Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea in Informatica

PROGETTAZIONE ED IMPLEMENTAZIONE DI UN SISTEMA DI AUTORIZZAZIONE ELETTRONICO PER IL TRASPORTO AUTO DI PERSONE DISABILI CON VALIDAZIONE BIOMETRICA

Relatore:
Chiar.mo Prof.
Luciano Bononi

Presentata da:
Lisa Cesario

II Sessione
Anno Accademico 2014-2015

Indice

Introduzione

Notizie riguardanti scandali relativi al utilizzo inappropriato di contrassegni per disabili sono all'ordine del giorno, a causa di quegli individui che si prodigano a falsificare contrassegni oppure ad utilizzarli anche in mancanza del disabile, eventualmente anche successivamente al decesso del medesimo. Tutto questo va inevitabilmente a discapito di tutti coloro che hanno reale diritto e necessità di usufruire delle agevolazioni.

Lo scopo di questa tesi è quindi quello di illustrare un possibile sistema per contrastare e possibilmente debellare questo malcostume diffusissimo in Italia. La proposta è quella di dematerializzare il pass cartaceo sostituendolo con un equivalente elettronico, temporaneo e associato non più ad una targa, ma all'individuo stesso. Per farlo si ricorrerà all'uso di tecniche di autenticazione attraverso sistemi biometrici, quali il riconoscimento facciale, vocale, di espressioni facciali e gestures.

Capitolo 1

Lo stato dell'arte

1.1 Il contrassegno per disabili

In seguito all'urbanizzazione di massa che ha coinvolto il mondo e l'Italia negli ultimi decenni, destreggiarsi nel traffico cittadino è diventata un'impresa ardua per tutti. A complicare le cose si aggiungono le continue modifiche alla viabilità, tra le quali spiccano l'inserimento di zone a traffico limitato e aree pedonali e l'inadeguato numero di parcheggi, non congruo in relazione all'ammontare di veicoli in circolazione. É questa la realtà con cui si confrontano ogni giorno tutti gli automobilisti che vivono nei grandi centri urbani. Tutto questo è aggravato dal fatto che in molte famiglie sono presenti uno o più soggetti con disabilità, motorie e non, che in un contesto così caotico e disordinato rischiano di rimanere tagliati fuori dalla società. Per venire incontro a queste esigenze sono state ideate una serie di norme *ad hoc* volte a incentivare e tutelare la mobilità di questa categoria di persone. In particolare l'articolo 381 comma 2 del decreto del Presidente della Repubblica 16 dicembre 1992, n. 495, recita:

> "Per la circolazione e la sosta dei veicoli a servizio delle persone invalide con capacita di deambulazione impedita, o sensibilmente ridotta, il comune rilascia apposita autorizzazione in deroga, previo specifico accertamento sanitario."

Questo significa che, una volta accertata l'effettiva disabilità di una persona, il Comune ove essa risiede è tenuto a rilasciare un documento che attesti l'effettivo

status di disabile e con esso il diritto ad usufruire delle agevolazioni stabilite dall'articolo. Questo documento è rilasciato sotto forma di *contrassegno di parcheggio per disabili*: si tratta di un cartoncino blu, fronte e retro, da esporre sul parabrezza del veicolo ospitante il soggetto svantaggiato. In particolare, il lato rivolto verso l'esterno dell'abitacolo mostra la data di scadenza, un numero identificativo e l'ente che lo ha rilasciato; all'interno si trovano tutte le generalità del titolare del contrassegno. É inoltre presente e ben visibile il "simbolo dell'accessibilità" - visibile nella figura 1 -, ovvero l'icona bianca e blu raffigurante un uomo sulla sedia a rotelle. Il decreto continua sottolineando che il contrassegno è legato al singolo individuo e non al veicolo su cui viaggia ed ha valore su tutto il territorio nazionale ed europeo. Questo implica che il contrassegno può essere utilizzato su qualsiasi vettura a prescindere dal fatto che essa sia o meno di proprietà del titolare del contrassegno. Ciò che conta è la presenza del disabile a bordo, alla guida oppure accompagnato da terzi.

Figura 1.1: Fac-Simile di un contrassegno europeo

1.2 La richiesta del contrassegno

Per essere idonei al rilascio del contrassegno per disabili è necessario richiedere all'Ufficio di Medicina Legale dell'Azienda Sanitaria Locale del Comune di residenza una certificazione medica che attesti cecità oppure la capacità di deambulazione impedita o sensibilmente ridotta. Il certificato può essere rilasciato anche a persone con invalidità temporanea, purché sia specificatamente espressa la presumibile durata della condizione di disabilità. Una volta ottenuto il certificato è necessario presentare una richiesta al Sindaco del Comune di residenza allegando tutta la documentazione medica. Ottenuto il contrassegno esso ha validità di cinque anni nel caso di invalidità permanente, mentre nel caso di invalidità temporanea farà fede la data riportata nella documentazione medica. In entrambi i casi è possibile rinnovare il contrassegno ripetendo tutti i passaggi necessari per il primo rilascio.

1.3 I vantaggi del pass

Una volta stabilita l'esistenza di una classe di persone che necessita di particolari esigenze in ambito di mobilità urbana, si passa quindi ad analizzare nel dettaglio tutti i vantaggi che vengono acquisiti da coloro che sono stati giudicati idonei al possesso di un contrassegno per disabili.

I veicoli al servizio della persona disabile possono circolare:

- nelle zone a traffico limitato (ZTL) quando è consentito l'accesso anche ad una sola categoria di veicoli adibiti a servizi di trasporto e pubblica utilità;
- nelle zone a traffico controllato (ZTC);
- nelle corsie preferenziali riservate a mezzi di trasporto pubblico e taxi;
- nelle aree pedonali urbane (APU) quando è autorizzato l'accesso ad una sola categoria di veicoli adibiti a servizi di trasporto e di pubblica utilità;

Il contrassegno consente ai veicoli al servizio della persona disabile di sostare:

- negli appositi spazi riservati nei parcheggi pubblici;
- illimitatamente nelle aree di parcheggio regolate da disco orario;

- nei parcheggi a pagamento gratuitamente secondo le norme stabilite dal Comune;

- nelle zone a traffico limitato, a traffico controllato e nelle aree pedonali urbane;

- nelle zone di divieto di sosta purché il veicolo non intralci la circolazione;

Tuttavia il contrassegno disabili non autorizza la sosta in luoghi in cui le principali norme di comportamento lo vietano, ovvero quando sono di intralcio o pericolo per la circolazione. Per chiarire il concetto, il veicolo marchiato di contrassegno non può parcheggiare in aree riservate *ad personam*, ovvero quelle aree riservate ad uno specifico soggetto disabile e nelle zone dove vige il divieto di sosta con rimozione forzata e il divieto di fermata.

1.4 Abusi del contrassegno

Non è difficile immaginare che i benefici messi a disposizione dal contrassegno disabili facciano gola a tutti i cittadini. Per questo non sono rari gli stratagemmi utilizzati per cercare di entrare in possesso di uno o più pass. In seguito riportiamo diverse situazioni verificatesi negli ultimi anni.

Usare il contrassegno di un defunto Analizziamo il caso in cui un cittadino abbia un parente disabile titolare di un regolare contrassegno per invalidi non temporaneo, valido quindi per cinque anni. Se in questo lasso di tempo il titolare del pass dovesse decedere, le targhe dei veicoli associati al contrassegno verrebbero eliminate da qualsiasi banca dati elettronica, ma nessuno provvederebbe a ritirare fisicamente il contrassegno. Questo vuol dire che i veicoli in questione non potrebbero varcare la soglia di una zona a traffico limitato regolata da Sirio, il vigile elettronico di Bologna, senza incorrere in una sanzione amministrativa; tuttavia nelle ore in cui Sirio è spento e nelle zone al di fuori della ZTL solo un controllo da parte dei vigili urbani potrebbe portare alla revoca del contrassegno.

Falsificare il pass disabili Sebbene falsificare il contrassegno disabili significhi incorrere nel reato di contraffazione e uso di atto falso, sono tutt'altro che rari i casi di pass fotocopiati ed esposti sul parabrezza della vettura. Ne sono un caso Caserta, Napoli e Verona, la cui Polizia Municipale ha persino allestito una pagina sul sito web volta ad disincentivare simili comportamenti.

Quello che viene alla luce da questi esempi è l'estrema facilità con la quale si riescono ad utilizzare i pass in maniera illecita, in quanto:

- si tratta di un oggetto cartaceo, fotocopiabile, duplicabile e falsificabile;
- può essere associato a più di una targa;
- non esiste nessuno strumento automatizzato in grado di verificare se il soggetto disabile è realmente a bordo del veicolo.

Nel corso dell'elaborato si provvederà a proporre un nuovo sistema di rilascio del contrassegno invalidi in modo da ovviare a questi inconvenienti. Tuttavia, prima di addentrarsi nel vivo della progettazione del sistema proposto, è bene soffermarsi un attimo sull'ultimo punto dell'elenco sovrastante, in quanto punto chiave della proposta.

1.5 Sistemi biometrici

Il fatto di pensare al contrassegno invalidi come un oggetto strettamente legato alla persona che soffre di disabilità, suggerisce in qualche modo la necessità di verificare che questa persona autentichi la sua presenza all'interno dell'autovettura. Normalmente nel campo della sicurezza informatica esistono tre tipi di autenticazione, in relazione a :

- *qualcosa che si conosce*, come una password o un PIN;
- *qualcosa che si possiede*, come un oggetto fisico, una carta magnetica o un token;
- *qualcosa che si è*, come l'impronta digitale, il volto, la voce.

La password e l'oggetto fisico non sono molto differenti dal contrassegno disabili già in uso. Banalmente quindi bisogna cercare la soluzione del problema altrove, ovvero ricorrere ad utilizzare qualcosa che appartiene univocamente all'individuo che vuole autenticarsi, ovvero se stesso. Il sistema biometrico è in assoluto il più sicuro strumento di autenticazione, in quanto non può essere rubato, dimenticato o duplicato in nessun modo. Ecco di seguito elencati i principali elementi usati per l'autenticazione biometrica.

L'impronta digitale L'impronta digitale è composta dal quel pattern di linee situate sulla punta di un dito. Esistono diversi approcci per la verifica dell'impronta digitale: alcuni emulano il tradizionale metodo di corrispondenza utilizzato dalla polizia, altri si basano sul riconoscimento di tutte e cinque le dita di una mano. Esistono una grande varietà di dispositivi atti a riconoscere le impronte digitali e il costo non è dei più alti.

La firma Il modo in cui un utente scrive la sua firma può essere utilizzato come un sistema di riconoscimento. Le caratteristiche di una firma, come la velocità e la pressione sono importanti tanto quanto la componente grafologica delle lettere. Sebbene le persone siano abituate a considerare la firma un'azione legata all'affermazione dell'identità di un individuo, questa tecnica non è molto utilizzata in ambito informatico.

