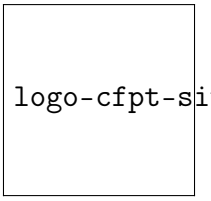

CFPT Informatique

Atelier Techniciens 1&2 - Rapport

ThermiScan

Étudiants :
Lorenzo Bauduccio
Tom Ryser
Kevin Moreno

Enseignant :
Francisco Garcia

11 mars 2020

Table des matières

1 Cahier des charges

1.1 Description du projet

Nous devons réaliser une application en Web qui doit nous permettre d'afficher une vidéo venant du smartphone CAT S60, qui possède une caméra thermique. L'application Web doit permettre de gérer plusieurs vidéos venant de plusieurs caméras, et permettre d'afficher une courbe de température.

1.2 Travail à réaliser

- Réaliser une application en Web qui s'approche un maximum d'un point de vue fonctionnel de l'application FLIR Tools Mobile.
  - Affichage d'un graphique montrant la température maximal, minimal et la moyenne.
  - Affichage de la vidéo.
  - Gestion de plusieurs vidéos venant d'une seule caméra.

1.3 Outils utilisés

- Visual Studio Code
- EasyPHP V 14.1
- MySQL
- Python 3
- Balsamiq Mockups 3

1.4 Comment avons-nous procédé ?

- Réalisation d'une maquette du front-end
- Création d'un exécutable à partir d'un programme python
- Création d'une base de données pour les utilisateurs, les caméras et les vidéos
- Réalisation du front-end avec easyPHP
- Ajout d'un script JS permettant d'afficher un graphique
- Mise au propre du code

1.4.1 Installation de Tesseract

1. Exécuter dans le cmd *pip install pytesseract*
2. Télécharger Tesseract OCR sur `https://digi.bib.uni-mannheim.de/tesseract/tesseract-ocr-w64-setup-v5.0.0-alpha.20200223.exe` et faire l'installation par défaut.
3. Déplacer le dossier contenu dans le .zip sous `C:\ProgramFiles`.
4. Dans la barre de recherche Windows, taper *"Modifier les variables d'environnement système"*.
5. Cliquer sur le bouton *"Variable d'environnement"* en bas de la page.
6. Cliquer sur *"Nouvelle..."* pour les variables d'administrateur.
7. Nom de la variable : `TESSDATA_PREFIX`.
8. Cliquer sur *"Parcourir le répertoire"* et sélectionner Tessdata sous `C:\ProgramFiles\tesseract-OCR\tessdata`

1.4.2 Installation de Python 3.8

1. Aller sur le site de python : `https://www.python.org/downloads/release/python-382/`
2. Prendre *Windows x86 executable installer*.
3. Au début de l'installation, sélectionner *Add Python 3.8 to PATH*.
4. À la fin de l'installation, cliquer sur *Disable Path Length limit*.
5. Après l'installation de Tesseract, Ouvrir le cmd et aller dans le dossier `pythonApp` du projet et exécuter dans le cmd *pip install -r requirements.txt*

2 Réalisations

2.1 Description du code

le fichier `background.py` détecte si une modification a eu lieu dans le dossier *toDo*, et envoie le chemin du fichier *video.mp4* à `extract.py`.

