Università Politecnica Delle Marche

Dipartimento di Ingegneria dell'Informazione

LAUREA TRIENNALE IN INGEGNERIA INFORMATICA E DELL'AUTOMAZIONE

Parola di Motoneurone:

Progettazione e sviluppo di una applicazione web per monitorare l'andamento della disartria nei pazienti con SLA

Design and development of a web application to monitor the progress of dysarthria in patients with ALS

Relatore:

Dott. Adriano Mancini

Candidato: 1053507

Lorenzo Di Carlantonio

Anno Accademico 2019 / 2020

The greatest enemy of knowledge is not ignorance,
is the illusion of knowledge.

- Stephen Hawking -

Sommario

Questa tesi illustra le fasi di progettazione e sviluppo del *framework* di strumenti per il progetto **Parola di Motoneurone** .

Il progetto Parola di Motoneurone, promosso dall'Università Politecnica delle Marche e che mi vede partecipe tra i suoi ideatori e realizzatori, ha vinto il premio SIN - Biogen in occasione del congresso della Società Italiana di Neurologia 2020. L'intento del progetto è quello di fornire ai clinici e specialisti della riabilitazione uno strumento con cui valutare e identificare tempestivamente la disartria nel paziente affetto da Sclerosi Laterale Amiotrofica (SLA) bulbare.

La SLA è una malattia neurodegenerativa ad andamento progressivo che si traduce nella graduale perdita delle funzioni motorie e comunicative del soggetto che ne è colpito. Gli esordi più comuni della malattia sono quello spinale e bulbare.

Attraverso il monitoraggio costante dei pazienti è più semplice per il clinico specialista individuare per tempo un *trend* di peggioramento dovuto ai sintomi di esordio della malattia, riuscendo così a proporre un percorso di cura efficace e personalizzato.

Per controllare periodicamente come evolve la malattia i pazienti sono chiamati a sottoporsi ad esami strumentali, parliamo ad esempio di esami spirometrici per la valutazione delle funzionalità respiratorie oppure a valutazione mediante scale. È difficile mappare i progressi compiuti dal paziente se le valutazioni strumentali sono tutte frammentarie con risultati che seppur correlati non vengono integrati in nessun sistema informatico e le valutazioni cliniche sono per lo più occhiometriche e dipendenti dall'esperienza e dalla condizione emotiva

dell'esaminatore.

Il *framework* che compone il progetto Parola di Motoneurone, si concretizza in un insieme di applicazioni che hanno come propria finalità la promozione di una soluzione di monitoraggio della SLA da remoto e basata su misure oggettive.

Di seguito verranno trattati gli aspetti progettuali dei software e l'implementazione delle tecnologie che hanno portato me ed il team di colleghi con cui collaboro alla realizzazione del primo strumento orbitante attorno a Parola di Motoneurone, la *WebApp* per acquisizione dati multimediali.

Nel primo capitolo il lettore viene avvicinato al mondo della SLA e alle principali metodologie di monitoraggio dell'evoluzione della malattia attualmente utilizzate.

Nei capitoli secondo e terzo si indicano i processi che hanno portato all'individuazione delle specifiche richieste per l'adozione clinica del software e le tecnologie utilizzate per la sua realizzazione.

Seguono la descrizione della struttura del software e dei sistemi riservati al clinico ed al paziente. In conclusione, la tesi vuole essere anche una breve guida all'uso del software indicandone, inoltre, i possibili sviluppi futuri.

Le tecnologie richieste per la realizzazione della *WebApp* sono ormai ampiamente utilizzate in ambiti di vario genere. Si parla, infatti, di tecnologie che vedono come protagonisti i flussi di dati provenienti dai sensori di uno smartphone, le immagini catturate da comuni telecamere RGB e l'eloquio del paziente registrato da un microfono.

I dati raccolti dal dispositivo sono trasmessi ad una piattaforma *cloud* che li analizza e genera un responso utile al clinico per individuare precocemente l'andamento della malattia nel paziente, il grado di miglioramento a seguito della terapia attivata e la possibilità di monitorare con continuità l'evoluzione della malattia analizzando il *trend* del paziente in cura.

Il primo software rilasciato abilita l'utilizzatore ad un protocollo di riabilitazione remota in grado di acquisire i dati relativi allo *stream* audio/video di un paziente durante lo svolgimento delle sessioni di esercizio prescritte dal clinico. Il paziente registrerà la propria sessione di esercizi attraverso uno smartphone in

modalità *video-selfie* o altro dispositivo dotato di fotocamera. Attraverso i dati generati, si potranno addestrare algoritmi di Intelligenza Artificiale (IA) in grado di estrapolare informazione clinica rilevante.

In una prima fase di studio verranno selezionati ed inseriti in un programma di sperimentazione gruppi di pazienti che hanno acconsentito alla registrazione delle loro sessioni di riabilitazione.

I dati raccolti verranno conservati in formato digitale, così da consentire la costruzione di un *dataset* per l'addestramento degli algoritmi di IA.

Successivamente, si procederà con la validazione clinica dei risultati delle analisi degli algoritmi realizzati confrontandoli con le attuali tecniche di monitoraggio.

Una volta validato il software e gli algoritmi di IA si potranno modificare le modalità di conservazione dei dati, così da garantire la *privacy* del paziente; l'elaborazione dei dati ottenuti verrà, infatti, memorizzata in forma anonima e criptata accessibile solo al medico curante.

Nella sezione Link Utili della presente tesi verranno forniti tutti i collegamenti ipertestuali ai software cui ci si riferisce nella loro ultima versione disponibile.

Indice

Capitolo 1

Background clinico: la disartria nel paziente affetto da Sclerosi Laterale Amiotrofica

Questo primo capitolo intende avvicinare il lettore al mondo della Sclerosi Laterale Amitrofica (SLA), descrivere ciò che la malattia comporta nei soggetti che ne sono colpiti e le attuali metodologie di monitoraggio.
Si propone dunque una descrizione generale della SLA, il grado di incidenza sul territorio nazionale italiano e i segnali e sintomi di esordio caratteristici.
Successivamente, ci si sofferma sulla SLA ad esordio bulbare e sulla descrizione della disfagia come uno dei sintomi peculiari.

1.1 La Sclerosi Laterale Amiotrofica

La SLA, conosciuta anche come "malattia di Charchot", dal neurologo francese che per primo la definì verso la fine dell'800, è una malattia neurodegenerativa tipica dell'età adulta che si traduce in una progressiva perdita delle funzioni muscolari e bulbari.

Nei paesi occidentali, ed in particolare sul territorio Italiano, l'incidenza della SLA è compresa fra 1,7 e 2,3 casi per 100.000 abitanti all'anno, soglia molto

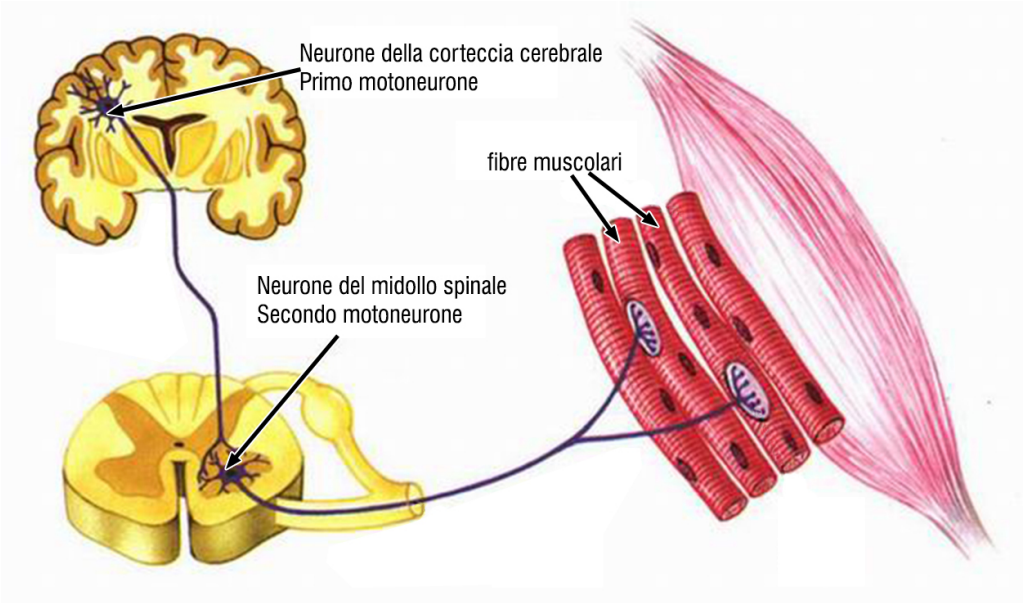

Figura 1.1: *Rappresentazione del I° e del II° motoneurone*[1]

maggiore rispetto a quanto rilevato da misurazioni analoghe effettuate in altri paesi. In altri contesti si parla infatti di una incidenza pari a 1 caso su 100.000 abitanti all'anno [2].

