

Università degli Studi di Trento  
Ingegneria dell'Informazione ed Organizzazione d'Impresa  
Comunicazioni multimediali  
A.A. 2019/2020

# **REPORT FINALE VINTAGE CAM**

**TEAM Foxfire**

Lyachi Sara - 178534  
Neri Carlotta - 196145  
Rossetti Elisabetta - 171255





# INDICE

ABSTRACT	1
INTRODUZIONE	2
PROJECT PLANNING	2
MOCKUP DELL'INTERFACCIA GRAFICA	8
GESTIONE DELLE CONTINGENZE	8
CONCLUSIONI E LESSONS LEARNED	9
APPENDICE A	11
CREDITS	20

## ABSTRACT

L’obiettivo di questo progetto consiste nell’implementazione di una serie di filtri “insta-like” statici e dinamici da applicare a file immagini e sequenze video.

Con l’obiettivo di rendere il progetto il più personale possibile il team “Firefox”, composto da tre risorse, ha deciso di seguire un tema di interesse comune, progettando filtri che creano un effetto vintage. Da qui l’idea di definire il progetto “*Vintage Cam*”.

L’attività progettuale è partita da una fase preliminare di **pianificazione** dell’intero progetto, dove si è proceduto ad un’accurata analisi di project management, che ha permesso di definire nel dettaglio lo scope del progetto, di trasformare gli obiettivi identificati in risorse necessarie alla loro realizzazione e di convertire le risorse individuate in calendario di attività.

In parallelo alla fase di pianificazione si è proceduto ad una fase di **gestione del cambiamento**, in modo tale da imporre dei margini di sicurezza sulle stime di pianificazione in termini di effort o di durata per poter affrontare con opportune azioni correttive le contingenze che si sono verificate nel corso dell’esecuzione delle varie attività progettuali.

Il codice sorgente di ogni filtro realizzato, così come i file immagini e video utilizzati nell’applicazione, si trovano nell’appendice A e sono stati forniti su un’apposita repository su GitHub: *layachisara/Vintage-Cam*

# INTRODUZIONE

Il seguente report intende descrivere nel dettaglio le varie fasi del ciclo di vita del progetto realizzato, partendo da una descrizione della fase preliminare di project planning e della parallela gestione delle contingenze. A chiusura del report vengono presentate delle considerazioni conclusive con una descrizione delle lessons learned.

## PROJECT PLANNING

La definizione degli obiettivi è una prerogativa per l'avvio di un progetto di sviluppo software. Essa infatti permette di stabilire i confini progettuali, garantendo che le attività individuate rappresentino una baseline del lavoro che deve essere svolto.

Tuttavia, non si deve dimenticare che la realizzazione di un progetto software è accompagnato inevitabilmente dall'insorgenza di imprevisti che devono essere affrontati con un opportuno approccio reattivo, noto come *contingency plan*, che consenta di ridurre l'impatto degli eventi negativi nel rispetto dei vincoli e di evitare il fallimento dell'intero progetto.

Ed è proprio quello che il nostro team ha affrontato in vari steps dell'attività: nonostante un'attenta pianificazione in termini di effort e di durata dei vari tasks il team si è trovato a dover applicare delle strategie di *remediation* verso le contingenze che hanno portato ad una rivalutazione e ridefinizione di alcuni milestones stabiliti in fase di *planning*.

I piani di contingenza, le opportune strategie di remediation messi in atto e i conseguenti cambiamenti al *project plan* verranno discussi in un'opportuna sezione del documento.

Nella fase della pianificazione, il team aveva stabilito come **obiettivo del progetto** la realizzazione di un'applicazione semplice che permettesse all'utente di applicare a immagini e video selezionabili una serie di filtri “*insta-like*” statici e dinamici con effetto “*vintage*”.

Nell'ottica di questo obiettivo, il team ha proceduto alla stesura del documento di scope identificando i requisiti funzionali e non funzionali dell'applicativo da realizzare. In particolare, l'analisi dei requisiti svolta dal team ha permesso di identificare i seguenti **requisiti funzionali**:

- l'applicazione deve consentire all'utente di selezionare video in formato MP4 e foto in formato JPEG tramite un'interfaccia utente immediata e di facile comprensione;
- l'elaborazione dei files consiste nell'applicazione di filtri statici e dinamici che vanno a imprimere effetti “*vintage*” agli elementi selezionati;
- l'applicazione deve consentire all'utente di selezionare i seguenti filtri:
  1. **Classic seppia:** genera un effetto seppia all'immagine attraverso l'applicazione di un'apposita matrice impostata con opportuni parametri;

2. **Dark Vintage**: applica un filtro gaussiano e una texture sull'immagine, che viene in seguito convertita in scala di grigi;
3. **Dirty Gray**: applica all'immagine un gray filter e aggiunge un successivo effetto di sharpening;
4. **Dreaming on**: aggiunge una texture all'immagine creando effetto flare;
5. **Polaroid**: permette di applicare una cornice con testo al file selezionato, creando un effetto di foto "polaroid";
6. **Stile retrò**: applica una variazione dell'effetto seppia al file e aggiunge un successivo sharpening, creando un effetto di foto antica;
7. **Sweet dusty**: imprime un effetto di blurring all'immagine e in seguito applica una texture che crea effetto di immagine impolverata;
8. **Old film**: filtro dinamico che sovrappone una sequenza video con effetto antico a un file video;
9. **Old tv**: filtro dinamico che va ad applicare a una sequenza video un effetto sfarfallio.