2.1.1 background.py

```
1  import extract  # For the image script
2
3  import logging  # To create easy log file
4  import time  # For the time.sleep
5  import threading # For
6  import os
7  import shutil #To move the folder
8  import pytesseract as pyt
9
10 # For the file detection, It doesn't work on 64-bit
11 from watchdog.observers import Observer
12 from watchdog.events import FileSystemEventHandler
13
14 # configuration for the logging function
15 logging.basicConfig(filename='app.log',
16                     filemode='a',# filemode='a' is for append
17                     format='%(asctime)s - %(message)s',
18                     datefmt='%d-%b-%y %H:%M:%S')
19
20 #class that wait for an event, when it get's an event it call the ↩
       Handler class and give him the event
21 class Watcher:
22     DIRECTORY_TO_WATCH = "../web/file/toDo"
23
24     def __init__(self):
25         self.observer = Observer()
26         pass
27
28     def run(self):
29         event_handler = Handler()
30         self.observer.schedule(event_handler,
31                                self.DIRECTORY_TO_WATCH,
32                                recursive=True)
33         self.observer.start()
34         try:
35             while True:
36                 time.sleep(5)
37         except:
38             self.observer.stop()
39             print("Error")
40
41 #called if an event occured
42 class Handler(FileSystemEventHandler):
43     @staticmethod
44     def on_any_event(event):
```

```
45
46          # global is used to tell that we will modifiy the global ↩
                variable
47          global folderName
48
49          # event_type can be : created, modified, deleted
50          # is_directory is used to tel if the event concern a ↩
                folder or not by a boolean
51          if event.event_type == 'created' and event.is_directory ↩
                == False:
52
53              # example of event.src_path '../web/file/toDo\
54              # 8675ab54b50cc355a5665dd33827806c14fbe71b\video.mp4'
55              path = event.src_path.split('/')
56              print(event.src_path)
57              path2 = path[3].split('\\')
58
59              # example of source '../web/file/toDo/
60              # 8675ab54b50cc355a5665dd33827806c14fbe71b'
61              source = path[0] + "/" + path[1] + "/" + path[2] + ↩
                    "/" + path2[0] + "/" + path2[1]
62
63              # example of destination '../web/file/done'
64              #destination is used when moving the folder at the end.
65              destination = path[0] + "/" + path[1] + "/" + path[2] ↩
                    + "/" + "done"
66
67              # test if the file is the video.mp4, and the folder ↩
                    is diffrent than the last one (to prevent the ↩
                    script from running twice on the same video)
68              if path2[-1] == 'video.mp4' and folderName != ↩
                    path2[-2]:
69                  folderName = path2[-2]
70                  print('event type: ' + event.event_type + ' and ↩
                        is a directory: ' + str(event.is_directory) + ↩
                        ' name is: ' + path2[-1] + ' source is :' + ↩
                        folderName)
71
72                  # prepare the function that will be called as a ↩
                        thread, and then start the thread
73                  thread = ↩
                        threading.Thread(target=extract.extractFrames(event.src_path, ↩
                        source))
74                  thread.start()
75
76                  # wait here for the end of the thread before ↩
                        continuing
77                  thread.join()
78                  logging.warning('The file at %s has been added' % ↩
                        event.src_path)
79
80
81                  # Try to move the folder now that the script is ↩
                        almost finished.
82                  try:
83                      shutil.move(source, destination)
```

```
84                      print('The folder as been moved.')
85                  except:
86                      pass
87                      print("An exception occurred, could not move ↩
                            the folder")
88
89
90
91
92  if __name__ == '__main__':
93      folderName = ''
94      scriptDirectory = os.path.dirname(os.path.realpath(__file__))
95      print('start: background script for Thermiscan project!')
96      w = Watcher()
97      w.run()
```

le fichier `extract.py` lit la vidéo dans le chemin spécifié par `background.py`, et il en extrait les images pour les envoyer à `image.py`.

2.1.2 extract.py

```
1  import image
2
3  # Importing all necessary libraries
4  import cv2 # For the text recognition
5  import os
6  import csv # To write the csv
7
8  # Read the video from specified path
9  path = "..//video//vidTest.mp4"
10 def extractFrames(path, folderSource):
11     print('start extract.py')
12     cam = cv2.VideoCapture(path)
13     try:
14         # creating a folder named data
15         if not os.path.exists('/data'):
16             os.makedirs('/data')
17
18     # if not created then raise error
19     except OSError:
20         print('Error: Creating directory of data')
21
22     # frame counter
23     currentframe = 0
24     print(cam)
25
26     while (True):
27
28         # reading from frame
29         ret, frame = cam.read()
30
31         if ret:
32             # if video is still left continue creating images
33             name = './data/frame' + str(currentframe) + '.jpg'
```

```
34
35              # writing the extracted images to be processed in ↩
                    image.py
36              cv2.imwrite(name, frame)
37              image.read_break_image(currentframe, frame, ↩
                    folderSource)
38              currentframe += 1
39          else:
40              break
41
42      # Release all space and windows once done
43      print('extract done')
44      cam.release()
45      cv2.destroyAllWindows()
```