La principale caratteristica clinica della SLA è il coinvolgimento dei motoneuroni superiori (I° motoneurone) e inferiori (II° motoneurone) in più regioni del tronco encefalico e del midollo spinale (Fig. 1.1).

La compromissione dei motoneuroni nella corteccia motoria primaria, nel tratto controspinale e nel midollo spinale, provoca una graduale degenerazione delle condizioni del paziente [3].

I motoneuroni sono le cellule responsabili della contrazione della muscolatura volontaria preposta in primo luogo al movimento e presiedono anche funzioni vitali come deglutizione, fonazione e respirazione: la loro degenerazione comporta la paralisi progressiva dei muscoli da loro innervati [4].

Dal punto di vista clinico, il coinvolgimento dei motoneuroni superiori compromette la funzionalità degli arti e conduce alla spasticità, alla debolezza e a vivaci riflessi tendinei profondi.

Il coinvolgimento dei motoneuroni inferiori causa la presenza di fascicolazioni, atrofia e debolezza; ma si evidenziano anche fatica nella deglutizione e difficoltà nella parola, fino alla perdita della capacità di comunicare verbalmente quando la degenerazione coinvolge i motoneuroni del tronco encefalico.

A causa delle complicazioni che la malattia genera nel paziente, oltre il 50% dei malati cade in uno stato depressivo prolungato aggravando ulteriormente il quadro clinico complessivo [5].

1.1.1 Esordi delle malattia

Le manifestazioni cliniche della SLA sono il risultato della compromissione dei motoneuroni I° e II°, ma i segni e i sintomi della malattia, in particolare all'esordio, possono essere così eterogenei da rendere spesso difficoltosa una diagnosi accurata [6].

L'esordio può essere di tipo **spinale** o di tipo **bulbare** anche se in molti casi, specie nella fase iniziale, si presenta con sintomi aspecifici.
Nel dettaglio l'esordio:

- **Spinale** - nella quale vengono compromessi i motoneuroni del midollo spinale - interessa circa il 70% dei pazienti con SLA e si presenta con sintomi legati alla debolezza muscolare e atrofia focale, in cui i sintomi iniziali possono essere distali o prossimali degli arti superiori e inferiori. A poco a poco gli arti atrofici possono sviluppare spasticità, che colpisce l'abilità manuale e l'andatura;

- **Bulbare** - nei casi in cui la lesione è legata ai motoneuroni del tronco cerebrale/bulbare - la malattia interessa il restante 30% dei pazienti con SLA e si presenta con disartria e disfagia per solidi o liquidi. I sintomi agli arti, nella stragrande maggioranza dei casi, si verificano entro i primi 2 anni.

Tuttavia, questa distinzione di carattere clinico, utile per definire la comparsa della malattia, non appare sempre così netta nell'evoluzione della stessa in quanto le due forme possono sovrapporsi.

La progressione clinica, i sintomi di esordio e la gravità della malattia possono variare fortemente da un paziente all'altro.

È possibile che vi siano una pluralità di muscoli colpiti, una velocità variabile nel peggioramento del paziente e una diversa entità nella paralisi del soggetto.

Purtroppo diagnosticare la SLA è un compito difficile che richiede numerose indagini mediche e per ogni paziente l'evoluzione può essere valutata solo attraverso il controllo neurologico periodico (ogni 2-3 mesi), poiché non esiste un esame specifico con elevata accuratezza diagnostica e prognostica.

La diagnosi di SLA avviene dunque per esclusione: un neurologo esperto richiede indagini mediche e valutazioni cliniche ripetute nel tempo in un percorso diagnostico che prevede test neurologici ed esami strumentali. In Italia esistono dei centri clinici specializzati per le diagnosi di SLA e accreditati per la certificazione e la definizione del piano terapeutico assistenziale per la malattia (D.M. 279/2001) [7].

1.1.2 SLA bulbare e disartria

Nelle SLA bulbari la disartria è uno dei sintomi e segni di esordio della malattia e nonostante la sua evoluzione possa essere correlata anche al peggioramento di altri sintomi e segni bulbari come la disfagia [8], è una condizione ancora poco studiata e caratterizzata.

La disartria è un disturbo motorio del linguaggio, classificabile a seconda della neuropatologia sottostante e responsabile del mancato controllo dei muscoli che permettono di realizzare le articolazioni dell'eloquio, dunque, del direzionamento del flusso d'aria atto alla produzione della parola con conseguente deficit nel linguaggio e nella intelligibilità.

La disartria può essere differenziata in sei tipologie principali: disartria flaccida, spastica, atassica, ipocinetica, ipercinetica e mista, ognuna delle quali legata ad una specifica disfunzione motoria/cerebrale/cerebellare [9].

La valutazione funzionale della disartria è fondamentale (i) ad individuare il momento opportuno per indicare strategie di compenso alla disabilità comunica-

tiva, (ii) a monitorare l'andamento della malattia, (iii) come misura di *outcome* nei *trials* clinici [10].

1.2 Monitoraggio della disartria nei pazienti con SLA bulbare

Una modalità di valutazione della disartria nei pazienti affetti da SLA avviene mediante l'uso di specifiche scale di valutazione ed autovalutazione.

Il "profilo Robertson" [11] è una scala non sommativa per la valutazione clinica della disartria, composta da 71 *item* che valutano vari aspetti clinici correlati con l'articolazione verbale (Fig. 1.2).

Essendo una scala non sommativa i 71 *item* si comportano come se fossero scale *mono-item*.

La scala ha dei tempi di somministrazione non inferiori ad 1 ora e diversi sono stati i tentativi di ridurre la complessità e gli *item* investigativi.

Gli *item* sono raggruppati nelle seguenti sottocategorie:

- fonazione;
- respirazione;
- muscolatura bucco-facciale;
- diadococinesi;
- riflessi;
- articolazioni;
- intelligibilità;
- prosodia.

Ulteriormente alla scala di Robertson viene somministrata al paziente la compilazione di una scheda di autocertificazione (Fig: 1.3), ovvero una valutazione soggettiva delle caratteristiche della parola e delle difficoltà di eloquio nelle situazioni sociali, nonché le strategie di compenso adottate e le relazioni degli interlocutori.

In letteratura la maggior parte delle valutazioni della disartria nel paziente si basa su valutazioni qualitative come le scale sopra descritte che spesso sono limitate all'analisi delle sole caratteristiche vocali [12]. Altre metodologie utilizzano supporti tecnologici [13] che però non forniscono un quadro dettagliato delle caratteristiche della disartria del paziente con SLA.

Accurate analisi in tempo reale prevedono l'adozione di sensori. Alcuni dei sensori che vengono utilizzati sono reti palatali (Fig. 1.4) che però raramente si prestano ad una continuativa azione di monitoraggio durante la pratica clinica [14].

La natura fisica ed ingombrante della sensoristica, seppur miniaturizzata, provoca disagio nel paziente, immette nella loro fruizione, operazioni di costante manutenzione, personalizzazione e ricalibrazione che non consentono una adeguata scalabilità nell'uso continuativo di questi strumenti per un monitoraggio efficace e continuativo del paziente.

	ITEM		1	2	3	4
Muscolatura facciale	1. Simmetria facciale a riposo	a) Faccia				
	2. Cambio di espressione su sorriso					
	3. Mantenimento chiusura labiale a riposo	b) Labbra				
	4. Stiramento labbra					
	5. Protrusione labbra					
	6. Chiusura labiale durante la conversazione					
	7. Bocca aperta / bocca chiusa	c) Mandibola				
	8. Lateralizzazione mandibola a destra					
	9. Lateralizzazione mandibola a sinistra					
	10. Protrudere la lingua	d) Lingua				
	11. Retrarre la lingua					
	12. Muovere la lingua a destra					
	13. Muovere la lingua a sinistra					
	14. Passare la lingua sopra i denti					
	15. Punta lingua contro guancia destra					
	16. Punta lingua contro guancia sinistra					
	17. Alzare punta lingua nella bocca					
	18. Alzare punta lingua fuori della bocca					
	19. Elevazione velo del palato su /a/	e) Velo				
	20. Elevazione velo del palato su serie di /a/					