La forma della mano Capire la geometria che compone una mano significa analizzare e misurare la sua forma. Questa tecnica biometrica offre un buon bilanciamento tra le *performance* e la facilità d'uso. É un sistema molto accurato e flessibile e si può adattare a diversi scenari d'uso, per questi motivi sta diventando molto utilizzato.

La retina Un sistema che cerca di riconoscere l'identità di un individuo tramite la sua retina mira ad analizzare il livello dei vasi sanguigni situati nel retro dell'occhio. Grazie ad una particolare tecnologia, queste tecnica utilizza una sorgente di luce a bassa intensità per scannerizzare il pattern nella retina. Si tratta di un metodo abbastanza preciso, ma richiede che l'utente focalizzi l'occhio su un determinato punto e si posizioni molto vicino allo scanner. É

poco indicato per coloro che indossano gli occhiali e per chi si sente a disagio ad essere a stretto contatto con una fonte di luce.

L'iride Un analisi biometrica basata sull'iride prende in esame l'anello colorato che circonda la pupilla. É una tecnica meno intrusiva rispetto all'analisi della retina in quanto non richiede uno stretto contatto tra il lettore e l'occhio. Inoltre funziona anche tra gli occhiali.

Il volto Il riconoscimento facciale analizza le caratteristiche della faccia. Richiede ovviamente una videocamera in grado di acquisire l'immagine del volto dell'utente che intende autenticarsi.

La voce L'autenticazione tramite la voce non è basata sul riconoscimento della voce bensì sul riconoscimento dell'impronta vocale.

Capitolo 2

Progettazione

2.1 Requisiti

Dall'analisi emersa nel capitolo uno è evidente che una delle principali falle di sicurezza nel sistema di rilascio del contrassegno disabili è costituita dal fatto che i medesimi sono cartacei, quindi duplicabili, e associati a più targhe, quindi difficili da controllare. Pertanto si propone di dematerializzare il pass sostituendo ad esso un equivalente elettronico rilasciato esclusivamente dopo aver verificato la presenza del disabile a bordo del veicolo. Per richiedere il contrassegno, l'accompagnatore del disabile si servirà di un'applicazione mobile, attraverso la quale richiederà di volta in volta l'autorizzazione ad usufruire del pass , il quale diventerà quindi temporaneo e durerà solo il numero di ore richieste dall'utente. Questo vuol dire che in un istante di tempo può esistere uno ed un solo contrassegno associato ad un particolare soggetto svantaggiato. La verifica della presenza a bordo del soggetto disabile avviene attraverso un test volto a riconoscere un insieme di tratti che combinati tra loro portano all'identificazione univoca dell'individuo. In particolare il sistema proporrà all'utente tre dei quattro tipi di azioni che combinate insieme costituiranno il test:

- **Riconoscimento Facciale**: esso si è rivelato indispensabile per l'identificazione del volto del soggetto disabile. É un test di tipo strettamente biometrico ed imprescindibile ai fini dell'identificazione del soggetto.

- **Riconoscimento Vocale**: si propone al soggetto di ripetere una sequenza di numeri o parole . Questo test è sia volto all'identificazione del soggetto mediante il timbro della voce, sia al riconoscimento delle parole pronunciate.
- **Riconoscimento delle espressioni facciali** : il soggetto svantaggiato dovrà riprodurre una serie di espressioni facciali comunicate dal sistema.
- **Riconoscimento delle *gestures*** : si propone di riprodurre con le mani un insieme di *gestures* .

La scelta di combinare insieme diversi tipi di azioni da svolgere ha una duplice funzione. La prima è sicuramente quella di aumentare il grado di sicurezza del sistema in quanto più il test è complesso più difficile eluderlo. Dall'altro lato non si corre il rischio di escludere dal rilascio del pass soggetti con un determinato tipo di disabilità che non permetterebbe loro di svolgere correttamente la *challenge*. Banalmente, per un individuo affetto da mutismo sarebbe pressoché impossibile autenticarsi vocalmente, mentre uno affetto da paralisi degli arti non gradirebbe un test in cui gli viene chiesto di muovere le mani. Oltre all'effettiva impossibilità fisica, questo risulterebbe anche una presa in giro nei confronti di chi già sta vivendo una situazione di disagio.

L'accompagnatore si occuperà di registrare il test al momento della richiesta del contrassegno, utilizzando l'applicazione proposta per creare un video e inviarlo al sistema. Questo implica che chi guida e il soggetto disabile siano nello stesso posto. In seguito verrà analizzato in dettaglio il funzionamento del sistema proposto.

2.2 Use Case Diagram

Il sistema necessita di un attore che si occupi di gestire la componente anagrafica di tutti i soggetti con disabilità. Dopo aver preso in considerazione il fatto che solo un medico legale può rilasciare una certificazione che attesti l'effettiva disabilità dell'individuo e che questo passo non può in alcun modo essere informatizzato, si propone al Comune di Bologna di occuparsi della registrazione su *database* di ogni singolo disabile. Nello specifico, oltre ai dati sensibili e le informazioni relative al tipo di disabilità, il sistema avrà il compito di memorizzare delle registrazioni

vocali e un set di foto del volto in modo tale da creare un campione univoco necessario per accertare in un secondo momento l'identità del soggetto. Inoltre, proprio per il fatto che i server dell'amministrazione comunale avranno accesso ai dati dei disabili, essi si occuperanno anche della creazione del contrassegno disabili e della relativa *challenge* associata ad esso. Se il test risulterà essere stato eseguito correttamente, il pass verrà convalidato e l'utente sarà liberò di circolare con il disabile a bordo.

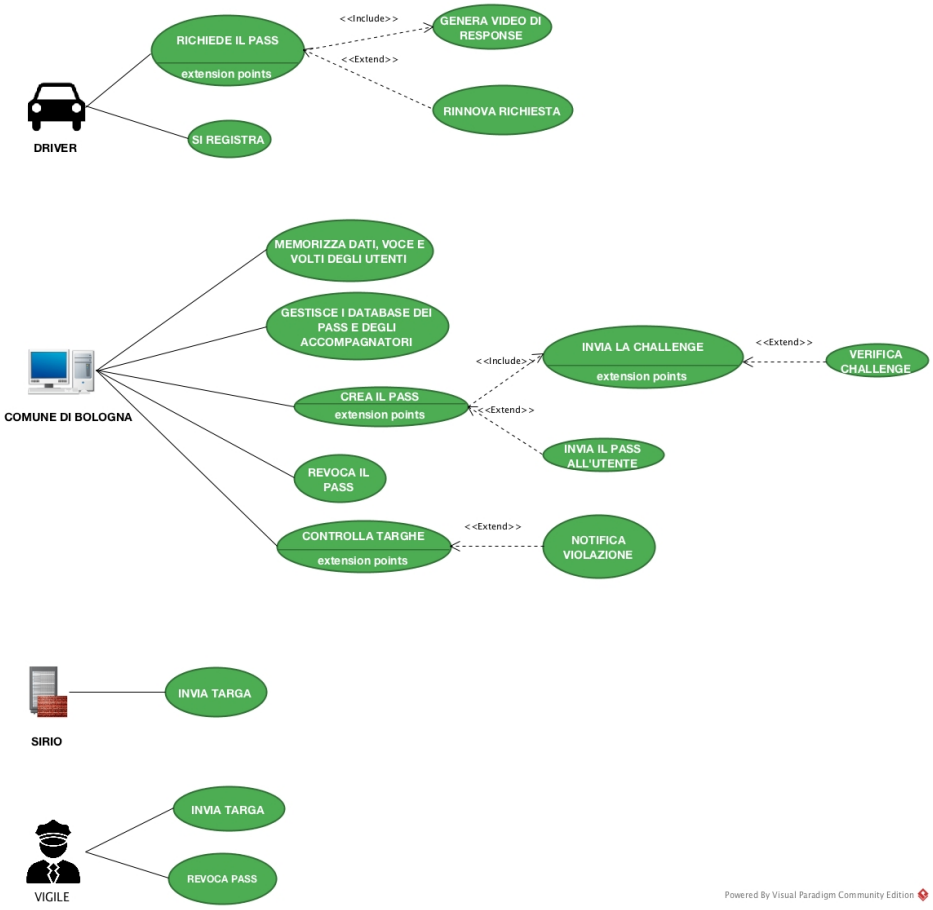

Figura 2.1: Diagramma dei casi d'uso

L'accompagnatore, che nel diagramma dei casi d'uso è indicato con l'etichetta *Driver*, dovrà innanzitutto registrarsi al servizio, fornendo dati sensibili e reali. La targa del veicolo invece può essere comunicata in seguito e cambiata senza limiti. Questo non è di per sè un grosso problema in termini di sicurezza, in quanto, come

già sottolineato nell'analisi dei requisiti, l'univocità del contrassegno è strettamente legato al fattore temporale e al disabile che si accompagna: è evidente che una persona disabile o non, non può essere in due autoveicoli contemporaneamente.

Il sistema contempla anche il caso in cui il disabile si trovi già in una zona a traffico limitato. In questo caso l'accompagnatore prenoterà il pass e si occuperà di convalidare il pass entro un periodo di tempo stabilito. In caso di mancato invio del test il server provvederà ad inviare eventuali contravvenzioni.

2.3 Class Diagram

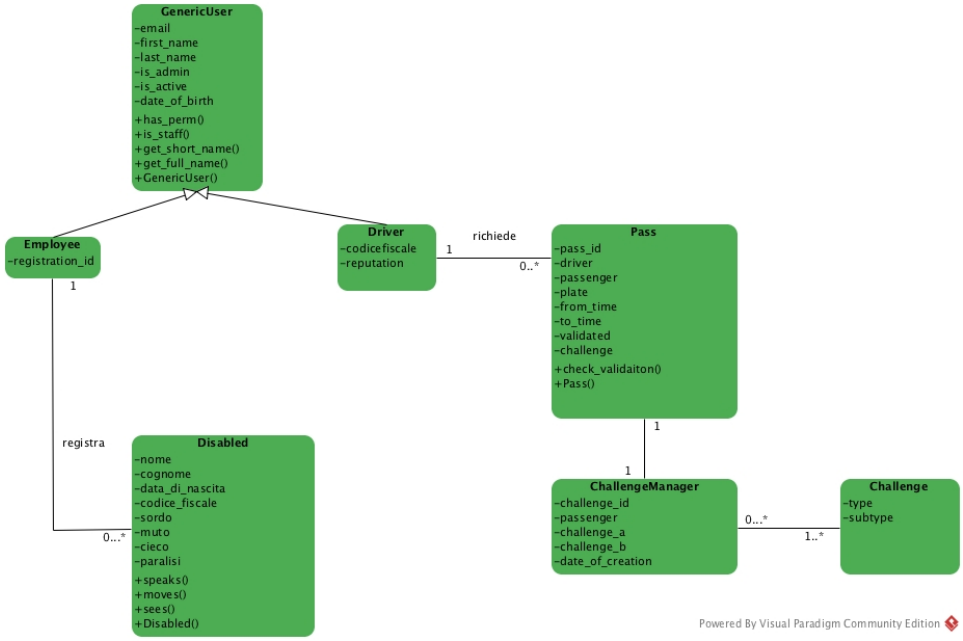

Figura 2.2: Diagramma di Classe di un pass

Nel diagramma sovrastante sono raffigurate solo le classi cardine del progetto.