Per quanto riguarda i **requisiti non funzionali** in una prima fase sono stati identificati i seguenti:

- il linguaggio di programmazione con cui implementare i filtri è Java;
- l'interfaccia grafica GUI dell'applicazione viene implementata con JavaFX;
- l'applicazione deve garantire la portabilità su qualsiasi sistema operativo Android;
- l'applicazione deve supportare il caricamento di immagini in formato JPEG e video formato MP4 direttamente acquisiti dall'utente.

Dopo aver identificato i requisiti del progetto si è proceduto all'identificazione degli outputs, definendo in questo modo cosa il progetto prevede di realizzare e le relative tempistiche. In particolare sono stati definiti i **deliverables** principali, cioè risultati unici, misurabili e verificabili del lavoro svolto, che andranno poi a rappresentare nel diagramma di Gantt gli elementi di interconnessione tra le varie attività. I deliverables individuati sono i seguenti:

1. produzione del project planning;
2. sviluppo dell'applicazione Vintage Cam;
3. produzione e consegna del report finale.

Parallelamente all'identificazione dei deliverables progettuali sono stati anche identificati i **key milestones**, ovvero eventi significativi e critici nel progetto e nello schedule dello stesso che possono essere utilizzati per rappresentare punti di verifica e di monitoraggio dello stato di avanzamento delle attività progettuali. In particolare, i milestones identificati sono i seguenti:

1. approvazione del tema del progetto da parte del team e in seguito degli stakeholders coinvolti;
2. produzione dello scope document;
3. realizzazione ed approvazione del Mockup;

4. approvazione del prototipo dell'applicazione;
5. approvazione dell'applicativo funzionante;
6. realizzazione dello User Manual.

Una volta identificati gli obiettivi e i confini del progetto sono state definite le attività che devono essere portate avanti nel progetto usando una decomposizione gerarchica del lavoro che deve essere svolto dal team per realizzare gli obiettivi ed i deliverables identificati. Il risultato di questa analisi si è concretizzato nella realizzazione della **Work Breakdown Structure (WBS)**, che viene di seguito presentata nella figura 1:

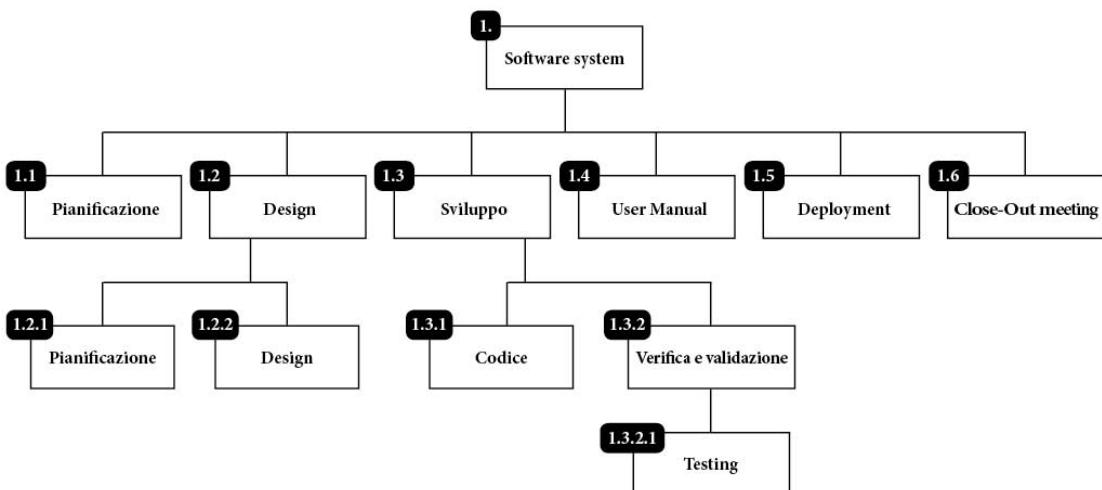


Figura 1: Work Breakdown Structure

Alla strutturazione delle attività progettuali nella WBS è seguita una fase di stima in termini di *ammontare di lavoro* per il completamento di ogni attività, di *risorse* richieste ed allocabili, espresse come manpower, e di *durata* di ogni task individuato.

Per svolgere la **stima dell'effort** è stata utilizzata un'equazione semplificata; questo perché la pianificazione di un progetto porta con sé variabilità derivanti dall'incapacità di stimare avvenimenti futuri. L'equazione utilizzata è la seguente:

$$\text{DURATA} = \text{EFFORT}/\text{MANPOWER}$$

dove la variabile MANPOWER viene espressa nel seguente modo:

$$\sum_{i=1}^N p_i$$

con N = numero di risorse e  $p_i$  = percentuale di disponibilità.

