Wordcloud generator

1 Indice

1 Indice 2
2 Introduzione 4
2.1 Informazioni sul progetto 4
2.2 Abstract 4
2.3 Scopo 4
2.3.1 Scopo professionale 4
2.3.2 Scopo didattico 4
3 Analisi 5
3.1 Analisi del dominio 5
3.2 Analisi e specifica dei requisiti 6
3.3 Use case 9
3.4 Pianificazione 10
3.5 Analisi dei mezzi 12
3.5.1 Software 12
3.5.2 Librerie 12
3.5.3 Hardware 12
4 Progettazione 13
4.1 Design delle interfacce 13
4.2 Design procedurale 15
4.2.1 Diagramma di flusso 15
5 Implementazione 17
5.1 File guigenerator.py 17
5.1.1 Generazione GUI 17
5.2 Input per il testo 18
5.2.1 Metodo di input per le parole 18
5.2.2 Parole escluse e importanti 19
5.2.3 Selezione del font 19
5.3 Input per l'immagine 20
5.3.1 Path per l'immagine iniziale 20
5.3.2 Bordo dell'immagine 20
5.3.3 Colore del bordo 21
5.4 Algoritmo di floading 22
5.5 Wordcloud 23
5.6 Download dell'immagine 24
5.7 File requirements.txt 25
6 Test 26
6.1 Protocollo di test 26
6.2 Risultati test 32
6.3 Mancanze/limitazioni conosciute 32
7 Consuntivo 33
8 Conclusioni 34
8.1 Sviluppi futuri 34
8.2 Considerazioni personali 35

9 Bibliografia 36
9.1 Sitografia 36
10 Glossario 38
11 Indice delle figure 39
12 Allegati 39

2 Introduzione

2.1 Informazioni sul progetto

In questo capitolo raccogliere le informazioni relative al progetto, ad esempio:

- Allievi: Alessandro Curiale, Christian Monga, Edoardo Ratti
- Docente: Geo Petrini
- SAMT sezione informatica modulo 306
- 27.01.2023 – 05.05.2023

2.2 Abstract

Data la poca fantasia presente nei manifesti abbiamo ben pensato di provare noi a riempire ciò che mancava con una nostra semplice inventiva. La nostra idea è stata quella di provare a sviluppare un wordcloud.

Per crearlo abbiamo iniziato pensando quali potrebbero essere i parametri che il nostro applicativo potesse supportare, creato un'interfaccia grafica e aggiunto ad essa ciò che poteva essere realizzabile. Abbiamo utilizzato una struttura di lavoro dove tutti potessero fare qualcosa di diverso in modo tale di arrivare alla fine e riunire tutto assieme.

La parte che ha dedicato più tempo è stata quella che si occupava di preparare una maschera sulla quale inserire le parole, ciò perché abbiamo utilizzato un algoritmo già esistente, la quale abbiamo passato i parametri prelevati inizialmente.

Una volta finito il progetto con successo abbiamo messo assieme la conclusione che il vantaggio principale è nostro prodotto è un applicativo, al contrario delle numerose altre proposte online, inoltre a ciò si tratta di un software gratuito.

2.3 Scopo

2.3.1 Scopo professionale

Il nostro progetto ha lo scopo di ricreare delle immagini usando le parole, queste immagini possono essere personalizzate tramite delle opzioni presenti sull'interfaccia.

2.3.2 Scopo didattico

Imparare a gestire un progetto di gruppo utilizzando una modalità Agile e imparare a utilizzare Python.

3 Analisi

3.1 Analisi del dominio

Ultimamente quando vediamo manifesti e pubblicità ci sembrano un po' spogli, senza fantasia, allora il nostro team di sviluppo ha pensato a come dare un po' inventiva al marketing.
Idealmente vogliamo sviluppare un classico Wordcloud Generator per poterlo fornire a scuole e ditte.

Sicuramente alle scuole tornerebbe utile per fare brainstorming, dunque aumentare l'interesse degli allievi nell'esprimere le proprie opinioni, ma il nostro obbiettivo principale è quello di tornare utili alle ditte per fornire una migliore pubblicità hai clienti.
Anche se questo tipo di applicativo è già presente online a noi interessa riprodurlo per poterlo manipolare a nostro piacimento.

Il software è molto semplice, chiunque è capace di utilizzare un computer è anche in grado di utilizzarlo. Inoltre forniremo un breve manuale di utilizzo per coloro che si potrebbero trovare in difficoltà.

Prima di tutto, per entrare nell'ottica del dominio, occorre saper scegliere il linguaggio di programmazione, nel nostro caso abbiamo scelto di utilizzare Python, questo perché permette di gestire le librerie in modo semplice.
Delle molte librerie disponibili online utilizzeremo opencv per gestire le immagini, kivy per quanto riguarda le interfacce e tkinter per implementare delle funzionalità che con kivy non possiamo implementare.

Inoltre utilizzeremo Beautifulsoup per poter leggere dalle pagine web i testi in modo semplice, filetype per verificare che i file passati siano del tipo corretto, numpy per aiutarci con la creazione della maschera, wordcloud per generare l'immagine con le parole e infine matplotlib per mostrare l'immagine nell'applicazione.

3.2 Analisi e specifica dei requisiti

Requisito	Req-001	**Priorità**	1	**Versione**	1.0
Nome	Upload immagine				
Note	L'utente trascina e rilascia nello spazio apposito l'immagine su cui lavorare				

Requisito	Req-002	**Priorità**	1	**Versione**	1.0
Nome	Parti da mantenere fisse dell'immagine				
Note	L'utente sceglie quali parti trasformare in parole e quali no				
Sotto requisiti					
001					

Requisito	Req-003	**Priorità**	1	**Versione**	1.0
Nome	L'utente inserisce le parole da rappresentare nelle immagini				
Note	L'utente sceglie le parole da inserire nell'immagine				
Sotto requisiti					
001	L'utente può inserirle tramite testo				
002	L'utente può inserirle tramite file				
003	L'utente può inserirle tramite URL di una pagina che verrà scaricata				

Requisito	Req-004	**Priorità**	1	**Versione**	1.0
Nome	Si può personalizzare il font family				
Note	L'utente può scegliere il font family del testo scritto nell'immagine				
Sotto requisiti					
001	L'utente sceglie da un drop down il font family da applicare				

Requisito	Req-005	**Priorità**	1	**Versione**	1.0
Nome	Il colore delle parole si adatta con il colore dell'immagine				
Note	Il colore delle parole deve rispecchiare il colore dell'immagine di base				

Requisito	Req-006	**Priorità**	1	**Versione**	1.0
Nome	Set di parole bloccate				
Note	Il programma contiene già un set di parole bloccate che non verranno mostrate nell'immagine				
Sotto requisiti					
001	L'utente inserisce delle parole che non vuole che vengano mostrate				

Requisito	Req-007	**Priorità**	1	**Versione**	1.0
Nome	Parole con importanza maggiore				
Note	L'utente può inserire delle parole che devono essere mostrate più grande di come sarebbero normalmente				
Sotto requisiti					
001	L'utente sceglie da un drop down il font family da applicare				

Requisito	Req-008	**Priorità**	1	**Versione**	1.0
Nome	Aggiunta del bordo all'immagine				
Note	L'utente può scegliere di aggiungere un bordo all'immagine creata				
Sotto requisiti					
001	Il bordo può avere un colore				
002	Il bordo deve avere uno spessore a scelta				

Requisito	Req-009	**Priorità**	1	**Versione**	1.0
Nome	Aggiornamento in tempo reale dell'immagine				
Note	L'immagine si deve aggiornare in tempo reale seguendo le varie modifiche apportate dall'utente				

Requisito	Req-010	**Priorità**	1	**Versione**	1.0
Nome	Scelta risoluzione immagine				
Note	Si può scegliere la risoluzione dell'immagine una volta scaricata				
Sotto requisiti					
001	La risoluzione dell'immagine ha delle proporzioni definite da cui scegliere				

Requisito	Req-011	**Priorità**	1	**Versione**	1.0
Nome	Scelta formato dell'immagine				
Note	L'utente può scegliere il formato con cui deve essere scaricata l'immagine				
Sotto requisiti					
001	Il formato può essere .jpg o .png o .webp				

Spiegazione elementi tabella dei requisiti:

ID: identificativo univoco del requisito
Nome: breve descrizione del requisito
Priorità: indica l'importanza di un requisito nell'insieme del progetto, definita assieme al committente. Ad esempio, poter disporre di report con colonne di colori diversi ha priorità minore rispetto al fatto di avere un database con gli elementi al suo interno. Solitamente si definiscono al massimo di 2-3 livelli di priorità.
Versione: indica la versione del requisito. Ogni modifica del requisito avrà una versione aggiornata.
Sulla documentazione apparirà solamente l'ultima versione, mentre le vecchie dovranno essere inserite nei diari.
Note: eventuali osservazioni importanti o riferimenti ad altri requisiti.
Sotto requisiti: elementi che compongono il requisito.

