Se è un file scrive direttamente (sovrascrive se gia esiste) mentre se è una cartella gestisce eventuali duplicati. Output è un dizionario contente l'associazione tra immagine e testo scannerizzato insieme ad altre info. Dest è la destinazione in cui scrivere l'output. Se è un file scrive tutto li, se è una cartella salverà le scansioni in quella dir. Prefix è il prefisso che avrà il file di output per evitare duplicati nella stessa cartella.

Alla fine il metodo ritorna il file di destinazione in cui ha scritto.

# Test

## Protocollo di test

### Test funzionali

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-001  REQ-001 | **Nome:** | Acquisizione di immagini in formato PNG o JPG/JPEG | |
| **Descrizione:** | Verificare che l'OCR prenda solo PNG e JPG/JPEG | | | |
| **Prerequisiti:** | Avere installato l’OCR e avere un’immagine di un testo in formato PNG, JPG/JPEG e un altro formato | | | |
| **Procedura:** | 1. Utilizzare l’OCR con un’immagine JPEG, JPG, PNG e un altro formato   > ocr.py img/engtxtjpeg.jpeg  > ocr.py img/engtxtpng.png  > ocr.py img/itatxtjpg.jpg  > ocr.py img/error.tiff   1. Guardare il file di log per capire quali sono i formati accettati | | | |
| **Risultati attesi:** | L’OCR dovrebbe accettare solo le immagini PNG e JPG/JPEG e per i file con un formato diverso stampa il messaggio d’errore. | | | |
| **Dato** | | | **Risultato** |
| Con PNG | | | OK |
| Con JPG | | | OK |
|  | Con JPEG | | | OK |
|  | Altro formato | | | FAILED |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-002  REQ-002 REQ-003 | **Nome:** | Rilevamento del testo dall’immagine con un algoritmo OCR | |
| **Descrizione:** | L'OCR deve rilevare il testo correttamente dall'immagine | | | |
| **Prerequisiti:** | Avere installato l’OCR e avere un’immagine di un testo in formato JPG/JPEG o PNG. | | | |
| **Procedura:** | 1. Utilizzare l’OCR con un’immagine che rispetta le specifiche che si possono trovare nella documentazione 2. Guardare l’immagine e il file di output e vedere se il testo è uguale | | | |
| **Risultati attesi:** | Il testo rilevato dall’immagine deve coincidere con il testo dell’immagine | | | |
| **Dato** | | | **Risultato** |
| Con PNG o JPG/JPEG | | | Output testo dall’immagine su file TXT |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-003  REQ-004 | **Nome:** | Rilevamento del testo in inglese, italiano e qualsiasi altra lingua | |
| **Descrizione:** | L'OCR deve rilevare il testo correttamente dall'immagine | | | |
| **Prerequisiti:** | Aver installato l’OCR e avere un’immagine di un testo in formato JPG/JPEG o PNG aggiungendo l’argomento –lang | | | |
| **Procedura:** | 1. Utilizzare l’OCR con un’immagine aggiungendo il parametro –l o –lang   > ocr.py img/itatxtjpg.jpg -l eng  > ocr.py img/itatxtjpg.jpg -l ita  > ocr.py img/itatxtjpg.jpg -l fra   1. Guardare il file di log per capire se la lingua inserita è accettata | | | |
| **Risultati attesi:** | L’OCR dovrebbe accettare solo le lingue Italiano e Inglese e per le altre lingue stampa un messaggio d’errore. | | | |
| **Dato** | | | **Risultato** |
| Con PNG/JPG/JPEG in inglese | | | OK |
| Con PNG/JPG/JPEG in italiano | | | OK |
| Altra lingua | | | FAILED |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-004  REQ-005 | **Nome:** | Inserire multipli file di input e ricevere un file di output singolo | |
| **Descrizione:** | Fare il rilevamento di più immagini e scrivere l’output in un singolo file TXT | | | |
| **Prerequisiti:** | Avere un’immagine in formato PNG o JPG/JPEG | | | |
| **Procedura:** | 1. Utilizzare l’OCR con più immagini   > ocr.py img/engtxtjpeg.jpeg img/engtxtpng.png img/itatxtjpg.jpg  > ocr.py img/\*  > ocr.py \*.\*   1. Controllare se nel file di output ci sono i contenuti di tutte le immagini | | | |
| **Risultati attesi:** | Si ottiene un singolo file di output | | | |
| **Dato** | | | **Risultato** |
| Input: img1.png img2.png, img3.jpeg | | | Un file di output .TXT che contiene il contenuto di tutte le immagini inserite |
| Input: img.jpg img/ img1/ img.png | | |
|  | Input: \*.\* | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-005  REQ-006  REQ-007 | **Nome:** | Statistiche scansioni | |
| **Descrizione:** | Controllare che dopo l’esecuzione dell’OCR vengano calcolate le statistiche e controllare che i dati statistici vengano visualizzati tramite il parametro opzionale | | | |
| **Prerequisiti:** | Avere un’immagine in formato PNG o JPG/JPEG | | | |
| **Procedura:** | 1. Utilizzare l’OCR   > ocr.py img/engtxtpng.png –stats | | | |
| **Risultati attesi:** | Alla fine dell’esecuzione a terminale dev’essere stampata la statistica del file.  Le statistiche devono coincidere con il file di output dell’immagine | | | |
| **Dato** | | | **Risultato** |
|  | Con il parametro –stats | | | Stampa le statistiche al terminale |
|  | Senza parametri | | | Esegue l’OCR normalmente senza mostrare le statistiche |
|  | Input: img.txt | | | Le statistiche stampate al terminale coincidono con le statistiche del file di output dell’immagine |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-006  REQ-008 | **Nome:** | Guida utilizzo | |
| **Descrizione:** | Con il parametro –h/--help viene stampata a terminale la guida di utilizzo | | | |
| **Prerequisiti:** | Utilizzare il programma ocr.py | | | |
| **Procedura:** | 1. Utilizzare il comando OCR con il parametro –h o -help   > ocr.py –help | | | |
| **Risultati attesi:** | Viene stampata a terminale la guida di utilizzo | | | |
| **Dato** | | | **Risultato** |
| Con il parametro –h / --h / --help | | | La guida di utilizzo |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-007  REQ-012 | **Nome:** | Dipendenze SW incluse |
| **Descrizione:** | Eventuali librerie esterne devono essere incluse e facilmente installabili | | |
| **Prerequisiti:** | Con l’exe devono essere installate e incluse tutte le dipendenze | | |
| **Procedura:** | 1. Eseguire il .exe dell’OCR 2. Controllare che tutte le dipendenze siano installate | | |
| **Risultati attesi:** | Tutte le dipendenze devono essere incluse e installate con l’exe | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-008  REQ-013 | **Nome:** | Attendibilità dei dati inseriti dall’utente | |
| **Descrizione:** | I parametri inseriti dall’utente sono verificati e corretti | | | |
| **Prerequisiti:** | Avere il programma funzionante | | | |
| **Procedura:** | 1. Utilizzare il comando OCR con tutti i parametri   > ocr.py img/engtxtpng.png –lang eng  > ocr.py img/engtxtpng.png –dest C:\Scans\  > ocr.py img/engtxtpng.png -prefix scan  > ocr.py img/engtxtpng.png –stats | | | |
| **Risultati attesi:** | L’OCR dovrebbe accettare solo i parametri dest, lang, prefix e stats. Per qualsiasi altro parametro (tranne –help/-h) dà errore. | | | |
| **Dato** | | | **Risultato** |
| -dest / -d | | | OK |
| -lang / -l | | | OK |
| -prefix / -p | | | OK |
| -stats | | | OK |
| Qualsiasi altro parametro | | | FAILED |