Il calcolo della durata dei vari tasks è stato eseguito presupponendo che l'attività di progetto si svolga su un totale di 5 giorni a settimana per un ammontare

di 8 ore al giorno. Le risorse disponibili sono tre, tutte occupate per un 50% dell'ammontare totale delle ore lavorative. Sono stati volutamente esclusi i sabati e le domeniche, ipotizzando la necessità di ciascuna risorsa di dedicarsi ad altre attività extra-progettuali o allo studio e preparazione di altri esami, ed i giorni festivi. In quest'ottica è possibile calcolare la durata richiesta per ciascuna delle attività nel seguente modo:

1. **PIANIFICAZIONE PROGETTO:** 168 man-hours con 3 risorse allocate al 50%  
DURATA = 168 man-hours/1.5 men = 112 ore = 14 giorni
2. **DESIGN:** 156 man-hours con 3 risorse allocate al 50%  
DURATA = 156 man-hours/1.5 men = 104 ore = 13 giorni
3. **SVILUPPO E TESTING:** 396 man-hours con 3 risorse allocate al 50%  
DURATA = 396 man-hours/1.5 men = 264 ore = 33 giorni
4. **USER MANUAL:** 24 man-hours con 3 risorse allocate al 50%  
DURATA = 24 man-hours/1.5 men = 16 ore = 2 giorni
5. **DEPLOYMENT:** 36 man-hours con 3 risorse allocate al 50%  
DURATA = 36 man-hours/1.5 men = 24 ore = 3 giorni
6. **CLOSE-OUT MEETING:** 12 man-hours con 3 risorse allocate al 50%  
DURATA = 12 man-hours/1.5 men = 8 ore = 1 giorno

Le stime probabilistiche ottenute sono state utilizzate per realizzare un diagramma **PERT**, rappresentato nella figura 2, che evidenzia il percorso logico da seguire nello svolgimento delle varie attività e, sulla base delle stime probabilistiche calcolate, permette di verificare la probabilità del piano di finire entro una certa data.

Nell'ottica di una strategia coerente di **change management**, partendo dal presupposto che i milestones identificati devono essere misurabili in termini di successo o insuccesso, il team ha previsto per ciascuno di essi un **contingency plan**, ovvero una strategia ad hoc da implementare nel caso in cui fossero subentrati eventi che avrebbero portato alla rivisitazione dei milestones prestabiliti e alla gestione ottimale dei rischi e delle incertezze associati a questi eventi.

A questo punto si può realizzare uno scheduling definitivo delle attività progettuali che si è concretizzato in un diagramma di Gantt, mostrato nella figura 3.

In esso sono stati messi in evidenza i seguenti aspetti:

- inizio e fine di un'attività;
- dipendenze tra le varie attività;
- allocazione e livellamento delle risorse;
- realizzazione e consegna degli output progettuali