3.3 Use case

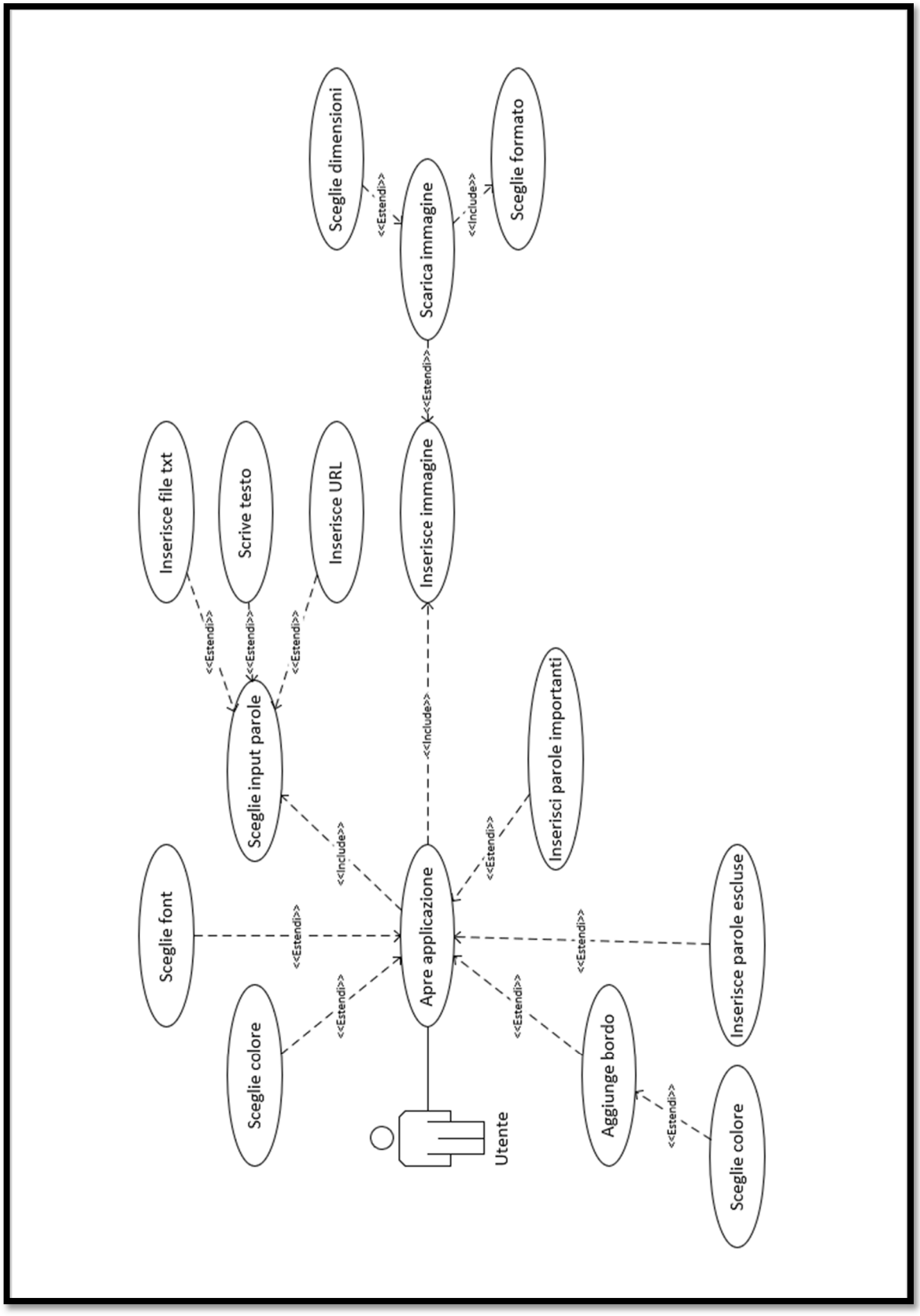

Figura 1 UseCase

3.4 Pianificazione

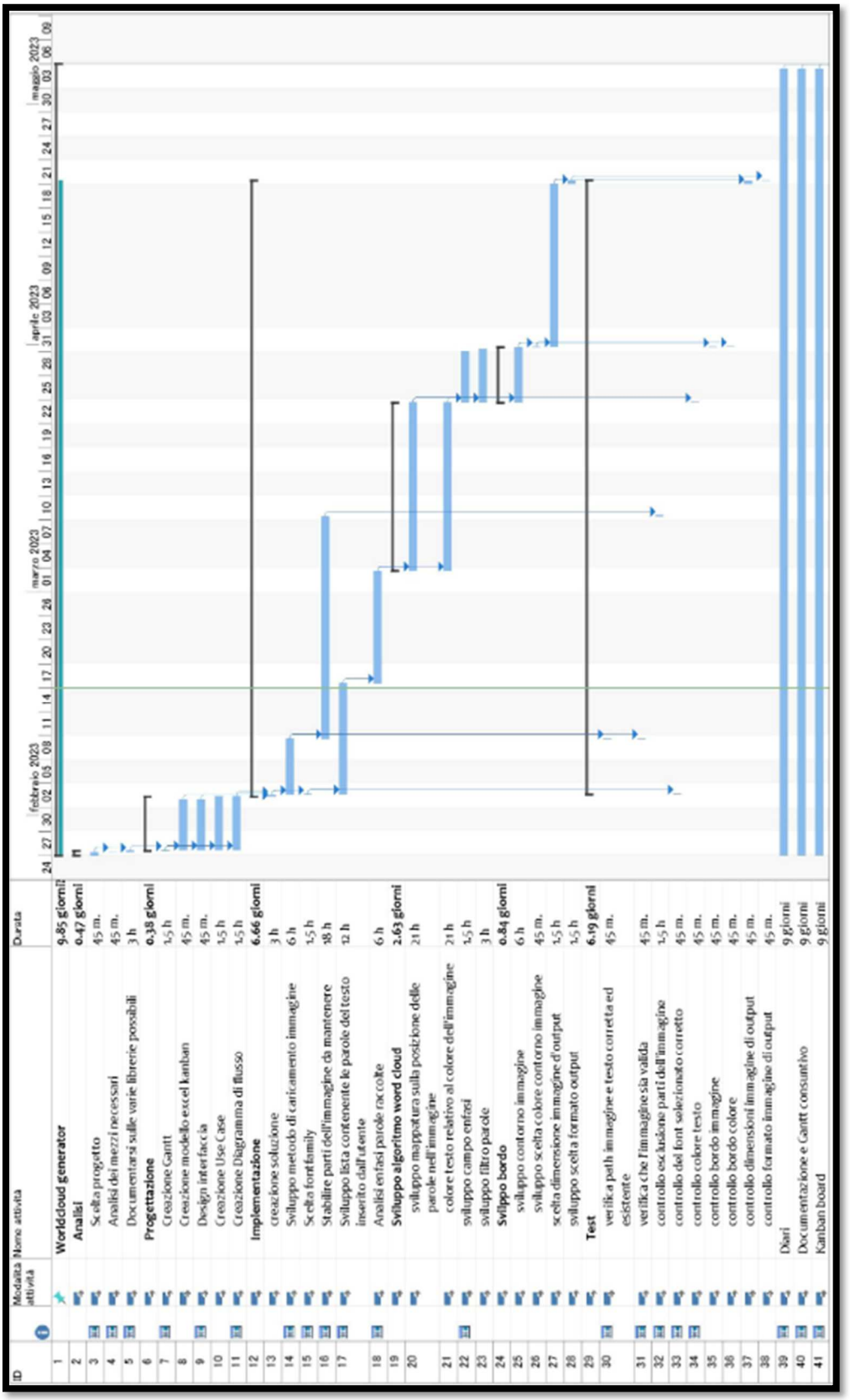

Figura 2 Diagramma di Gantt

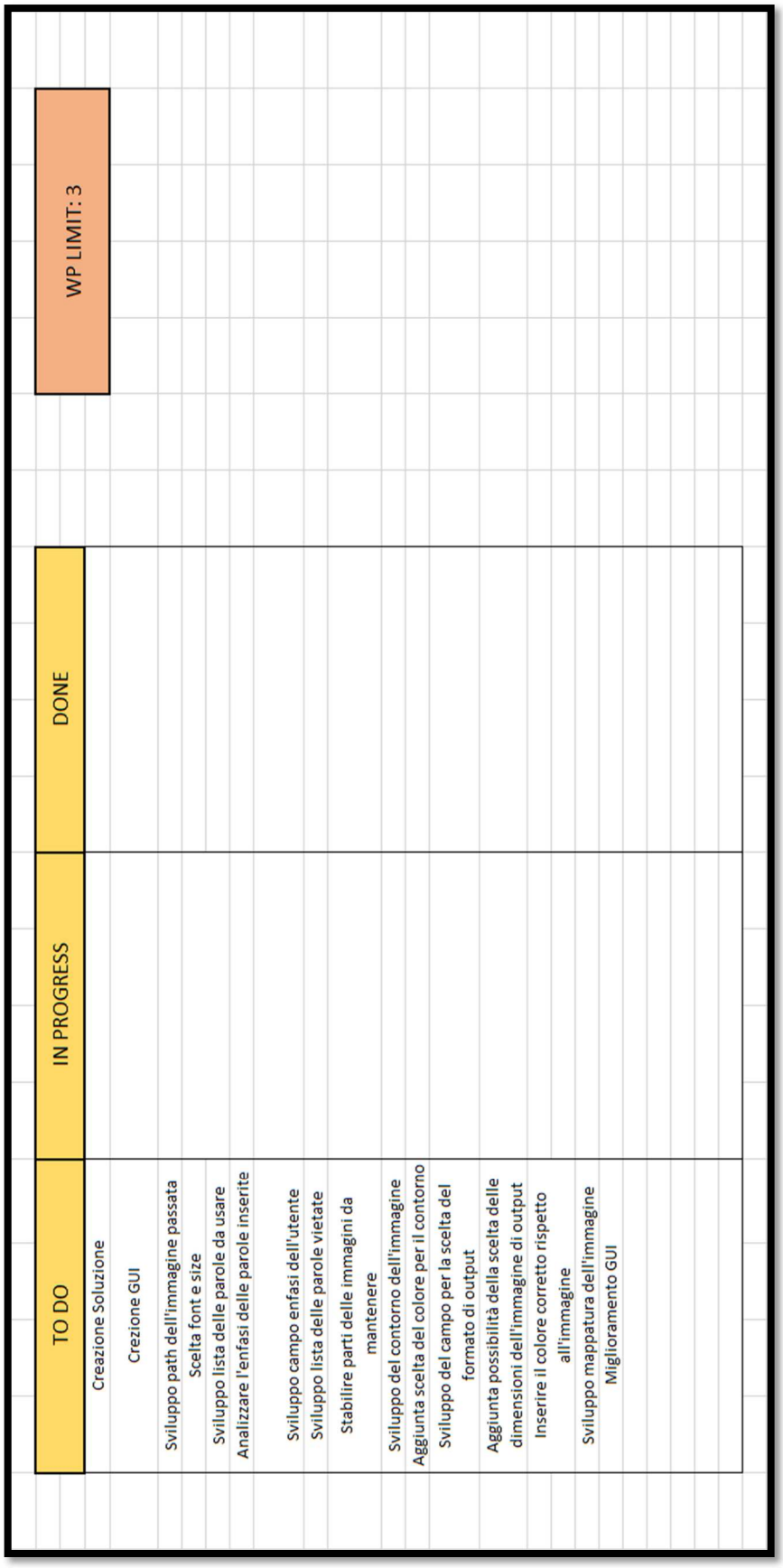

Figura 3 Kanban Board

3.5 Analisi dei mezzi

3.5.1 Software

Python 3.10.6
Visual Studio Code 3.10.9

3.5.2 Librerie

Kivy 2.1.0
Tkinter
OpenCV 4.7.0.72
Matplotlib 3.7.1
Numpy 1.24.2
Pillow 9.5.0
Validators 0.20.0
Wordcloud 1.9.1.1
Filetype 1.2.0
Beautifulsoup 4.11.2

3.5.3 Hardware

Il nostro prodotto è pensato per essere utilizzato da un computer con installato Windows.
Per la realizzazione di questo progetto abbiamo utilizzato tre computer scolastici con installato Windows 10.