Un *Generic User* è una classe che rappresenta tutti gli utenti che possono essere abilitati ad autenticarsi nel sistema. L'autenticazione avviene fornendo email e password. Si tratta di una generalizzazione di due grandi gruppi di utenti: *Employee* e *Driver*. I primi rappresentano tutti gli impiegati del Comune di Bologna autorizzati a registrare i dati dei *Disabili*. Quest'ultima classe invece gestisce l'anagrafica di tutti i soggetti affetti da disabilità. Gli accompagnatori, *driver*, sono

tutti coloro, che effettuando login esclusivamente tramite l'applicazione mobile, possono richiedere la creazione di un nuovo contrassegno per disabili. Ad ogni *Pass* è associato una classe che si occupa di gestire la creazione e la verifica delle *challenge*, il *Challenge Manager*. Esso è composto da due e solo due *Challenge*. Ogni *Challenge* è composta da un tipo, che indica la categoria di riferimento (vocale, espressione, *gesture*), e un sottotipo che indica la specifica azione da compiere.

2.4 State Diagram

Analizziamo con un diagramma di stato il "ciclo di vita" di un oggetto *Pass*.

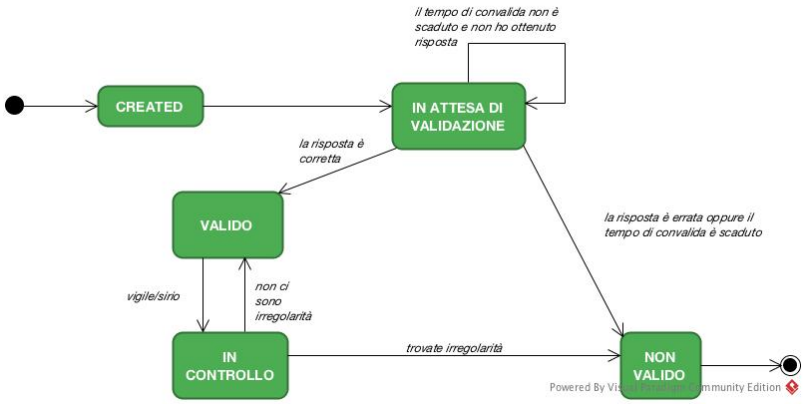

Figura 2.3: Diagramma di stato di un pass

Per prima cosa l'oggetto viene creato ed inizializzato con il codice fiscale dell'accompagnatore, il codice fiscale dell'utente disabile, il numero di targa e la durata del contrassegno in base alla richiesta effettuata dall'utente. A questo punto il pass entra nello stato di "attesa di validazione", ovvero esso non è ancora valido, in quanto l'utente non ha effettuato il test. Fino a che il tempo che è stato stabilito congruo per convalidare il contrassegno non è scaduto, l'oggetto pass esiste e l'utente potrà inviare il suo video di risposta alla challenge. Trascorso quell'intervallo di tempo se non è stata ancora effettuata la convalida il pass sarà dichiarato non valido e verrà distrutto. In questo caso l'utente dovrà ripetere tutta la procedura per la richiesta del pass. Invece, in caso di test effettuato correttamente l'oggetto

pass sarà considerato valido a tutti gli effetti. Periodicamente verrà controllata la validità del contrassegno, sia da agenti esterni, come SIRIO oppure la polizia municipale, e in caso di irregolarità il pass verrà automaticamente dichiarato non valido e distrutto.

2.5 Sequence Diagram

Si utilizza un diagramma di sequenza per simulare la richiesta di un oggetto pass: Il primo passo è effettuato dall'accompagnatore del disabile, il quale attra-

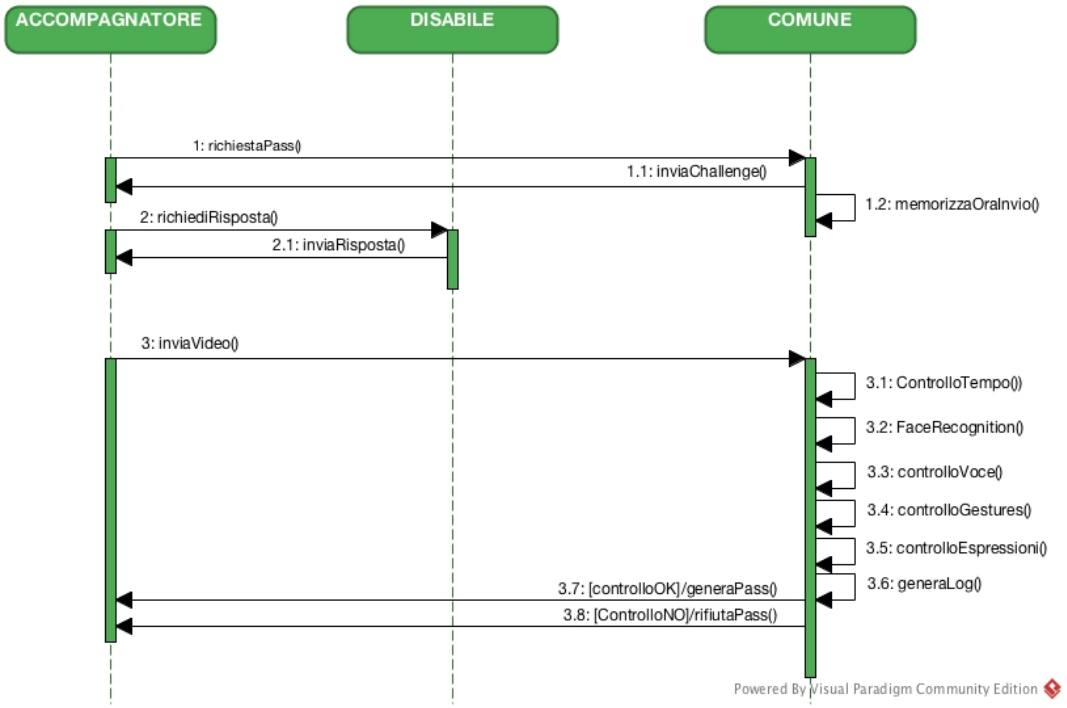

Figura 2.4: Diagramma di sequenza della richiesta di un pass

verso l'applicazione proposta compilerà un form comunicando il suo codice fiscale, il codice fiscale del disabile che vuole accompagnare, la targa del veicolo al quale associare il pass e invia tutto ai server del Comune di Bologna. Questi, dopo aver accertato l'integrità dei dati e l'effettiva esistenza del disabile nel database provvedono a calcolare una *challenge* congrua al tipo di disabilità e ad inviarla all'utente dell'applicazione. Successivamente lo scambio di messaggi avverrà tra

l'accompagnatore e il disabile: il primo comunicherà al secondo il tipo di test che dovrà effettuare, eventualmente mostrandogli i gesti da eseguire; dopodiché verrà registrato un video con il volto del disabile, le frasi oppure i movimenti mostrati. Attraverso la applicazione l'accompagnatore avrà il compito di inviare il video appena registrato al Comune di Bologna e attendere la risposta. Il primo controllo che sarà effettuato dai server sarà verificare l'effettiva validità temporale del contrassegno: come già mostrato in precedenza nel diagramma di stato di un pass, se il tempo di convalida è scaduto, esso verrà automaticamente dichiarato non valido e distrutto. Se così fosse, si eviterebbero dispendiosi calcoli necessari nelle operazioni successive. Terminata la verifica temporale, il sistema procederà ad analizzare il video cercando dapprima la corrispondenza tra il volto presente nel file e l'utente disabile comunicato. Dopodiché , esso provvederà in successione a controllare la componente vocale del video e la corrispondenza delle *gestures* e delle espressioni con quelle richieste. Terminato il controllo l'utente riceverà in risposta un messaggio contenente l'eventuale validazione del pass.

Capitolo 3

Il server

3.1 Funzionalità

Come è emerso dall'analisi dei requisiti e dai diagrammi UML illustrati nel capitolo precedente, il server contiene il vero e proprio *core* dell'applicazione: infatti qui si trovano tutte le meccaniche e gli algoritmi necessari al riconoscimento e all'autenticazione di un soggetto disabile. Questa scelta implementativa deriva dalla consapevolezza che, in generale, gli algoritmi di *computer vision* richiedono l'utilizzo di molte risorse il che andrebbe discapito del consumo della batteria di un qualsiasi *smartphone*. Inoltre per questioni di *privacy* sarebbe stato pressoché impossibile riuscire ad ottenere il permesso di accedere ed elaborare direttamente tramite l'applicazione i dati di tutti i soggetti disabili registrati, in quanto trattasi di elementi sensibili. In questo capitolo verranno analizzate tutte le tecnologie e gli algoritmi utilizzati a lato server, tenendo presente che alcune parti sono state implementate al solo scopo di testare l'applicazione più agevolmente.

3.2 Django Framework

La storia Il *Django Project* nasce da World Online, un giornale online che si occupava di programmare numerose applicazioni web dovendo rispettare rigide scadenze giornalistiche: spesso capitava di dover trasformare schemi concettuali di applicazioni web in servizi pronti all'uso per il pubblico, seguendo tutte le norme

di *best pactices* per lo sviluppo web. Precisamente nel 2003, due sviluppatori di World Online, Adrian Holovaty e Simon Wilson iniziarono ad utilizzare *Python* per sviluppare un *framework* che potesse aiutarli a produrre in tempi rapidi un numero sempre maggiore di applicazioni. Due anni dopo, World Online decise di rendere questo progetto *open-source*: nacque così Django. E' stato utilizzato per molti anni da compagnie del calibro di Instagram, Pinterest, e Mozilla.

La struttura di un'applicazione Django Al momento della creazione ogni applicazione sviluppata con il *framework* Django racchiude una serie di file di partenza, creando una struttura di questo tipo:

```
laMiaApplicazione/
    manage.py
    mysite/
        __init__.py
        settings.py
        urls.py
        wsgi.py
```

Più precisamente, analizzando nel dettaglio :

- `manage.py` è un *utility* per linea di comando che permette di interagire con il progetto sviluppato in Django in vari modi;
- `init.py` è un file vuoto che dice a Python che questa precisa *directory* deve essere considerata un pacchetto di Python;
- `settings.py` contiene tutte le configurazioni del progetto di Django. In particolare vengono specificate tutte le applicazioni installate e i moduli necessari a far funzionare il progetto. Inoltre vengono fornite tutte le informazioni relative al database utilizzato (per default Django utilizza SQLite3);
- `urls.py` è un file che contiene una sorta di "tabella dei contenuti" del progetto, ovvero l'insieme di tutte le URL del progetto;
- `wsgi.py` contiene le specifiche di come un web server comunica con un'applicazione web.