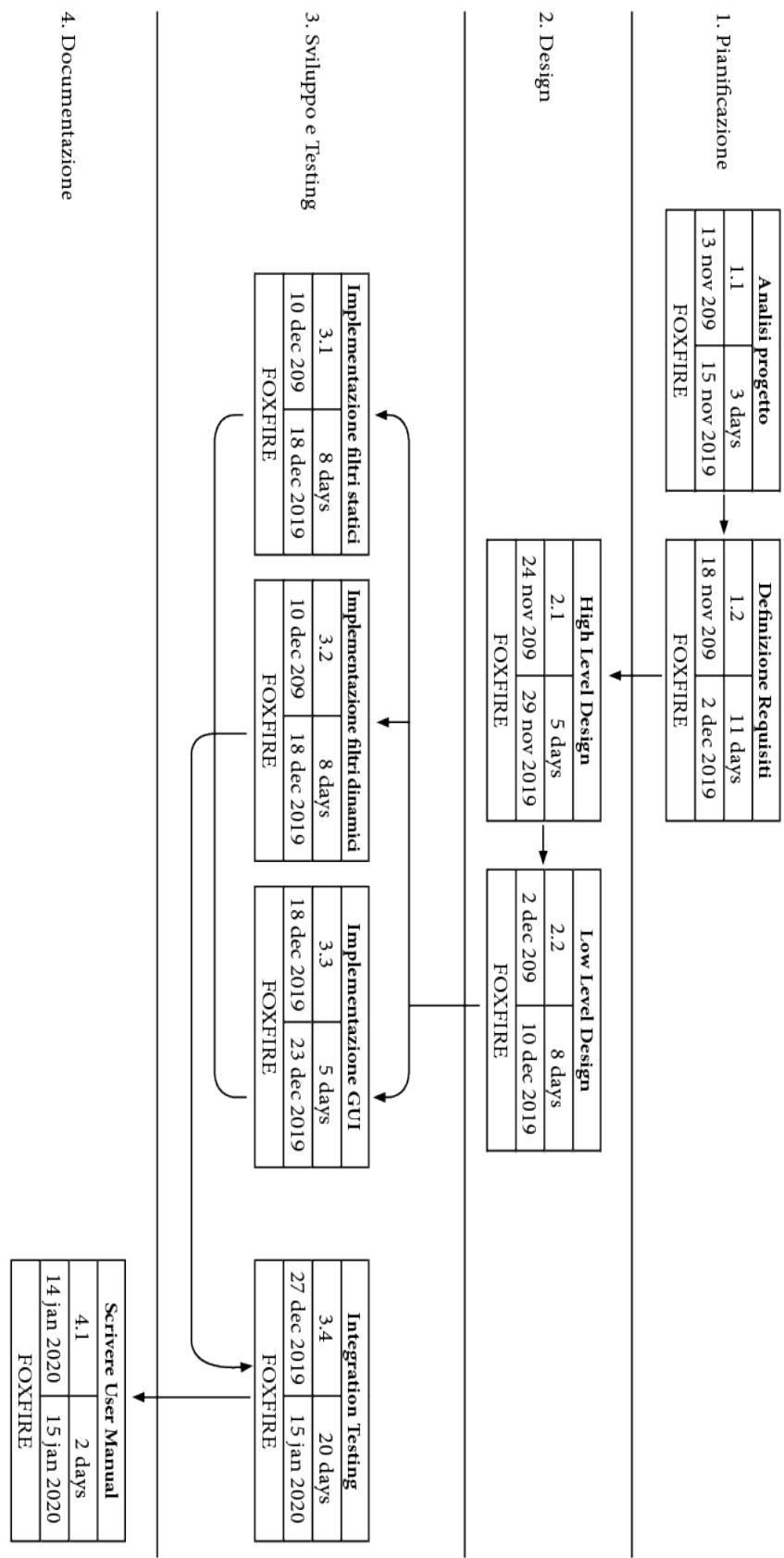


Figura 2: PEERT

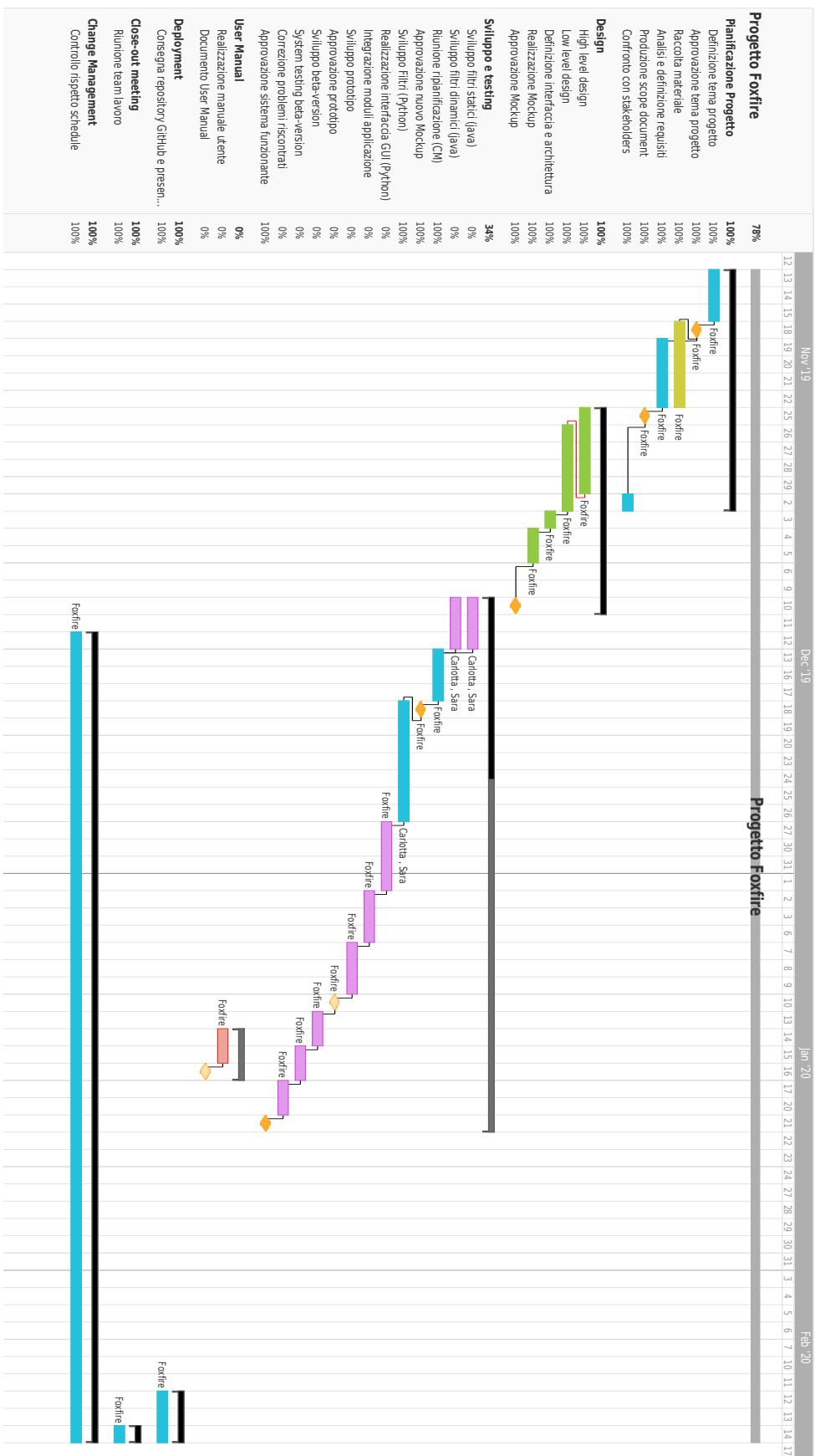


Figura 3: Gantt

## MOCKUP DELL'INTERFACCIA GRAFICA

L'iniziale pianificazione del progetto prevedeva la realizzazione di un mockup, ovvero di un prototipo che riproducesse fedelmente l'interfaccia grafica dell'applicazione. Il team aveva stabilito che l'importanza del mockup consistesse nel testare la funzionalità dell'applicativo, consapevole che realizzare la copia verosimile di un prodotto fosse utile sia per il team sia per il cliente finale.