HW computer utilizzato:

- CPU: I7-9700
- Scheda video RTX-2060
- 32GB di RAM DDR4

4 Progettazione

4.1 Design delle interfacce

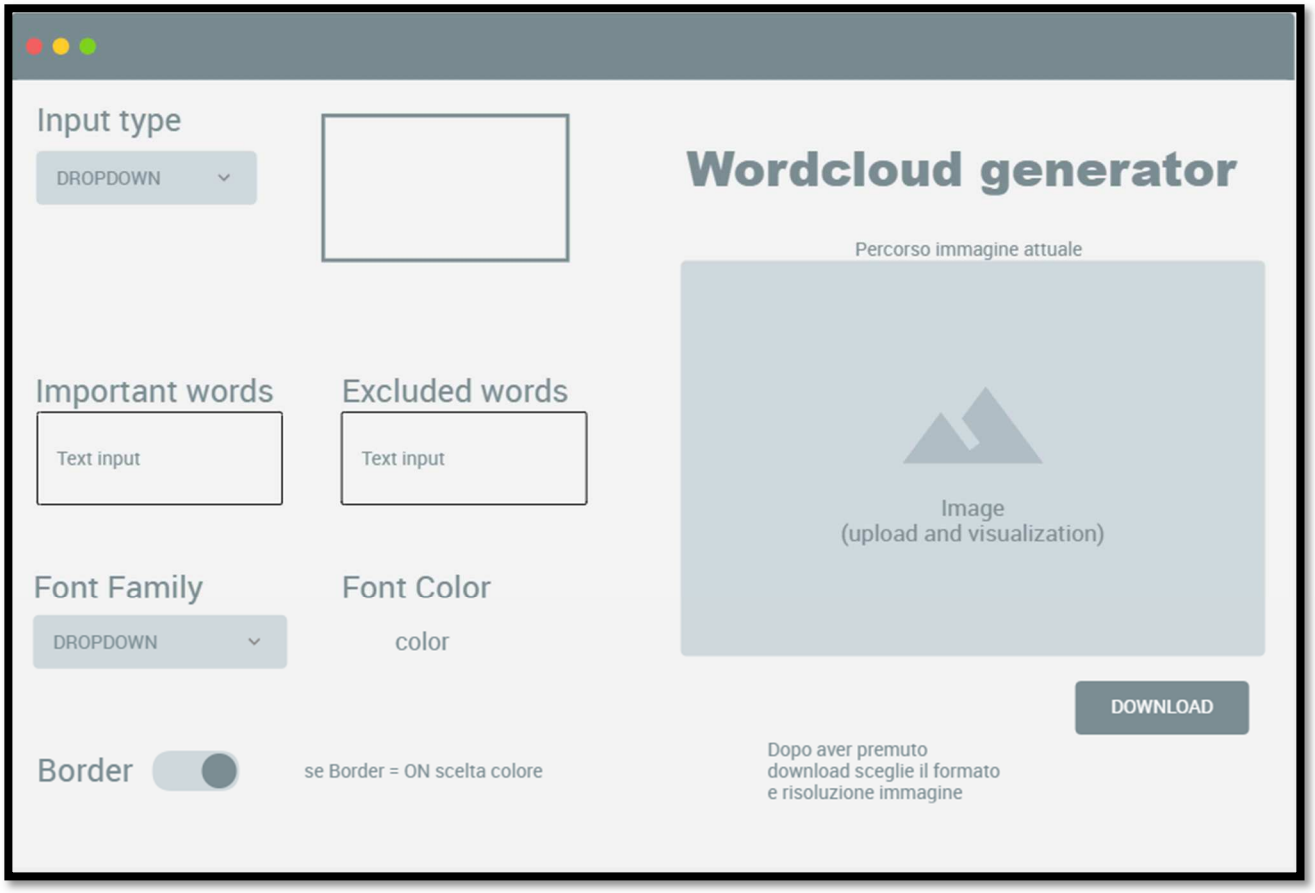

Figura 4 Design iniziale interfaccia

Questa è stato il nostro primo design dell'interfaccia per la nostra applicazione, tuttavia nel corso del progetto abbiamo capito quanto fosse scomoda e disordinata, per questo abbiamo velocemente fatto un altro design e lo abbiamo implementato nella nostra applicazione.

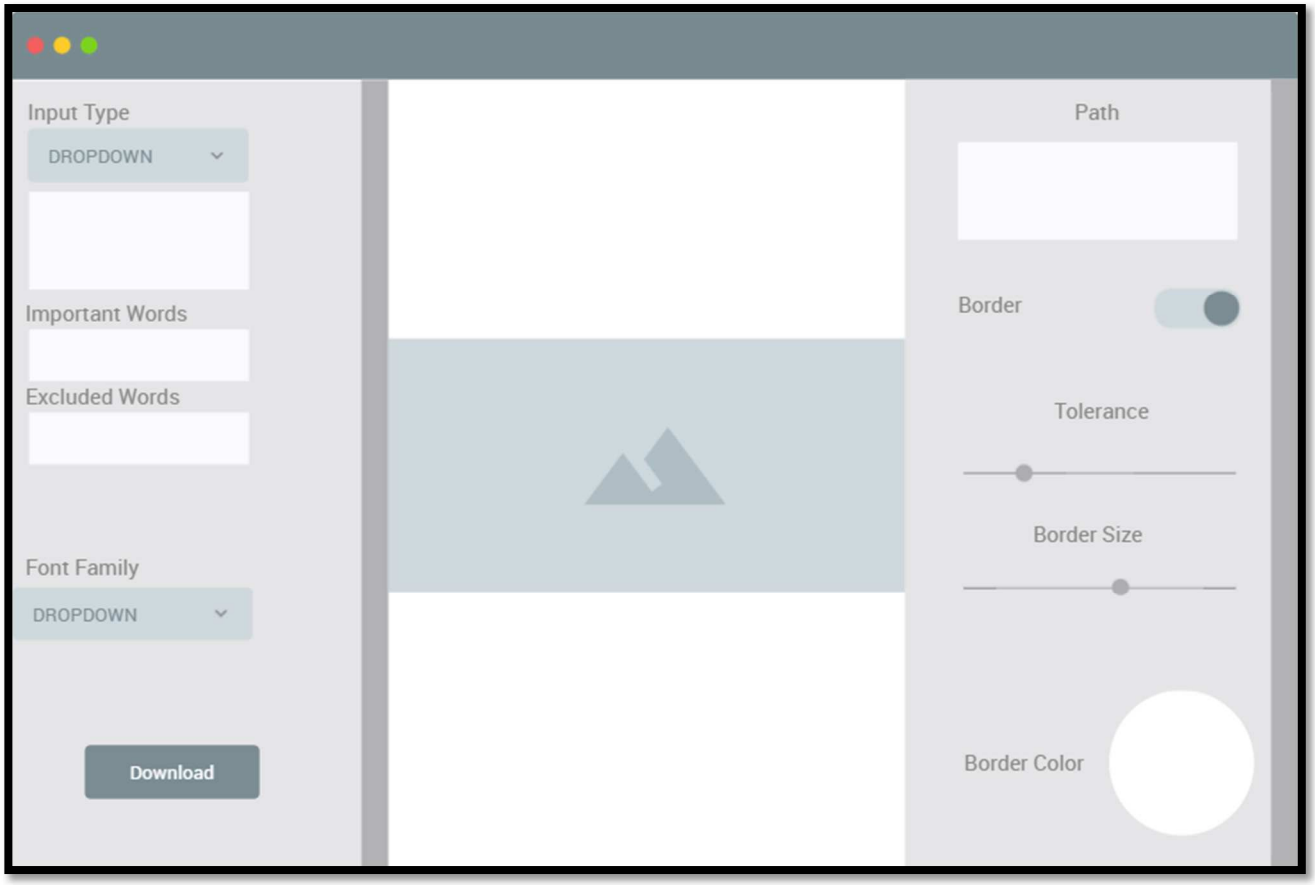

Figura 5 Secondo design per la GUI

Questa interfaccia è strutturata in tre sezioni, a sinistra tutti gli input di tipo testuale e il pulsante per scaricare l'immagine, al centro l'immagine selezionata e a destra gli input da dare per l'immagine.
Le due colonne di input saranno scorrevoli.

