Deduplicatore di immagini

Titolo del progetto: Deduplicatore di immagini
Alunno/a: Edoardo Ratti
Classe: I3BB
Anno scolastico: 2022/2023
Docente responsabile: Geo Petrini

1 Introduzione

1.1 Informazioni sul progetto

- Allievo: Edoardo Ratti Docente: Geo Petrini
- SAMT sezione infomatica modulo 306
- 09.09.2022 -- 23.12.2022

1.2 Abstract

Con il miglioramento della tecnologia i dispositivi sono e saranno sempre dotati di memorie più grandi ma anche di file più dettagliati e di conseguenza di dimensioni maggiori; dunque, rimane il bisogno di limitare lo spazio utilizzato al minimo possibile evitando gli sprechi.
Io per ovviare al problema ho deciso di sviluppare un applicativo in grado di classificare le immagini in base alla loro similitudine, allo scopo di poter cancellare quelle doppie e quelle molto simili.
Inoltre alla risoluzione di questo problema il programma è anche in grado di darci un'idea delle nostre immagini, permettendo anche di poterle visualizzare senza aprile, dunque risparmiare tempo.

1.3 Scopo

Lo scopo del progetto è quello di separare le immagini simili raggruppandole in base alla loro similitudine, questo può per esempio aiutare l'utente a ottimizzare lo spazio, consigliando quali immagini sono ridondanti.
Inoltre, lo sviluppo del progetto comporta ad approfondimento delle conoscenze nel linguaggio Java e della sua libreria opencv.
Parlando del progetto in generale posso anche dire di aver incrementato le mie capacità inerenti alla progettazione, allo sviluppo di design e tutto l'insieme di file documentativi come documentazione e diari.

2 Analisi

2.1 Analisi del dominio

Il mio applicativo funge da riduttore di spazio inutile, è pensato per le per persone con grandi quantità di fotografie, le quali hanno problemi di spazio, non sapendo quali immagini sono realmente simili.
Il programma per risolvere il problema raggruppa tutte le immagini simili tra di loro, allo scopo di avere una visione completa di ciò che si dispone. Il software è adatto a tutti, è molto semplice e intuitivo da utilizzare e non necessità competenze particolari per il suo utilizzo, ma semplicemente il saper utilizzare in compilatore.

2.2 Analisi e specifica dei requisiti

Il prodotto in questione che funge da riduttore di spazio utilizzato è stato commissionato per far fronte al mercato della fotografia, dunque per alcuni mestieri come fotografi e fanartist.
Esso sarà disposto di un'interfaccia molto semplice ed intuitiva per facilitare il lavoro degli utenti, ma anche relativamente spoglia per adattare più facilmente nuove feature.

ID: REQ-001	
Nome	Formati supportati
Priorità	1
Versione	1.0
Note	JPG, PNG, WEBP

ID: REQ-002	
Nome	Scansione ricorsiva
Priorità	1
Versione	1.0

ID: REQ-003	
Nome	Rilevamento di immagini
Priorità	1
Versione	1.0
Sotto requisiti	
001	Rilevamento immagini identiche
002	Rilevamento immagini simili

ID: REQ-004	
Nome	Mostrare lista immagini raggruppate
Priorità	2
Versione	1.0
Note	Anteprima immagini + nome

ID: REQ-005	
Nome	Possibilità di salvare i risultati
Priorità	2
Versione	1.0

ID: REQ-006	
Nome	Gestione scansione
Priorità	2
Versione	1.0
Sotto requisiti	
001	Interruzione
002	Ripresa