	ITEM		1	2	3	4
Diadococinesi	1. Aprire e chiudere la bocca rapidamente	N° in 5 sec.:				
	2. Protrudere e retrarre le labbra rapidamente	N° in 5 sec.:				
	3. Protrudere e retrarre la lingua rapidamente	N° in 5 sec.:				
	4. Alzare e abbassare punta lingua rapidamente	N° in 5 sec.:				
	5. Muovere la lingua da parte a parte rapidamente	N° in 5 sec.:				
	6. Ripetere "u-i" rapidamente	N° in 5 sec.:				
	7. Ripetere "pa..." rapidamente	N° in 5 sec.:				
	8. Ripetere "ta..." rapidamente	N° in 5 sec.:				
	9. Ripetere "ka..." rapidamente	N° in 5 sec.:				
	10. Ripetere "kala ..." rapidamente	N° in 5 sec.:				
	11. Ripetere "p.t.k...."rapidamente	N° in 5 sec.:				

	ITEM	1	2	3	4
Fonazione	1 Attacco su /a/				
	2. Emissione /a/ prolungata				
	3. Emissione/a/ intensità elevata				
	4. Crescendo su /a/				
	5. Diminuendo su /a/				
	6. Serie di /a/				
	7. Scala ascendente su /a/				
	8. Scala discendente su /a/				
	9. Glissato ascendente su /a/				
	10. Glissato discendente su /a/				
	11. Adeguata intensità di conversazione				
	12. Qualità vocale				

Figura 1.2: *Alcuni item della scala di valutazione del profilo Robertson. Si noti come la scala valutativa presenti solo 4 possibili stime: Scarso (1), Discreto (2), Buono (3), Ottimo (4). Le valutazioni vengono effettuate spuntando la casella corrispondente nella colonna a destra dell'esercizio eseguito [11].*

COGNOME_____ ***NOME***_____ ***DATA***_____

Questionario di Autovalutazione della Disartria
(Yorkston, Bombardier e Hammen, 1994; vers.it. Schindler e Gulli, 2002)

Item	Mai	*Quasi mai*	Qualche *volta*	Quasi *sempre*	Sempre
Caratteristiche della parola					
1. La mia voce suona rauca e aspra					
2. Il mio modo di parlare è lento					
3. Il mio modo di parlare è troppo forte o troppo debole					
4. Il mio modo di parlare ha una qualità nasale					
5. Ho difficoltà a parlare quando sono di fretta					
6. Quando sono stanco/a parlo peggio					
7. Quando parlo con estranei hanno difficoltà a comprendermi					
8. I miei familiari hanno difficoltà a comprendermi					
Strategie di compenso					
1. Faccio "tradurre" quello che dico ad amici o familiari quando gli estranei non mi capiscono					
2. Quando parlo con le persone mi assicuro che mi vedano bene (ad es. sto in luoghi ben illuminati o mi assicuro che siano di fronte a me)					
3. Se qualcuno non ha capito o ha capito male quello che ho detto, ripeto il messaggio usando altre parole					
4. Sottolineo l'argomento della conversazione in modo tale che il mio interlocutore mi capisca meglio					
5. Quando qualcuno non ha capito ripeto più forte o più lentamente					
6. Dico agli altri di segnalarmi quando hanno difficoltà a capirmi					
7. Evito di parlare con estranei a causa dei miei problemi di espressione					
8. Quando sono coinvolto/a in una importante conversazione, spengo la radio, la TV o altre sorgenti di rumore					
Totali:					

Figura 1.3: *Alcuni item della scala di autovalutazione della disartria [11].*

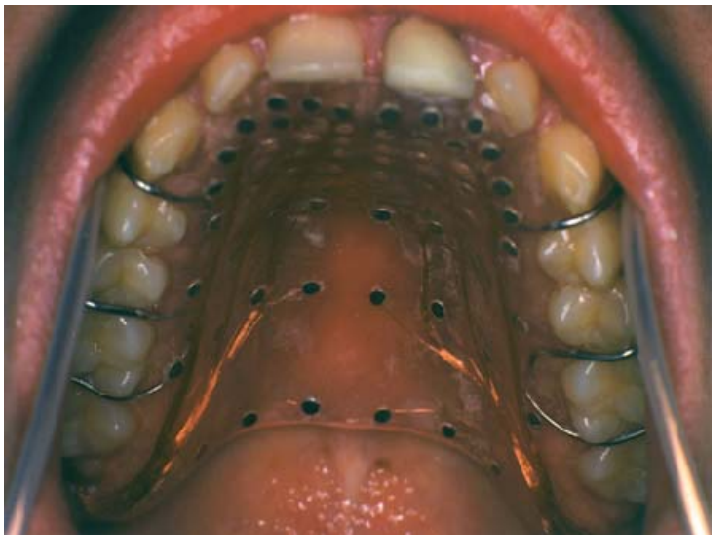

Figura 1.4: *Sensori utilizzati per l'elettropalatografia, il sensore permette di registrare il comportamento della lingua durante l'eloquio. L'elettropalatografia combinata alla logopedia convenzionale sono utili nel trattamento delle disartrie moderate e gravi ma la sensoristica deve essere realizzata ad hoc su ogni paziente, riducendo la scalabilità e le tempistiche di intervento [15].*

Capitolo 2

Parola di Motoneurone: descrizione ed analisi dei requisiti

Il secondo capitolo di questa tesi espone le caratteristiche fondamentali di Parola di Motoneurone e le necessità di clinici e pazienti che la soluzione andrà a soddisfare. Vengono descritte le funzionalità che si intendono implementare nel *framework*, le modalità di fruizione e la rappresentazione dei requisiti funzionali e non funzionali. Segue la descrizione delle fasi di progettazione del *framework* e si conclude con la relazione dettagliata del primo prototipo realizzato. Al termine del capitolo si forniscono i dettagli implementativi ed una panoramica delle tecnologie utilizzate.

2.1 Descrizione generale del progetto

Il progetto Parola di Motoneurone si pone come obiettivo quello di favorire il monitoraggio frequente dei pazienti affetti da SLA bulbare così da poter registrare e catalogare in modo oggettivo, non invasivo e da remoto, la progressione della loro disartria.

Il progetto Parola di Motoneurone è stato presentato durante il convegno della Società Italiana di Neurologia (SIN) 2020, ottenendo in questa sede il primo premio sponsorizzato da SIN e Biogen[1] denotando in questo ambiente il forte bisogno di sviluppare nuove tecnologie a supporto del monitoraggio e della prognosi della SLA.

Parola di Motoneurone garantirà ai clinici uno strumento per identificare con immediatezza i cambiamenti nelle *performance* dei pazienti ed accelerare l'attuazione di eventuali strategie correttive e compensatorie.

La prima interfaccia proposta è quella riservata al paziente. L'utente ha la necessità di visualizzare ed avviare la sua sessione di esercizi, durante la loro esecuzione, il software provvede a registrarli e a salvarli temporaneamente nel dispositivo. A sessione conclusa, i dati raccolti vengono impacchettati ed inviati alla seconda infrastruttura della soluzione, l'ambiente *cloud*.

Il *framework* si compone di tre ambienti distinti (Fig. 2.1), interconnessi tra loro mediante comuni protocolli di comunicazione delle informazioni (TCP/IP[2] e HTTPS[3])

In questa tesi viene trattato il primo dei 3 ambienti, la *WebApp* di acquisizione dei dati multimediali e l'implementazione di alcune logiche dell'ambiente *cloud* quali (i) login, (ii) registrazione e (iii) conservazione dei dati multimediali.

[1] Biogen: azienda di biotecnologie che agisce a livello globale

[2] TCP/IP: *Transmission Control Protocol*, è un protocollo di rete a *livello di trasporto* (4° ISO/OSI) che si occupa del trasporto a "pacchetti" dell'informazione. *Internet Protocol*, è un protocollo a *livello di rete* che si occupa dell'instradamento dell'informazione

[3] HTTPS: *HyperText Transfer Protocol over Secure Socket Layer*, è uno dei protocolli di comunicazione delle informazioni all'interno di una rete di telecomunicazioni

Figura 2.1: *Flusso delle informazioni:*
1) Il paziente si registra in modalità video-selfie ed invia i dati al cloud
2) Il cloud analizza i dati e li predispone per la consultazione del clinico

Il software per acquisire i dati delle sessioni di esercizio del paziente è stato concepito mirando alla semplicità d'uso. Al paziente, basterà infatti, scaricare una applicazione sul proprio *device* che, mantenendo accesi videocamera e microfono, lo guiderà nell'esecuzione di compiti vocali (ad es., vocalizzazioni, ripetizioni di sillabe in sequenza) o di azioni precise (ad es., aprire e chiudere la bocca velocemente). Inoltre, per esaminare l'intelligibilità del soggetto, come ultimi esercizi da svolgere, questi dovrà leggere una serie di parole nel miglior modo possibile.