4.2 Design procedurale

4.2.1 Diagramma di flusso

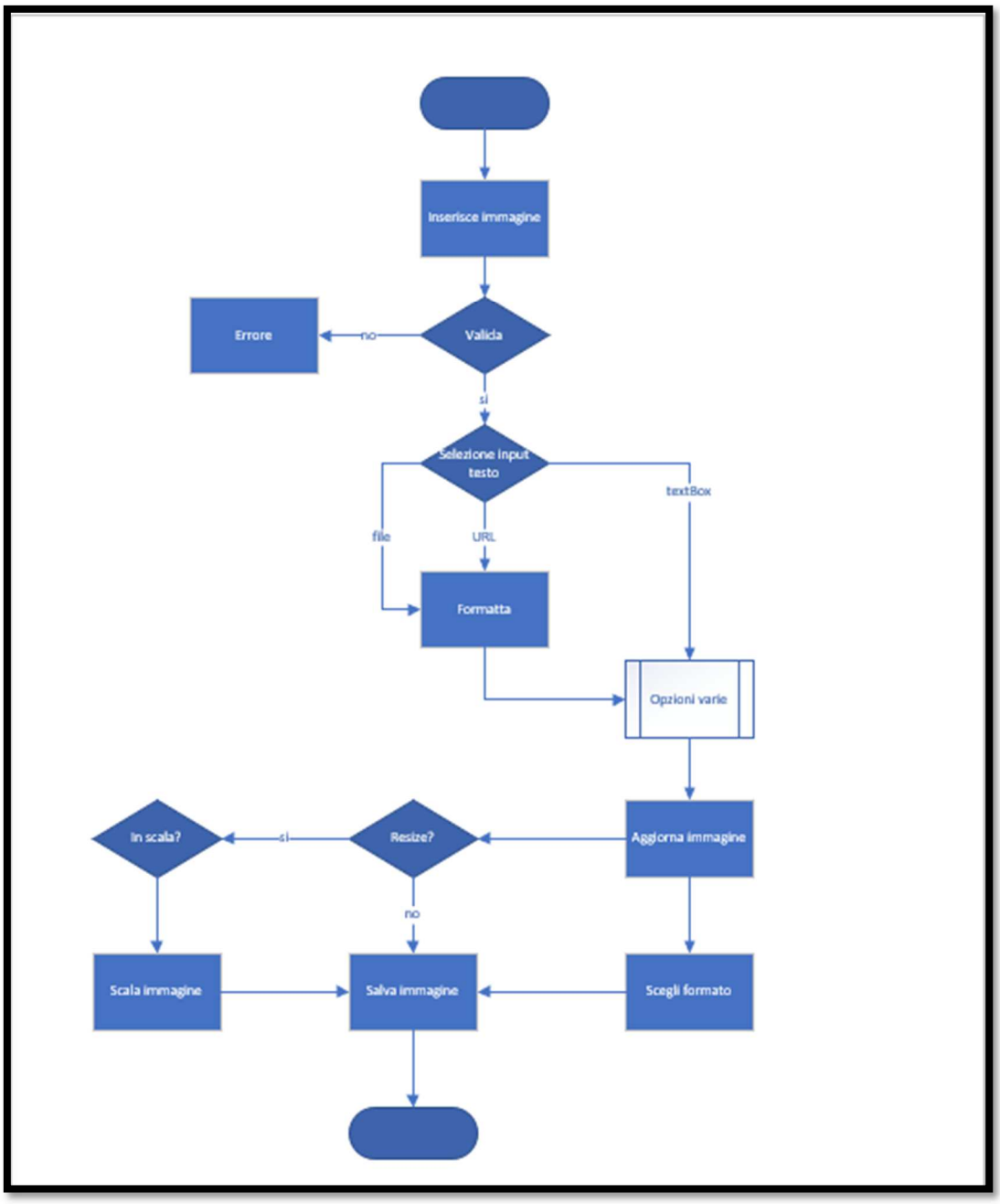

Figura 6 Diagramma di flusso

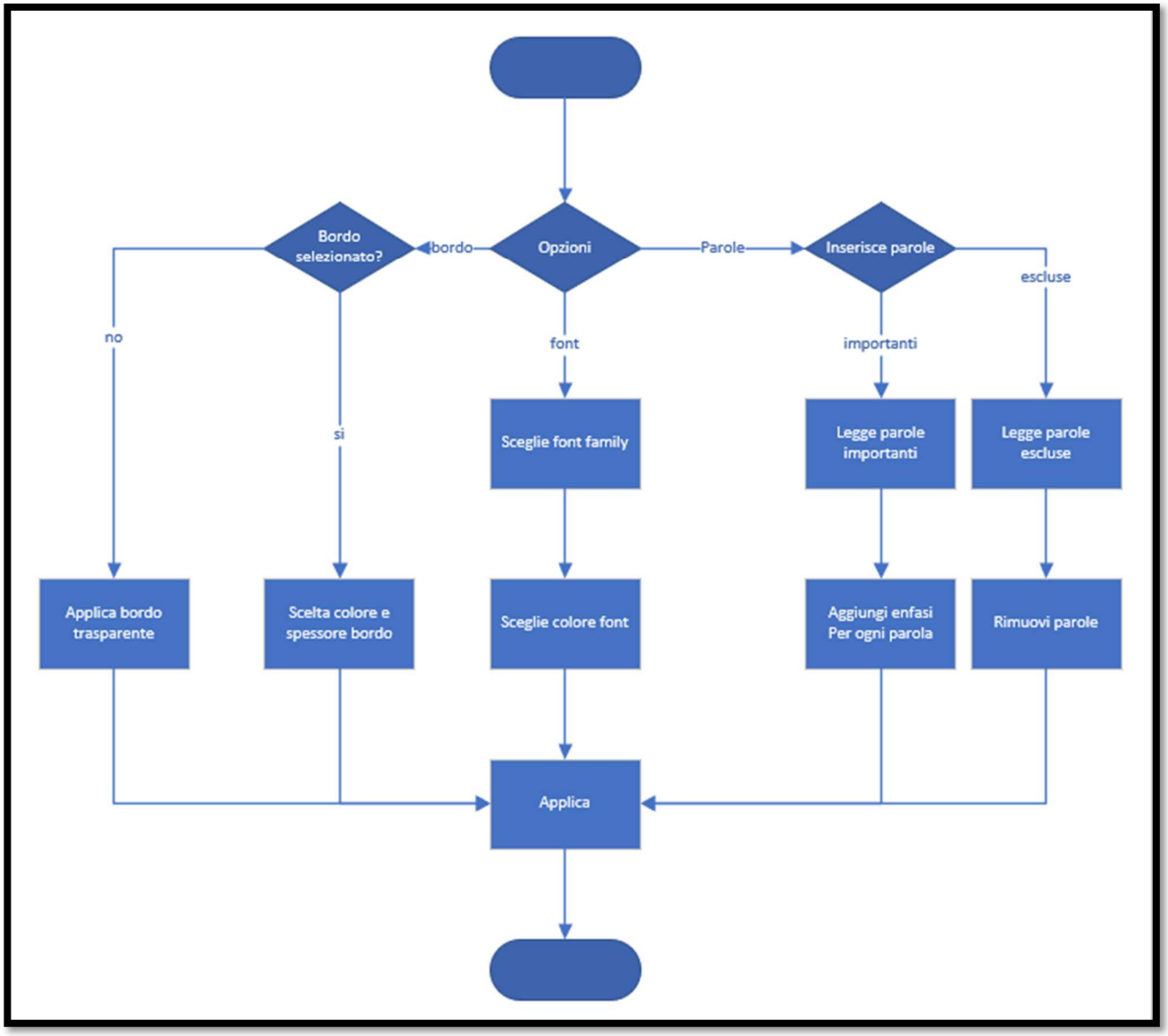

Figura 7 Diagramma opzioni varie

5 Implementazione

5.1 File guigenerator.py

Questo file è il file principale del nostro progetto, infatti si occupa di istanziare tutte le altre classi presenti, le collega e genera l'interfaccia principale.

5.1.1 Generazione GUI

Per generare la nostra interfaccia grafica abbiamo usato kivy, per prima cosa abbiamo creato un widget padre della nostra applicazione che verrà creato all'avvio con all'interno le tre sezioni: TextOptions, ImageSelected e ImageOptions.
Ad ognuna di queste sezioni, al momento della creazione, vengono associati dei widget che contengono il vero contenuto di esse, per esempio nel TextOptions saranno presenti i widget contenenti gli input per il testo.
Questa associazione viene gestita all'interno dei costruttori:

```
imageSelector = ImageSelector(IMAGE_OPTIONS[0])
border = Border(IMAGE_OPTIONS[1])
imageSelection = ImageSelection()

class ImageOptions(BoxLayout):
    def __init__(self, **kwargs):
        super(ImageOptions, self).__init__(**kwargs)
        self.add_widget(imageSelector)
        Logger.info(f'[guigenerator.py] Aggiunta sezione image options con successo')
        self.add_widget(border)
        Logger.info(f'[guigenerator.py] Aggiunti elementi border con successo')
        colorPicker = ColorPicker()
        colorPicker.bind(color=border.on_color)
        self.add_widget(colorPicker)
        Logger.info(f'[guigenerator.py] Aggiunto elemento color picker con successo')
```

In questo modo vengono istanziate le classi, vengono collegate all'interno del programma e vengono aggiunte all'interfaccia.
Per la realizzazione dell'interfaccia abbiamo utilizzato principalmente i "BoxLayout" con all'interno dei "Button", "Label", "TextInput", "Spinner" e "Image".

5.2 Input per il testo

Per l'input del testo abbiamo creato una cartella contenente 5 file, ognuno di essi fa delle operazioni differenti per permettere la memorizzazione del testo e delle sue proprietà.

5.2.1 Metodo di input per le parole

L'utente può inserire le parole che dovranno venir rappresentate nell'immagine in tre modi:

- Tramite testo semplice
- Tramite file
- Tramite URL

Per questa scelta abbiamo implementato uno "Spinner" con queste tre opzioni, ogni volta che l'utente seleziona un tipo di input viene richiamato una funzione che prende come argomento il tipo di input selezionato.
All'interno di essa viene controllato inizialmente quale tipo di input è stato selezionato e in base a questo viene chiamata la funzione apposita per poter memorizzare il testo.