Model-View-Controller Parte fondamentale dell'applicazione sia a lato server che a lato client è il *Model-View-Controller*(MVC). Si tratta di un design pattern che assegna agli oggetti facenti parte di un'applicazione uno di questi tre ruoli: *model* (modello), *view* (vista) oppure *controller*. Ciascuno di questi tre tipi di oggetti è separato dagli altri da confini astratti e comunica con essi solo varcando questi confini.

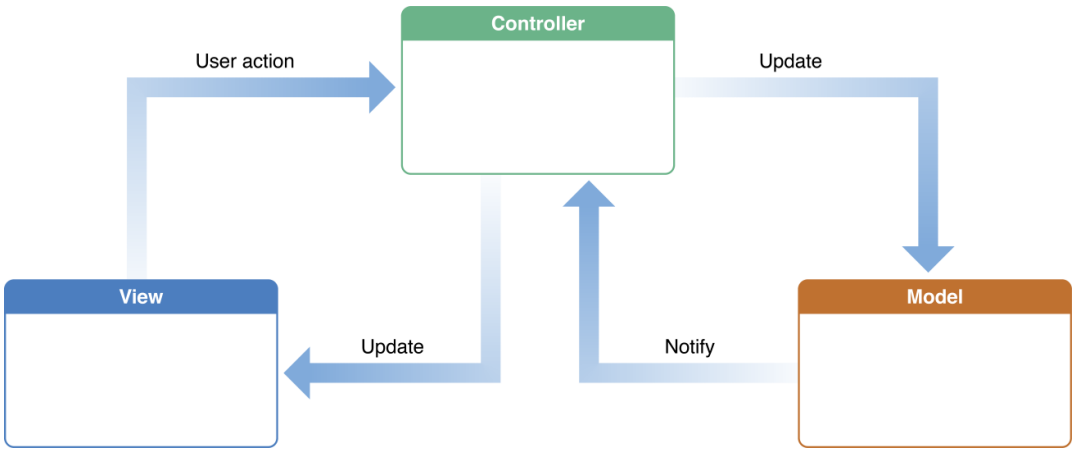

Figura 3.1: Funzionamento del pattern MVC

Il *modello* encapsula i dati specifici di un'applicazione e definisce la logica e i calcoli necessari per manipolarli e processarli. Si potrebbe prendere come esempio un personaggio di un videogioco oppure un contatto in un'applicazione di rubrica. Un modello può avere relazioni uno-ad-uno oppure uno-a-molti con gli altri oggetti dell'applicazione. Concettualmente il modello non dovrebbe avere nessun tipo di collegamento con la vista dell'oggetto che presenta i suoi dati e permette all'utente di modificare i dati. Le azioni dell'utente nella vista che crea o modifica i dati sono comunicati attraverso un *controller* e il risultato dell'operazione creerà o modificherà oppure aggiornerà un oggetto del modello. Quando un oggetto del modello cambia notifica al *controller* quali aggiornamenti sono appropriati per gli oggetti della vista.

Un oggetto che appartiene alla *vista* compone la parte dell'applicazione che gli utenti possono vedere. Una vista sa quando disegnare se stessa sullo schermo e può rispondere alle azioni dell'utente. Il principale compito di una vista è quello di visualizzare i dati dal modello e attivare la modifica di essi.

Il *controller* è un oggetto che agisce come intermediario tra una o più viste di un'applicazione e i suoi modelli. Un *controller* inoltre è in grado di gestire il ciclo di vita degli altri oggetti e eseguire *setup* . Esso capta le azioni che un utente ha svolto in un oggetto vista e comunica dati nuovi o modificati al modello. Allo stesso modo, quando un modello cambia un controller deve comunicare alla vista il nuovo dato affinchè possa essere mostrato.

Django utilizza massicciamente una reinterpretazione di questo paradigma di programmazione. Il *controller* è costituito essenzialmente dal *framework*, ovvero tutte le meccaniche che inviano la richiesta alla vista appropriata in base alla configurazione delle URL. La *view* invece si occupa di descrivere i dati che vengono presentati all'utente, ma non come vengono visti. Si tratta di una sorta di funzione *callback* per una particolare URL che decide quali dati vengono presentati. Inoltre Django utilizza le *view* come *delegate* per un particolare tipo di *template* il quale descrive come il dato viene mostrato all'utente (banalmente, il codice html). Proprio la presenza dei *template* ha dato vita all'acronimo MTV, ovvero *Model Template View*.

L'applicazione L'applicazione scritta con l'ausilio del *framework* Django è a sua volta suddivisa in due moduli:

- **mobileserver** si occupa di ricevere ed analizzare tutte le richieste che vengono inviate dal *client*. Inoltre in questo modulo sono presenti tutti gli algoritmi di *computer vision* che verranno analizzati nel paragrafo successivo.
  - `request_pass` Questa funzione si occupa di creare un nuovo pass memorizzando nel database i dati inviati dal client. In base al tipo di disabilità dell'utente segnalato dall'accompagnatore, vengono stabilite challenge opportune, create e memorizzate a loro volta nel database. Infine il nuovo pass viene inviato al client sotto forma di JSON;
  - `driver_register` Questa funzione si occupa di registrare l'accompagnatore e abilitarlo all'utilizzo dell'applicazione;
  - `driver_login` Questa funzione si occupa di gestire il login dell'accompagnatore;

  - `check_challenge` Questa funzione prende in input la challenge da analizzare e il video e restituisce l'esito dell'analisi. Qui vengono invocate le funzioni di *computer vision*;

  - `upload_file` Questa funzione carica i file e i dati ricevuti dal client nel server, e salva il *path* del video nel database, precisamente nel campo del test associato al pass. Una volta salvato il video, verrà invocata la funzione check_challenge che si occuperà di verificare se il test è stato superato oppure no. Il valore di ritorno inviato al client è un file JSON contenente l'identificativo del pass appena esaminato e un campo indicante l'esito della verifica.

- **account** Sviluppato soprattutto ai fini di *testing*, questo modulo si occupa della registrazione e gestione degli individui disabili. E' composto da una parte front-end che simula l'interfaccia che dovrebbe avere un ipotetico programma utilizzato dagli impiegati del Comune di Bologna. Contiene quindi un database sia di tutti i disabili, sia di tutti gli impiegati.

3.3 L'analisi del volto

3.3.1 OpenCV

OpenCV (*Open Source Computer Vision Library*) è una libreria completamente *open source* rilasciata sotto licenza BSD utile per la realizzazione di software nell'ambito della *computer vision* e del *machine learning*. Scritta originariamente in C e C++, vanta innumerevoli interfacce in C, C++, Java e Python, che permettono il supporto completo nelle diverse piattaforme Linux, Microsoft, OsX, Android e IOS. Esistono anche dei *wrapper* per C# e Ruby. Tra i più di 2500 algoritmi che compongono la libreria, sicuramente quelli più interessanti ai fini dello sviluppo del progetto sono quelli che si occupano dell'individuazione degli oggetti, del riconoscimento facciale e dell'analisi dei video.

3.3.2 Face Detection

Dato un video da analizzare, il primo passo per la verifica dell'identità del soggetto svantaggiato al quale rilasciare il pass disabili è senza ombra di dubbio l'individuazione del volto. Esistono innumerevoli approcci al problema. Tuttavia i principali algoritmi che saranno esplorati in dettaglio saranno il Local Binary Pattern e Viola Jones.

Local Binary Pattern

La tecnica LBP (*Local Binary Pattern*) risulta molto efficace ed estremamente veloce per l'analisi delle caratteristiche della *texture* di un'immagine. In particolare ad ogni pixel p viene associato un valore binario, o 0 o 1, in relazione alla suo valore rispetto al pixel centrale di cui p costituisce l'intorno: se p ha un valore superiore oppure uguale al valore del pixel centrale pc allora si assegnerà a p il valore 1, 0 altrimenti. Questo concetto è stato esteso per gestire intorni di dimensione variabile di pixel, scegliendo i pixel campione tramite interpolazione. Ecco come nascono i pattern. Tra tutti i *pattern* che si formano, quelli di maggiore interesse per analizzare le immagini sono i *pattern* uniformi, ovvero quei pattern che contengono al più due transizioni 0-1 1-0. Considerare solo i *pattern* uniformi permette un notevole risparmio in termini di memoria.

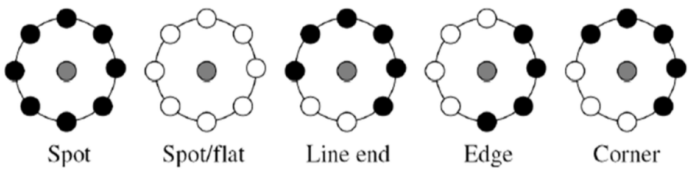

Figura 3.2: Principali *pattern* uniformi

Il secondo passo verso la costruzione di un vettore di *pattern* è costituito dal calcolo dell'istogramma, eventualmente normalizzato, per l'immagine. Per fare ciò, l'immagine viene dapprima decomposta in k *sottofinestre* dalle quali vengono estrapolati tutti i *pattern* dai quali si calcola l'istogramma parziale. Concatenando gli istogrammi parziali si ottiene il vettore di *pattern* per l'identificazione di un volto.

Figura 3.3: Analisi di un volto con *pattern* uniforme e non

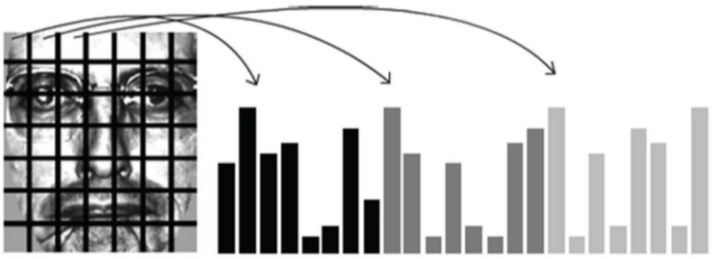

Figura 3.4: Esempio di istogramma del volto

Vantaggi LBP:

- Approccio semplice
- Computazionalmente semplice
- Più veloce dell'algoritmo di Viola Jones
- Tolleranza dei cambi di illuminazione

Svantaggi LBP:

- usare larghe regioni di immagini aumenta la possibilità di errore;
- non è preciso;
- si può usare solo per immagini in scala di grigio;

Viola e Jones

L'algoritmo per l'individuazione di oggetti proposto da Paul Viola e Michael Jones si basa su un approccio di *machine learning* che propone di costruire un *cascade classifier* il quale verrà poi utilizzato per riconoscere oggetti in altre immagini. Nel nostro caso specifico è stato costruito un classificatore per individuare il volto. Inizialmente l'algoritmo necessita di numerose immagini, suddivise in positive , ovvero le immagini del volto, e negative, ovvero le immagini senza volto. Queste saranno alla base del nostro classificatore.

Una volta costruito un set di immagini è necessario estrarre delle *features* da esse. In particolare, nella *face detection* vengono utilizzate le *feature Haar*, dalle quali deriva il nome h*aar cascadeclassifier*. Ogni *feature Haar* consiste in due o più rettangoli bianchi e neri e il valore di ogni *features* corrisponde alla differenza tra la somma del valore di grigio del pixel con le regioni bianche e nere del rettangolo.