Di seguito, nella figura 4, la rappresentazione del mockup realizzato:

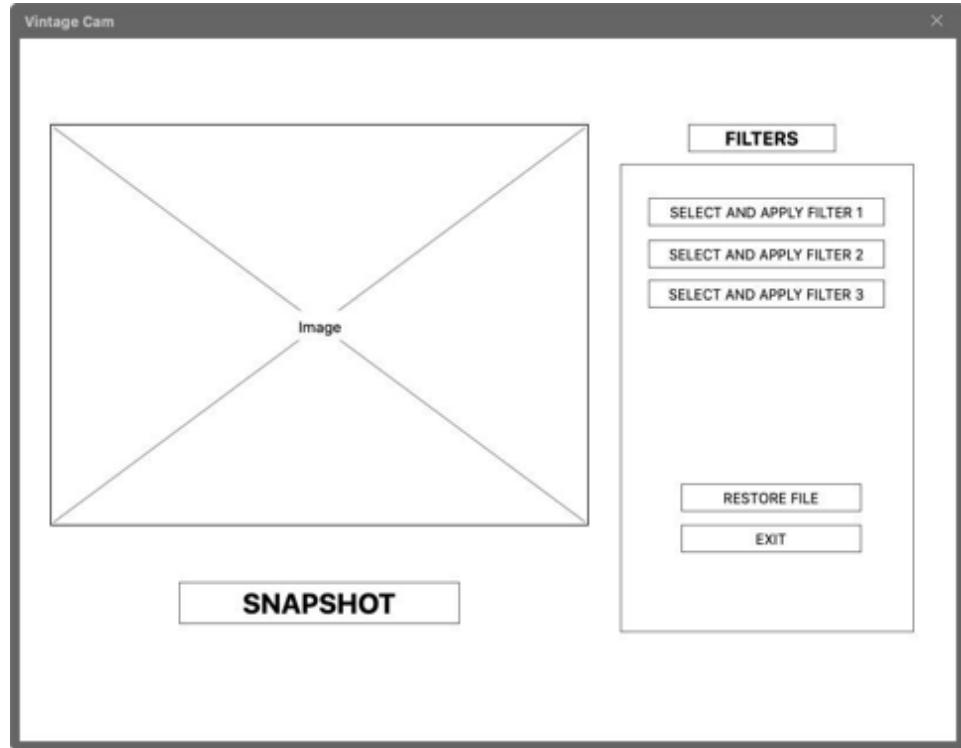


Figura 4: Mockup interfaccia

## GESTIONE DELLE CONTINGENZE

Nel concreto, l'attività che ha mostrato maggiori criticità è stata la fase di sviluppo, durante la quale si sono verificati vari imprevisti che hanno aumentato la durata stimata per la realizzazione dei milestones associati e di conseguenza quella delle attività correlate. Questa fase prevedeva inizialmente la realizzazione dei filtri statici e dinamici utilizzando come linguaggio di programmazione Java. Dopo aver iniziato il lavoro di scrittura del codice, le risorse del team si sono rese conto che non erano in linea con le tempistiche stabilite. Una riunione del team ha permesso di ricondurre la motivazione di tale ritardo alle limitate competenze nell'utilizzo del linguaggio di programmazione Java e nella realizzazione dell'interfaccia grafica con JavaFX. Si è quindi deciso di gestire tale contingenza andando a rivalutare il linguaggio di programmazione da utilizzare per la realizzazione dei

filtr, scegliendo un linguaggio che ogni membro del gruppo riteneva più adeguato, ovvero Python.

Per ovviare al ritardo registrato a seguito di questo evento e comprimere lo scheduling del piano garantendo la consegna del prodotto finito entro la data prestabilita, è stato deciso all'unanimità di eliminare dalla pianificazione l'attività di scrittura dello user manual, in quanto considerata poco utile in vista dell'obiettivo del progetto rivalutato. Si è inoltre ritenuto utile rivedere le stime della fase di testing, per la quale si è deciso di allocare una sola risorsa e di svincolare le altre due che quindi si sono potute occupare, con un surplus di ore lavoro, alla fase di implementazione del codice.

Un'ulteriore problematica incontrata durante lo svolgimento del progetto ha riguardato la realizzazione dell'interfaccia grafica dell'applicazione. Nel corso dello svolgimento del progetto una delle tre risorse ha dovuto ridurre la disponibilità oraria personale, in quanto si è trovata coinvolta in un'attività lavorativa full time. Questo ha imposto sia una riallocazione delle due risorse residue in termini di effort richiesto, sia una rivalutazione dei milestones progettuali, al fine di garantire la realizzazione dell'obiettivo finale entro le tempistiche previste dallo schedule. Il team ha affrontato tale contingenza decidendo di non realizzare l'interfaccia grafica dell'applicazione, ma di presentare il codice sorgente dei filtri realizzati.

