

FaceDetection

1. Introduzione

- Informazioni sul progetto
- Abstract
- Scopo

2. Analisi

- Analisi del dominio
- Analisi dei mezzi
- Analisi e specifica dei requisiti
- Use case
- Pianificazione

3. Progettazione

- Design dell'architettura del sistema
- Design dei dati e database

4. Implementazione

5. Test

- Protocollo di test
- Risultati test
- Mancanze/limitazioni conosciute

6. Consuntivo

7. Conclusioni

- Sviluppi futuri
- Considerazioni personali

8. Sitografia

9. Allegati

Introduzione

Informazioni sul progetto

- **Progetto svolto da:** Gionata Battaglioni, Lucas Previtali
- **Mandanti del progetto:** Luca Muggiasca
- **Docente Responsabile:** Luca Muggiasca
- **Scuola:** Arti e Mestieri Trevano
- **Sezione:** Informatica
- **Classe:** I3AA
- **Data d'inizio:** 16.03.2018
- **Termine della consegna:** 18.05.2018

Abstract

Lucas and I decided to take over the project that we had carried out in groups. Since the project is not finished we have chosen to resume it and put it in place. To do this we decided to restart the whole project from the beginning. It was not easy to redesign everything because initially this project was designed to be carried out by four people and we are two. So we had some problems with load balancing. The result was what I expected since the beginning, our approach worked very well except, as said before, we had to invest maybe too much time consulting with various guides.

Scopo

Lo scopo del progetto è quello di realizzare un sistema di riconoscimento facciale tramite una webcam. Questo sistema serve per quantificare il numero di persone che visitano uno stand di Espoprofessioni, visto che il progetto é stato ripreso per la modifica di eventuali errori. Lo scopo resta lo stesso ma non servirà più per Espoprofessioni ma per contare il numero di persone in base alle faccie riconosciute in generale. Appunto il sistema deve essere in grado di riconoscere le persone che entrano nel campo visivo della webcam.

Analisi

Analisi del dominio

Il progetto completo verrà presentato alla prossima edizione di Espoprofessioni, perciò sarà presentato principalmente a un pubblico di ragazzi e ragazze non per forza appassionati di informatica (in generale il gli “utenti” saranno variegati). Il pubblico presente sarà lì soprattutto per vedere diverse professioni e molti saranno studenti. Per capire il funzionamento del sistema non bisogna essere per forza esperti o appassionati di informatica.

Analisi e specifica dei requisiti

Il committente richiede una pagina web che effettua un riconoscimento facciale. Una volta che la faccia viene riconosciuta dalla pagina viene salvata all'interno di una variabile e a sua volta salvata in un database. In base al numero di persone riconosciute e al tempo che le persone rimangono ferme davanti alla web cam viene redatto un grafico. Mentre un secondo grafico viene redatto in base al numero di persone che sono state riconosciute dalla webcam e alla fascia oraria.

ID	REQ-001
Nome	Valutazione della libreria da utilizzare
Priorità	1
Versione	1.0
Note	
Sub-ID	Requisito
001	Valutare le varie librerie che permettono il tracking

ID	REQ-002
Nome	Creazione di un supporto grafico del programma
Priorità	1
Versione	1.0
Note	
Sub-ID	Requisito
001	Implementare una rappresentazione della webcam
002	Organizzare e realizzare una interfaccia grafica

ID	REQ-003
Nome	Creazione pagina Web per grafici
Priorità	1
Versione	1.0
Note	
Sub-ID	Requisito
001	Implementare la lettura dal DB
002	Creazione dei grafici: Numero di persone rilevate in ogni ora del giorno. Tempo medio di una persona di fronte all'obiettivo.
003	Utilizzare un form di log in per il REQ-005

