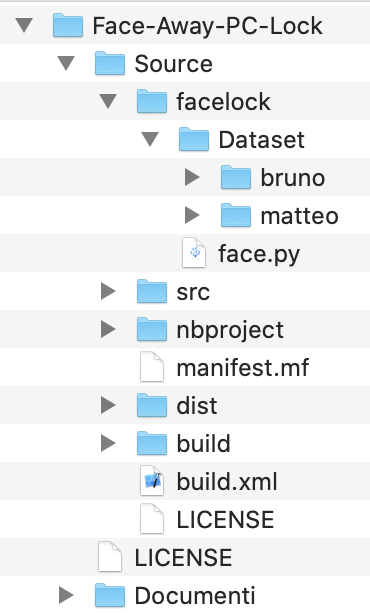
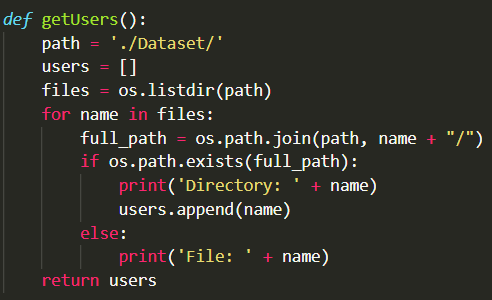
# Lavori svolti

|  |  |  |  |
| --- | --- | --- | --- |
| **Orario** | **lavoro svolto** | **persone** | **ID Lavoro** |
| **8:20 – 9:00** | Definito standard per lo sviluppo | Matteo, Jonas | 1 |
| **9:00 – 10:30** | Sviluppata funzione per ricevere il numero di utenti | Matteo | 2 |
| **8:20 - 11:35** | Aggiunta di utenti tramite l’interfaccia grafica | Luca, Bruno | 3 |

# Lavori

1. Abbiamo deciso di sviluppare le cartelle del programma nella seguente maniera:  
   L’utilizzo delle seguenti directory è:
   * **Face-Away-PC-Lock:** Directory principale del progetto, che contiene tutto.
   * **Documenti:** Contiene i documenti del progetto, tra cui i requisiti, i diari e la documentazione.
   * **Source:** Programma effettivo.
   * **Facelock:** Contiene tutti gli algoritmi in python per riconoscere i volti.
   * **Dataset:** set di immagini da paragonare a quello che vede la webcam, dentro a questa directory ci saranno delle cartelle con il nome del profilo da controllare.
   * **Resto:** Il resto delle cartelle (in Source) non sono altro che cartelle del progetto NetBeans per l’interfaccia grafica.
2. È stata sviluppata la funzione *getUser*che si occupa di ritornare il numero di utenti basandosi sul numero di cartelle all’interno della cartella Dataset. Per essere sicuro che il numero di utenti sia corretto non si basa semplicemente sul numero di file ma sul numero di directory, di cui verranno poi presi i nomi e ritornati sotto forma di lista. Il procedimento esatto che esegue è il medesimo:
   * Prende la lista di tutto quello che è contenuto all’interno della cartella Dataset (che conterrà le immagini degli utenti).
   * Grazie ad un join, metodo che consente di aggiungere elementi ad una path, ottiene la path completa (ovvero quella assoluta) del elemento corrente.
   * Stampa a terminale se si tratta di una directory o di un file (solo debug).
   * Se si tratta di una directory aggiunge il nome della directory, che corrisponde a quello dell’utente, alla lista di utenti che l’algoritmo effettivo dovrà poi andare a controllare.

Gestire le path in python ci ha richiesto di importare *os*, un modulo che si occupa di gestire in maniera molto comoda i file con le relative path, dando a disposizione metodi molto utili.

1. È stata inserita la possibilità di aggiungere utenti tramite la GUI, i quali necessitano un nome ed una faccia registrata. Sono stati eseguiti i seguenti controlli:
   * Trim del nome
   * Controllo della presenza sia del nome che della faccia registrata, per poter registrare l’utente.

# Problemi riscontrati

2.   
  
  
  
  
Abbiamo riscontrato un problema con il modulo os, più specificatamente all’inizio utilizzavamo il metodo *isdir().* Quando provavamo ad utilizzarlo funzionava unicamente se il file contenente l’algoritmo era situato all’interno della directory da controllare, per ovviare ho cercato una soluzione alternativa e sono riuscito a trovare *exists()*. Lo scopo di questo metodo è dire se una directory (che gli passiamo tramite parametro) esiste o meno, quindi prendendo il percorso assoluto dell’elemento da verificare e passandoglielo possiamo sapere se si tratta di una directory o di un file.

# Programma di massima per la prossima giornata di lavoro

# Punto rispetto alla pianifica