2.3 Use case

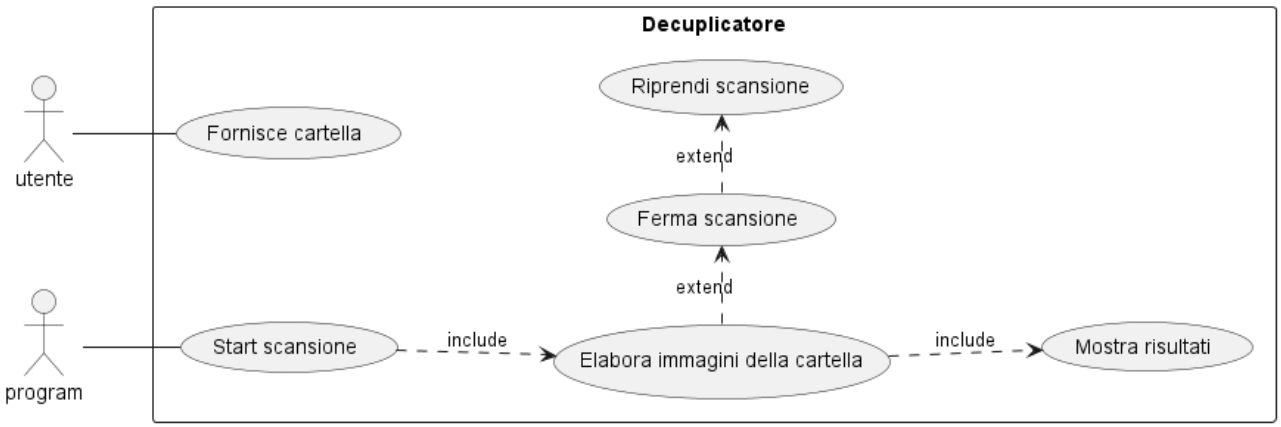
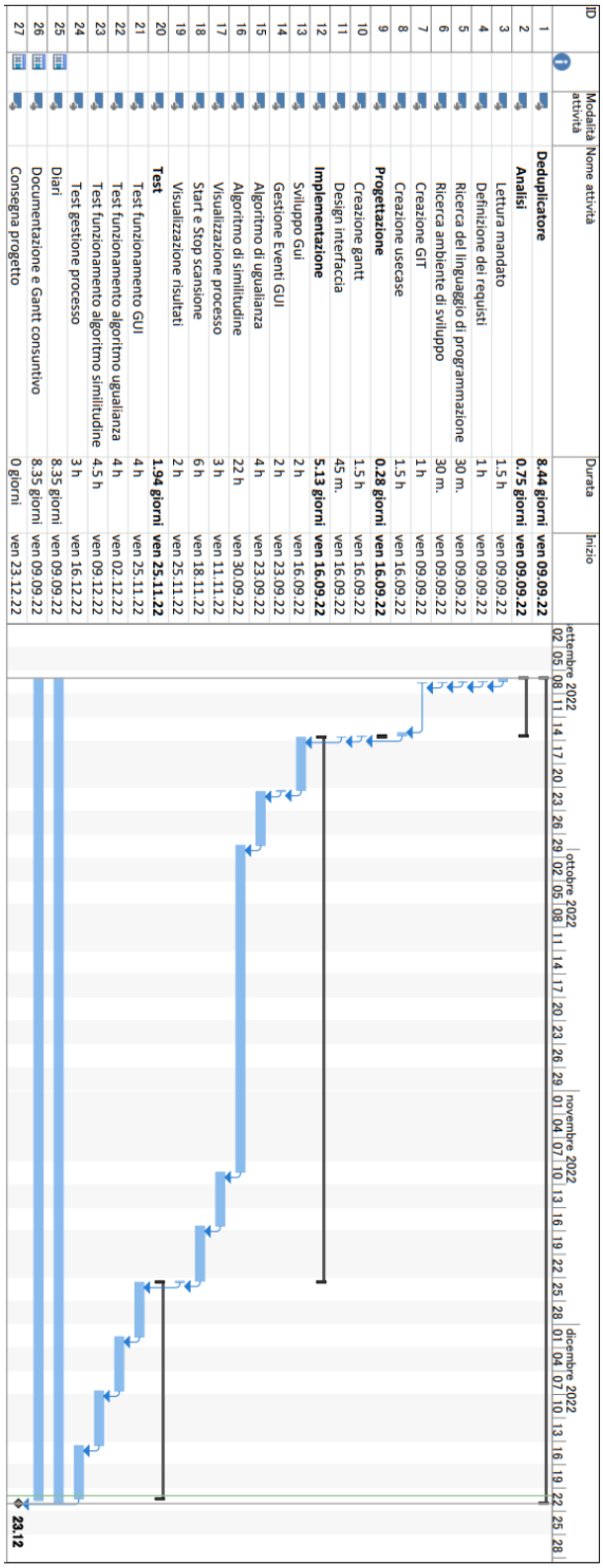


Figura 1 – UseCase

2.4 Pianificazione



ID	Modalità attività	Nome attività	Durata	Inizio
1		**Deduplicatore**	**8.44 giorni**	**ven 09.09.22**
2		**Analisi**	**0.75 giorni**	**ven 09.09.22**
3		Lettura mandato	1.5 h	ven 09.09.22
4		Definizione dei requisti	1 h	ven 09.09.22
5		Ricerca del linguaggio di programmazione	30 m.	ven 09.09.22
6		Ricerca ambiente di sviluppo	30 m.	ven 09.09.22
7		Creazione GIT	1 h	ven 09.09.22
8		Creazione usecase	1.5 h	ven 16.09.22
9		**Progettazione**	**0.28 giorni**	**ven 16.09.22**
10		Creazione gantt	1.5 h	ven 16.09.22
11		Design interfaccia	45 m.	ven 16.09.22
12		**Implementazione**	**5.13 giorni**	**ven 16.09.22**
13		Sviluppo Gui	2 h	ven 16.09.22
14		Gestione Eventi GUI	2 h	ven 23.09.22
15		Algoritmo di uguaglianza	4 h	ven 23.09.22
16		Algoritmo di similitudine	22 h	ven 30.09.22
17		Visualizzazione processo	3 h	ven 11.11.22
18		Start e Stop scansione	6 h	ven 18.11.22
19		Visualizzazione risultati	2 h	ven 25.11.22
20		**Test**	**1.94 giorni**	**ven 25.11.22**
21		Test funzionamento GUI	4 h	ven 25.11.22
22		Test funzionamento algoritmo uguaglianza	4 h	ven 02.12.22
23		Test funzionamento algoritmo similitudine	4.5 h	ven 09.12.22
24		Test gestione processo	3 h	ven 16.12.22
25		Diari	8.35 giorni	ven 09.09.22
26		Documentazione e Gantt consuntivo	8.35 giorni	ven 09.09.22
27		Consegna progetto	0 giorni	ven 23.12.22

Figura 2 -- Diagramma di Gantt

2.5 Analisi dei mezzi

Durante lo svolgimento del progetto con l'ausilio del computer scolastico sono a andato a ricercare alcuni dati che mi servivano al compimento del codice, in particolare mi serviva una libreria in grado di riconoscere i pattern tra oggetti, tra diversi algoritmi di definizione dei pattern ho scelto SIFT, questo perché molti altri pacchetti erano a pagamento e richiedevano licenze.