5.2.1.1 Memorizzazione del testo semplice

Questa è l'operazione più semplice, infatti viene leggiamo il testo contenuto nel "TextInput" e lo salviamo all'interno di una variabile.

5.2.1.2 Memorizzazione del testo da file

Per memorizzare il testo da un file abbiamo usato delle funzioni della libreria "os" per aprire il file, leggerlo e salvarlo all'interno di una variabile.

5.2.1.3 Memorizzazione del testo da URL

Per memorizzare il testo da una pagina web, invece, abbiamo installato "BeautifulSoup" che ci permette di aprire la pagina e di salvare tutto il contenuto, evitando i vari tag HTML e Javascript, in modo piuttosto semplice.
Inoltre, essendo in un ambiente scolastico, abbiamo dovuto creare una funzione che aggiunge le variabili d'ambiente per il proxy nel caso in cui non fossero presenti. Questo perché senza queste variabili l'applicazione non riesce ad andare su internet a prendere il contenuto della pagina web.

Dopo aver memorizzato tutte le varie parole esse vengono separate e controllate singolarmente. Questo controllo consiste nel vedere se le parole presenti sono delle parole da escludere o meno, se risultano come parole "vietate" esse verranno semplicemente cancellate.
Per finire bisogna calcolare l'enfasi di queste parole prima di poterle inserire nell'immagine. Per farlo abbiamo creato una nuova funzione che, tramite un array, memorizza la parola e quante volte è presente all'interno del testo. Una volta riempito questo array esso viene ordinato in modo crescente così da avere alla prima posizione la parola più presente.
In seguito, tramite un'altra funzione, il contenuto di questo array viene scritto all'interno di un file base così da permetterci di andare a leggerlo per prendere il contenuto per riempire l'immagine.

5.2.2 Parole escluse e importanti

Oltre alle parole escluse di base abbiamo inserito due input per permettere all'utente di scegliere quali parole escludere e a quali parole dare più importanza.
Questa funzionalità viene gestita da due file appositi che permettono di memorizzare queste parole e di confrontarle con il testo da scrivere nell'immagine, in questo modo possiamo applicare le modifiche necessarie per rendere il testo completo e corretto con le preferenze dell'utente.

5.2.3 Selezione del font

Per la selezione del font abbiamo inizialmente creato un "dictionary" associativo con il nome del font che si collega al percorso del file. In seguito kivy mette a disposizione il "LabelBase" che ci permette di associare il nome del font al percorso del suo stesso file e quindi lo facciamo per ogni font che abbiamo aggiunto.
La scelta del font da parte dell'utente avviene grazie ad uno "Spinner", il suo valore attuale viene letto, salvato e applicato al label con la scritta "Font Family" per far vedere all'utente un esempio dell'applicazione del font ad un testo.

Figura 8 Font family selezionato si applica al label

5.3 Input per l'immagine

Nella parte centrale della nostra interfaccia è presente l'immagine che l'utente vuole modificare mentre nella colonna più a destra sono presenti tutti i vari input necessari alla personalizzazione di essa.

5.3.1 Path per l'immagine iniziale

L'utente può inserire la path per l'immagine da utilizzare e per questo c'è bisogno di controllare se questa path è effettivamente un'immagine o meno.
Noi abbiamo fatto anche in modo che, nello stesso momento in cui l'utente inserisce un nuovo carattere nel "TextInput" per la path, il programma ricerca automaticamente se esiste l'immagine e, quando la trova, la mostra a schermo.
Per fare ciò abbiamo utilizzato la proprietà "on_text" che ci permette di richiamare un metodo ogni volta che viene inserito un nuovo carattere.
Questa funzione, da noi chiamata "visualizer", ne richiama subito una nuova per poter leggere e memorizzare la path inserita dall'utente.
Dopo averla memorizzata avviene un controllo dove verifichiamo che:

1. La path sia valida
2. La path sia un file
3. Il file sia un'immagine

Se questi tre controlli vanno a buon fine allora la path verrà utilizzata per mostrare l'immagine mentre se falliscono verrà mostrata un'immagine di default creata da noi che mostra semplicemente la scritta "Image not found".

5.3.2 Bordo dell'immagine

Una volta selezionata l'immagine l'utente inizialmente deve scegliere la tolleranza con cui vengono definiti i bordi dell'immagine, in seguito può scegliere se applicare o no il bordo all'immagine finale e che colore dargli.
Il bordo verrà comunque visualizzato sull'immagine nell'applicazione per permettere di capire all'utente quali parti può mantenere e quali no.
Per permettere di far vedere in tempo reale all'utente i cambiamenti applicati all'immagine, alla creazione dell'applicazione abbiamo creato un "Clock" dove ogni decimo di secondo viene richiamata la funzione che ricarica l'immagine con le nuove proprietà.

5.3.3 Colore del bordo

Per il colore del bordo abbiamo usato il "ColorPicler" messo a disposizione da kivy che ci permette di scegliere un colore, utilizzando RGB, per il bordo.

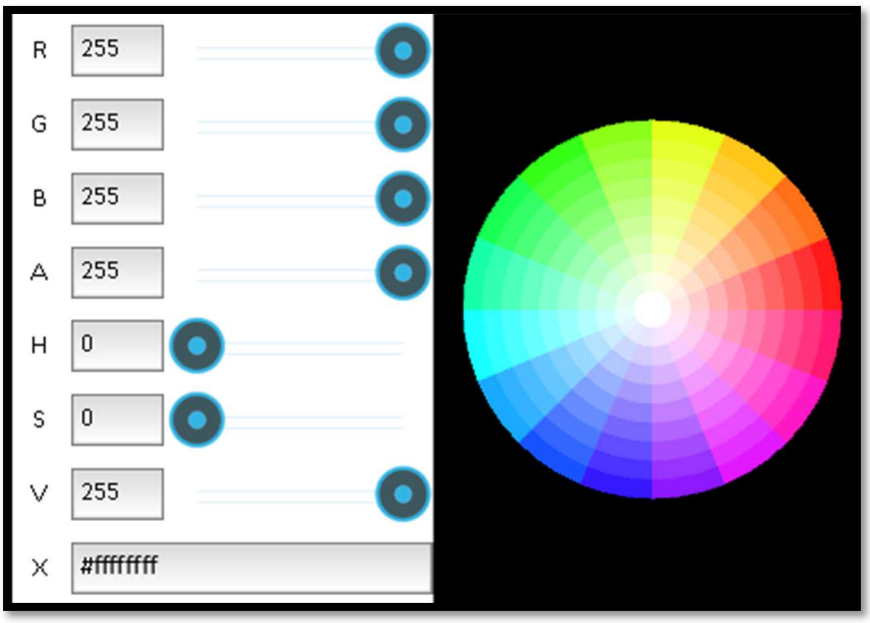

Figura 9 ColorPicker per il colore del bordo

5.4 Algoritmo di floading

Questo è l'algoritmo più importante che abbiamo realizzato all'interno del nostro progetto, è l'algoritmo che permette all'utente di scegliere la parte dell'immagine dove dovranno venir inserite le parole.

La classe "imagepartselector" possiede 4 variabili globali, 3 sono dedicate alle path, la prima che contiene la larghezza del bordo mentre le altre 3 il percorso delle immagini temporanee che creiamo.

```
BORDERWINDOW = 4
TEMP_PATH = "./pictures/imageMod.png"
MASK_PATH = "./pictures/.mask.png"
DELTA_PATH = "./pictures/.delta.png"
```

La funzione "createBorderImage", contenuta nel file delle proprietà del bordo, si occupa di creare l'immagine con il bordo.
Questa immagine verrà utilizzata per il nostro alogritmo di floading.
In seguito abbiamo creato una nuova funzione "updateMask" che legge l'immagine creata in precedenza, crea una maschera in bianco e nero della parte in cui verranno inserite le parole e crea un'immagine delta che servirà in seguito per dare il colore alle parole.

Per fare la scelta della parte dell'immagine abbiamo utilizzato la funzione "on_touch_down" che, ad ogni click dell'utente sull'immagine chiama il metodo "updateMask".
A questo punto il metodo crea un'immagine completamente bianca e memorizza le dimensioni dell'immagine originale, della finestra dell'applicazione, le coordinate dove l'utente a premuto con il mouse e le calcola per controllare se sono all'interno dell'immagine o meno.
Se il controllo fallisce non verrà fatto nulla, altrimenti inizierà l'algoritmo.
Come prima funzione viene richiamata "highlightArea", una funzione iterativa che sostituisce quella ricorsiva classica di floading visto che Python ha una limitazione per quanto riguarda il numero delle iterazioni.

Essa si occupa di evidenziare la zona desiderata dall'utente con l'aiuto di altre 3 funzioni dall'intuitivo scopo: "isHighlighedPixel", "isBorder" e "isInArea"
Queste si occupano di controllare se:

1. Un determinato pixel dell'immagine è già stato colorato
2. Una coordinata si trova nel bordo della figura tracciata
3. Una coordinata si trova nello spazio dell'immagine

Grazie a questi controlli l'algoritmo riesce a disegnare in nero, all'interno della maschera, le parti da mantenere e a disegnare, con lo stesso colore dell'immagine originale, le parti da mantenere sull'immagine delta.

5.5 Wordcloud

Per l'algoritmo per la generazione delle parole sull'immagine abbiamo importato la libreria Wordcloud che lo fa per noi. Questa scelta è dovuta alla mancanza di tempo per provare ad implementare un algoritmo creato da zero da noi.
Per questa operazione abbiamo creato un nuovo file "generatecloud.py", questo file contiene semplicemente una funzione che prende tutte le proprietà inserite dall'utente in precedenza e richiama la classe WordCloud che genererà l'immagine completa.

In particolare inizialmente viene letto il file di testo contenente le parole inserite dall'utente, in seguito viene memorizzata tramite "numpy" l'immagine delta per poi poter creare la mappa dei colori dell'immagine e infine viene memorizzato il colore del bordo da utilizzare.