le fichier `image.py` est appelé pour traiter une image et enregistrer ses valeurs dans un CSV.

2.1.3 image.py

```
1  import csv
2  import re
3  import os
4  import sys
5  import numpy as np
6  from PIL import Image
7  import cv2
8  import pytesseract as pyt
9  #uncomment the next line to execute on Windows
10 pyt.pytesseract.tesseract_cmd = r'C:\\Program ↩
       Files\\Tesseract-OCR\\Tesseract.exe'
11 regex = r"\d+\,\d+"
12
13 def read_break_image(frame, image, folderSource):
14
15     # Set minimum and max HSV values to display
16     lower = np.array([0, 0, 250])#0,0,0
17     upper = np.array([179, 70, 255])#179,70,255
18
19     # Create HSV Image and threshold into a range.
20     hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
21     mask = cv2.inRange(hsv, lower, upper)
22     output = cv2.bitwise_and(image,image, mask= mask)
23     img_to_read = cv2.bitwise_not(output)
24     img_to_read = img_to_read[0:110, 10:240]
25
26     cv2.imwrite('out.png', img_to_read)
27     text = pyt.image_to_string(img_to_read)
28     write_csv(text, frame, folderSource)
29
30
31 def write_csv(row, frame, folderSource):
32     with open(folderSource + '/value.csv', 'a', newline='') as ↩
           outfile:
```

```
33          writer = csv.writer(outfile,  delimiter=';',
34                              quotechar=' ', quoting = ↩
                                   csv.QUOTE_NONNUMERIC)
35          #data_writer.writerow(['Min', 'Moy', 'Max'])
36
37          matches = re.findall("\d+\,\d+", row)
38          if len(matches) == 3:
39              writer.writerow([frame, matches[2], matches[0], ↩
                    matches[1]])
40              #print(frame, 'OK')
41          else:
42              print('dead')
43              print(matches)
44
45  if __name__ == '__main__':
46      #print('image file.started.')
47      read_break_image("capture.png")
48      #print('End of image.py script')
```

3 Conclusion

3.1 Bilan

Dans l'ensemble, nous sommes plutôt satisfaits de notre projet. Nous avons atteint les buts les plus importants.

3.2 Problèmes rencontrés

3.2.1 Téléphone ne permettant pas la diffusion en direct

Le problème majeur rencontré dès le début du projet est l'indisponibilité de live streaming du Cat S60. Selon les pages qu'on parcourait, il était mentionné que la Cat S60 possédait une option de live streaming, ce qui n'est au final pas le cas, contrairement à la nouvelle génération du Cat, le S61. Étant donné que le cahier des charges était de faire un live directement via une application web, nous nous sommes retrouvé à devoir modifier le cahier des charges, qui est passé d'un live sur une application web à juste afficher une vidéo.

3.2.2 Affichage d'un graphique

Plusieurs problèmes se sont joint lors de la création du graphique. Premièrement, il nous fallait trouver un script JS permettant d'afficher un graphique à partir de valeur venant d'un fichier CSV. Lors de nos premiers tests, le problème principal était l'importation de nos valeurs dans les tableaux, les graphiques ne possédant pas de tableau souple au niveaux des données entrées. Nous avons essayé avec un graphique

venant de `www.amcharts.com` et un autre venant de `www.highcharts.com`. En fin de compte, Nous nous sommes occupé d'en faire un, en reprenant des parties de code provenant des anciens tests.

Table des figures