2.5.1 Software

- Apache NetBeans IDE 12.4
- Java JDK 17.0.5
- Opencv 4.5.5
- PlanUML 1.2022.14
- Microsoft Word
- Microsoft Project

2.5.2 Hardware

- CPU: i7-9700
- RAM: 32GB

3 Progettazione

3.1 Design delle interfacce

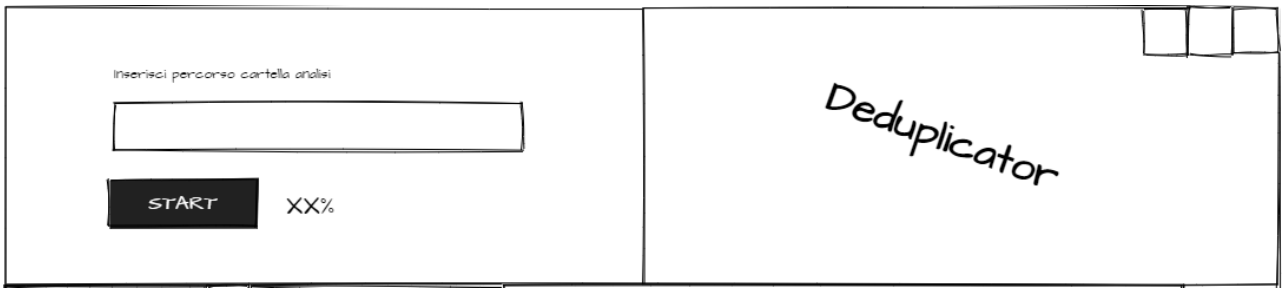


Figura 3 -- Design MainFrame

Interfaccia costituita da una parte che permette la gestione del processo, dove sarà possibile cambiare percorso, far partire e fermare la scansione con tanto di visibilità della percentuale di esecuzione.
Premendo sullo start il pulsante cambierà funzione diventando “STOP” e facendolo un'altra volta “RESTART”.

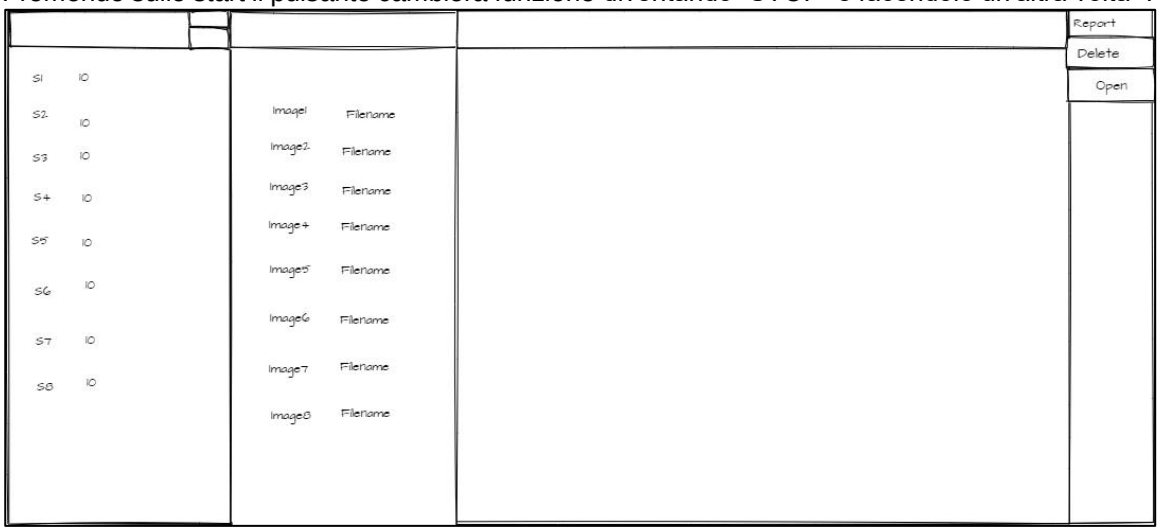


Figura 4 -- ImageMenuPanel

Nella seconda parte è possibile selezionare le singole serie da visualizzare tramite l’input predisposto, tale comportamento porta alla visualizzazione delle immagini contenute in una serie specifica, inoltre sono presenti dei pulsanti per il report della serie, cancellazione immagine e download immagine.

3.2 Design procedurale

Figura 5 -- Diagramma UML delle classi

4 Implementazione

Installare opencv e javacv con le varie JAR e DLL nel progetto.

4.1 Deduplicatore

Caricare opencv nella classe

```
static{ System.loadLibrary(Core.NATIVE_LIBRARY_NAME); }
```

Figura 6 -- Caricamento della libreria

Creare una costante stringa contenente tutte le estensioni possibili desiderate.

```
static final String[] EXTENSIONS = new String[]{
    "jpg", "png", "webp","PNG"
};
```

Figura 7 -- Array esensioni

Creare il filtro con l'ausilio del metodo *accept* che ci permette di stabilire il filtro sulle estensioni scelte.

```
static final FilenameFilter FILTER = new FilenameFilter() {
    @Override
    public boolean accept(final File dir, final String name) {
        for (final String ext : EXTENSIONS) {

            if (name.endsWith("." + ext)) {
                return (true);
            }
        }
        return (false);
    }
};
```

Figura 8 -- Filtro estensioni

```java
//tolleranza percentuale della scansione
public static final int TOLLERANCE = 30;
//contrassegna un elemento di misuration come cancellato
public static final int DELETED_ELEM = -2;
//path della cartella principale
public final String ROOTPATH;
//lista predisposta a conenere le path assolute di tutte
//le cartelle ricorsivamente
public List<File> directories = new ArrayList<>();
//misurazione calcolate dall'analisi
public int[][] misuration;
//lista predisposta a contenere tutte le immagini presenti in ROOTPATH
public List<File> images;
//si tratta del quantitativo di immagini
public int size;
//numero di immagini simili a un'immagine in base alla tolleranza
public int[] similar;
//percentuale di processo
public float percentage = 0;
```