Dopo tutte queste operazioni viene richiamao "WordCloud" per generare l'immagine.

```
    wc = WordCloud(
                    width=size[1],
                    height=size[0],
                    font_path=getFont(),
                    stopwords=STOPWORDS,
                    mask=python_mask,
                    background_color="white",
                    contour_color=borderColor,
                    contour_width=getBorderSize()
                ).generate(text)
```

Come si può vedere da questa parte di codice per generare l'immagine abbiamo passato come argomenti la larghezza dell'immagine e la sua altezza, in seguito impostiamo il font prendendolo tramite una funzione importata, poi, oltre alle parole già scartate dal nostro file iniziale, abbiamo scelto di vietare anche le parole (in inglese) che questa libreria offre di default.
Come opzioni per l'immagine abbiamo inserito la maschera creata in precedenza, il colore del background dell'immagine finale, il colore del bordo e la sua grandezza.
La funzione "getBorderSize" è collegata alla scelta dell'utente per la presenza o meno del bordo, se l'utente non vuole il bordo questa funzione ritornerà 0 e quindi non verrà disegnato alcun bordo sull'immagine.
Infine tramite la funzione "generate" della libreria generiamo l'immagine con il testo inserito dall'utente.

Dopo aver generato l'immagine verrà salvata nel percorso dell'immagine che viene visualizzata a schermo, cosicché l'utente potrà visualizzare in tempo reale il risultato finale.

5.6 Download dell'immagine

Per scaricare l'immagine abbiamo aggiunto un pulsante download nella schermata principale che, tramite tkinter, apre una finestra nell'esplora risorse di Windows dove si può salvare l'immagine all'interno di una cartella specifica o semplicemente nella cartella "Download".

Sono disponibili tre formati:

- .png
- .jpg o .jpeg
- .webp

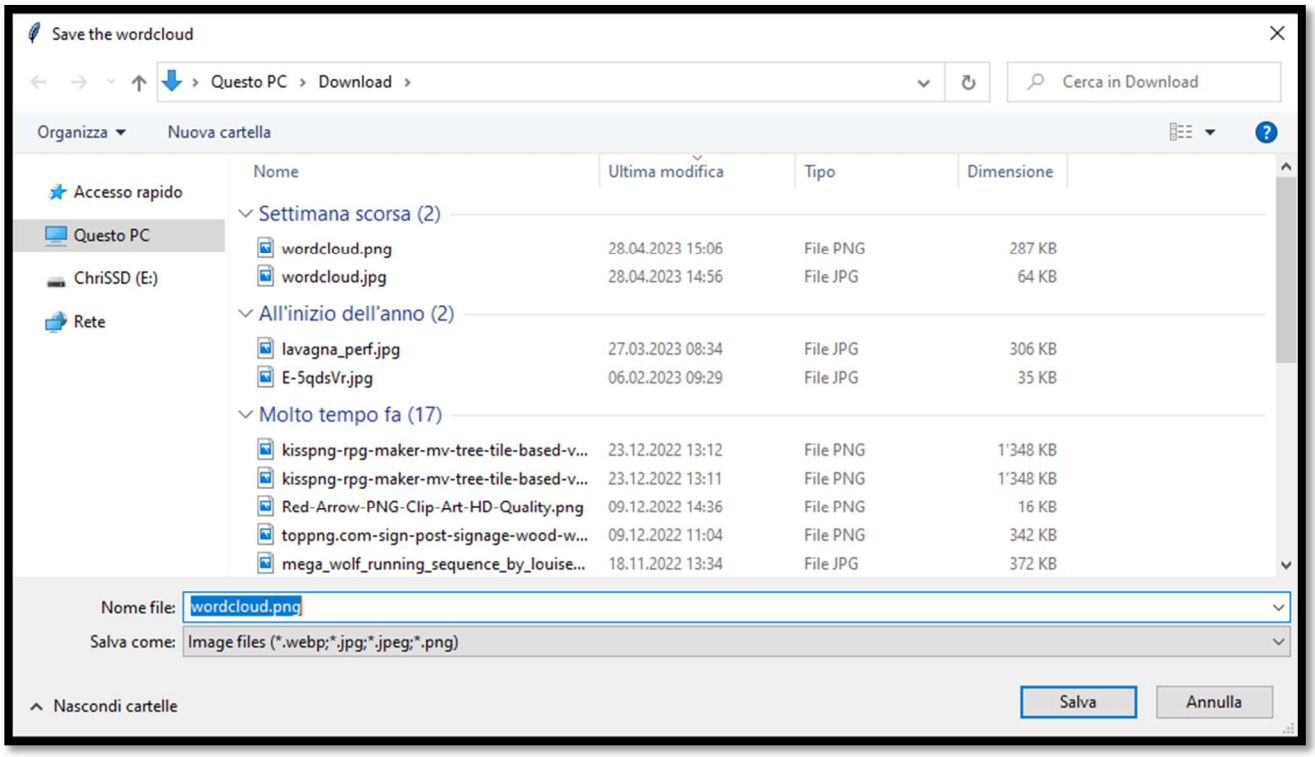

Figura 10 Finestra di download per l'immagine

5.7 File requirements.txt

Per ultima cosa, per garantire il funzionamento dell'applicativo con la versione 3.10.6 di Python, abbiamo creato il file "requirements.txt".
Questo file contiene tutte le varie librerie che abbiamo installato con la propria versione e.
Per creare questo file serve questo comando:

```
pip freeze > requirements.txt
```

Esso prende tutte le librerie che trova nei file del progetto e le scrive all'interno del file.

6 Test

6.1 Protocollo di test

Test Case **Riferimento**	TC-001 REQ-001	**Nome**	Test upload dell'immagine
Descrizione	Caricare un'immagine sull'applicazione		
Prerequisiti	Poter avviare l'applicazione		
Procedura	1. Avviare l'applicazione 2. Trascinare e rilasciare nello spazio apposito l'immagine 3. Controllare se l'immagine viene visualizzata a schermo		
Risultati attesi	L'immagine viene visualizzata a schermo		

Test Case **Riferimento**	TC-002 REQ-002	**Nome**	Test delle parti d'immagine da non cambiare
Descrizione	Testare che l'utente possa scegliere le parti da non cambiare all'interno dell'immagine		
Prerequisiti	Aver verificato che funzioni l'inserimento dell'immagine		
Procedura	1. Avviare l'applicazione 2. Caricare un'immagine 3. Premere sulla parte che si vuole mantenere dell'immagine 4. Far generare l'immagine composta dalle parole 5. Verificare che le parole vengano scritte solo nella parte selezionata in precedenza		
Risultati attesi	L'utente può scegliere le parti che vengono modificate e l'immagine generata comprenderà solamente quella parte selezionata		

Test Case **Riferimento**	TC-003 REQ-003	**Nome**	Test delle parole inserite tramite input di testo
Descrizione	Testare che funzioni il metodo di inserimento manuale da parte dell'utente delle parole		
Prerequisiti	Aver verificato che funzioni l'inserimento delle immagini. Aver implementato il drop down di scelta del metodo di input delle parole		
Procedura	1. Scegliere l'opzione di input tramite testo inserito dall'utente 2. Scrivere all'interno del text box apposito 3. Controllare che l'immagine contenga tutte le parole 4. Scrivere anche parole bloccate e controllare che non vengano mostrate		
Risultati attesi	L'utente inserisce un testo all'interno dello spazio apposito e nell'immagine compaiono tutte le parole inserite tranne quelle bloccate		

Test Case **Riferimento**	TC-004 REQ-003	**Nome**	Test delle parole inserite tramite file di testo
Descrizione	Testare che funzioni il metodo di inserimento delle parole tramite un file di testo		
Prerequisiti	Aver verificato che funzioni l'inserimento delle immagini. Aver implementato il drop down di scelta del metodo di input delle parole		
Procedura	1. Scegliere l'opzione di input con l'upload del file di testo 2. Rilasciare nello spazio apposito un file di testo contenente delle parole 3. Controllare che l'immagine contenga tutte le parole presenti all'interno del file tranne quelle bloccate		
Risultati attesi	L'utente inserisce un file di testo all'interno dello spazio apposito e nell'immagine compaiono tutte le parole inserite tranne quelle bloccate		

Test Case **Riferimento**	TC-005 REQ-003	**Nome**	Test delle parole inserite tramite URL di una pagina da scaricare
Descrizione	Testare che funzioni il metodo di inserimento delle parole tramite un link web		
Prerequisiti	Aver verificato che funzioni l'inserimento delle immagini. Aver implementato il drop down di scelta del metodo di input delle parole		
Procedura	1. Scegliere l'opzione di input tramite URL di una pagina 2. Inserire un URL all'interno dello spazio apposito 3. Controllare che l'immagine contenga tutte le parole tranne i vari tag HTML e le parole bloccate		
Risultati attesi	L'utente inserisce un URL all'interno dello spazio apposito e nell'immagine compaiono tutte le parole presenti in quella pagina tranne quelle bloccate e i tag HTML		

Test Case **Riferimento**	TC-006 REQ-004	**Nome**	Test del font family corretto
Descrizione	Testare che una volta selezionato un tipo di font family esso venga effettivamente utilizzato		
Prerequisiti	Aver verificato che funzioni l'inserimento delle immagini. Aver verificato il funzionamento dei vari input delle parole		
Procedura	1. Inserire un'immagine 2. Inserire del testo 3. Selezionare un font family 4. Controllare che sull'immagine venga effettivamente utilizzato il font family selezionato		
Risultati attesi	L'utente sceglie il font e quest'ultimo viene effettivamente utilizzato nell'immagine creata		