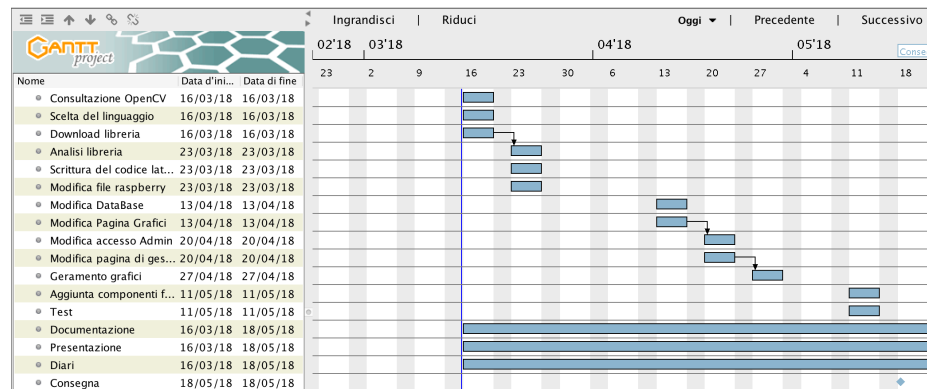
ID	REQ-004
Nome	Ricerca di nuovi volti
Priorità	1
Versione	1.0
Note	
Sub-ID	Requisito
001	Scegliere la libreria migliore per il tracking dei volti: openCV, tracking.js, emgu.cv
002	Implementare il codice per l'individuazione dei dati tramite libreria.
003	La webcam deve eseguire la ricerca di nuovi volti ogni 15 secondi e se rileva dei volti nelle coordinate vicine a quelle vecchie, non ne terrà conto.

ID	REQ-005
Nome	Salvataggio sul Database
Priorità	1
Versione	1.0
Note	
Sub-ID	Requisito
001	Allestire un database capace di contenere i dati
002	Il Database deve essere compatibile sia per le pagine web e sia con l'applicazione scritta in C#

ID	REQ-006
Nome	Utilizzo del prodotto su RaspBerry
Priorità	1
Versione	1.0
Note	
Sub-ID	Requisito
001	Allestire un webserver Linux su RaspBerry
002	Trasportare l'intero codice del progetto su RaspBerry

Pianificazione

Questo é il Gantt che abbiamo realizzato in base alla lista dei requisiti che abbiamo redatto e al tempo a disposizione.



Analisi dei mezzi

Come prodotti fisici abbiamo usato i seguenti:

Prodotto	Caratteristiche
Raspberry Pi3	so. raspbian 9.0
tastiera K200	Logitech
Mouse	Lenovo
Monitor HDMI	Asus
Webcam (C270)	Logitech

Il Raspberry lo abbiamo utilizzato unicamente come mezzo di supporto del programma. Appunto il programma vero e proprio é stato realizzato una libreria per C# chiamata OpenCV.

Pacchetto	Versione
OpenCV	3.4.1
Microsoft Visual Studio 2017 Enterprise	15.7

Analisi dei costi

Costo per persona:

È stimato che un apprendista al nostro stesso livello guadagni circa 80 franchi all'ora.

Costo per ora:	Ore	Totale
60	72	4320 fr.

Costo totale (dipendenti):

Essendo quattro persone a lavorare in questo progetto, i costi vanno moltiplicati.

Costo per ora	Ore	persone	Totale
60	48	2	8640 fr.

Costo totale:

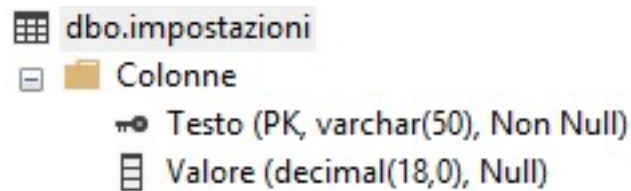
Facendo una somma dei vari totali e aggiungendo il costo di 29 fr. per la webcam arriviamo al costo totale finale di questo progetto cioè 8669 fr.





Progettazione

Design dei dati e database

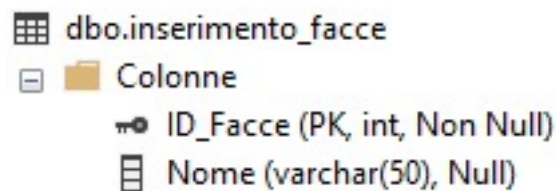
Il database che abbiamo realizzato non é molto complesso. Presenta soltanto le tabelle necessarie al corretto funzionamento. Che sono:





Impostazioni



 dbo.impostazioni
 Colonne
 Testo (PK, varchar(50), Non Null)
 Valore (decimal(18,0), Null)

Inserimento delle facce



 dbo.inserimento_facce
 Colonne
 ID_Facce (PK, int, Non Null)
 Nome (varchar(50), Null)