Figura 9 -- Attributi della classe Deduplicatore

Questo metodo ricorsivo permette di ottenere un'alberatura completa in base alla root path selezionata in modo ricorsivo.

```java
public void getDirectories(String path){
    File root = new File(path);
    File[] list = root.listFiles();

    for (File f : list){
        if (f.isDirectory()){
            directories.add(f);
            getDirectories(f.getAbsolutePath());
        }
    }
}
```

Figura 10 -- getDirectories

Questo metodo invece permette di ottenere tutte le immagini contenute in una cartella in base ad un filtro.

```java
public List<File> getImages(){
    List<File> fileslist = new ArrayList<>();

    for (File dir : directories ) {
        File[] temp = dir.listFiles(FILTER);
        fileslist.addAll(Arrays.asList(temp));
    }
    return fileslist;
}
```

Figura 11 -- getImages

Metodo che permette di rimuove le serie ridondanti, ovvero con dati identici ma in posizione dell'array *misuration* differenti.

```
public void removeRedundantSeries() {
    countBetterSimilitude();
    for (int i = 0; i < size; i++) {
        for (int j = 0; j < size; j++) {
            if (misuration[i][j] >= TOLLERANCE && similar[i] >= similar[j]) {
                deleteRow(j);
            }
        }
    }
}
```

Figura 12 -- removeRedundantSeries

Questo seguente particolare metodo ha il compito di calcolare quante immagini simili abbia effettivamente ogni immagine e quanto lo siano.

```
public void countBetterSimilitude() {
    similar = new int[size];
    for (int i = 0; i < size; i++) {
        int simCnt = 0;
        for (int j = 0; j < size; j++) {
            if (misuration[i][j] >= TOLLERANCE) {
                simCnt += misuration[i][j];
            }
        }
        similar[i] = simCnt;
    }
}
```

Figura 13 -- countBetterSimilitude

Il sottostante metodo serve a eliminare assegnando il valore di *DELETED_ELEM* a tutte le immagini che sono già state assegnate a una serie e quelle con un valore minore di *TOLLERANCE*.

```
public void removeRedundantFiles(){
    for (int i = 0; i < size; i++) {
        if(IntStream.of(misuration[i]).anyMatch(x -> x == -1)){
            for (int j = 0; j < size; j++) {
                if(misuration[i][j] != -1){
                    if(misuration[i][j] < TOLLERANCE){
                        misuration[i][j] = DELETED_ELEM;
                    }else{
                        for (int k = 0; k < size; k++) {
                            if(misuration[i][j] > misuration[k][j] && i != k && misuration[k][j] != -1){
                                misuration[k][j] = DELETED_ELEM;
                            }
                        }
                    }
                }
            }
        }
    }
}
```

Figura 14 -- removeRedundantFiles

Si tratta del metodo più importante dell'intero programma, esso è proprio quello che va a calcolare la similitudine tra le immagini; infatti, esso possiede due parametri stringa contenenti le paths delle immagini che permetteranno di ottenere degli oggetti *Mat* appartenenti alla libreria installata.
Nelle due variabili descriptor1 e 2 verranno immagazzinati i numeri di descrittori delle immagini, ovvero il numero di punti simili trovati nella scansione che poi verranno rapportati per ottenere una percentuale di similitudine attendibile.

```
public static int compareImage(String p1, String p2) {
    SIFT detector = SIFT.create(0, 3);

    Mat img1 = Imgcodecs.imread(p1);
    Mat img2 = Imgcodecs.imread(p2);
    MatOfKeyPoint keypoints1 = new MatOfKeyPoint();
    MatOfKeyPoint keypoints2 = new MatOfKeyPoint();
    Mat descriptors1 = new Mat();
    Mat descriptors2 = new Mat();

    detector.detectAndCompute(img1, new Mat(), keypoints1, descriptors1);
    detector.detectAndCompute(img2, new Mat(), keypoints2, descriptors2);

    double max = Math.max(descriptors2.rows(), descriptors1.rows());
    double min = Math.min(descriptors2.rows(), descriptors1.rows());
    System.out.println("Testing  " + min / max * 100);
    return (int) ((min / max) * 100);
}
```

Figura 15 -- compareImage

Il metodo *analyseImage* è quello che mette assieme tutti gli altri metodi visti in precedenza allo scopo di paragonare tutte le immagini con se tutte le altre e già rimuovendo in gran parte i calcoli di dati ridondanti grazie alla consapevolezza che i dati sono specchiati in diagonale e che questa fornirà solo valori del 100%.
Alla fine del metodo c'è la stampa dell'array bidimensionale *misuration*, per poter contemplare i dati ottenuti dall'analisi.

```java
public void analyseImage() {
    getListSize(images);
    int numberOfOperation = (size * (size - 1)) / 2;
    misuration = new int[size][size];
    for (int i = 0; i < size; i++) {
        for (int j = 0; j < size; j++) {
            if (images.get(i).equals(images.get(j))) {
                misuration[i][j] = -1; //valore non valevole nel contesto per segnalare la base
            } else if (misuration[j][i] == 0) {
                long timer = System.currentTimeMillis();
                misuration[i][j] = compareImage(images.get(i).toString(), images.get(j).toString());
                misuration[j][i] = misuration[i][j];
                timer = System.currentTimeMillis() - timer;
                float increment = (float) ((float) timer / (float) ((float) timer * (float) numberOfOperation)) * (float) 100;
                //System.out.println("timer" + timer + "totTimer, " + (timer * numberOfOperation) + "Increment, " + increment);
                percentage = percentage + increment;
                //System.out.println("Percentuale " + percentage);
                raisePercChange();

            }
        }
    }
    removeRedundantSeries();
    removeRedundantFiles();
    for (int[] i : misuration) {
        for (int j : i) {
            System.out.printf("%03d ", j);
        }
        System.out.println("");
    }
}
```