I dati così raccolti verranno inviati al secondo strumento del *framework*, l'ambiente *cloud*. Questo ambiente riceve i dati e li consegna ad una serie di algoritmi di Intelligenza Artificiale (IA) (ad es., *Computer Vision* o analisi spettrografica della voce) così da generare delle valutazioni sugli esercizi effettuati dal paziente.

Una volta generati i report, questi potranno essere prelevati dal terzo ed ultimo sistema del *framework*, l'app per i clinici specialisti. Lo specialista potrà navigare nelle schede di dettaglio dei pazienti in cura, visualizzando uno storico delle sessioni effettuate ed una rappresentazione grafica del *trend* del paziente tra una sessione di esercizio e l'altra.

Grazie ad una corretta organizzazione delle informazioni ed alla possibilità di confrontare agevolmente i dati delle sessioni effettuate, sarà più semplice per lo specialista intraprendere azioni correttive e migliorare il percorso di monitoraggio del paziente.

La possibilità di dotare il paziente di uno strumento con cui poter effettuare i test clinici, da remoto e in completa autonomia, permette di erogare un maggior

numero di valutazioni, senza sovraccaricare la struttura clinica di competenza e senza obbligare il paziente a frequenti spostamenti presso la struttura stessa. Ciò deve essere considerato come un valore aggiunto all'uso del *framework* sopratutto nell'attuale periodo di restrizioni dovute all'emergenza sanitaria a causa della pandemia globale da COVID-19.

L'adozione del software presso le strutture sanitarie consentirà l'erogazione del servizio di monitoraggio con maggiore costanza, a frequenze settimanali o addirittura giornaliere, per i propri pazienti.

Sarà possibile impiegare al meglio le competenze del personale specializzato operando attraverso la valutazione clinica dei report generati dall'applicazione, senza che questi vengano coinvolti in operazioni di ricezione e trattamento dei pazienti in struttura.

2.2 Obiettivo della tesi

Con la presente tesi si vogliono descrivere le fasi di realizzazione del prototipo dell'applicazione web per l'acquisizione dei dati prodotti dal paziente durante lo svolgimento delle sue sessioni di esercizio. Successivamente il prototipo sarà integrato con le specifiche che si andranno a delineare a seguito della sua adozione presso i primi centri specializzati per il trattamento della SLA.

Pilastri del progetto sono il monitoraggio oggettivo, non invasivo e da remoto della progressione della disartria nei pazienti affetti da SLA bulbare.

L'ambiente *cloud*, sviluppato in collaborazione con altri colleghi dell'UNIVPM, è il cuore del sistema; esso, infatti, provvede alla trasformazione dei dati grezzi in informazioni rilevanti per il clinico. Mediante algoritmi di *computer vision* e *speech analysis*, i dati ricevuti vengono analizzati in modo da ottenere dettagli oggettivi sulle *performace* del paziente tra i vari esercizi eseguiti. L'analisi effettuata si va ad accorpare ai precedenti risultati andando, di fatto, a tracciare il trend evolutivo della disartria nel paziente in disanima.

Maggiore è la frequenza di esercizio, maggiori sono le probabilità di evidenziare peggioramenti ed applicare di conseguenza le terapie compensatorie.

Per permettere una comprensione migliore dei dati analizzati, questi verranno resi disponibili al clinico mediante il terzo ed ultimo elemento del sistema, l'interfaccia tecnica. Attraverso una *dashboard* dettagliata, i clinici specializzati potranno visualizzare in forma grafica i dati di ogni paziente, così da poter individuare agevolmente l'evoluzione della disartria.

2.3 Analisi dei requisiti

In questa sezione si procede analizzando nel dettaglio le funzionalità richieste per la messa in opera dei singoli sistemi, dividendo i requisiti funzionali da quelli non funzionali. In ultimo vengono fornite a integrazione delle varie sezioni, grafici e figure esplicative.

2.3.1 Requisiti funzionali

Si procede con la definizione dei requisiti funzionali richiesti dai 3 sistemi di interesse.

Use cases del paziente

Il sistema che si intende fornire al paziente, deve essere accessibile, possibilmente eseguibile su tutte le piattaforme disponibili (*Windows*, *Apple*, *Android*, ecc) e di semplice usabilità.

Si richiede, dunque, che il sistema sia distribuito nella forma di *Web Application*, accessibile attraverso i *browser* più diffusi per ogni sistema operativo (Chrome, Firefox, Safari, Opera, ecc).

- Il sistema deve consentire le comuni funzionalità di *login* e *logout* dalla piattaforma.

- Il sistema deve permettere la registrazione di un nuovo account (tipologia paziente)

La registrazione di un nuovo account prevede il compimento di due fasi:
I) Registrazione account:

- definizione email e password;
- invio di email di conferma alla casella di posta elettronica specificata;
- accettazione dei termini e condizioni di servizio e successiva convalida della email di registrazione.

II) Inserimento informazioni utente:

- successivamente alla verifica della email il sistema dovrà richiedere l'inserimento di ulteriori informazioni da parte dell'utente, come nome, cognome, età e sesso.

- Il sistema deve poter comunicare con la telecamera ed il microfono del dispositivo.

- Il sistema deve determinare *codec* e *mimeType* supportati dal dispositivo ed applicare la migliore combinazione che garantisca un discreto livello di standardizzazione dei dati raccolti

- Il sistema non deve necessariamente essere collegato ad internet nella fase di esercizio, ma deve potersi connettere durante il *download* degli esercizi da eseguire e durante l'*upload* delle registrazioni degli stessi.

- Ogni esercizio proposto deve essere avviato a seguito di una azione esplicita dell'utente che avvierà un countdown (es. pressione del bottone *play*). Allo scadere del *countdown* sarà possibile registrare l'esercizio.

- Il meccanismo di *countdown* deve poter essere interrotto, ovvero ripristinato, così da consentire all'utente la migliore esecuzione possibile dell'esercizio

- L'esercizio deve essere mostrato a schermo in modo chiaro e conciso in forma testuale e, ove necessario, in forma audiovisiva.

- La *WebApp* deve mostrare costantemente, in una porzione dello schermo ben visibile, lo *stream* video dell'utente così che questo possa posizionarsi in maniera ottimale per l'esecuzione della sessione di esercizi.

Al termine della sessione è necessario inviare i dati raccolti verso le unità di elaborazione dell'ambiente *cloud*. Durante questo processo è necessario informare in modo chiaro l'utente di non chiudere o terminare la *WebApp* e di attendere il completamento dell'operazione. Inoltre, è necessario che venga inibita ogni altra funzionalità della *WebApp* per tutta la durata dell'*upload*.

L'*upload* dei file deve seguire queste fasi:
I) Salvataggio in locale degli esercizi eseguiti;
II) Al termine della sequenza degli esercizi il dispositivo deve inviare, attraverso il protocollo HTTPS, i file di esercizio archiviati localmente;
III) Al termine dell'*upload*, e successivamente alla risposta di avvenuta ricezione, la *WebApp* deve rimuovere i file temporaneamente salvati nel dispositivo.

Al termine delle operazioni di *upload* delle registrazioni l'utente potrà iniziare una nuova sessione di esercizio o navigare le altre funzionalità dell'app.

Use cases del clinico

Il sistema riservato all'uso da parte di personale clinico specializzato dovrà mostrare l'elenco dei pazienti in cura e i dettagli di ognuno in pagine a sé stanti.

Il sistema possiederà due ruoli utente con funzionalità specifiche. Di seguito vengono descritti i due livelli di utenza e le loro funzionalità:

- **Specialista di reparto**
  - Modifica del proprio profilo;
  - CRUD[1] pazienti;
  - CRUD schede di esercizio per i pazienti in cura;
  - Assegnazione delle schede di esercizio ai pazienti in cura;
  - Selezione del paziente di cui si vogliono visualizzare i dettagli;
  - Ricerca di un paziente attraverso le tag[2] associate;
  - Visualizzazione pagina paziente.

[1] CRUD: *Create*, *Read*, *Update*, *Delete*, sono le 4 operazioni di base tipiche in ogni struttura dati.

[2] Tag: marcatore testuale di facile utilizzo per la definizione di filtri ed aggregazioni

- **Responsabile del reparto ospedaliero**
  - Eredita le funzionalità dello **specialista di reparto**;
  - CRUD completo di clinici e pazienti;
  - CRUD Tag;
  - CRUD dettagli entità Reparto;
  - Assegnazione dei pazienti ai clinici.

La visualizzazione della pagina di un paziente permette le seguenti funzionalità:

- Visualizzare le sessioni effettuate e rappresentazione grafica del *trend* di esecuzione;
- Visualizzare per ogni esercizio i dati restituiti dall'elaborazione del segnale grezzo;
- Permettere il confronto tra indici assimilabili.