Test Case Riferimento	TC-007 REQ-005	**Nome**	Test del colore delle parole
Descrizione	Testare che il colore delle parole rispecchi quello dell'immagine iniziale		
Prerequisiti	Aver verificato che funzioni l'inserimento delle immagini. Aver verificato che le parole vengano scritte nell'immagine		
Procedura	1. Inserire delle parole nell'input di testo 2. Controllare con l'immagine iniziale che i colori siano stati mantenuti		
Risultati attesi	I colori delle parole rispecchiano i colori iniziali dell'immagine		

Test Case Riferimento	TC-008 REQ-006	**Nome**	Test delle parole bloccate
Descrizione	Testare che le parole bloccate non vengano mostrate sull'immagine		
Prerequisiti	Aver verificato che le parole vengano visualizzate nell'immagine		
Procedura	1. Inserire nell'input di testo delle parole bloccate 2. Controllare che non vengano mostrate sull'immagine		
Risultati attesi	Dopo aver inserito le parole bloccate esse non verranno mostrate sull'immagine		

Test Case Riferimento	TC-009 REQ-007	**Nome**	Test delle parole con importanza maggiore
Descrizione	Testare che le parole con importanza maggiore definite dall'utente vengano effettivamente mostrate più grandi delle altre		
Prerequisiti	Aver verificato che le parole vengano visualizzate nell'immagine		
Procedura	1. Inserire nel campo di testo delle parole enfatizzate del testo 2. Controllare che nell'immagine la parola inserita all'interno del campo di testo venga visualizzata più in grande		
Risultati attesi	La parola inserita verrà mostrata con un font di dimensioni maggiori rispetto alle altre parole		

Test Case **Riferimento**	TC-010 REQ-008	**Nome**	Test del bordo
Descrizione	Testare che quando viene abilitato il bordo esso appaia in modo corretto		
Prerequisiti	Aver verificato che le parole vengano visualizzate nell'immagine		
Procedura	1. Abilitare il bordo dell'immagine 2. Controllare che sull'immagine sia apparso un bordo nero 3. Scegliere un colore per il bordo 4. Controllare che il colore venga applicato in modo corretto 5. Impostare uno spessore al bordo 6. Controllare che lo spessore venga applicato correttamente all'immagine		
Risultati attesi	Una volta abilitato il bordo all'immagine esso verrà automaticamente mostrato in modo corretto, quando si cambierà il colore, esso si aggiornerà automaticamente al colore selezionato e infine quando si imposterà uno spessore il bordo si aggiornerà automaticamente ingrandendosi o rimpicciolendosi		

Test Case **Riferimento**	TC-011 REQ-009	**Nome**	Test dell'aggiornamento in tempo reale dell'immagine
Descrizione	Testare che l'immagine si aggiorni automaticamente ad ogni cambiamento effettuato dall'utente		
Prerequisiti	Aver verificato che tutti gli input a disposizione dell'utente funzionino		
Procedura	1. Caricare una nuova immagine 2. Inserire del testo e controllare che ad ogni parola l'aggiornamento dell'immagine 3. Cambiare font e controllare che venga cambiato in tempo reale anche nell'immagine 4. Abilitare il bordo e controllare che si aggiorni automaticamente mostrandolo di colore nero 5. Cambiare il colore al bordo e controllare che si aggiorni automaticamente usando il colore selezionato		
Risultati attesi	L'immagine, qualunque cosa faccia l'utente, si aggiorna sempre in tempo reale		

Test Case Riferimento	TC-012 REQ-010	**Nome**	Test della risoluzione a scelta dell'immagine
Descrizione	Testare che l'utente possa scegliere una risoluzione a suo piacimento e che questa venga effettivamente applicata		
Prerequisiti	Aver verificato che l'applicazione funziona correttamente		
Procedura	1. Caricare una nuova immagine 2. Inserire del testo per creare una nuova immagine 3. Premere il pulsante download 4. Scegliere vari tipi di risoluzione e scaricare l'immagine 5. Controllare che ogni immagine rispecchi la risoluzione data		
Risultati attesi	L'immagine sarà sempre della risoluzione data dall'utente al momento del download		

Test Case Riferimento	TC-013 REQ-011	**Nome**	Test del formato dell'immagine una volta scaricata
Descrizione	Testare che l'immagine, una volta scaricata, sia del formato scelto precedentemente dall'utente		
Prerequisiti	Aver verificato che l'applicazione funzioni correttamente e in modo completo		
Procedura	1. Caricare una nuova immagine 2. Inserire del testo per creare una nuova immagine 3. Premere il tasto download 4. Scegliere una risoluzione base dell'immagine 5. Scegliere il formato 6. Controllare che una volta scaricata sia effettivamente di quel formato		
Risultati attesi	L'immagine, una volta scaricata, sarà del formato scelto al momento del download da parte dell'utente		

6.2 Risultati test

Test Case	Risultato ottenuto	Stato
TC-001	Il nostro programma non permette di trascinare e rilasciare l'immagine per poterla caricare. Tuttavia l'immagine può essere caricata scrivendo il suo percorso, in questo modo il programma funziona.	Non Passato
TC-002		Passato
TC-003		Passato
TC-004		Passato
TC-005		Passato
TC-006		Passato
TC-007		Passato
TC-008		Passato
TC-009		Passato
TC-010		Passato
TC-011		Passato
TC-012	L'immagine generata dall'applicazione ha una dimensione fissa generata dalla libreria wordcloud, l'utente non può sceglierla.	Non Passato
TC-013		Passato

6.3 Mancanze/limitazioni conosciute

Non abbiamo implementato il trascinamento e rilascio delle immagini sull'applicazione, questo perché abbiamo provato inizialmente a farlo ma non avevamo trovato nessun modo per poterlo implementare con kivy; quindi, abbiamo deciso di lasciarlo da parte e riprenderlo una volta finito il progetto se avanzava ancora tempo.

Non siamo riusciti a cambiare la risoluzione dell'immagine generata con le parole, quindi l'utente avrà sempre una dimensione standard decisa dalla libreria wordcloud.

L'utente può scegliere una sola parte dell'immagine da mantenere visto che appena si preme con il cursore in una determinata area, l'immagine si genera istantaneamente e viene mostrata all'utente. L'ideale sarebbe stato avere un'applicazione dove l'utente potesse scegliere più parti differenti da mantenere.

Il testo può essere composto solamente da caratteri UTF-8, non è possibile inserire caratteri, per esempio, arabi o cinesi.

Se l'immagine è troppo piccola l'algoritmo non riesce a inserire le parole all'interno.

7 Consuntivo

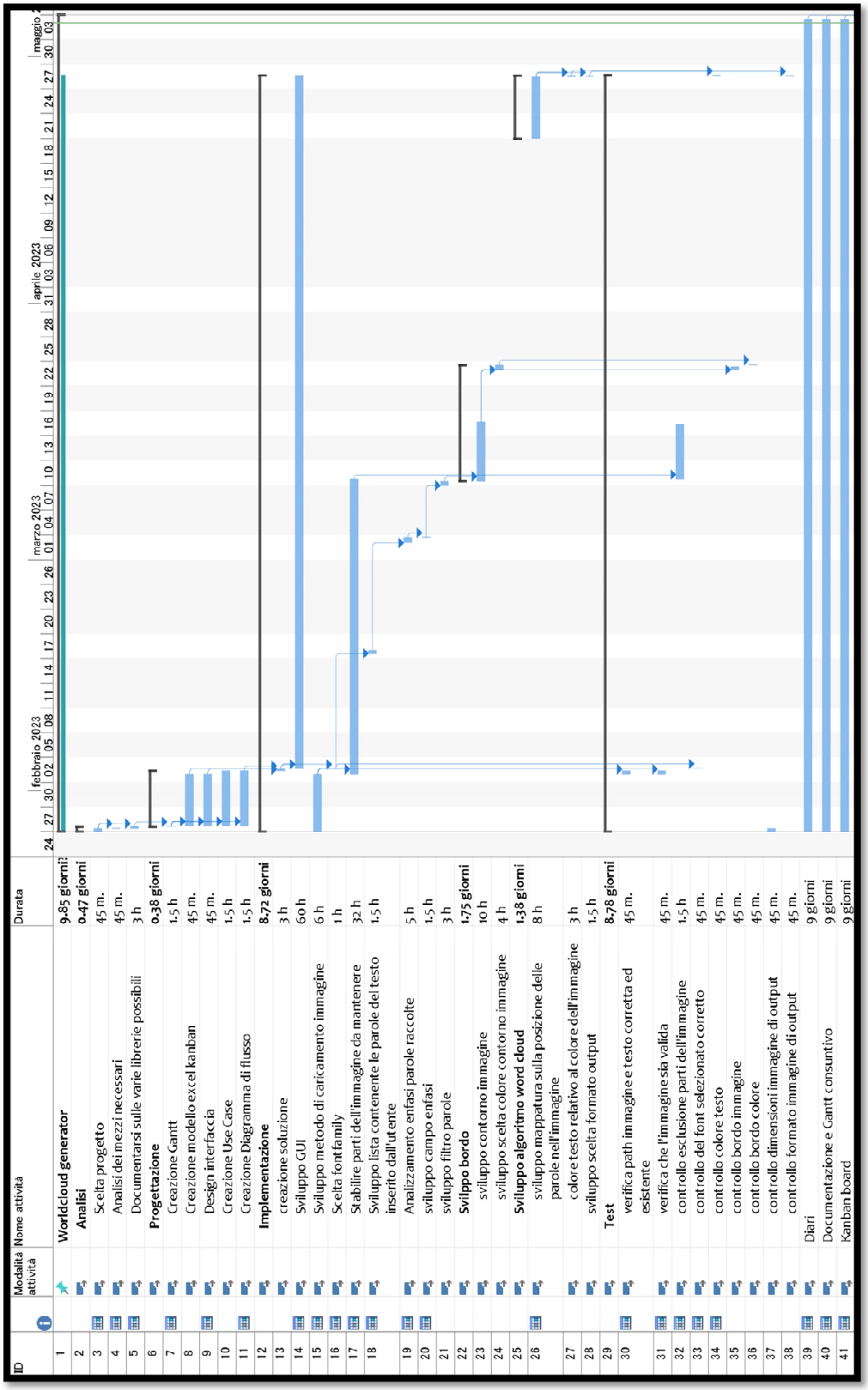

Figura 11 Gantt consuntivo

8 Conclusioni

Siamo molto contenti del risultato ottenuto, non pensiamo che esso possa cambiare il mondo, ma nemmeno che sia totalmente inutile. Anche se già presente nel web siamo convinti che avere una versione applicativa di un Wordcloud generator sia molto comodo, inoltre pensiamo che la nostra soluzione possa essere sviluppata ulteriormente in modo piuttosto semplice.