Figura 16 -- analyseImage

Il seguente metodo serve a notificare alla classe genitore del cambiamento della percentuale di esecuzione che cambia ciclicamente all'analisi di due immagini.

```java
private void raisePercChange() {
    PropertyChangeEvent event = new PropertyChangeEvent(this, "c", 0, percentage);
    PropertyChangeListener listener = this.getPropertyChangeListeners()[0];
    listener.propertyChange(event);
}
```

Figura 17 -- raisePercChange

Codesto svolge la funzione di offrire al client l'analisi dei dati anche su un file di testo oltre che nell'interfaccia durante l'esecuzione.

```java
public void outReport() {
    String report = "";
    for (int i = 0; i < misuration.length; i++) {
        if (IntStream.of(misuration[i]).anyMatch(x -> x == -1)) {
            report += images.get(i).getName() + System.lineSeparator() + System.lineSeparator();
            for (int j = 0; j < misuration.length; j++) {
                int misure = misuration[i][j];
                if (misure >= Deduplicatore.TOLLERANCE) {
                    report += "\t-" + images.get(j).getName() + System.lineSeparator();
                }
            }
            report += System.lineSeparator();
        }
    }
    Path outFile = Paths.get("log.txt");
    byte[] b = report.getBytes();
    try {
        Files.write(outFile, b);
    } catch (IOException e) {
        throw new IllegalArgumentException("Error out report file");
    }
}
```

Figura 18 -- outReport

4.2 ImageMenuPanel

Proprietà utilizzate in questa classe

```
private Deduplicatore dec;
private Component[] imageButtons;//Contiente i pulsanti relativi alle immagini specifici di una serie
private Component[] serieButtons;//Contiene i pulsanti relativi alle serie
private int buttonSelectedIndex;//Indice del pulsante premuto corrente
private Component buttonSelected;//Pulsante corrente
private String root = "";
```

Figura 19 -- Attributi della classe ImageMenuPanel

Metodo che mostra le serie sull'interfaccia e assegna ad ogni pulsante inerente ad una serie l'evento che permetterà la selezione di quest'ultimo. Alla fine, è presente un pezzo di codice che stabilisce le dimensioni dello spinner presente nell'interfaccia in base alla quantità di serie disponibili.

```
public void displaySeries() {
    updateComponentTreeUI(this.getParent());
    seriesPanel.removeAll();
    filesPanel.removeAll();

    for (int i = 0; i < dec.misuration.length; i++) {
        if (IntStream.of(dec.misuration[i]).anyMatch(x -> x == -1)) {
            JButton button = new JButton(dec.images.get(i).getName());
            seriesPanel.add(button);

            //Creazione evento
            button.addActionListener(new java.awt.event.ActionListener() {
                @Override
                public void actionPerformed(java.awt.event.ActionEvent evt) {
                    seriesButtonPressedEvent(evt);
                }
            });
        }
    }
    SpinnerNumberModel model = new SpinnerNumberModel(0,0,seriesPanel.getComponentCount(),1);
    seriesSpinner.setModel(model);
    serieButtons = seriesPanel.getComponents();
}
```

Figura 20 -- displaySeries

Proprio come il metodo spiegato sopra questo secondo serve a svolgere il medesimo compito, ma la sostanziale differenza è che questo mostra invece le immagini al posto delle serie.

```
private void seriesButtonPressedEvent(java.awt.event.ActionEvent evt) {
    updateComponentTreeUI(this.getParent());
    int index = getFileIndex(((JButton) evt.getSource()).getText());
    buttonSelectedIndex = index;
    switchJToolBarButtons(true);
    imgdeleteButton.setEnabled(false);
    filesPanel.removeAll();
    for (int i = 0; i < dec.misuration.length; i++) {
        int misure = dec.misuration[index][i];
        if (misure >= Deduplicatore.TOLLERANCE) {
            JButton button = new JButton(dec.images.get(i).getName());
            filesPanel.add(button);
            button.addActionListener(new java.awt.event.ActionListener() { //Creazione evento
                @Override
                public void actionPerformed(java.awt.event.ActionEvent evt) {
                    showImage(evt);
                    buttonSelectedIndex = getFileIndex(((JButton) evt.getSource()).getText());
                    buttonSelected = button;
                    imgdeleteButton.setEnabled(true);
                }
            });
        }
    }
    imageButtons = filesPanel.getComponents();//components array
    try {
        //image display
        scaleImage(dec.images.get(index));
    } catch (IOException ex) {
        throw new IllegalArgumentException("Image not displayable");
    }
}
```

Figura 21 -- seriesButtonPressedEvent

I sottostanti due metodi hanno la funzione di filtrare la ricerca all'interno dell'interfaccia il primo filtra in base alla presenza della sottostrigna scritta nel campo di testo nelle immagini disponibili, mentre nel secondo caso tramite lo spinner sarà possibile selezionare una serie in base al numero di indice. Il numero zero è quello che mostra tutte le serie.

```
private void filterImages(){
    String text = filesSearchField.getText();
    for (Component button : imageButtons) {
        if (((JButton) button).getText().contains(text)) {
            button.setVisible(true);
        } else {
            button.setVisible(false);
        }
    }
}

private void filterSeries(){
    Object number = seriesSpinner.getValue();
    if((Integer)number == 0){
        for(Component button : serieButtons){
            button.setVisible(true);
        }
    }else{
        for (Component button : serieButtons) {
            button.setVisible(false);
        }
        serieButtons[(Integer)number - 1].setVisible(true);
    }
}
```