## CONCLUSIONI E LESSONS LEARNED

L'attività progettuale ha permesso ai membri del team di trarre importanti insegnamenti in termini sia di acquisizione di know how, tecnici e teorici, sia di rafforzamento di competenze in campo di gestione del ciclo di vita di un progetto. Nel corso dello svolgimento dei vari tasks previsti infatti si è fin da subito imposta la necessità di creare un vero e proprio clima di team tra tutte le risorse coinvolte, ricreando le dinamiche di un vero e proprio progetto di sviluppo. In particolare, ci siamo impegnate fin da subito ad impostare relazioni positive e una comunicazione chiara, franca e aperta tra di noi, cercando di realizzare sessioni di lavoro frequenti, con una partecipazione che prevedeva l'impegno e la responsabilizzazione di ciascuna risorsa nei confronti del progetto e del lavoro altrui, mantenendo comunque un rispetto di fondo per gli impegni di ciascun membro del team. Fondamentale per la buona riuscita del progetto è stata la ricerca del consenso e della condivisione da parte di tutti i componenti che prevedeva azioni di processo decisionale, problem solving di gruppo e frequenti kick-off meeting come punti di confronto fondamentali.

Le contingenze incontrate nel corso dello svolgimento del progetto sono state affrontate con passaggi di consegne tempestivi e sincronizzati in modo tale da riuscire a gestire gli imprevisti in maniera ottimale e favorire l'avanzamento del progetto.

Di seguito, nella figura 5, si fornisce una rappresentazione grafica degli obiettivi progettuali effettivamente realizzati rispetto a quelli pianificati in fase di project planning, in modo tale da poter prevedere le attività residue che a seguito di un kick-off meeting finale il team di sviluppo ha deciso di completare.

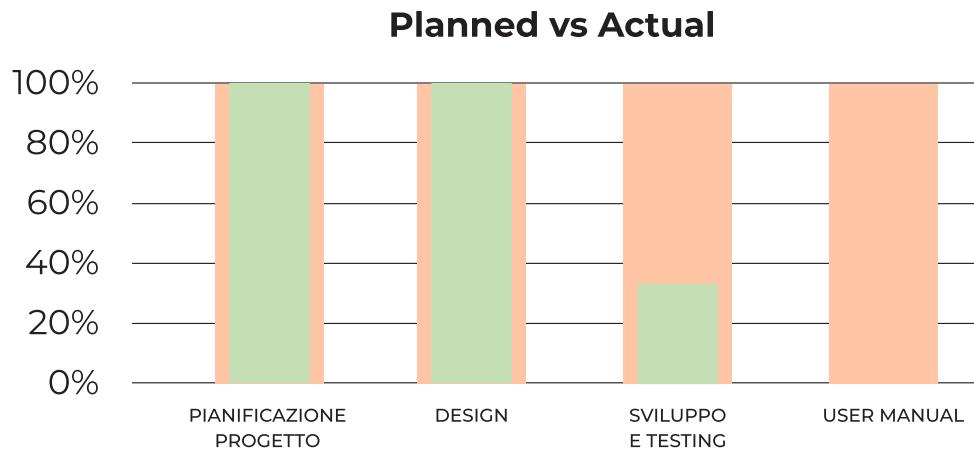


Figura 5: Reporting delle performances

Come fase successiva alla consegna del progetto, nell’ottica di un piano progettuale realistico e finalizzato al completamento di tutte le attività previste nello scheduling, il team ha deciso di definire i tasks rimasti incompiuti. Le attività residue che dovranno essere svolte per modificare l’impostazione del progetto da “as is” a “to be” sono le seguenti:

- Realizzazione interfaccia grafica dell’applicazione che consenta all’utente di applicare filtri tempo-varianti che vanno a calibrare i parametri di saturazione, contrasto e brightness del file selezionato;
- Realizzazione dello User Manual;
- Implementare il “Filtro Polaroid” in modo tale da permettere all’utente di poter scrivere nella cornice un messaggio personalizzato.

# APPENDICE A

## CODICE FILTRI

### 1. CLASSIC SEPIA

```
import cv2
import numpy as np
from PIL import Image

# read image
input = cv2.imread('foto-input/foto-3.jpg')

# Apply Sepia matrix
array = np.array(input)
seppiaMatrix = np.array([[0.393, 0.769, 0.189], [0.349, 0.686, 0.168], [0.272, 0.534, 0.131]])
filter = cv2.transform(array, seppiaMatrix)
# Check which entries have a value greater than 255 and set it to 255
filter[np.where(filter > 255)] = 255

# Create an image from the array
output: Image
output = Image.fromarray(filter)

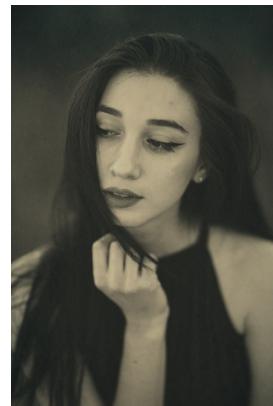
# conversion from Pil to cv
output_np = np.array(output)
output_cv = cv2.cvtColor(output_np, cv2.COLOR_RGB2BGR)

# add texture
mask = cv2.imread('Texture/blu-texture.jpg')
mask_resized = cv2.resize(mask, (output_cv.shape[1], output_cv.shape[0]))
output = cv2.addWeighted(output_cv, 0.7, mask_resized, 0.1, 0)

cv2.imshow('Classic sephia', output)
cv2.waitKey(0)
cv2.imwrite("foto-output/classic-seppia.jpg",output)
cv2.destroyAllWindows()
```