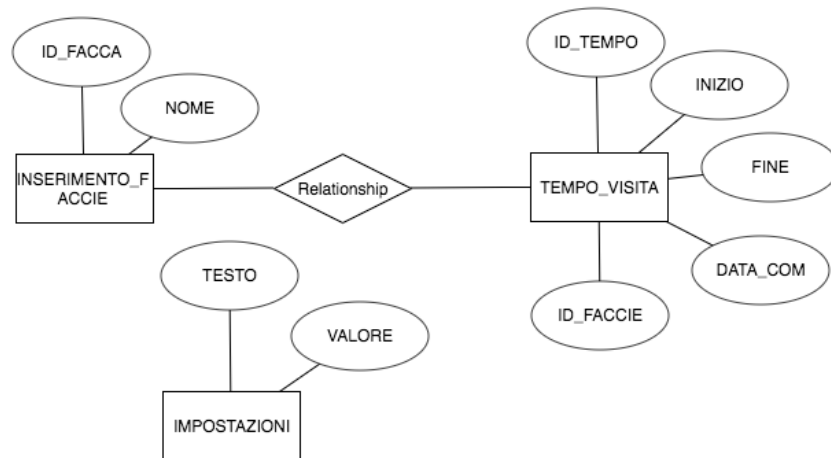
Tempo della visita



 dbo.tempo_visita
 Colonne
 ID_TempVis (PK, FK, int, Non Null)
 Inizio (time(7), Non Null)
 Fine (time(7), Non Null)
 DataCom (date, Non Null)
 ID_Facce (int, Non Null)

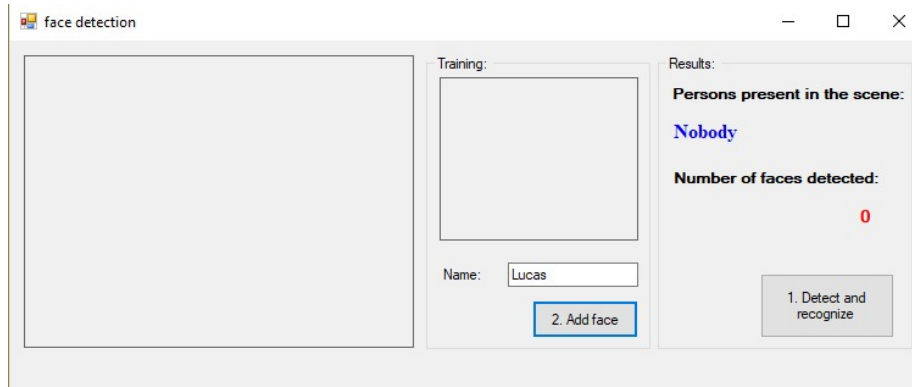
Schema E-R, schema logico e descrizione.

Questo é il diagramma ER del database generato per consentire lo scambio dei dati tramite la pagina web dei grafici e l'applicazione in C#.

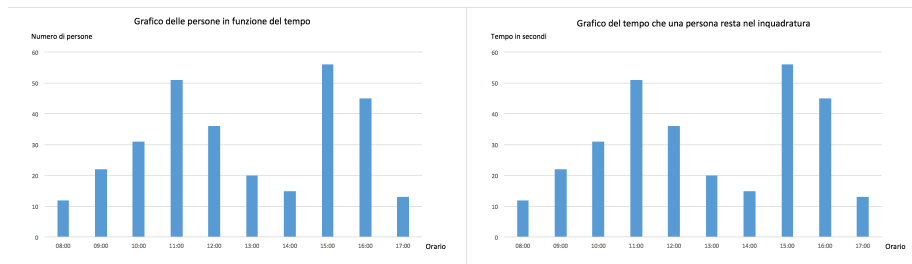


Design delle interfacce

Prima di iniziare a scrivere il codice abbiamo pensato a un approccio molto simile a quello del progetto precedente ma con dei cambiamenti. Avendo scelto un'altra libreria abbiamo cambiato anche il linguaggio di scrittura del codice. Quindi l'interfaccia grafica si presenta così:



Per quello che concerne la pagina dei grafici abbiamo deciso di tenere la stessa struttura ma di cambiare un po' la grafica, ponendo i due grafici come se fossero un confronto quindi uno da parte all'altro e non più uno sopra e uno sotto.