Figura 22 -- Metodi per filtrare i record

Semplicemente serve a cambiare lo stato dei tre pulsanti alla destra dell'interfaccia tra abilitato e non abilitato, allo scopo di agevolare il lavoro di quei tre.

```
public void switchJToolBarButtons(boolean b) {
    reportButton.setEnabled(b);
    imgdeleteButton.setEnabled(b);
    imgdlButton.setEnabled(b);
}
```

Figura 23 -- switchJToolBarButtons

ScaleImage, si occupa di fare in modo che l'immagine che deve comparire sia mostrata in scala.

```
private void scaleImage(File f) throws IOException {
    BufferedImage img = ImageIO.read(f);
    int w;
    int h;
    int max = Math.max(img.getHeight(), img.getWidth());
    if(max == img.getWidth()){
        w = imageLabel.getWidth();
        h = (int)((((float)img.getHeight()) / (float)img.getWidth()) * w);
    }else{
        h = imageLabel.getHeight();
        w = (int)((((float)img.getWidth()) / (float)img.getHeight()) * h);
    }
    Image dimg = img.getScaledInstance(Math.round(w), Math.round(h), Image.SCALE_DEFAULT);
    imageLabel.setIcon(new ImageIcon(dimg));
}
```

Figura 24 -- scaleImage

4.3 MainFrame

Per cominciare ho creato una piccola GUI contenente anche imageMenuPanel che permette di interagire in maniera ottimale con il programma.

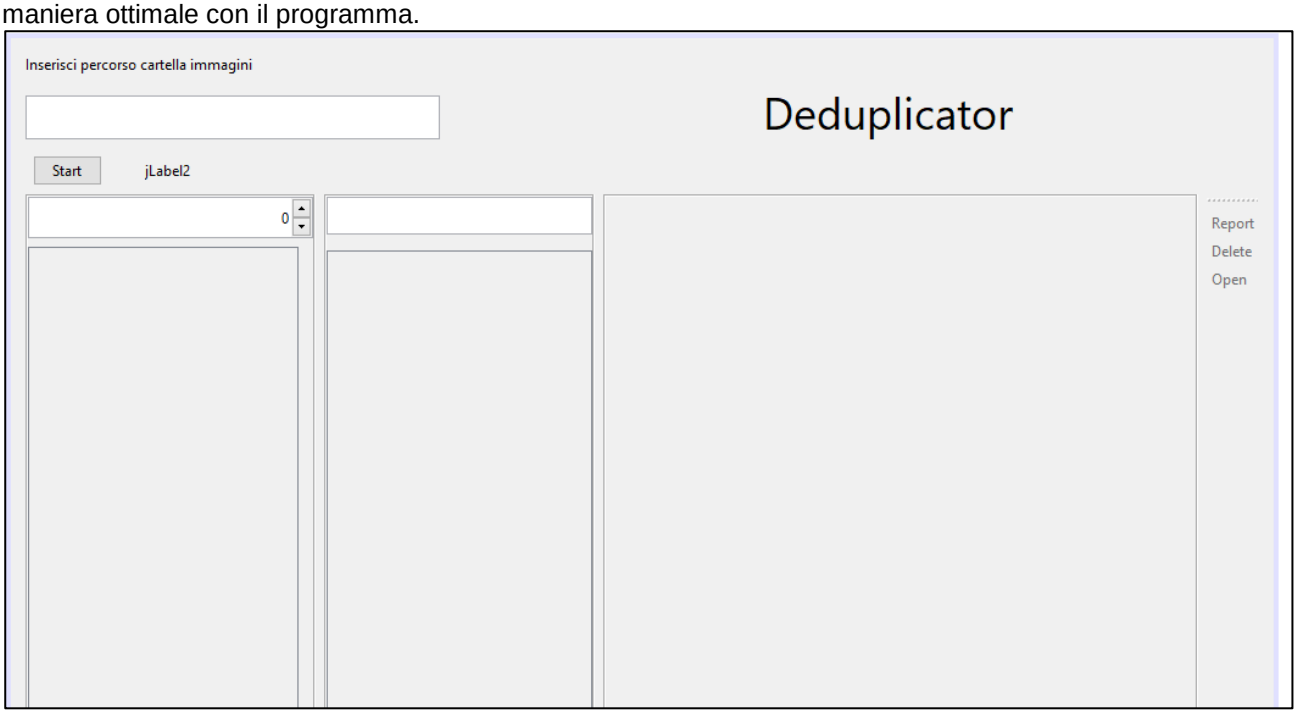


Figura 25 -- GUI

```
private Deduplicatore dec;
private int buttonState = 0; //Indica lo stato del pulsante tra start, stop e resume
private String root;
```

Figura 26 -- Attributi della classe MainFrame

Metodo che permette di scriver la percentuale di esecuzione all'interno della GUI, purtroppo il metodo non è funzionante, per qualche motivo non fa la stampa nell'etichetta, però è comunque in grado di stampare il numero corretto nel terminale.

```
private void percentageChange(java.beans.PropertyChangeEvent evt) {
    float perc = 0;
    jLabel2.setText(perc + "%");
    Object percentage = evt.getNewValue();
    if (percentage instanceof Float) {
        perc = (Float)percentage;
        System.out.println("Percentuale: " + Math.round(perc) + "%");
        jLabel2.setText(perc +  ""); // genera errore
        jLabel2.updateUI();
        updateComponentTreeUI(this);
        //repaint();
    }
}
```