input



output

## 2. DARK VINTAGE

```
import cv2
import numpy as np

# read image
input = cv2.imread('foto-input/foto-7.jpg')

# apply dark vintage filter
# Gaussian filter
rows, cols = input.shape[:2]
kernel_x = cv2.getGaussianKernel(cols,170)
kernel_y = cv2.getGaussianKernel(rows,170)
kernel = kernel_y * kernel_x.T
filter = 255 * kernel / np.linalg.norm(kernel)
vintage_im = np.copy(input)
# for each channel in the input image, we will apply the above filter
for i in range(3):
    vintage_im[:, :, i] = vintage_im[:, :, i] * filter

# add texture
mask = cv2.imread('Texture/blu-texture.jpg')
mask_resized = cv2.resize(mask, (vintage_im.shape[1], vintage_im.shape[0]))
dst = cv2.addWeighted(vintage_im, 0.7, mask_resized, 0.3, 0)

# convert to gray
gray_im= cv2.cvtColor(dst, cv2.COLOR_BGR2GRAY)

cv2.imshow('Dark vintage', gray_im)
cv2.waitKey(0)
cv2.imwrite("foto-output/dark-vintage.jpg",gray_im)
cv2.destroyAllWindows()
```



input



output

### 3. DIRTY GRAY

```
import cv2
import numpy as np

input = cv2.imread("foto-input/foto-1.jpg")

# gray filter
output_gray = cv2.cvtColor(input, cv2.COLOR_BGR2GRAY)

# sharpening filter
Kernel = np.array([[-1, -1, -1], [-1, 9, -1], [-1, -1, -1]])
output_sharpen = cv2.filter2D(output_gray, -1, Kernel)

cv2.imshow('Dirty Gray', output_sharpen)
cv2.waitKey(0)
cv2.imwrite("foto-output/dirty-gray.jpg",output_sharpen)
cv2.destroyAllWindows()
```



input



output

#### 4. DREAMING ON

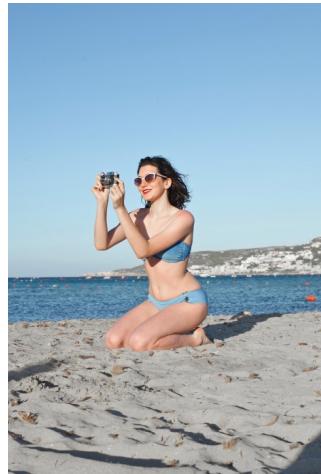
```
import cv2

# read image
input = cv2.imread('foto-input/foto-6.jpg')

# add texture
mask = cv2.imread('Texture/texture-dream.jpg')
mask_resized = cv2.resize(mask, (input.shape[1], input.shape[0]))
output1 = cv2.addWeighted(input, 0.7, mask_resized, 0.6, 0)
# mask1 = cv2.imread('Texture/texture-2.jpeg')
# mask_resized1 = cv2.resize(mask1, (dst.shape[1], dst.shape[0]))
# dst1 = cv2.addWeighted(dst, 0.7, mask_resized1, 0.1, 0)

# add texture
mask = cv2.imread('Texture/blu-texture.jpg')
mask_resized = cv2.resize(mask, (output1.shape[1], output1.shape[0]))
output2 = cv2.addWeighted(output1, 0.7, mask_resized, 0.2, 0)

cv2.imshow('Dreaming On', output2)
cv2.waitKey(0)
cv2.imwrite("foto-output/dreaming-on.jpg",output2)
cv2.destroyAllWindows()
```



input



output

## 5. POLAROID

```
import cv2
import numpy as np

# read image
input = cv2.imread('foto-input/foto-2.jpg')

# add BLURRING: image blurring is achieved by convolving the image with a low-pass filter kernel.
# It is useful for removing noise. It actually removes high frequency content (eg: noise, edges) from the image.
# So edges are blurred a little bit in this operation (there are also blurring techniques which don't blur the edges).
#blur = cv2.GaussianBlur(input,(13,13),cv2.BORDER_DEFAULT)
Kernel = np.ones((4, 4), np.float32) / 16
blur_img = cv2.filter2D(input, -1, Kernel)

mask = cv2.imread('Texture/polaroid-texture.jpg')
mask_resized = cv2.resize(mask, (blur_img.shape[1], blur_img.shape[0]))
out = cv2.addWeighted(blur_img, 0.7, mask_resized, 0.5, 0)

# create an image with a single color (red)
red_img = np.full((682,512,3), (0,0,150), np.uint8)
mask_resized = cv2.resize(red_img, (out.shape[1], out.shape[0]))

# add the filter with a weight factor of 20% to the target image
fused_img = cv2.addWeighted(out, 0.8, mask_resized, 0.2, 0)
WHITE = [255,255,255]
white_border_img = cv2.copyMakeBorder(fused_img, 60,60,20,20, cv2.BORDER_CONSTANT, value=WHITE)

# add text
position = (120,700)
cv2.putText(
    white_border_img, #numpy array on which text is written
    "Happy Birthday", #text
    position, #position at which writing has to start
    cv2.FONT_HERSHEY_SIMPLEX, #font family
    1, #font size
    (0, 0, 0, 0), #font color
    3) #font stroke

cv2.imshow("Polaroid", white_border_img)
cv2.waitKey(0)
cv2.imwrite("foto-output/polaroid.jpg",white_border_img)
cv2.destroyAllWindows()
```