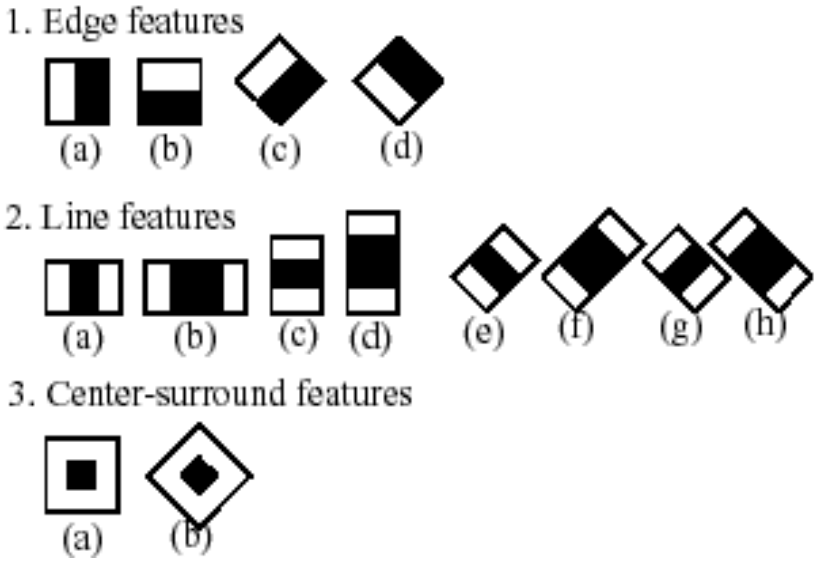

Figura 3.5: Esempi di haarfeatures

Tuttavia, non tutte le *features* trovate sono utili alla fine dell'individuazione del volto. Per questo è necessario applicare ogni singola *features* a tutte le immagini che compongono il nostro *training set*. Per ciascuna *features* l'algoritmo trova la miglior soglia per classificare le facce positive e negative. Il classificatore finale è una somma di tutti i classificatori costruiti parzialmente per ogni singola immagine. Originariamente le *features* venivano applicate ad un'intera immagine una per una,

controllando tutte le volte se fosse presente una faccia. E' evidente di come questo metodo fosse estremamente dispendioso in termini di tempo e di memoria. Per ovviare a questo problema Viola e Jones introducono il concetto di *Cascade of Classifiers*, ovvero, invece che applicare ogni singola *features* su un'immagine, esse vengono raggruppate in differenti passi di classificazione. Se esiste una regione che non è una faccia essa viene esclusa dall'analisi dei successivi gruppi di *features*. Se il test viene superato si effettuerà il controllo sul secondo gruppo di *features* e così via. Se la parte di immagine supera tutti i test significa che nell'immagine è presente una faccia.

Vantaggi Haar Cascades:

- è l'algoritmo più usato per la *face detection* in *real time*;
- estremamente accurato grazie alla presenza di classificatori;
- bassa possibilità di falsi positivi;

Svantaggi Haar Cascade:

- occorre molto tempo per il *training*;

Poichè l'obiettivo del progetto era quello di avere maggiore accuratezza è stato utilizzato il metodo di Viola Jones.

Esempio algoritmo di face detection con haarcascades.

```
2 video = cv2.VideoCapture(video_path)
  while video.isOpened():
4   (rval, img) = video.read()
    img = cv2.flip(img, 1, 0)
6   mini = cv2.resize(img, (img.shape[1]/size,
                            img.shape[0]/size))
8   gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

10
```

```
    faces = faceClassifier.detectMultiScale(
12      gray,
        scaleFactor=1.1,
14      minNeighbors=5,
        minSize=(30, 30),
16      flags=cv2.cv.CV_HAAR_SCALE_IMAGE
    )
18  for (x, y, w, h) in faces:
        # Face is detected
20      cv2.rectangle(img,(x, y),(x + w, y + h),(0, 255, 0),4)
```

OpenCV fornisce la funzione `detectMultiScale()` che prende in input un video o un'immagine opportunamente trasformata e convertita in scala di grigio e restituisce in output i volti che sono stati rilevati. Gli altri parametri della funzione sono :

- `minSize` : determina la misura minima delle facce che sono rilevanti per l'analisi. In questo caso è stato scelto un volto delle dimensioni di 30 x 30 pixels;
- `scaleFactor`: determina il rapporto tra le misure dei volti che devono essere analizzati. In questo caso il valore è 1.1, non il più veloce, ma molto accurato.
- `minNeighbors`: determina il grado di confidenza del *face detector*.
- `flags`: flag aggiuntivi. In questo caso `CV_HAAR_SCALE_IMAGE` indica che l'algoritmo deve scalare l'immagine piuttosto che modificare il detector.

3.3.3 Face Recognition

Riconoscere le persone, generalmente, è un compito semplice per gli esseri umani. Esperimenti hanno dimostrato che i neonati sono capaci di distinguere i volti conosciuti sin dai primi giorni di vita. Questo è possibile grazie al fatto che nel cervello umano sono presenti delle cellule specializzate atte ad analizzare specifiche caratteristiche di una determinata scena, come le linee, gli angoli e i movimenti fino a formare dei *pattern* di riconoscimento.

Questo suggerisce che l'identità di un volto è codificata soprattutto dalle relazioni spaziali fra i suoi tratti (naso, bocca, occhi), quindi tra tutte le caratteristiche collegate assieme.

In linea di massima, riconoscere le persone automaticamente con un calcolatore funziona allo stesso modo. Lo scopo del riconoscimento facciale automatico è di associare ad un'immagine ricevuta in input la classe di appartenenza. Quindi dato un set di immagini, chiamato *learning set*, ciascuna identificata con un'etichetta che si riferisce all'identità del soggetto raffigurato e un set di immagini, chiamato *test set*, senza etichetta alcuna, l'obiettivo prefisso è quello di identificare l'identità di ciascuna persona. Il più semplice approccio alla risoluzione di questo problema è il metodo del *nearest neighbour classifier*. Secondo questo paradigma, è buona norma normalizzare le immagini di entrambi i set, in modo da ottenere una media uguale a zero e una varianza unitaria. In questo modo il risultato non dipenderà dall'intensità delle fonti di illuminazione. Dopo aver compiuto l'operazione di normalizzazione si procede alla classificazione dell'immagine del test set alla quale verrà associata l'etichetta dell'immagine del *learning set* che ha più punti in comune con essa. Tuttavia questa procedura ha notevoli svantaggi: prima di tutto se il *learning set* e il *test set* sono memorizzati sotto diverse condizioni di illuminazione i punti potrebbero non coincidere; in secondo luogo è un algoritmo computazionalmente molto dispendioso.

La libreria OpenCV viene incontro alle esigenze degli sviluppatori di software nell'ambito della *computer vision* fornendo la classe *FaceRecognizer* che implementa una serie di funzionalità specifiche per il riconoscimento facciale.

Prima di addentrarsi nell'esplicare il funzionamento dell'algoritmo è utile soffermarsi un attimo sulla creazione del set di immagini.

La classe FaceRecognizer in OpenCV

```
1
class FaceRecognizer : public Algorithm
3 {
public:
5   //! virtual decostruttore
   virtual ~FaceRecognizer() {}
7
```

```
    // Trains a FaceRecognizer.
    virtual void train(InputArray src, InputArray labels) = 0;

    // Predict() cerca di capire a chi appartiene una precisa immagine
    virtual void predict(InputArray src, int &label,
        double &confidence)

    createEigenFaceRecognizer(int num_components=0,
        double threshold=DBL_MAX)

};
```

Eigenface e PCA

Questo algoritmo, proposto da Matthew Turk e Alex Pentland, propone di trasformare le immagini che raffiguranti individui in un set di immagini, raccogliendo alcuni tratti distintivi del volto. E' importante sottolineare che le caratteristiche estratte dal volto non devono necessariamente coincidere con la nozione intuitiva di elementi che compongono una faccia, come ad esempio il naso, la bocca, le labbra o i capelli. In termini matematici Turk e Pentland propongono di trovare i componenti principali della distribuzione delle facce, o gli autovettori che compongono la matrice di covarianza del set delle immagini. Questi autovettori possono essere pensati come un set di features che insieme caratterizzano la variazione tra le varie facce che compongono il set. Ogni area dell'immagine si riferisce ad un autovettore, quindi è possibile unire gli autovettori per mostrare una sorta di faccia fantasma: una *eigenface*. Ciascuna immagine facente parte del *training set* può essere rappresentata esattamente in termini di una combinazione lineare delle *eigenface*, di conseguenza è possibile creare un numero di *eigenfaces* pari al numero di immagini presenti nel set. Tuttavia, ai fini di ottenere un algoritmo efficiente e senza sprechi di risorse, è consigliabile utilizzare solo quelle *eigenfaces* che hanno il maggior autovettore e che rappresentano la varianza del set di immagini. L'algoritmo di riconoscimento può essere suddiviso nelle seguenti fasi:

Figura 3.6: Esempi di eigenfaces

- Inizializzazione, ovvero acquisire il *training set* delle immagini dei volti e calcolare le *eigenfaces* che definiscono lo spazio delle facce. Le immagini che compongono il set sono opportunamente normalizzate eseguendo un'operazione di *crop* sui volti, in modo da creare solo gli autovettori necessari. Questa operazione di *training* va ripetuta ogni qual volta vengono aggiunti nuovi soggetti.

- Quando si inserisce una nuova immagine essa viene proiettata su ciascuna delle *M-eigenface* e in base a questo confronto si calcola un set di pesi.

- Determina se l'immagine è una faccia oppure no, controllando se l'immagine è sufficientemente somigliante allo spazio delle facce.

- Se è una faccia, classifica il pattern del peso a seconda che sia una persona conosciuta oppure no

La libreria OpenCV fornisce il metodo `createEigenFaceRecognizer()` che permette di creare un classificatore basato sull'individuazione Eigenfaces.

3.3.4 Emotion Recognition

Le espressioni facciali sono uno dei modi più immediati e potenti usati dagli esseri umani per comunicare le proprie emozioni e sensazioni. Riconoscerle automaticamente tuttavia è un compito abbastanza arduo, a causa della complessità del volto e della sua dinamicità. A questo scopo uno dei sistemi più utilizzati è il *Facial Action Coding System* (FACS), sviluppato da Ekman e Friesen. Piuttosto che classificare le immagini in categorie di emozioni, come "felice", "triste" o

"sorpreso", l'obiettivo di questo sistema di codifica è quello di individuare le azioni muscolari che compongono un'espressione e di rilevare l'attività dei muscoli facciali, basandosi su un video in cui è presente un individuo. Ekman e Friesen hanno individuato all'incirca 46 distinte azioni (*action units*) , ciascuna corrispondente all'attività di un singolo muscolo oppure di un gruppo muscolare che produce una distorsione della morfologia della faccia visualizzata. Per questi motivi, analizzare automaticamente le espressioni facciali è un problema molto complesso e ampiamente studiato.