Figura 27 -- percentageChange

5 Test

5.1 Protocollo di test

Test Case: **Riferimento**:	TC-001 REQ-001	**Nome:**	Rilevamento formato immagini
Descrizione:	Controllo se l'immagini hanno un formato supportato		
Prerequisiti:			
Procedura:	1. Cominciare una scansione con formati supportati e non supportati		
Risultati attesi:	Nel listing vengono mostrate solo le immagini con formati supportati		

Test Case: **Riferimento**:	TC-002 REQ-002	**Nome:**	Test di ricorsività
Descrizione:	Controllo se il programma raggiunge tutte le immagini		
Prerequisiti:	TC-001		
Procedura:	1. Mettere nella root delle cartelle immagini e cartelle con a loro volta altre immagini		
Risultati attesi:	Tutte le immagini sono presenti nel listing		

Test Case: **Riferimento**:	TC-003 REQ-003	**Nome:**	Test di comparazione immagini
Descrizione:	Controllo se l'immagine uguale o simile		
Prerequisiti:	TC-001, TC-002		
Procedura:	1. Cominciare una scansione con immagini simili, uguali e differenti 2. Controllare I risultati		
Risultati attesi:	Immagine identica: =100% - Immagine simile: ≠ 100%, >0%		

Test Case: **Riferimento**:	TC-004 REQ-004	**Nome:**	Mostra lista files
Descrizione:	La lista compare		
Prerequisiti:	TC-003		
Procedura:	1. Cominciare una scansione 2. Selezionare una serie		
Risultati attesi:	Il pannello cambia mostrando i file della serie selezionata		

Test Case: TC-005 **Riferimento**: REQ-005		**Nome:**	Apertura immagini
Descrizione:	È possibile aprire le immagini		
Prerequisiti:	TC-004		
Procedura:	1. Cominciare una scansione 2. Premere sul pulsante “apri”		
Risultati attesi:	Sarà aperta l’immagine		

Test Case: TC-006 **Riferimento**: REQ-006		**Nome:**	Interruzione e restart scansione
Descrizione:	Possibilità di gestire la scansione		
Prerequisiti:	TC-004		
Procedura:	1. Cominciare una scansione 2. Interrompere la scansione 3. Ricominciare la scansione		
Risultati attesi:	Ogni volta è cambiato lo stato della scansione in maniera corretta		

5.2 Risultati test

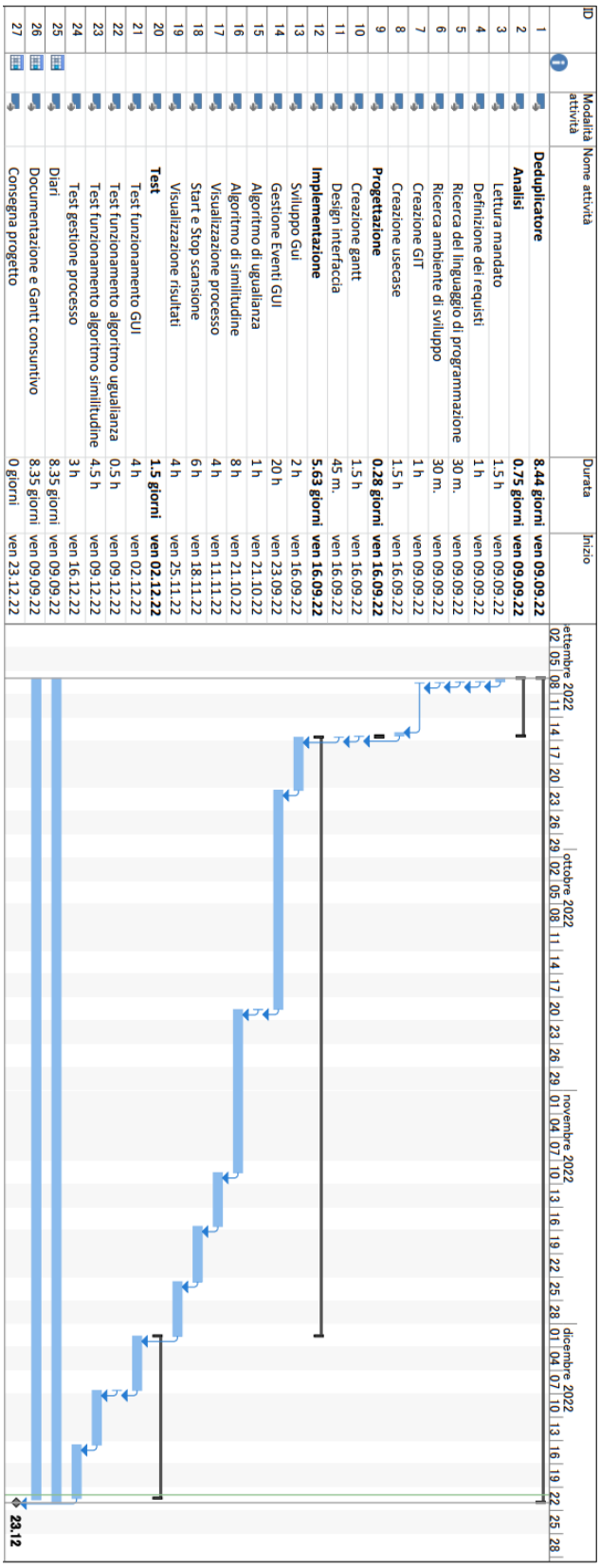
Test Case:	Risultato:
TC-001	Passato
TC-002	Passato
TC-003	Passato
TC-004	Passato
TC-005	Passato
TC-006	Non passato

Per quanto riguarda il TC-006, unico test non passato, ho pensato a una miglioria applicabile, ovvero un migliore utilizzo delle funzioni di threading, relativo al cambiamento di posizione della creazione del thread.