Funzionalità *cloud*

Riguardo i processi che deve intraprendere il sistema di elaborazione, è necessario che:

- Si possano prendere in carico le analisi di diversi stream audio/video;
- Siano differenziate le analisi per ogni esercizio;
- Siano prodotti responsi standardizzati e codificati in formato *json*[1];
- Siano salvati in un database i risultati delle analisi ed associati al paziente a cui appartengono;
- Vengano distrutti i dati grezzi, a meno che non vi siano ulteriori permessi che ne approvino la conservazione per una successiva consultazione;

[1] JSON: formato di interscambio dei dati di particolare efficacia per la sua notazione sintetica. È frequentemente utilizzato per il trasporto delle informazioni su protocolli HTTPS

- Venga segnalata la presenza di dati *malformed* o non completi così da consentire l'adozione di politiche di risoluzione.

2.3.2 Requisiti non funzionali

Di seguito verranno illustrate le specifiche non funzionali per la realizzazione di Parola di Motoneurone. A tal proposito si intende utilizzare uno *stack* tecnologico basato sui servizi Google.

Per quanto riguarda la messa in sicurezza dei dati si farà affidamento ai protocolli di trasferimento dell'informazione criptati come HTTPS e WSS[1].

Come integrazione dei criteri di sicurezza adottati si prevede la cifratura attraverso primitive crittografiche come l'***sha-256*** dei dati destinati all'archiviazione nei *database*. L'accesso alle risorse sarà veicolato a mezzo di specifiche *Access Control List (ACL)* di cui si prevede una prima divisione in:

- **Amministratore di sistema** - operatori della società che fornirà i servizi connessi con Parola di Motoneurone;
- **Specialista di reparto** (clinico);
- **Responsabile di reparto**;
- **Paziente**.

I sistemi collegati a Parola di Motoneurone dovranno in futuro implementare anche le seguenti funzionalità:

- La *WebApp* deve mostrare una pagina in cui sono raccolte le informazioni condivise dall'utente e deve permetterne la modifica. (cambio password, rimozione metodi di accesso attivi, cambio email, accesso a due fattori, ecc);
- L'utente può in ogni momento eseguire una azione di conclusione anticipata dell'esercizio. In questo caso l'esercizio non verrà trasmesso e non sarà storicizzato nel profilo dell'utente;

[1] WSS: *Web Socket Secure*, è un protocollo per il trasferimento delle informazioni in tempo reale

- L'interfaccia utente deve permettere di visionare lo storico delle sessioni effettuate;
- L'interfaccia grafica deve essere minimale e concisa, bisogna evitare che durante l'esecuzione degli esercizi l'utente possa inavvertitamente concludere una sessione o invalidare il contenuto delle registrazioni effettuate;
- L'interfaccia deve consentire l'interazione differita tra medico e paziente. Entrambi si potranno avvalere di strumenti come "chat, chat-bot e reminder".

2.4 Descrizione dell'ambiente tecnologico - Ambiente Google

2.4.1 Flutter

Per la realizzazione della *WebApp* si è scelto un approccio *Google based*. La scelta del linguaggio per la realizzazione del Front End è ricaduta su Dart [16] con il suo *framework* Flutter [17].

La scelta di utilizzare Flutter si basa sulla sua semplicità d'uso. Il *framework* permette di creare delle interfacce dalla grafica accurata e fluida senza bisogno di eccessive implementazioni, possiede nativamente una forte standardizzazione e propone due modelli ampiamente affermati di resa grafica, la libreria Cupertino per i *Widget* Apple e la libreria Material per quelli Android.

Ultimamente Flutter si è aperto anche alla compilazione per il web, dotando il *framework* di utili librerie per la transcodifica in JavaScript. Proprio quest'ultima tecnologia è alla base della *WebApp* che si è realizzata per questo progetto.

Flutter web permette di costruire una *WebApp* fornendo tutti gli strumenti necessari affinché questa venga riconosciuta dai browser come una *Progressive WebApp* (PWA).

Come tale, la PWA può essere installata sul dispositivo del paziente senza bisogno di passare da alcuno *store* simulando una vera e propria applicazione Nativa.

Attraverso il *Service Worker* installato nel browser, la *WebApp* rimane in ascolto degli eventuali aggiornamenti distribuiti sull'*host* e mantiene attive tutte le funzionalità dell'app.

Con l'aggiornamento a Flutter 2 il supporto per la transcodifica per il web è uscita dalla fase sperimentale ed è pronta per la distribuzione di soluzioni software a livello *production*.

Attraverso questo aggiornamento è stata data agli sviluppatori la possibilità di sfruttare anche il paradigma *Null Safety* che permette di strutturare il codice in maniera più sintetica e "pulita".

Come si evince dalla figura 2.3 il software rispecchia tutti i requisiti minimi affinché possa essere distribuita secondo il *pattern Null Safety*. Questo *pattern* assicura che ogni porzione di codice sia robusta alla presenza di valori non definiti (appunto *Null*) che comprometterebbero l'esecuzione delle istruzioni.

La struttura fortemente modulare di Flutter permette di costruire applicazioni facilmente scalabili. Inoltre, la possibilità di transcodifica consente di realizzare un unico codice da cui successivamente generare le *build cross-platform* di cui si necessita.

Unica accortezza sarà quella di diversificare le librerie utilizzate in modo che siano compatibili con il dispositivo *target* per cui si intende generare la distribuzione.

I componenti base che Flutter ci mette a disposizione sono i *Widget* e questi si dividono tra "StateLess" o "StateFull". Il primo è un componente senza stato, ossia un contenitore di informazioni che non variano nel tempo, l'altro invece è un contenitore che possiede uno stato e che si ridisegna ad ogni suo cambiamento.

L'albero delle dipendenze che si viene a creare sfrutta il *pattern top-down* per cui una modifica dei nodi più alti dell'albero provoca l'update dei componenti figli fino ad arrivare alle foglie del grafo.

2.4.2 Firebase

La *WebApp* non presenta un vero e proprio Backend, ma si è scelto di adottare il servizio *Firebase* che funge da "Backend as a Service" (BaaS), ossia l'implemen-

tazione di tutte le meccaniche normalmente affidate al backend vengono prese in carico da un servizio *cloud* che provvede a fornire le funzionalità di base di cui si necessita. Dato che la *WebApp* non prevede implementazioni particolari a carico del backend, se non il semplice CRUD di dati, si è scelto di utilizzare Firebase come BaaS, lasciando per gli algoritmi più particolari la possibilità di essere strutturati come microservizi e distribuiti attraverso le *Functions* di *Google Cloud*.

2.4.3 Authentication

Grazie all'adozione di Firebase, tutta la parte che riguarda l'autenticazione è presa in carico dal servizio *Authentication*. Il servizio consente di selezionare una moltitudine di metodi di autenticazione e di implementarli semplicemente nell'applicazione attraverso l'SDK fornito dal servizio di Google.

Per lo sviluppo di Parola di Motoneurone si è scelto di sfruttare l'autenticazione attraverso due servizi:

- **Email/Password** - Il servizio permette di definire un nuovo account tramite l'inserimento di email e password. Successivamente alla creazione del nuovo account, l'utente dovrà verificare la proprietà della casella di posta elettronica dichiarata. A seguito di ciò l'account sarà attivo ed utilizzabile dall'utente.

- **Account Google** - Il servizio permette di sfruttare il proprio account Google per accedere all'applicazione. I metodi di login tramite *social* permettono una più rapida fruizione dei servizi consentendo allo sviluppatore di accedere anche ai dati che l'utente ha condiviso sul suo account (Per lo sviluppo di questo progetto non è stato fatto uso dei dati contenuti nell'account *social* ma si è solo data dimostrazione di come sia possibile sfruttare anche questo meccanismo di autenticazione).

Per poter adottare questi sistemi di autenticazione, Google richiede che venga dimostrata la proprietà dell'host che ospita l'applicazione fornendo un file **.html**

da inserire nella cartella *root* del progetto così che i *crawler*[1] del servizio possano trovarlo ed associarlo al servizio di autenticazione.

Inoltre viene richiesto che l'applicazione sia accessibile attraverso il protocollo HTTPS e che vengano specificati i domini autorizzati ad accedere al servizio di autenticazione.