Implementazione

Supporto

Inizialmente abbiamo optato per utilizzare un raspberry dove avremmo caricato tutti i file inerenti al progetto. Così abbiamo recuperato un raspberry e lo abbiamo scaricato i programmi necessari per far sì che il raspberry sia in grado di eseguire i file .exe, abbiamo avuto bisogno di fare questa scelta perché aprire scaricare visual studio sul raspberry risultava troppo pesante e lo avrebbe rallentato troppo. In questo modo siamo in grado di avere una piattaforma fissa, cioè avere una postazione fissa e in caso facilmente trasferibile.

Applicazione di riconoscimento facciale

L'applicazione è la parte principale del progetto. Il programma una volta avviato è in grado di riconoscere le facce che trova davanti alla webcam, inoltre è in grado di salvare le facce scattando una foto. Una volta che la foto è stata scattata è possibile scrivere il proprio nome in un campo di testo. Una volta che il nome è stato scritto il nostro programma riconoscerà sempre la persona salvata scrivendo il nome sopra il rettangolo di riconoscimento.

Ecco in poche parole cosa è in grado di fare la nostra applicazione:

1. Id univoco per ogni persona rilevata dalla WebCam.
2. Orario nel quale una persona viene specchiata.
3. Orario di fine del tracking
4. Giorno in cui è stato eseguito il tracking
5. Salvataggio della persona sotto forma di fotografia e indicizzazione della persona tramite al nome

L'applicazione è stata sviluppata usando il linguaggio c# con Visual Studio 17, quindi il prodotto è un file eseguibile .exe. Per poter svolgere il riconoscimento facciale abbiamo usato la libreria EmguCV, che è un'estensione di OpenCV in grado di lavorare con c#. Questo rende più semplice almeno la parte riguardante il riconoscere una faccia (o anche gli occhi) rispetto a qualsiasi altro oggetto. Il programma non è solo in grado di riconoscere le facce ma anche di associarle a un nome, quindi di distinguere una persona da un'altra. Questo è fatto salvando le varie facce in formato bitmap e salvando i nomi in un documento di testo. Le facce vengono salvate con il nome "face" seguito dal numero della faccia che rappresenta (la prima faccia viene salvata con il nome **face1** la seconda con **face2** e così via). I nomi vengono salvati separati dal carattere %.

All'avvio del programma per far partire il riconoscimento facciale è necessario premere il pulsante 1. Dopo averlo fatto verrà avviata la webcam e quindi il frame. A questo punto l'applicazione cercherà se sono già presenti delle facce

registrate per poter eseguire il riconoscimento anche di facce salvate in precedenza.

```
// carico il file haarcascade che svolge il face detection
face = new HaarCascade("haarcascade_frontalface_default.xml");
try
{
    // carico eventuali facce già riconosciute in precedenza
    string Labelsinfo = File.ReadAllText(Application.StartupPath +
        "/TrainedFaces/TrainedLabels.txt");
    // i nomi sono separati dal carattere "%"
    string[] Labels = Labelsinfo.Split('%');
    // il primo valore contenuto nel file è il numero di facce registrate
    NumLabels = Convert.ToInt16(Labels[0]);
}
```

Per far capire all'utente finale meglio il funzionamento dell'applicazione, attorno ad ogni faccia viene disegnato un rettangolo rosso.

```
currentFrame = grabber.QueryFrame().Resize(320, 240,
    Emgu.CV.CvEnum.INTER.CV_INTER_CUBIC);
[...]
foreach (MCvAvgComp f in facesDetected[0])
{
    result = currentFrame.Copy(f.rect)
        .Convert<Gray, byte>().Resize(100, 100,

    Emgu.CV.CvEnum.INTER.CV_INTER_CUBIC);
    //disegno del rettangolo attorno alle facce
    currentFrame.Draw(f.rect, new Bgr(Color.Red), 2);
}
```

Inoltre sopra ad ogni rettangolo viene stampato il nome della persona.

```
if (trainingImages.ToArray().Length != 0)
{
    EigenObjectRecognizer recognizer = new EigenObjectRecognizer(
        trainingImages.ToArray(),
        labels.ToArray(),
        3000,
        ref termCrit);
    name = recognizer.Recognize(result);
    // scrive il nome di ogni faccia riconosciuta
    currentFrame.Draw(name, ref font, new Point(f.rect.X - 2,
        f.rect.Y - 2),
        new Bgr(Color.LightGreen));
}
```