5.3 Mancanze/limitazioni conosciute

Durante l’implementazione delle funzionalità ho avuto un problema con l’utilizzo delle thread, non sono riuscito ad ottenere risultato atteso nel TC-006, e di conseguenza nemmeno a far comparire la percentuale di esecuzione a schermo.

6 Consuntivo



ID	Modalità attività	Nome attività	Durata	Inizio
1		**Deduplicatore**	**8.44 giorni**	**ven 09.09.22**
2		**Analisi**	**0.75 giorni**	**ven 09.09.22**
3		Lettura mandato	1.5 h	ven 09.09.22
4		Definizione dei requisti	1 h	ven 09.09.22
5		Ricerca del linguaggio di programmazione	30 m.	ven 09.09.22
6		Ricerca ambiente di sviluppo	30 m.	ven 09.09.22
7		Creazione GIT	1 h	ven 09.09.22
8		Creazione usecase	1.5 h	ven 16.09.22
9		**Progettazione**	**0.28 giorni**	**ven 16.09.22**
10		Creazione gantt	1.5 h	ven 16.09.22
11		Design interfaccia	45 m.	ven 16.09.22
12		**Implementazione**	**5.63 giorni**	**ven 16.09.22**
13		Sviluppo Gui	2 h	ven 16.09.22
14		Gestione Eventi GUI	20 h	ven 23.09.22
15		Algoritmo di ugualianza	1 h	ven 21.10.22
16		Algoritmo di similitudine	8 h	ven 21.10.22
17		Visualizzazione processo	4 h	ven 11.11.22
18		Start e Stop scansione	6 h	ven 18.11.22
19		Visualizzazione risultati	4 h	ven 25.11.22
20		**Test**	**1.5 giorni**	**ven 02.12.22**
21		Test funzionamento GUI	4 h	ven 02.12.22
22		Test funzionamento algoritmo ugualianza	0.5 h	ven 09.12.22
23		Test funzionamento algoritmo similitudine	4.5 h	ven 09.12.22
24		Test gestione processo	3 h	ven 16.12.22
25		Diari	8.35 giorni	ven 09.09.22
26		Documentazione e Gantt consuntivo	8.35 giorni	ven 09.09.22
27		Consegna progetto	0 giorni	ven 23.12.22

Figura 28 -- Gantt Consuntivo

Il gantt consuntivo ha alcuni cambiamenti, ma uno è stato quello più grande. Durante la pianificazione ero convinto che la parte di creazione dell'algoritmo di similitudine avrebbe richiesto l'intervento maggiore di tempo nel progetto, ma infine non è stato così infatti una parte che pensavo fosse molto semplice, ovvero la parte riguardante la gestione eventi della GUI è stata quella che mi ha fatto occupare la maggior parte del tempo, infatti era quella dove ho riscontrato più problemi, avendo il numero maggiore di attività da svolgere, anche se piccole richiedevano una breve ricerca di informazioni non indifferente.

7 Conclusioni

7.1 Sviluppi futuri

Come future migliorie io punto sull'aumento di features relative alle serie e alle immagini, come il report e l'apertura immagine, l'aumento di scansioni simultanee, utilizzo di un esplora risorse al posto di un campo di testo per la selezione del percorso, aumento delle informazioni relative ai record.

7.2 Considerazioni personali

Dopo questo periodo semestrale dove dedicavo una parte del mio tempo a questo progetto posso dire di essere contento del mio risultato. Penso che il mio progetto anche se non sia di fondamentale utilità, o meglio non si tratti di un applicativo che si usi moltissimo possa essere un cambiamento, ha cambiato in parte il mio modo di pensare, programmare, ma sicuramente mi ha aperto gli occhi.
Ha fatto approfondire le mie conoscenze nel linguaggio java: ho imparato ad installare librerie e sfruttare meglio le possibilità che offre, come l'utilizzo di liste, thread ed eventi. Adesso sono in grado di sviluppare un'interfaccia grafica e di implementare al suo interno controlli e funzioni.
Sicuramente la mia soluzione ha dei difetti, ma io sono felice di portali dietro, sono arrivato fino al risultato che ho ottenuto solo grazie agli sforzi che ho dato, dunque agli errori che ho portato avanti allo scopo di risolverli. Credo di aver ottenuto un buon risultato, anche se non sono riuscito a completarlo con tutte le caratteristiche che volevo implementare comunque sono arrivato ad un prodotto funzionante.
'

8 Glossario

Termine	Descrizione
Fanartist	Professione, creatore di disegni e immagini stilizzati
Spinner	Controllo input che permette di scalare i numeri in base a due pulsanti

9 Bibliografia

9.1 Sitografia

- https://wireframepro.mockflow.com, Wireframepro
- https://github.com/bytedeco/javacv, Github
- https://sourceforge.net/projects/opencvlibrary/files/4.5.5/opencv-4.5.5-vc14_vc15.exe, Souce Forge
- https://docs.opencv.org/4.x/d5/dde/tutorial_feature_description.html, Opencv
- https://docs.opencv.org/4.5.5/d7/d60/classcv_1_1SIFT.html, Opencv
- https://docs.opencv.org/3.4/db/d39/classcv_1_1DescriptorMatcher.html, Opencv
- https://docs.opencv.org/4.5.5/d2/d6e/classcv_1_1StereoMatcher.html, Opencv

9.2 Indice delle figure

10 Allegati

Elenco degli allegati, esempio:

- Diari di lavoro
- Deduplicatore.java, MainFrame.java, ImageMenuPanel.java
- Mandato e QdC
- Prodotto