8.1 Sviluppi futuri

Abbiamo ipotizzato molti possibili sviluppi futuri, tra questi abbiamo l'idea di aumentare la gamma delle immagini sulla quale è possibile applicare l'effetto wordcloud, che può sembrare una cosa banale, ma richiede diverso tempo per trovare o sviluppare un ricercatore di bordi più avanzato.

Durante l'implementazione avevamo anche pensato all'inserimento delle parole in obliquo, anche di permettere all'utente di decidere il colore delle parole oppure di scrivere le parole all'interno di altre.

Inoltre uno dei primi sviluppi che sarebbero da applicare è sicuramente quello di aggiungere la possibilità di fare il Drag and drop per il caricamento dell'immagine e del file di testo.

8.2 Considerazioni personali

Alessandro:
In questo progetto sono riuscito a raggiungere gli obbiettivi che mi ero prefissato: quello di imparare ad usare Python, completare il core del progetto e lavorare in team.
Posso dire che la parte più difficile per me è stata quella di lavorare con l'interfaccia usando kivy, infatti ringrazio Christian che spesso è venuto in mio aiuto per la parte grafica, mentre per la parte di programmazione e di requisiti mi sono fatto aiutare da Edoardo. Posso dire di essere molto soddisfatto del nostro lavoro di squadra, soprattutto di Edoardo, perché pensavo di avere alcuni problemi nel dialogare con lui, essendo che spesso parla prima che qualcuno finisca la frase, ma per tutta la durata di questo progetto non è quasi mai capitato.
Durante la realizzazione di questo progetto mi sono reso utile sia creando che supportando i miei compagni e ho conquistato la loro fiducia.

Christian:
Durante questo progetto ho imparato molte cose e la più importante è sicuramente quella di aver imparato a lavorare in team. Nella realizzazione della nostra applicazione ho capito quanto fosse importante suddividerci il lavoro, parlarci e aiutarci a vicenda. Non nascondo che ci siano state delle difficoltà nel metterci d'accordo sul da farsi ma penso che siamo sempre riusciti a risolvere queste situazioni con professionalità e rimanendo uniti.
Inoltre ho imparato in modo autonomo ad utilizzare Python, con anche l'utilizzo di librerie, cosa che ritenevo particolarmente importante da un po' di anni e sono veramente contento di aver avuto finalmente l'opportunità di utilizzarlo in un progetto così importante.
Per concludere posso dire di essere molto soddisfatto del lavoro svolto insieme ai miei due compagni e sono sicuro che questa esperienza mi tornerà utile in futuro in un ambiente professionale.

Edoardo:
La risoluzione di questo progetto mi ha insegnato diverse cose, tra le quali sicuramente è l'utilizzo del linguaggio Python. L'ho trovato interessante non avendolo mai utilizzato prima, anche se ha portato alcune considerazioni negative dal mio punto di vista.
Ma tutto sommato sono contento della mia esperienza con esso.
Il progetto in sé essendo il primo di gruppo mi ha tranquillizzato molto, infatti vedendo i requisiti con la certezza di dover fare tutto da solo è spaventosa, soprattutto considerano la sua grandezza.

9 Bibliografia

9.1 Sitografia

- https://www.fluidui.com/editor/live/, Fluid 27.01.2023
- https://medium.com/, Medium 03.02.2023
- https://kivy.org/doc/stable/api-kivy.uix.behaviors.drag.html, Kivy 03.02.2023
- https://stackoverflow.com/questions/37573848/kivy-drag-n-drop-get-file-path, Stackoverflow 03.02.2023
- https://stackoverflow.com/questions/56609278/how-to-open-new-window-on-button-press-event-in-kivy, Stackoverflow 10.02.2023
- https://kivy.org/doc/stable/api-kivy.uix.screenmanager.html, Kivy 10.02.2023
- https://kivy.org/doc/stable/_modules/kivy/uix/spinner.html, Kivy 10.02.2023
- https://www.geeksforgeeks.org/how-to-add-custom-fonts-in-kivy-python, Geeksforgeeks 10.02.2023
- https://www.1001freefonts.com/new-fonts-3.php, FreeFonts, 10.02.23
- https://stackoverflow.com/questions/15138614/how-can-i-read-the-contents-of-an-url-with-python, Stackoverflow 10.02.23
- https://www.geeksforgeeks.org/remove-all-style-scripts-and-html-tags-using-beautifulsoup, Geeksforgeeks 10.02.2023
- https://kivy.org/doc/stable/api-kivy.metrics.html, Kivy 17.02.2023
- https://pypi.org/project/opencv-python, Pypi 17.02.2023
- https://www.tutorialspoint.com/how-to-compute-the-area-and-perimeter-of-an-image-contour-using-opencv-python, Tutorialpoint 17.02.2023
- https://kivy.org/doc/stable/api-kivy.uix.slider.html, Kivy 03.03.2023
- https://kivy.org/doc/stable/api-kivy.uix.image.html, Kivy 03.03.2023
- https://docs.opencv.org/4.x/d3/df2/tutorial_py_basic_ops.html, OpenCV 03.03.2023
- https://stackoverflow.com/questions/889333/how-to-check-if-a-file-is-a-valid-image-file, Stackoverflow 03.03.2023
- https://stackoverflow.com/questions/3845362/how-can-i-check-if-a-key-exists-in-a-dictionary, Stackoverflow 10.03.2023
- https://thispointer.com/python-get-first-value-in-a-dictionary/, Thispointer 10.03.2023
- https://www.askpython.com/python/array/reverse-an-array-in-python, Akspython 10.03.2023
- https://www.digitalocean.com/community/tutorials/python-add-to-list, Digitalocean 10.03.2023
- https://www.geeksforgeeks.org/adding-text-on-image-using-python-pil/, Geeksforgeeks 10.03.2023
- https://onelinerhub.com/python-pillow/how-to-rotate-text, Onlinerhub, 10.03.2023
- https://github.com/kivy/kivy/issues/3292, Github 24.03.2023
- https://kivy.org/doc/stable/api-kivy.uix.colorpicker.html, Kivy 24.03.2023

- https://stackoverflow.com/questions/64807163/importerror-cannot-import-name-from-partially-initialized-module-m, Stackoverflow 24.03.2023
- https://www.saltycrane.com/blog/2011/11/how-get-username-home-directory-and-hostname-python/https://www.saltycrane.com/blog/2011/11/how-get-username-home-directory-and-hostname-python/, Saltycrane 31.03.2023
- https://www.makeuseof.com/python-convert-image-file-format/#:~:text=You%20can%20simply%20convert%20the,image%20in%20the%20identified%20format, Makeuseof 31.03.2023
- https://docs.python.org/3/library/queue.html#queue-objects, Python 31.03.2023
- https://kivy.org/doc/stable/guide/events.html, Kivy 31.03.2023
- https://www.youtube.com/watch?v=vRbSnlRyJNQ, Youtube 21.04.2023
- https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.savefig.html, Matplotlib 21.04.2023
- https://kivy.org/doc/stable/api-kivy.core.window.html, Kivy 21.04.2023
- https://stackoverflow.com/questions/8218608/scipy-savefig-without-frames-axes-only-content, Stackoverflow 21.04.2023

10 Glossario

Termine	Significato
Agile	Modalità di svolgimento di un progetto dove il progetto viene diviso in compiti (task) e ogni compito viene portato avanti a piccoli incrementi chiamati sprint.
Beautifulsoup	Libreria di Python da noi utilizzata per leggere il testo dalle pagine web.
Filetype	Libreria di Python utilizzata per verificare i file passati dall'utente.
Floading	Si tratta di un algoritmo che permette, tramite un punto di input, di colorare una parte di immagine contenuta in un bordo.
GUI	Graphical User Interface, l'interfaccia che si presenta all'utente una volta aperta l'applicazione.
Kanban Board	Lavagna contenente tre colonne: "TO DO", "IN PROGRESS", "DONE". Utilizzata per conoscere l'andamento del progetto visto che si spostano le attività tra le varie colonne.
Matplotlib	Libreria di Python per la creazione di grafici e immagini.
Numpy	Libreria di Python che da supporto per lavorare con matrici e array.
Kivy	Framework per Python per la realizzazione di interfacce grafiche.
Opencv	Libreria di Python utilizzata per lavorare con le immagini.
Pillow	Python Imaging Library, libreria che aggiunge supporto per la lavorazione delle immagini.
Python	Linguaggio di programmazione, orientato ad oggetti, utilizzato per questo progetto.
Tkinter	Libreria di Python utilizzata per lo sviluppo di applicativi con interfaccia grafica.
Wordcloud	Stile di immagine dove un soggetto viene riempito di parole in base alla loro presenza in un testo.

11 Indice delle figure

Figura 1 UseCase 9
Figura 2 Diagramma di Gantt 10
Figura 3 Kanban Board 11
Figura 4 Design iniziale interfaccia 13
Figura 5 Secondo design per la GUI 14
Figura 6 Diagramma di flusso 15
Figura 7 Diagramma opzioni varie 16
Figura 8 Font family selezionato si applica al label 19
Figura 9 ColorPicker per il colore del bordo 21
Figura 10 Finestra di download per l'immagine 24
Figura 11 Gantt consuntivo 33

12 Allegati

- Diari di lavoro
- Codici sorgente
- Manuale di utilizzo
- QdC
- Prodotto