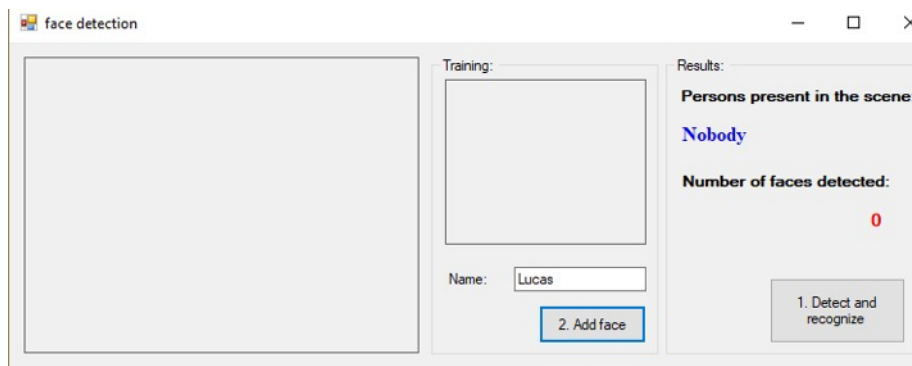
Un'ulteriore funzionalità dell'applicazione è quella di interfacciarsi con un database (con sql server), inserendo per ogni faccia alcuni dati interessanti:

1. Nella tabella inserimento_facce viene inserito un id incrementale e il nome delle varie facce salvate
2. Nella tabella tempo_visita vengono inseriti per ogni faccia un id, e alcuni dati relativi agli orari e al tempo di visita, che saranno poi utilizzati dalla pagina dei grafici

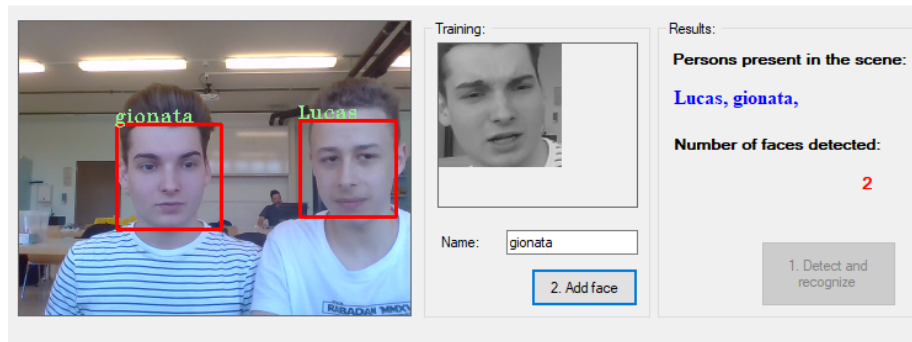
```
connectionString=
"Server=LUCASPC;Database=facedetection;Trusted_Connection=True";
using (SqlConnection cnn = new SqlConnection(connectionString))
{
    sql = "insert into inserimento_facce ([ID_Facce], [Nome])
    values (@first, @last)";

    sql1 = "select count(*) from inserimento_facce";
    SqlCommand cnt = new SqlCommand(sql1, cnn);
    cnn.Open();
    using (SqlCommand cmd = new SqlCommand(sql, cnn))
    {
        Int32 count = Convert.ToInt32(cnt.ExecuteScalar());
        cmd.Parameters.AddWithValue("@first", count);
        cmd.Parameters.AddWithValue("@last", textBox1.Text);
        int row = cmd.ExecuteNonQuery();
    }
}
```

Questo è l'interfaccia dell'applicazione:



mentre questa è l'interfaccia dell'applicazione già avviata e mentre sta processando:



Creazione pagina Grafici

La pagina grafici esegue una continua sincronizzazione sul DataBase affinché tutti i dati siano sempre aggiornati. La sua funzione è quella di mostrare 2 grafici: 1. Mostrare la quantità di persone specchiati nella WebCam per ogni fascia oraria. 2. Mostrare il tempo medio di tracking per ogni fascia oraria.

Tramite una ricerca su Internet siamo venuti a conoscenza di una libreria specializzata nel display di grafici non troppo dispendiosa per quanto riguarda le nostre singole conoscenze personali: Chart.js.