2.4.4 FireStore

Il servizio database utilizzato per lo sviluppo di Parola di Motoneurone è Google FireStore. Di seguito viene fornito lo schema che si è definito per la rappresentazione delle varie entità che concorrono al conseguimento delle logiche di business. Il database offerto da Google permette di mantenere sincronizzati i dati tra i vari *client* attraverso *listener* in tempo reale. I *pillars* che la documentazione di FireStore espone sono:

- Flexibility - grazie ad un modello gerarchico è possibile definire oggetti strutturati in forma di documenti organizzati in collezioni;

- Expressive querying - si ha la possibilità di interrogare i documenti anche se questi presentano proprietà nidificate ad ogni livello di profondità. Si ha la possibilità di eseguire *query* concatenate e filtrate su più parametri, anche nidificati;

- Realtime updates - si potranno informare in tempo reale i *client* che si sono sottoscritti ai cambiamenti di stato delle informazioni conservate nel database;

- Offline support - sarà possibile offrire il servizio indipendentemente allo stato della rete del *client*. Quando si tornerà ad avere una connessione disponibile verrà aggiornato lo stato dei dati remoti.

[1] Crawler: Applicazioni utilizzate dai browser che scandagliano le pagine internet censite in rete per l'indicizzazione dei loro contenuti

- Designed to scale - essendo un servizio ospitato su *cloud* è possibile sfruttare tutte le meccaniche di scalabilità e replica offerte da Google per mantenere l'applicazione sempre reattiva e raggiungibile.

Google distribuisce l'SDK con cui è possibile interfacciarsi con i suoi servizi in diversi linguaggi di programmazione. Per questo progetto è stato scelto di utilizzare l'sdk distribuito per il linguaggio Dart.

2.4.5 Storage

Il servizio per conservare i file di esercizio dei pazienti è Google *Storage*. Il servizio di archiviazione prevede nativamente sistemi di accesso autenticato alle informazioni contenute, fornisce API per il *download* e l'*upload* dei file da e verso i server Google.

Funzionalità chiave del servizio è la robustezza delle operazioni, grazie a sistemi di controllo della connettività di rete, il servizio è in grado di trasferire dati indipendentemente dalla qualità della rete, interrompendo e riavviando il trasferimento qualora fosse necessario. Inoltre, *Storage* si integra perfettamente con le meccaniche di sicurezza del servizio *Authentication* con il quale è possibile definire una utenza automatizzata e legata al singolo *client* che permette l'autorizzazione alla lettura e alla scrittura dei file sul server.

Per il progetto Parola di Motoneurone è stato utilizzato questo servizio per l'archiviazione degli esercizi eseguiti dai pazienti. Ad ogni paziente è stata associata una *directory* personale in cui storicizzare tutti gli esercizi compiuti.

Successivamente, i file verranno analizzati tramite algorimti di IA e i risultati verranno inseriti in formato **.json** nel database FireStore.

2.4.6 Functions

Il servizio *Functions* di Firebase permette l'esecuzione di funzioni *serverless* direttamente su *cloud*. Il servizio permette di definire *trigger* in grado di scatenare specifiche funzioni. Questa funzionalità sarà utilizzata per scatenare l'esecuzione degli algoritmi di IA successivamente all'*upload* dei file di esercizio.

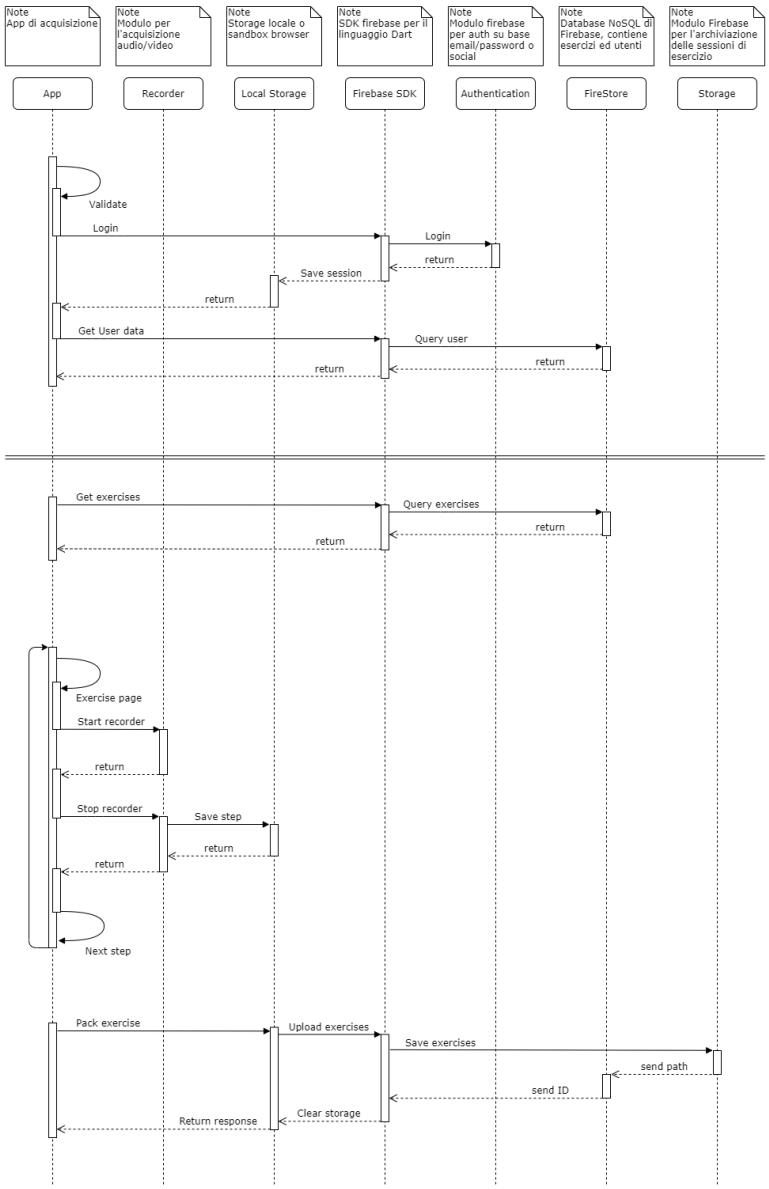

Figura 2.2: *Flussi della WebApp di acquisizioni degli stream audio/video dei pazienti in cura*

```
C:\Users\loren\IdeaProjects\motorneuron_word (dev -> origin)
λ flutter build web

 Building with sound null safety

Compiling lib\main.dart for the Web...
```

Figura 2.3: *Porzione della cli di IntelliJ che indica che il software utilizza il paradigma Null Safety*

Capitolo 3

Sviluppo del progetto

In questo capitolo si descrive la struttura del progetto Parola di Motoneurone e le funzionalità ad esso associate. Si descriverà la struttura individuata per il *database* e la catena di eventi che si scatena dal *login* dell'utente fino alla conclusione degli esercizi. Successivamente si descrivono le operazione che si susseguono alla ricezione dei file di esercizio fino alla distribuzione dei dati elaborati sulla *dashbord* del clinico.

3.1 Architettura a microservizi

Grazie ai servizi *Funtions* offerti da Firebase è possibile sviluppare le logiche di business attraverso il *pattern serverless* e microservizi. Verranno dunque strutturate tutte le funzionalità di analisi dei dati attraverso funzioni che eseguiranno gli algoritmi di IA in grado di estrapolare informazioni clinico rilevanti dai file di esercizio sostenuti dai pazienti.
Attraverso la piattaforma *cloud* offerta da Google ed in particolare i servizi Firebase, Parola di Motoneurone potrà garantire continuità e robustezza nella fruizione dei servizi.

Come si evince dalla figura 3.1, il *core* dell'applicazione è supportato dalla piattaforma *cloud* cui fanno capo sia la *WebApp* di acquisizione riservata ai pazienti, sia il sistema di consultazione del clinico specialista.

Figura 3.1: *Architettura cloud di Parola di Motoneurone*

All'interno del *cloud* verranno archiviati gli *stream* audio/video dei pazienti, successivamente, verranno analizzati estraendone informazione clinica rilevante.

3.2 Modellazione del *database*

Conclusa la fase di analisi dei requisiti si può affrontare la modellazione della struttura dei dati che serviranno ad attuare le logiche di business dell'applicazione.