input



output

## 6. STILE RETRÒ

```
import cv2
import numpy as np
from PIL import Image

# read image
input = cv2.imread('foto-input/foto-5.jpg')
# apply seppia filter
array = np.array(input)
seppiaMatrix = np.array([[0.393, 0.769, 0.189], [0.349, 0.686, 0.168],[0.272, 0.534, 0.131]])
filter = cv2.transform(array, seppiaMatrix)
# Check which entries have a value greater than 255 and set it to 255
filter[np.where(filter > 255)] = 255
output_pil: Image
output_pil = Image.fromarray(filter)
output_np = np.array(output_pil)
output_cv = cv2.cvtColor(output_np, cv2.COLOR_RGB2BGR)

# apply sharpening
Kernel = np.array([[-1, -1, -1],[-1, 9, -1],[-1, -1, -1]])
output_sharpen = cv2.filter2D(output_cv, -1, Kernel)

# print output
cv2.imshow('Stile Retro', output_sharpen)
cv2.waitKey(0)
cv2.imwrite("foto-output/stile-retro.jpg",output_sharpen)
cv2.destroyAllWindows()
```



input



output

## 7. SWEET DUSTY

```
import cv2
import numpy as np

# read image
input = cv2.imread('foto-input/foto-4.jpg')

# add blur
kernel = np.ones((4,4), np.float32) / 16
blur_img = cv2.filter2D(input, -1, kernel)

# add texture
mask = cv2.imread('Texture/texture-dust.jpg')
mask_resized = cv2.resize(mask, (blur_img.shape[1], blur_img.shape[0]))
output = cv2.addWeighted(blur_img, 0.7, mask_resized, 0.2, 0)

cv2.imshow('Sweet Dusty', output)
cv2.waitKey(0)
cv2.imwrite("foto-output/sweet-dusty.jpg",output)
cv2.destroyAllWindows()
```



input



output

## 8. OLD FILM

```

import cv2
import numpy as np
def dirty_gray(frame):
    ## gray filter
    output_gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    ## sharpening filter
    Kernel = np.array([[-1, -1, -1], [-1, 9, -1], [-1, -1, -1]])
    output_sharpen = cv2.filter2D(output_gray, -1, Kernel)
    return output_sharpen

mask_filter= cv2.VideoCapture('Texture/old-film-texture.mp4')
cap = cv2.VideoCapture('video-input/video-3.mp4',0)
width_frame = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
height_frame = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
size = (width_frame,height_frame)
# Define the codec and create VideoWriter object.
fourcc = cv2.VideoWriter_fourcc(*'m', 'p', '4', 'v')
out = cv2.VideoWriter('video-output/old-film.mp4', fourcc, 20.0, size,0)
while cap.isOpened():
    ret, frame = cap.read()
    ret1, frame1 = mask_filter.read()
    if ret == True and ret1 == True:

        h, w, c = frame.shape
        h1, w1, c1 = frame1.shape
        # adapt the mask to the video
        if h != h1 or w != w1:
            frame1 = cv2.resize(frame1, (w, h))

        both = cv2.addWeighted(frame, 0.9, frame1, 0.3, 0)
        output = dirty_gray(both)
        cv2.imshow('Old film', output)
        out.write(output)

        if cv2.waitKey(50) & 0xFF == ord('q'):
            break
    else:
        break

cap.release()
mask_filter.release()
out.release()
cv2.waitKey(0)
cv2.destroyAllWindows()

```

frame input



frame output



## 9. OLD TV

```

import cv2
import numpy as np

def dirty_gray(frame):
    ## gray filter
    output_gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    ## sharpening filter
    Kernel = np.array([[-1, -1, -1], [-1, 9, -1], [-1, -1, -1]])
    output_sharpen = cv2.filter2D(output_gray, -1, Kernel)

    return output_sharpen

mask_filter= cv2.VideoCapture('Texture/old-tv-texture.mp4',0)
cap = cv2.VideoCapture('video-input/video-6.mp4')
width_frame = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
height_frame = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
size = (width_frame,height_frame)
# Define the codec and create VideoWriter object
fourcc = cv2.VideoWriter_fourcc(*'m', 'p', '4', 'v')
out = cv2.VideoWriter('video-output/old-tv.mp4', fourcc, 20.0, size,0)
while cap.isOpened():
    ret, frame = cap.read()
    ret1, frame1 = mask_filter.read()
    if ret == True and ret1 == True:
        h, w, c = frame.shape
        h1, w1, c1 = frame1.shape
        # adapt the mask to the video
        if h != h1 or w != w1:
            frame1 = cv2.resize(frame1, (w, h))
        both = cv2.addWeighted(frame, 0.9, frame1, 0.4, 0)
        output = dirty_gray(both)                                frame input
        cv2.imshow('Old film', output)
        out.write(output)

        if cv2.waitKey(25) & 0xFF == ord('q'):
            break
    else:
        break

cap.release()                                         frame output
out.release()
mask_filter.release()
cv2.waitKey(0)
cv2.destroyAllWindows()

```



Credits:

Foto di Cottonbro da Pexels

Foto di Johan De Jager da Pexels

Foto di Daria Shevtsova da Pexels

Video di Kelly Lacy da Pexels

Video di Lisa Fotios da Pexels

Video di Cottonbro da Pexels