Per creare i grafici mi sono dapprima connesso al database ed ho preso ed inserito dentro degli array i dati della data attuale, dell'inizio e della fine di sessione.

```
$servername = "(local)";
$username = array("Database"=>"facedetection");
$conn = sqlsrv_connect($servername, $username);

if( $conn === false ){
    echo "Errore di connessione.\n";
    die( print_r( sqlsrv_errors(), true));
}

$sql = "SELECT Inizio, Fine FROM tempo_visita";
$result = sqlsrv_query($conn, $sql);
if ($result === FALSE) {
    die( print_r( sqlsrv_errors(), true) );
    echo "Errore di connessione";
}
```

In seguito ho fatto un counter che permettesse di calcolare quante persone utilizzano l'applicazione in vari fasce d'orario. Inoltre ho utilizzato una tecnica simile anche per calcolare la media della durata di sessione per ogni utente. Dopo aver eseguito tutti i calcoli ed aver inserito i risultati negli array, ho creato i

grafici ed inserito i dati. Per inserirli ho utilizzato più volte una tecnica di programmazione che permette l'utilizzo delle variabili presenti nella porzione di codice php e l'utilizzo diretto nella parte javascript, e l'ho inserita all'interno della parte "data" della creazione del grafico.

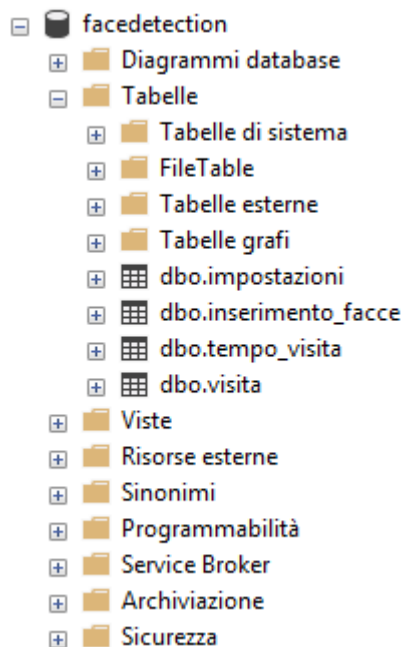
```
var dataNumeroVisite = {  
  labels: ["09:00", "10:00", "11:00", "12:00", "13:00", "14:00", "15:00",  
    "16:00", "17:00", "18:00",  
    "19:00", "20:00", "21:00", "22:00"],  
  
  datasets: [{  
    label: "Numero di visite",  
    backgroundColor: "rgba(255,99,132,0.2)",  
    borderColor: "rgba(255,99,132,1)",  
    borderWidth: 2,  
    hoverBackgroundColor: "rgba(255,99,132,0.4)",  
    hoverBorderColor: "rgba(255,99,132,1)",  
    data: [<?php echo $countUsers[0]; ?>,  
      <?php echo $countUsers[1]; ?>,  
      <?php echo $countUsers[2]; ?>,  
      <?php echo $countUsers[3]; ?>,  
      <?php echo $countUsers[4]; ?>,  
      <?php echo $countUsers[5]; ?>,  
      <?php echo $countUsers[6]; ?>,  
      <?php echo $countUsers[7]; ?>,  
      <?php echo $countUsers[8]; ?>,  
      <?php echo $countUsers[9]; ?>,  
      <?php echo $countUsers[10]; ?>,  
      <?php echo $countUsers[11]; ?>,  
      <?php echo $countUsers[12]; ?>,0]  
    }]  
};
```

Questo è il risultato ottenuto:



Creazione Database

Per far sì che le pagine comunicano e si scambino i dati tra di loro è stato necessario creare un database. Per crearlo abbiamo usato Microsoft SQL Server Managment Studio versione 17. Il database in questo modo sarà collegato a un server e quindi accessibile anche da remoto. Ecco come si presenta il database su MSSMS:



La tabella impostazioni si presenta così:

LUCASPC.facedetec...- dbo.impostazioni			
	Nome colonna	Tipo di dati	Consenti val...
?	Testo	varchar(50)	<input type="checkbox"/>
	Valore	decimal(18, 0)	<input checked="" type="checkbox"/>

Mentre le tabelle tempo_visita e inserimento_facce si presentano in questo modo:

LUCASPC.facedetec...- dbo.tempo_visita ▢ ✕			
	Nome colonna	Tipo di dati	Consenti val.
🔑	ID_TempVis	int	<input type="checkbox"/>
	Inizio	time(7)	<input type="checkbox"/>
	Fine	time(7)	<input type="checkbox"/>
	DataCom	date	<input type="checkbox"/>
	ID_Facce	int	<input type="checkbox"/>

LUCASPC.facedetec...- dbo.tempo_visita ▢ ✕			
	Nome colonna	Tipo di dati	Consenti val.
🔑	ID_TempVis	int	<input type="checkbox"/>
	Inizio	time(7)	<input type="checkbox"/>
	Fine	time(7)	<input type="checkbox"/>
	DataCom	date	<input type="checkbox"/>
	ID_Facce	int	<input type="checkbox"/>

Ora abbiamo un database in grado di comunicare i propri dati.