Al termine del processo di normalizzazione del *database*, sono state individuate 10 entità minime:

- **Ospedale** - Entità che rappresenta l'acquirente del software;
- **Reparto** - Entità che permette di modellare i vari reparti di cui l'ospedale si compone;
- **Clinico** - Ogni Reparto ha dei clinici che vi lavorano. Questa è l'entità che rappresenta lo specialista che seguirà i pazienti e che utilizzerà la *WebApp* dedicata alla visualizzazione dei dati elaborati dalla IA e con cui potrà migliorare la sua capacità di intervento;
- **Paziente** - Entità che rappresenta l'utilizzatore della *WebApp* di acquisizione associata a Parola di Motoneurone. Il suo clinico di riferimento definisce gli esercizi che deve compiere. Ad ogni sessione eseguita, il sistema provvederà a popolare la cartella di *Storage* ad esso associata;

- **Utente** - Entità di base che permette di inserire le informazioni condivise tra paziente e clinico. L'entità conterrà Nome, Cognome, Sesso e l'indice univoco associato all'account generato con **Authentication** di Firebase. Attraverso questa entità sarà possibile recuperare le informazioni di contatto successivamente al *login* dell'utente;

- **Tag** - Attraverso questa entità il sistema sarà in grado di offrire, lato clinico, un sistema di filtri ed aggregazioni tramite marcatori testuali;

- **Sessione** - Entità collegata alle sessioni effettuate da ogni paziente. Della sessione fanno parte anche le informazioni relative al *path* generato nel servizio **Storage** da cui sarà poi possibile estrarre gli esercizi eseguiti dal paziente ed inviarli alle interfacce di IA per la loro analisi;

- **Indici** - Attraverso questa entità sarà possibile definire un gruppo di indici di interesse clinico associati ad ogni esercizio. Questa entità rappresenta il *core* di Parola di Motoneurone poiché verrà popolata dagli algoritmi di IA a seguito dell'analisi degli esercizi audio/video eseguiti dai pazienti. Successivamente, l'entità verrà interrogata dall'app del clinico per estrapolare le informazioni di interesse e rappresentarle attraverso grafici e tabelle;

- **Esercizi** - Entità che raccoglie le informazioni degli esercizi che il paziente dovrà eseguire;

- **Template patologia** - Entità che si prevede possa aiutare i reparti ospedalieri a definire dei *template* di esercizi associabili a profili patologici ben distinti che aiuteranno i clinici nella prescrizione degli esercizi da far compiere ai propri pazienti.

Di seguito si mostra come le entità sono tra loro collegate e successivamente viene fornita una rappresentazione più dettagliata contenente le proprietà di ogni entità sopra descritta.

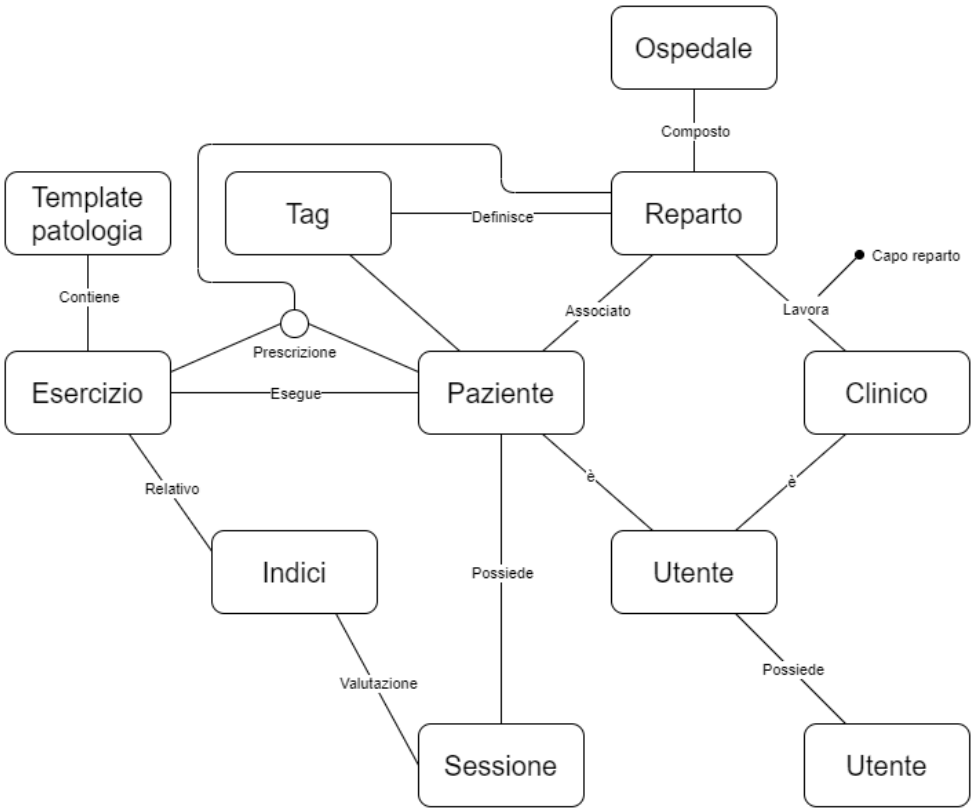

Figura 3.2: *Rappresentazione schematica dei collegamenti tra le varie entità di Parola di Motoneurone.*

Capitolo 4

Implementazione dell'applicazione Parola di Motoneurone

Al momento della stesura di questo documento è stata portata a termine la realizzazione del primo modulo del sistema Parola di Motoneurone, la *WebApp* di acquisizione riservata ai pazienti in cura. Di seguito si descrivono i passaggi implementativi che hanno portato alla distribuzione dell'applicazione ed alla sua prima adozione presso le strutture facenti capo ad *Ospedali riuniti di Ancona*.

4.1 Applicazione Web per il paziente

4.1.1 Inizializzazione dell'applicazione e *routing*

Come prima operazione la *WebApp* provvede ad inizializzare il servizio Firebase importando le librerie necessarie e avviando le meccaniche di sincronizzazione con l'account Firebase collegato all'applicazione.

Attraverso i parametri di configurazione, di cui si può vedere uno *snippet* di dettaglio in figura 4.1, l'applicazione si collega ai vari servizi in *cloud*.

Successivamente, vengono istanziate le classi `Network`, `Database` e `Auth` che permetteranno, rispettivamente, la connessione verso l'esterno, le *query* al *database* e le regole di ACL per la visualizzazione delle informazioni (Fig 4.2).

Il componente base dell'applicazione è il *MaterialApp* su cui viene configurato il tema principale e la sua variante per la *darkmode*.

Il sistema di *routing* all'interno della *WebApp* viene gestito tramite la configurazione del parametro `onGenerateRoute` (Fig 4.3).

4.1.2 Registrazione e *login*

Non appena il caricamento della configurazione di base termina, la *WebApp* propone all'utente una schermata con cui potrà effettuare il *login* o la registrazione (Fig 4.4 - immagine di sinistra).

Nel caso l'utente non possegga un account valido potrà crearne uno nuovo fornendo *email* e *password* e cliccare il pulsante "Usa queste credenziali per registrarti".

Il servizio Firebase *Authorization*, provvederà ad inviare una email di verifica alla casella di posta elettronica specificata.

L'utente non sarà in grado di accedere alla *WebApp* fintanto che non avrà correttamente dimostrato di avere pieno possesso della casella di posta elettronica (Fig. 4.5). Dovrà quindi entrare nella sua casella di posta e cliccare sul link di verifica inviatogli dal sistema (Fig 4.6). L'utente sarà portato sulla pagina mostrata in figura 4.7 ed il suo account verrà attivato.

Terminata la fase di verifica della casella di posta, l'utente potrà effettuare nuovamente il *login*.

A questo punto il sistema cercherà le informazioni associate all'account che si è appena loggato inoltrando una richiesta al servizio *cloud FireStore*.

Se il sistema risponderà positivamente allora l'utente verrà trasferito sulla pagina di inizio sessione, viceversa, verrà trasferito sulla pagina *continue registration* in cui potrà inserire le informazioni richieste per completare la registrazione (Fig. 4.8).

4.1.3 Interfacce di comunicazione verso l'ambiente esterno all'applicazione

Tutte le funzioni descritte nella sezione precedente sono rese possibili dalla classe `Auth` che espone all'applicazione i metodi di *login*, registrazione e associazione delle informazioni aggiuntive con l'account creato.

Le classi `Network`, `Database` e `Auth` sono progettate in modo da mantenere separate le funzionalità da loro esposte dalle logiche che le implementano.

In questo modo è possibile sostituire totalmente il sistema esterno che realizza le funzionalità senza che l'applicazione ne risenta internamente.

La realizzazione di queste interfacce permette di cambiare agevolmente i servizi *cloud* associati all'applicazione in favore di un nuovo *provider* di servizi, basterà infatti modificare le sole classi `Network`, `Database` e `Auth`, adattandole al nuovo ambiente, ma lasciando invariate le funzionalità da loro esposte al resto del sistema.

Lo sviluppatore potrà così utilizzare i metodi esposti da queste 3 classi sapendo che il loro funzionamento all'interno dell'applicazione non varierà anche se il contesto esterno dovesse mutare.

4.1.4 Pagina di esercizio

Conclusa la fase di registrazione/*login* e verificata la correttezza dell'account, l'utente viene portato nella pagina di inizio sessione. In questa pagina l'utente avrà la possibilità di iniziare una nuova sessione o di uscire dall'applicazione attraverso il pulsante *logout*.