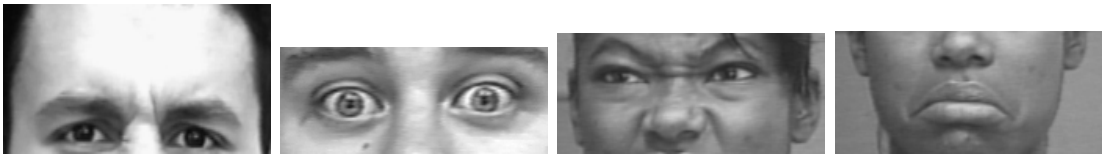

Figura 3.7: Esempi di FACS

Esistono due principali metodi per questo tipo di riconoscimento:

- Metodi *geometry-based*. Essi mostrano la forma e la posizione dei vari componenti della faccia che sono estratti dal video per formare un vettore di *features* che rappresenta l'intera geometria della faccia. A livello prestazionale non si discosta molto dagli algoritmi che usano approcci *aspect-based*
- Metodi *aspect-based*. Essi utilizzando i metodi del *LBP* e degli *Haar Cascades Classifiers* descritti in precedenza.

Il procedimento è del tutto analogo a quello utilizzato nell'algoritmo di *face detection*: si applica la funzione `detectMultiScale()` prima sul classificatore del volto; successivamente si utilizza il classificatori della bocca e degli occhi, al fine di individuare la loro conformazione. Infine si cerca tra le emozioni quale è quella rappresentata dal soggetto nel video.

3.3.5 Hand Gestures Recognition

Gesticolare è un'abitudine profondamente radicata nelle comunicazioni tra gli esseri umani. Basti pensare a quanto spesso le persone accompagnano le loro conversazioni al telefono con ampi movimenti delle mani. I gesti forniscono una

modalità di esprimere un'idea complementare al discorso parlato, ma separato da esso, pertanto sono un potente strumento della comunicazione tra gli esseri umani. Di conseguenza sembra lecito pensare che le *hand gestures* possano costituire uno strumento di comunicazione tra l'uomo e un calcolatore. Il punto nodale del problema è capire come rendere comprensibili al computer i gesti. Esistono fondamentalmente due tipi di approcci:

- *Data-Glove based.* Sono sistemi basati sulla raccolta dati attraverso dispositivi con particolari sensori che si occupano di digitalizzare il movimento della mano e delle dita. Raccogliere informazioni sulla configurazione e sul movimento della mano è estremamente semplice. Tuttavia, come è immaginabile, questi dispositivi sono costosi e richiedono una certa esperienza dell'utente.

- *Vision based.* Sono sistemi basati esclusivamente su una videocamera, senza necessità di supporti aggiuntivi. Questi sistemi tendono a far combaciare la visione biologica delle cose con i sistemi di visione artificiale.

Nei sistemi di riconoscimento di *gestures* basati sulla visione artificiale, il movimento della mano è registrato da una o più videocamere. Questo video viene decomposto in un set di features prendendo singoli frame. Talvolta quest'operazione di filtraggio può essere utilizzata anche per rimuovere dati non necessari, come ad esempio altre parti del corpo oppure elementi di background. Gli approcci alla soluzione del problema sono essenzialmente due.

- *Modello 3D della mano.* E' l'approccio indicato per monitorare i tentativi di interazioni realistiche in ambienti virtuali. Si tratta in pratica di tracciare il movimento della mano analizzando il problema in maniera inversa: ovvero, data un frame è necessario trovare i parametri del modello.

- *Modello 2D della mano.* Questi approcci si basano sull'utilizzare le caratteristiche di un'immagine per modellare la conformazione della mano e comparare questi parametri con le caratteristiche estratte dal video di input. Grazie a queste features 2D questo tipo di sistema gode di un notevole vantaggio in termini di performance in *real-time*. Uno dei metodi più semplici per l'individuazione di una *gestures* è quello di ricercare nell'immagine zone

del colore della pelle. Tuttavia questo ha il notevole svantaggio di dipendere dalle condizioni di illuminazione che modificano sensibilmente la gradazione di colore della pelle. Inoltre è inevitabile che nell'immagine debba essere presente un solo soggetto. Un altro approccio molto utilizzato è quello degli *eigenfaces*, già approfondito nel paragrafo riguardante la *Face Recognition*. Infine esiste un terzo metodo per il riconoscimento dei gesti. Esso consiste nel combinare insieme l'algoritmo di Viola e Jones, già citato nelle tecniche di individuazione dei volti, e l'algoritmo Adaboost. In questo caso i classificatori *haar* sono utilizzati per riconoscere una o più mani all'interno del video focalizzandosi in una certa area dell'immagine piuttosto che su ogni singolo pixel. Per questo motivo è necessario utilizzare l'algoritmo di *learning machine* Adaboost: esso è composto da una fase di *training* che permette, dato in input un set di elementi " positivi ", ovvero contenenti l'oggetto di interesse, e uno di "negativi", di scegliere via via le caratteristiche Haar migliori per classificare l'immagine. Il vantaggio di questo ultimo approccio è che l'algoritmo di Viola e Jones è 15 volte più veloce di qualsiasi altro metodo presentato in precedenza. Tuttavia la fase di *training* è computazionalmente molto onerosa.

3.4 L'analisi della voce

La voce umana è una parte indispensabile dell'identità di un individuo. Le tecniche per analizzare una voce in maniera automatica si basano sui diversi tipi di informazioni che possono essere raccolte a partire da segnali acquisiti normalmente da un microfono e digitalizzati. Ecco un elenco di dati che possono essere estratti da un semplice file audio:

- *Lo spettro*: Differenti interlocutori hanno uno spettro della voce, composto dalla posizione e dalla grandezza dei picchi, diverso per suoni simili. Una proprietà nodale dei metodi che analizzano lo spettro è la scala logaritmica, che mima il funzionamento dell'orecchio umano, aumenta la percentuale di successo del riconoscimento.

- *Stress, accento e intonazione*: il modo più semplice per estrarre queste caratteristiche è attraverso il picco, l'energia e la durata dell'informazione. Riuscire ad estrarre queste caratteristiche può essere utile al fine di migliorare la robustezza del sistema in quanto spesso gli esseri umani quando cercano di imitare la voce di un altra persona lo fanno concentrandosi sulle dinamiche dei picchi piuttosto che su valori istantanei.

- *La fonetica*: E' possibile caratterizzare diversi pattern utilizzando la sequenza di fonemi: è risaputo infatti che lo stesso fonema può essere pronunciato in modi differenti senza cambiare la semantica di un'istanza. Parte della fonetica risiede anche nelle inflessioni dialettali dell'utente . Lo svantaggio di questo sistema di analisi è la necessità di implementare un sistema automatico di riconoscimento delle parole e di costruire una raccolta dati sufficientemente esaustiva contenente tutti i dialetti esistenti.

- *Frequenza delle parole*: l'idea di fondo è quella di riconoscere gli interlocutori dal loro utilizzo delle parole. Alcune persone infatti tendono ad abusare di innumerevoli parole.

Le informazioni estratte dal sistema possono essere combinate tra loro ai fini di ottenere un sistema di riconoscimento più performante. Per questo motivo esistono sistemi *text-dependent*,che dipendono strettamente dal testo pronunciato. Infatti uno dei fattori critici del riconoscimento dell'interlocutore è l'incostanza dei canali registrati nella fase di *training* rispetto alla fase di *testing*,causati dal rapporto tra la differenza del segnale e il rumore, il tipo di microfono e dal modificare della voce con il passare del tempo. Per questo motivo il picco, la frequenza delle pause, l'uso di un determinato lessico e di una determinata pronuncia incidono sensibilmente nel riconoscimento di un individuo, in quanto si tratta di caratteristiche ad alta frequenza che non vengono intaccate dal rumore. Dopo aver scelto quali caratteristiche analizzare, il sistema può utilizzare due approcci per risolvere il problema del riconoscimento biometrico della persona attraverso input vocale.

Speaker identification Il sistema automatico basato su *speaker identification* si basa sul capire se la voce di colui che parla, appartiene ad un set predefinito di

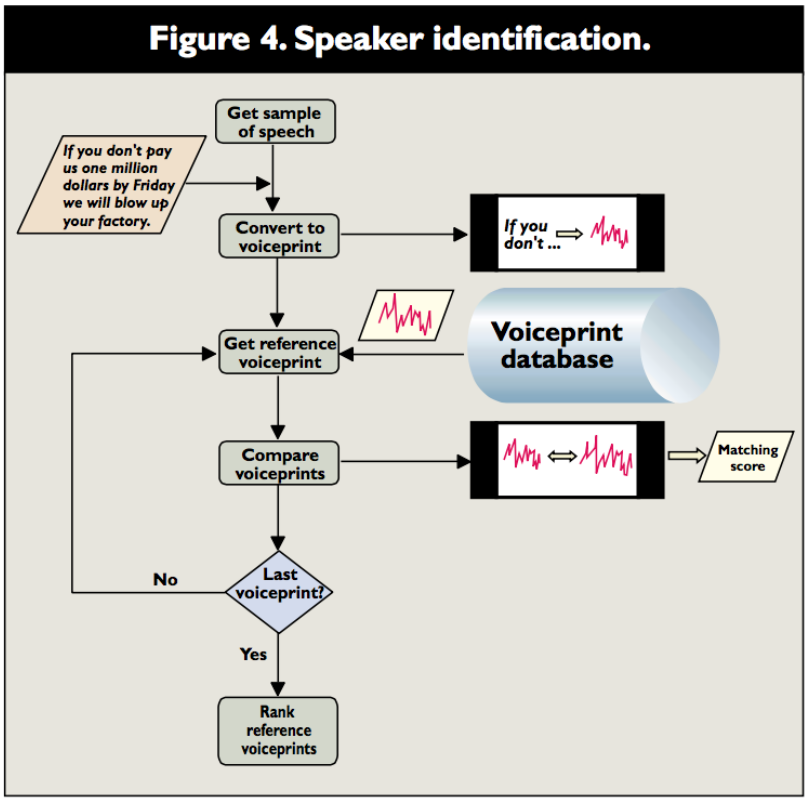

Figura 3.8: Schema di esempio di una tecnica di speaker identification

speaker conosciuti. Non viene fornito alcun identificativo e questo vuol dire che il riconoscimento si basa esclusivamente sull'impronta vocale fornita in input. Affinché questo metodo fornisca risultati attendibili, il set degli interlocutori acquisiti deve essere molto inferiore del potenziale numero di utenti che possono cercare di accedere al sistema. Questo è un sistema *text-indipendent*, in quanto al sistema interessa solo l'impronta vocale dell'utente, senza analizzare il lessico.