Test

Protocollo di test

Le tabelle sottostanti rappresentano i test che abbiamo svolto in base ai requisiti che abbiamo scelto e creato.

Test Case	TC-001
Nome	Creazione macchina virtuale
Riferimento	REQ-001
Descrizione	Creazione macchina virtuale per gestire le cartelle su raspberry
Prerequisiti	
Procedura	- Installare un programma per gestire le macchine virtuali, noi abbiamo usato VirtualBox. - Creare una macchina virtuale Windows. - Installare microsoft VisualStudio 2017.
Risultati attesi	Avere un ambiente di sviluppo pronto per iniziare la scrittura del codice

Test Case	TC-002
Nome	Supporto grafico dell'applicazione
Riferimento	REQ-002
Descrizione	Gestire il riconoscimento facciale
Prerequisiti	-
Procedura	- Scaricare la libreria OpenCV - Modificare la libreria con il linguaggio C#.
Risultati attesi	Avere un'applicazione che è in grado di riconoscere le facce e salvarle su un database

Test Case	TC-003
Nome	Pagina Web per grafici
Riferimento	REQ-003
Descrizione	Gestire i dati mandati dalla pagina web della webcam tramite dei grafici
Prerequisiti	-
Procedura	- Realizzare dei grafici tramite la libreria chart.js - Prendere i dati dal database e inserirli all'interno dei grafici.
Risultati attesi	I grafici vengono mostrati correttamente in base ai dati presi dal database.

Test Case	TC-004
Nome	Ricerca di nuovi volti
Riferimento	REQ-004
Descrizione	I volti vengono trovati dalla pagina della Webcam
Prerequisiti	Download libreria tracking.js
Procedura	- Aver installato la libreria tracking.js - Gestire il riconoscimento facciale tramite JavaScript
Risultati attesi	La pagina della Webcam è in grado di riconoscere i volti

Test Case	TC-005
Nome	Database
Riferimento	REQ-006
Descrizione	Realizzazione di un database che contiene i dati raccolti dalla pagina web della Webcam
Prerequisiti	L'applicazione del rilevamento facciale deve essere conclusa e funzionante

Test Case	TC-005
Procedura	- Scaricare un programma per creare il database, noi abbiamo utilizzato Heidi - Creare il database con gli stessi parametri della libreria presa e del codice scritto per l'applicazione della Webcam.
Risultati attesi	Il database riesce a prendere i dati delle pagine prescritte

Test Case	TC-006
Nome	Salvataggio delle persone sul DB
Riferimento	REQ-004
Descrizione	Questo test serve per verificare il corretto funzionamento dell'immissione dei dati all'interno del database
Prerequisiti	aver creato il database
Procedura	- Collegare il database alle pagine tramite php
Risultati attesi	I dati presi dalla webcam sono presenti all'interno del database

Test Case	TC-007
Nome	Utilizzo del prodotto su RaspBerry
Riferimento	REQ-005
Descrizione	Installazione del sistema operativo su Raspberry e fare in modo che il raspberry sia in grado di eseguire i file .exe
Prerequisiti	avere un Raspberry
Procedura	- Caricare l'immagine del sistema operativo sul raspberry e installare mono 3.7
Risultati attesi	Raspberry è in grado di eseguire file di estensione .exe.

Risultati test

I risultati dei test non sono male ma neanche eccellenti. I test TC-001, TC-002, TC-003, TC-004, TC-005, TC-006 sono passati. Il TC-007 non è passato. Si è ripresentato il problema che abbiamo avuto la scorsa volta. Il Raspberry non è abbastanza prestante per riuscire a reggere il nostro programma. Noi avevamo pensato che il problema fosse il fatto che il linguaggio utilizzato in precedenza potesse essere troppo pesante, quindi lo abbiamo cambiato. Ma il problema c'era ancora. Dopo vari test sul Raspberry abbiamo appurato che non è in grado di reggere un WebServer e un Applicazione scritta in C# allo stesso momento. Come soluzione abbiamo deciso di mantenere il progetto su un altro