### Test non funzionali

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-09  REQ-009 | **Nome:** | Utilità (applicazione) |
| **Descrizione:** | Controllo del funzionamento del programma | | |
| **Prerequisiti:** |  | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-010  REQ-010 | **Nome:** | Gestione degli errori |
| **Descrizione:** | Verifica se tutti gli errori sono gestiti correttamente | | |
| **Prerequisiti:** |  | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-011  REQ-011 | **Nome:** | Registro eventi/Logging |
| **Descrizione:** | Verifica se i dati corretti o errati registrati in un file di log;  Verifica se i dati sono salvati nel formato appropriato e raggruppati in modo utile | | |
| **Prerequisiti:** | Utilizzare l’OCR | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-012  REQ-013 | **Nome:** | Gestione delle eccezioni (batch) |
| **Descrizione:** | Controllo se tutti gli errori sono gestiti correttamente | | |
| **Prerequisiti:** |  | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-013  REQ-015 | **Nome:** | Organizzazione del programma |
| **Descrizione:** | Se il programma è strutturato in metodi ed è commentato | | |
| **Prerequisiti:** |  | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-014  REQ-016 | **Nome:** | Utilizzo di diagrammi di flusso |
| **Descrizione:** | Se ci sono i diagrammi di flusso | | |
| **Prerequisiti:** |  | | |

## 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.

**Legenda**: test riuscito test fallito

|  |  |  |
| --- | --- | --- |
| ID Test Case | Nome | Risultato |
| TC-001 | Acquisizione di immagini in formato PNG o JPG/JPEG |  |
| TC-002 | Rilevamento del testo dall’immagine con un algoritmo OCR |  |
| TC-003 | Rilevamento del testo in inglese, italiano e qualsiasi altra lingua |  |
| TC-004 | Inserire multipli file di input e ricevere un file di output singolo |  |
| TC-005 | Statistiche scansioni |  |
| TC-006 | Guida utilizzo |  |
| TC-007 | Dipendenze SW incluse |  |
| TC-008 | Attendibilità dei dati inseriti dall’utente |  |
| TC-009 | Utilità (applicazione) |  |
| TC-010 | Gestione degli errori |  |
| TC-011 | Registro eventi/Logging |  |
| TC-012 | Gestione delle eccezioni (batch) |  |
| TC-013 | Organizzazione del programma |  |
| TC-014 | Utilizzo di diagrammi di flusso |  |

## 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

## Bibliografia per articoli di riviste:

1. Cognome e nome (o iniziali) dell’autore o degli autori, o nome dell’organizzazione,
2. Titolo dell’articolo (tra virgolette),
3. Titolo della rivista (in italico),
4. Anno e numero
5. Pagina iniziale dell’articolo,

## Bibliografia per libri

1. Cognome e nome (o iniziali) dell’autore o degli autori, o nome dell’organizzazione,
2. Titolo del libro (in italico),
3. ev. Numero di edizione,
4. Nome dell’editore,
5. Anno di pubblicazione,
6. ISBN.

## Sitografia

1. URL del sito (se troppo lungo solo dominio, evt completo nel diario),
2. Eventuale titolo della pagina (in italico),
3. Data di consultazione (GG-MM-AAAA).

**Esempio:**

* http://standards.ieee.org/guides/style/section7.html, *IEEE Standards Style Manual*, 07-06-2008.

# 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
* …