Speaker verification Lo scopo di questo metodo è quello di determinare se la persona è chi dice di essere. Inizialmente verrà chiesto all'utente un identificativo, come inserire un numero di conto, un codice id, oppure, un input vocale nel caso in cui il sistema necessiti di riconoscere delle parole e non solo l'identità. Questo passaggio permetterà al sistema di risalire all'impronta vocale della persona che sta chiedendo l'autorizzazione. In seguito l'utente dovrà inviare un file vocale che

verrà confrontato con l'impronta memorizzata nel database: l'esito della verifica ovviamente inficerà sull'autenticazione dello *speaker*. Questo implica che l'utente deve fornire un identificativo e il sistema deve accettare o rifiutare gli utenti in base all'esito della verifica. La robustezza di questi approcci è valutata in termini di falsi negativi e falsi positivi.

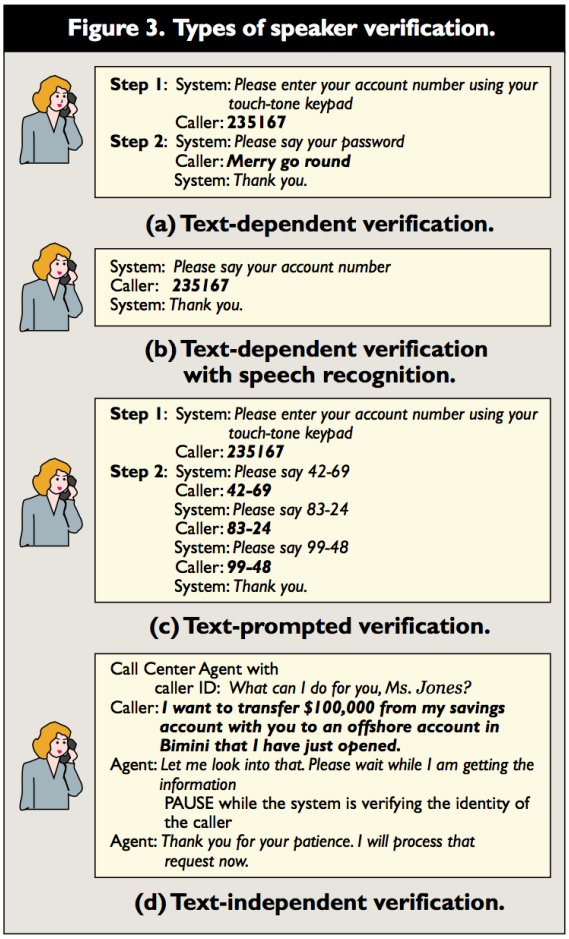

Figura 3.9: Schema di esempio di una tecnica di speaker verification

Capitolo 4

Il client

4.1 Funzionalità

Il vantaggio di racchiudere tutte le meccaniche di *computer vision* e gli algoritmi per l'analisi dei video nel lato server è stato sicuramente lo snellimento delle funzionalità da implementare a lato *client*. In particolare, in questo progetto, il compito delle applicazioni mobili è essenzialmente quello di inviare al *server* una serie di richieste tramite protocollo *HTTP* e gestire eventuali risposte. In questo capitolo analizzeremo un'applicazione realizzata per il sistema operativo *iOS*.

4.2 Caso di studio iOS

iOS è un sistema operativo per dispositivi mobili ideato e sviluppato da Apple Inc. ed installato su iPhone, iPod e iPad. Per sviluppare un'applicazione eseguibile su iOS, Apple fornisce un IDE di sviluppo chiamato Xcode il quale contiene una serie di *tool* per gestire e curare il software dalla creazione, al *testing*, all'ottimizzazione fino all'invio nell'App Store. Oltre alle funzionalità utili per gestire il flusso di lavoro ed organizzare al meglio il codice, Xcode include simulatore integrato che permette di testare l'applicazione su vari dispositivi virtuali, dall'iPhone all'iPad fino al recente Apple Watch. A partire dalla versione 7.0 è possibile installare e *debuggare* software direttamente tramite un dispositivo fisico, senza la necessità di aderire all'*Apple Developer Program*, ovvero un servizio a pagamento che Apple

mette a disposizione agli sviluppatori per fornire loro supporto e permettergli di pubblicare applicazioni sull'Apple Store. Xcode si rivela uno strumento essenziale anche per la cura del design e dell'interfaccia dell'applicazione, specialmente tramite l'uso dello *storyboard*.

4.2.1 La struttura : Storyboard

Uno *storybord* è una rappresentazione visuale dell'interfaccia utente di un'applicazione iOS. Esso è composto da una sequenza di scene, ciascuna delle quali rappresenta un *UIViewController* e le sue viste. Le transizioni tra le varie scene sono rappresentate con una segue.

Xcode fornisce un editor visuale per gestire gli *storyboard* e permette di definire il *layout* e il *design* della *user interface* semplicemente aggiungendo viste, bottoni, tabelle e testo. Benché nato per un approccio di tipo visuale, lo *storyboard* può essere acceduto e modificato anche tramite codice . Si mostrerà la struttura dell'applicazione attraverso l'analisi dello *storyboard*.

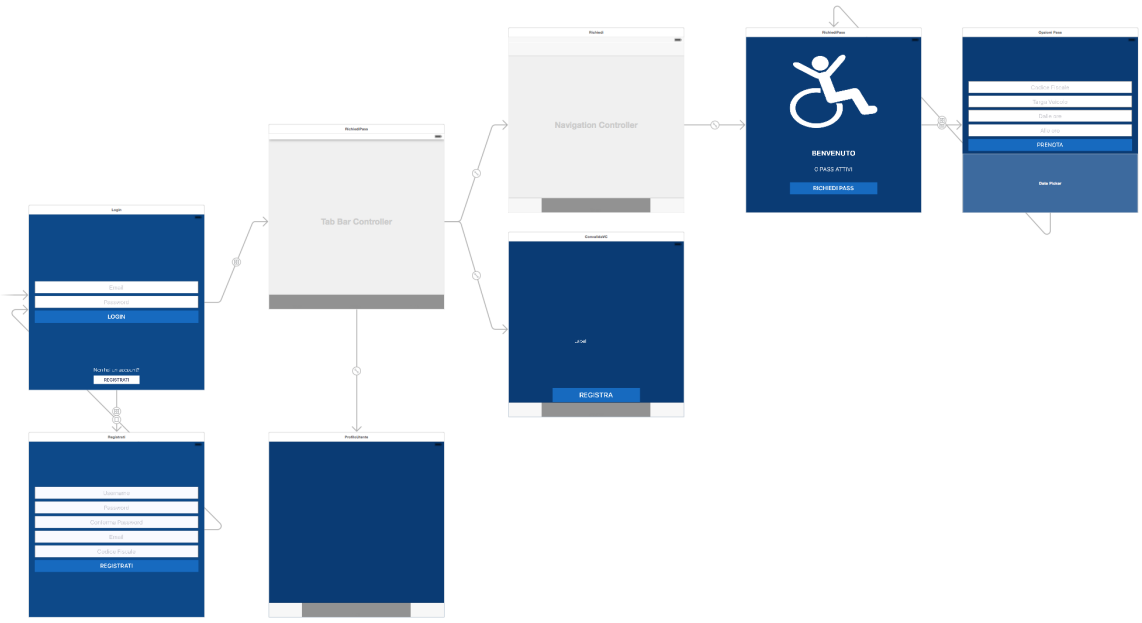

Figura 4.1: Lo storyboard

La schermata iniziale è costituita da un semplice *form* per gestire il *login* dell'utente accompagnatore. E' anche disponibile una funzione per registrarsi. Una volta effettuato il login si entra nel *core* dell'applicazione, costituito da un **Tab Bar View Controller**: esso implementa un particolare tipo di *view controller* che gestisce un'interfaccia costituita da una serie di etichette, o *tab*, racchiuse in una *Tab Bar* in fondo allo schermo. A ciascuna etichetta è collegata una particolare vista che apparirà sullo schermo in base alla selezione dell'utente. E' anche possibile associare un *view controller* annidato all'interno del *UITabBarController* per permettere all'utente di orientarsi meglio tra le viste. Nello specifico, nell'applicazione proposta sono presenti tre *tabs*: una per raccogliere le informazioni dell'utente, una per convalidare il contrassegno e infine una per richiederlo. Quest'ultima contiene un **UINavigationController**, il quale si occupa di creare uno *stack* di navigazione delle viste, ovvero un array di *view controller*, e di navigarle tramite *segue* normalmente associate ad un pulsante di *back* all'interno di un'apposita Navigation Bar. Una volta terminata la progettazione della *user interface* si procede all'implementazione delle classi che compongono l'applicazione. Generalmente, per sviluppare applicazioni iOS si utilizzano i linguaggi *Swift* e *Objective-C*. Quest'ultimo è il primo linguaggio di programmazione *object-oriented* usato per scrivere software per le piattaforme OSX e iOS. Esso eredita la sintassi e i tipi primitivi del C e aggiunge dichiarazioni per definire classi e metodi. Per fare un esempio, ogni *view controller* mostrato nello *storyboard* è implementato programmaticamente attraverso una sottoclasse di *UIViewController*. In seguito si analizzeranno nel dettaglio le classi di oggetti che sono maggiormente coinvolte nel progetto.

4.2.2 NsURLSession

Una classe utilizzata massicciamente all'interno dell'applicazione è sicuramente **NSURLSession**. Infatti essa permette di stabilire una connessione con il server e di inviare dati e richieste. La classe *NSURLSession* supporta nativamente la trasmissione di dati edi file e i protocolli FTP, HTTP e HTTPS. É possibile anche stabilire un protocollo di comunicazione personalizzato. In un'applicazione possono essere presenti una o più sessioni, ciascuna delle quali si occupa di una precisa

trasmissione dati, che prende il nome di *task*. Esistono tre tipi di *task*:

- *Data Task* il quale invia e riceve oggetti di classe NSData. Utilizzato specificamente per piccole richieste al server quali il *login*, *logout* e registrazione dell'utente, ma anche per la richiesta di creazione di un pass;
- *Upload Task* simili ai Data Task, ma inviano dati e file, e supportano l'*upload* in *background* quando l'applicazione non è attiva. Questo *task* viene utilizzato per inviare al server il video da analizzare;
- *Download Taks* scarica dati e file. Anch'essa supporta il *download* in *background*.

Ogni *task* è legato ad una NSURLSession e condivide con essa la sua configurazione, la quale definisce il comportamento della connessione:

- *Singleton shared session* è per le richieste molto semplici. Non è personalizzabile e si usa normalmente quando non sono richiesti particolari requisiti;
- *Default session* si comporta analogamente alla *shared session*, ma è personalizzabile e riceve i dati gradualmente tramite un delegate;
- *Ephimeral session* sono simili alle *default session*, ma non memorizzano *caches*, *cookies* o credenziali di autenticazioni sul disco;
- *Background session* permettono di svolgere l'*upload* e il download dei contenuti anche quando l'applicazione non è in stato di *running*.