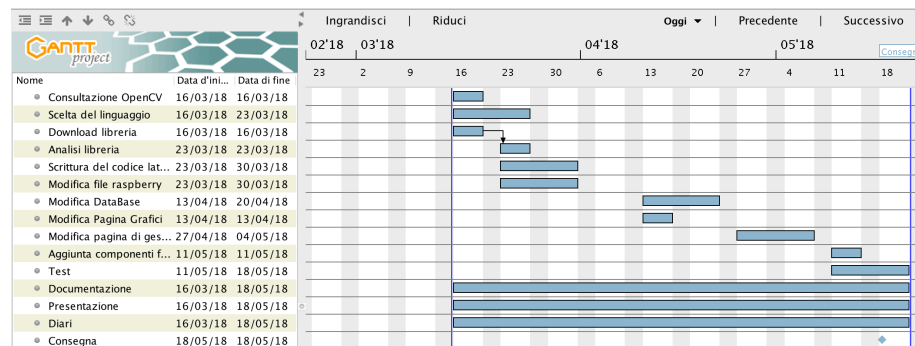
computer esterno.

Mancanze e limitazioni conosciute

Per noi é stata difficile la partenza, cioè suddividerci il lavoro e imparare a lavorare in coppia, é stato difficile perche durante l'anno abbiamo svolto lo stesso progetto ma con un gruppo di quattro persone mentre ora essendo stati soltanto in due, abbiamo fatto un po fatica a bilanciarci il carico di lavoro. Una volta capito il vero funzionamento del lavoro di squadra e una volta che abbiamo suddiviso i compiti siamo riusciti a arrivare ad avere un vero e proprio team organizzato. Per quanto riguarda le competenze informatiche abbiamo avuto qualche difficolta utilizzando di programmi o linguaggi che non abbiamo mai usato. Come per esempio raspberry, abbiamo dovuto installare il suo sistema operativo (raspbian) e trovare un modo per poter avviare i programmi con estensione exe.

Consuntivo

Ecco come sono andate le ore di lavoro, in base a quelle che abbiamo programmato prima dell'inizio del progetto.



Conclusioni

La soluzione che abbiamo portato ci soddisfa, é sicuramente molto più soddisfacente che la prima versione che abbiamo portato. Il nostro programma avrà sicuramente un impatto positivo con le persone che lo proveranno, porterà molto divertimento. Noi non pensiamo che il nostro programma cambierà il mondo ma siamo sicuri che porterà una piccola svolta, sicuramente più per noi stessi che per gli altri. Possiamo definire questo progetto un successo importante, più che successo questo progetto ci porta un forte orgoglio personale soprattutto perchè siamo riusciti a svolgere un applicazione molto più prestante di quella precedentemente realizzata con la metà dei collaboratori. Inoltre questo lavoro

é una grande aggiunta alla nostra crescita professionale, ci ha dato molto sia come competenze lavorative che come competenze sociali.

Sviluppi futuri

Come miglioria potrebbe essere implementato in una scala molto più grande rispetto che un semplice schermo con una webcam. Sarebbe bello poter collegare il nostro progetto su delle videocamere reali in modo da riuscire a riconoscere una quantità maggiore di volti. Mentre come sviluppo futuro vorremmo riuscire integrare la nostra applicazione nel mondo degli smartphone.

Considerazioni personali

Con questo progetto abbiamo imparato cosa vuol dire lavoro di coppia, di quanto esso sia estremamente importante e di come con un collaboratore sia più facile e motivante lavorare.

Sitografia

- <https://opencv.org/>, *OpenCv La libreria per implementare il riconoscimento facciale.
- <https://www.mono-project.com/docs/getting-started/install/linux/>, *Manuale di installazione del programma mono, grazie a questo programma il raspberry é in grado di avviare i file .exe.
- <https://www.raspberrypi.org/downloads/raspbian/>, *Sistema operativo Raspbian Abbiamo scaricato il sistema operativo direttamente dal sito del produttore di raspberry.
- <http://www.vemp.org/raspberrypi/preparare-una-card-sd-con-raspbian/>, *Programma per caricare il .img di raspbian su raspberry.
- <http://www.chartjs.org/>, *pagina per i grafici

Allegati

Elenco degli allegati:

- Diari di lavoro
- Guida utente / Manuale di utilizzo
- Guida di configurazione di raspberry