Esempio di utilizzo di NSURLSession.

```
2 NSURLSession *session = [NSURLSession sharedSession];
  NSURLSessionDataTask *task =[session
4                         dataTaskWithRequest:request
                          completionHandler: ^(NSData *data,
6                                 NSURLResponse *response,
                                  NSError *error) {
8
```

```
        NSString *responseStringWithEncoded = [
10              [NSString alloc] initWithData: data
                        encoding:NSUTF8StringEncoding];
12
        NSDictionary *result =
14              [NSJSONSerialization JSONObjectWithData:data
        options:NSJSONReadingMutableContainers error:&error];
16
// Save myPass in UserDefaults
18    NSLog(@"User Defaults : %@", result);
    NSUserDefaults *myActualPass =
20        [NSUserDefaults standardUserDefaults];
    [myActualPass setObject:result forKey:@"myActualPass"];
22    [myActualPass synchronize];
     } ];
24        [task resume];
 }
```

4.2.3 AVFoundation

AvFoundation è uno degli innumerevoli *framework* che possono essere usati per riprodurre e creare audio, foto e video tramite un'interfaccia scritta in Objective-C. In questo progetto è stata utilizzata per registrare gli input della videocamera e del microfono dello *smartphone*. Tutte queste operazioni vengono gestite da una **AVCaptureSession**, la quale ha il compito di coordinare il flusso dei dati dai dispositivi input al file di output. Tutto quello di cui si ha bisogno è :

- un'istanza di **AVCaptureDevice** la quale rappresenta il dispositivo fisico volto a catturare l'immagine, e contiene tutte le proprietà di esso. Esiste un oggetto particolare per ogni tipo di dispositivo: uno per gestire l'input del microfono e due per gestire la fotocamera frontale e quella sul retro. Si usa per configurare le proprietà dell'hardware e fornisce i dati alla AVCaptureSession;
- un'istanza di **AVCaptureInput** , ovvero un oggetto che gestisce il flusso di dati dei dispositivi di input;

- un'istanza di **AVCaptureOutput** che serve per gestire l'output e convertirlo in un'immagine, in un video oppure un file audio.

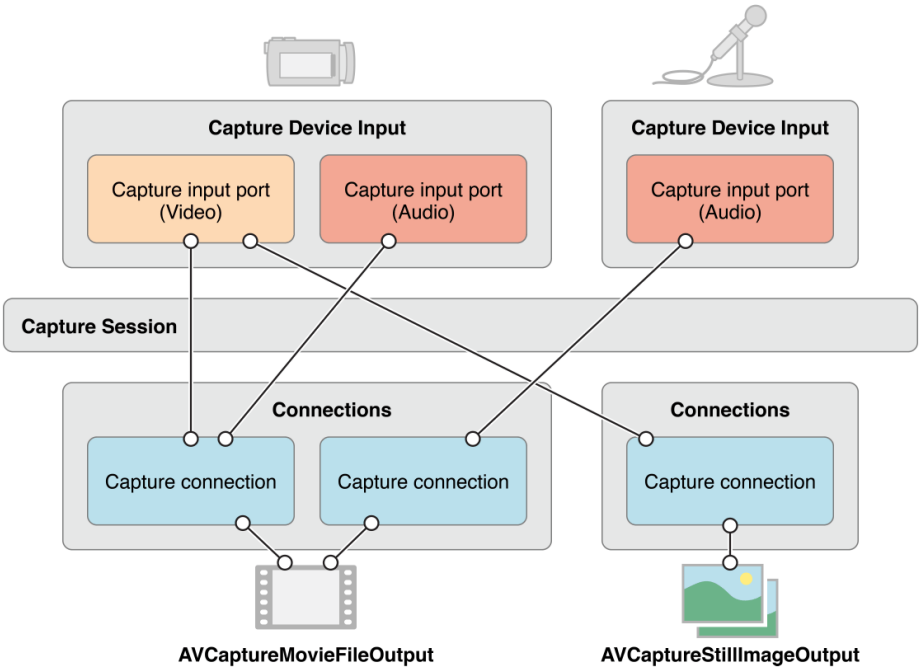

Figura 4.2: Struttura di una *AVCaptureSession*

Quando si aggiunge un input o un output ad una *AVCaptureSession*, si crea una connessione tra tutte le porte input compatibili a registrare dati e gli output registrati. Questa connessione è rappresentata da un **AVCaptureConnection**. Per registrare un video è necessario utilizzare un *delegate* conforme al protocollo **AVCaptureFileOutputRecordingDelegate**. Bisogna quindi implementare due metodi :

- `startRecordingToOutputFileURL:recordingDelegate` è il metodo che indica l'inizio della registrazione. I parametri dati in input sono rispettivamente il path del video da salvare e l'oggetto da usare come delegate. Nell'applicazione proposta questo metodo è invocato all'interno della funzione `(IBAction)takePhoto:(id)sender` in quanto per rendere l'interfaccia *user friendly* si è rivelato necessario collegare l'azione di registrare un video ad un bottone da premere per iniziare e fermare la registrazione;

- `captureOutput: didFinishRecordingToOutputFileAtURL: fromConnections: error:` è il metodo che indica le azioni da svolgere una volta terminata la registrazione del video. In input riceve il file che è stato appena scritto e il suo path, un array con tutte le *AVCaptureConnection* legate al video ed infine una variabile indicante eventuali errori. Appurato che non vi siano stati errori il video viene convertito in un flusso dati e inviato al server tramite una *NSURLSession*.

Capitolo 5

Possibili sviluppi futuri

Nei precedenti capitoli è stato esposto un possibile sistema per contrastare l'uso improprio del contrassegno per disabili. Tuttavia, la soluzione proposta deve essere considerata una base di partenza per la realizzazione di un software che garantisca sicurezza sia dal punto della protezione dei dati sensibili degli utenti, sia dal punto di vista della creazione di una *challenge* difficile da aggirare. Ecco quindi riportate in seguito una serie di idee che sono ritenute necessarie affinché il sistema effettivamente sostituirsi all'attuale pass cartaceo.

Innanzitutto si ritiene necessario il completamento della piattaforma per l'inserimento dei soggetti svantaggiati nel database del Comune. In particolare, bisognerà implementare un'interfaccia *user friendly* che permetta rapidamente agli impiegati del Comune di Bologna di registrare volti e voci. Per aumentare la sicurezza del sistema, questa piattaforma dovrà essere accessibile ai dipendenti esclusivamente tramite i computer degli uffici del Comune e solamente durante il proprio turno di lavoro. Queste limitazioni hanno lo scopo di cercare di evitare che un impiegato comunale possa connettersi tramite il proprio pc di casa e alterare i database corrompendo l'integrità del riconoscimento facciale e vocale.

Dopodiché risulta doveroso sviluppare la componente del *client* anche per sistemi operativi Android e Windows Phone, in modo da coprire la quasi totalità dei dispositivi mobili presenti in circolazione. Infatti, affinché il sistema possa effettivamente sostituire il contrassegno cartaceo,è necessario che tutti i cittadini con disabilità siano in grado di poter usufruire del servizio informatizzato,

indipendentemente dal modello di *smartphone* posseduto.

Un altro punto nodale che andrebbe approfondito è la gestione delle *challenge*. In particolare si ritiene che un solo test potrebbe non essere sufficiente per garantire l'effettiva presenza del disabile a bordo, in quanto l'accompagnatore potrebbe semplicemente effettuare il test prima di intraprendere il viaggio verso il centro. Per ovviare a questo si ritene necessario permettere all'applicazione mobile di generare richieste in *background* al server, in modo tale da ricevere una nuova *challenge* ogni t istante di tempo casuale compreso nelle ore di validità del pass. Il rischio di questa soluzione è che essendo t generato casualmente dal sistema, potrebbero essere generate n *challenge*, una per ogni minuto in cui il pass risulta attivo nel sistema. É evidente che questo approccio creerebbe un disagio sia nei confronti di una persona disabile, la quale verrebbe costantemente stressata dai continui test richiesti, sia nei confronti dell'accompagnatore il quale si ritroverebbe costretto ad accostare continuamente l'automobile per effettuare il test. Per superare questo scoglio, si propone di permettere all'applicazione di tracciare il percorso che il disabile dovrà attraversare per raggiungere il luogo di interesse: in questo modo risulterebbe possibile stimare il tempo di percorrenza del viaggio e permettere all'accompagnatore di posticipare l'esecuzione del test al momento in cui sarà giunto a destinazione. Questo meccanismo tornerebbe utile anche nella situazione in cui sia necessario richiedere un pass per andare a prelevare un soggetto affetto da disabilità che si trova già in centro urbano, permettendo all'utente di prenotare posticipare la verifica della prova a quando il disabile sarà effettivamente a bordo del veicolo.

Capitolo 6

Bibliografia

Articoli

Classsifying Facial Action di Paul Ekman, Joseph C.Hager,Marian Stewart Bartlett, Paul A. Viola, Terrence J. Sejnowski, Beatrice A. Golomb, Jan Larsen, Advances in Neural Information Processing Systems 8,823-829, 1996

A Practical Guide to Biometric Security Technology , di Simon Liu and Mark Silverman,IT Pro, IEEE, January-February 2001

State-of-the-Art in Speaker Recognition , di Enric Monte-Moreno , Marcos Faundez-Zanuy, May 2005, IEEE A&E SYSTEMS MAGAZINE

Eigenface-based facial recognition di Dimitri PISSARENKO,, December 1, 2002

EIGENFACES AND FISHERFACES , di Naotoshi Seo, University of Maryland ENEE633 Pattern Recognition Project 2-1

A Study of Various Face Detection Methods , di Ms. Varsha Gupta, Mr. Dipesh Sharma, International Journal of Advanced Research in Computer and Communication Engineering , May 2014, Vol. 3, Issue 5

Facial expression recognition from video sequences: temporal and static modeling di Ira Cohen,, Nicu Sebe, Ashutosh Garg, Lawrence S. Chen, and Thomas S. Huanga , Computer Vision and Image Understanding 91, 2003

Vision Based Hand Gesture Recognition di Pragati Garg, Naveen Aggarwal and Sanjeev Sofat, , World Academy of Science, Engineering and Technology 49, 2009

Eigenfaces for recognition di Matthew Turk and Alex Pentland, Journal of Cognitive Neuroscience, , Number 1, Volume3, 1991

Voice Biometrics di Judith A. Markowitz, COMMUNICATIONS OF THE ACM, No.9, Vol.43, September 2000

Sitografia

www.ilfattoquotidiano.it/2011/04/20/ di Gerardo Adinolfi, *Auto dei calciatori col pass invalidi. Di Vaio ai pm: "Ci sono cose più importanti"*, 20 aprile 2011

www.ilfattoquotidiano.it/2011/08/27/ Antonella Beccaria, *Scandalo pass invalidi dei calciatori. La Procura indaga sulle responsabilità di Atc*, 27 agosto 2011

www.poliziamunicipale.comune.verona.it/nqcontent.cfm?a_id=32481 , sito della polizia municipale di Verona

http://docs.opencv.org/ Documentazione OpenCV

http://www.aci.it/i-servizi/per-la-mobilita/aci-per-il-sociale/ sito dell'ACI