All'atterraggio nella pagina l'applicazione provvede a richiedere le autorizzazioni per l'utilizzo della webcam del dispositivo ed il suo microfono. Una volta che l'utente avrà dato il consenso all'uso di questi dispositivi, il sistema provvederà ad istanziare un nuovo tag HTML di tipo `<video/>`.

Successivamente, attraverso il listato visibile in figura 4.9, il tag HTML `<video/>` iniettato nell'applicazione verrà collegato all'oggetto `MediaStream` proveniente dalla fotocamera del dispositivo.

Questo verrà visualizzato a schermo interno come sfondo dell'intera applicazione. Sul *canvas* così generato verranno applicati degli effetti per migliorare la resa grafica finale e permettere al paziente una migliore lettura del testo dei compiti motori.

Successivamente, l'applicazione, attraverso la classe `Database`, provvederà a recuperare dal servizio *FireStore* gli esercizi che il paziente dovrà eseguire.

Attraverso il provider `Exercises`, il dato rappresentante gli esercizi verrà destrutturato e serializzato in modo che la *WebApp* possa interpretarlo coerentemente durante tutto il ciclo di vita dell'applicazione.

Iniziare una nuova sessione

Completato il download degli esercizi e collegato il modulo di acquisizione audio/video, l'utente potrà iniziare la sua sessione di esercizi.
Sul display del dispositivo verranno visualizzati:
- il testo del compito da eseguire;
- un pulsante per iniziare l'esecuzione dell'esercizio;
- una barra di stato per indicare il numero di esercizi completati.

Alla pressione del tasto *Play*, la *WebApp* farà partire un *countdown* di 3 secondi. In questa fase l'utente potrà decidere se fermare il *countdown* o lasciarlo scorrere.

Al termine del *countdown* l'applicazione inizierà la registrazione dello *stream* audio/video, mostrerà un *timer* indicativo del tempo trascorso dall'inizio dell'esercizio e darà la possibilità al paziente di concludere l'esecuzione dell'esercizio per procedere con il successivo.

Per ogni esercizio è previsto un tempo massimo di esecuzione, superato il quale la *WebApp* terminerà automaticamente l'esercizio passando al successivo. Di norma il tempo massimo di esecuzione è impostato su 30 secondi, ma questo dato può essere modificato al bisogno aggiornando la tupla corrispondente nel *database*.

Ad ogni step compiuto i dati multimediali raccolti verranno salvati in una posizione di memoria temporanea del *browser*.

Al termine di ogni esercizio, l'utente avrà la possibilità di ripetere lo *step* appena concluso oppure procedere con il successivo. Se l'utente deciderà di ripetere l'ultimo esercizio, la nuova esecuzione andrà a sovrascrivere i dati precedentemente raccolti.

Le operazioni sopra descritte si ripetono fino al completamento della sessione.

4.1.5 Terminare una sessione

Al termine della sessione, tutti gli *stream* raccolti vengono raggruppati ed inviati tramite una chiamata HTTPS in modalità multipart verso il servizio Firebase *Storage* (Fig. 4.10).

I file saranno salvati in un path così composto:

```
<id_user>/<timestamp>/step_<step_number>.<extension>
```

L'estensione del file sarà determinata dal *mimeType* meglio compatibile con il dispositivo in uso.

Scelta del mimeType

Data la variabilità di dispositivi su cui è possibile eseguire l'applicazione, la classe `WebcamRecorder` si occuperà della scelta della prima combinazione *mimeTypes/codec* supportata dal dispositivo.

Si è preferito determinare la combinazione *mimeTypes/codec* attraverso una doppia lista di valori *mimeTypes* e *codecs* (Fig. 4.11).

In questo modo si ha la possibilità di controllare maggiormente il formato con cui l'applicazione convertirà gli *stream* audio/video del dispositivo.

Attraverso questa soluzione sarà possibile determinare la combinazione che meglio si presta all'analisi con gli algoritmi di IA e di conseguenza limitare le combinazioni supportate dall'applicazione.

```
1  <body>
2    <!-- The core Firebase JS SDK is always required and must be listed
         first -->
3    <script src="https://www.gstatic.com/firebasejs/8.4.1/firebase-app.js"
         ></script>
4    <!-- Add SDKs for Firebase products that you want to use
5         https://firebase.google.com/docs/web/setup#available-libraries -->
6    <script src="https://www.gstatic.com/firebasejs/8.4.1/firebase-auth.js
         "></script>
7    <script src="https://www.gstatic.com/firebasejs/8.4.1/firebase-
         analytics.js"></script>
8    <script src="https://www.gstatic.com/firebasejs/8.4.1/firebase-storage
         .js"></script>
9    <script src="https://www.gstatic.com/firebasejs/8.4.1/firebase-
         firestore.js"></script>
10   <script>
11     // Your web app's Firebase configuration
12     var firebaseConfig = {
13       apiKey: "<apiKey>",
14       authDomain: "neuron-word.firebaseapp.com",
15       databaseURL: "https://neuron-word.firebaseio.com",
16       projectId: "neuron-word",
17       storageBucket: "neuron-word.appspot.com",
18       messagingSenderId: "<messagingSenderId>",
19       appId: "<appId>"
20     };
21     // Initialize Firebase
22     firebase.initializeApp(firebaseConfig);
23     firebase.analytics();
24     var storage = firebase.storage();
25   </script>
26   <!-- This script installs service_worker.js to provide PWA
         functionality to
27        application. For more information, see:
28        https://developers.google.com/web/fundamentals/primers/service-
              workers -->
29   <script>
30     if ('serviceWorker' in navigator) {
31       window.addEventListener('flutter-first-frame', function () {
32         navigator.serviceWorker.register('flutter_service_worker.js?v
               =4025127195');
33       });
34     }
35   </script>
36   <script src="main.dart.js" type="application/javascript"></script>
37   <noscript>Javascript  necessario per poter visualizzare correttamente
         questa app</noscript>
38 </body>
```

Figura 4.1: *In alto gli script per importare le librerie di firebase (3 - 9), al centro lo snippet del codice per la configurazione di firebase (10 - 25), in basso l'Inizializzazione del service worker per la PWA (29 - 34)*

```dart
class App extends StatelessWidget {
  final Future<FirebaseApp> _initialization = Firebase.initializeApp();

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return FutureBuilder(
      future: _initialization,
      builder: (context, snapshot) {
        if(snapshot.hasError)
          return showError("firebase initialization error");
        if(snapshot.connectionState == ConnectionState.done) {
          Network();
          Database();
          Auth();
          return MaterialApp(
            title: APP_NAME,
            theme: ThemeData.light(),
            darkTheme: ThemeData.dark(),
            themeMode: ThemeMode.system,
            onGenerateRoute: Routes.onGenerateRoute,
          );
        }
        return showLoader();
      },
    );
  }
```

Figura 4.2: *Inizializzazione della WebApp - codice DART.*
Si noti l'adozione del costruttore FutureBuilder per attendere il caricamento delle configurazioni di firebase

```dart
class Routes {
  // Route name constants
  static const String Loading = '/';
  static const String Login = '/login';
  static const String Exercise = '/exercise';
  static const String Verification = '/verification';
  static const String ContinueRegistration = '/registration';

  static String? currentRoute;
  static Route<dynamic> onGenerateRoute(RouteSettings settings){
    dynamic args = settings.arguments;
    String? pageName = settings.name;
    if(pageName == '/') pageName = Routes.Loading;
    currentRoute = pageName;
    return MaterialPageRoute(builder: (context) {
      Map<String, CustomRoute> routes = getRoutes(args);
      if(routes.keys.contains(pageName)) {
        if(!routes[pageName!]!.requireLogin || (routes[pageName]!.
            requireLogin
            && Auth.instance.isRouteAuthorized(pageName)))
          return routes[pageName]!.builder(context);
      }
      return routes["404"]!.builder(context);
    });
  }

  /// The map used to define our routes, needs to be supplied to [
      MaterialApp]
  static Map<String, CustomRoute> getRoutes(dynamic args) {
    return {
      Routes.Loading: CustomRoute((context) => LoadingPage(), false),
      Routes.Login: CustomRoute((context) => LoginPage(), false),
      Routes.Verification: CustomRoute((context) =>
          EmailVerificationPage(), false),
      Routes.Exercise: CustomRoute((context) => ExercisePage(), true),
      Routes.ContinueRegistration: CustomRoute((context) =>
          ContinueRegistrationPage(), true),
      "404": CustomRoute((context) => NoPage(), false),
    };
  }
}
class CustomRoute {
  WidgetBuilder builder;
  bool requireLogin;
  CustomRoute(this.builder, this.requireLogin);
}
```

Figura 4.3: *Il dettaglio del file Routes.dart - con questa classe vengono definite le rotte dell'applicazione*

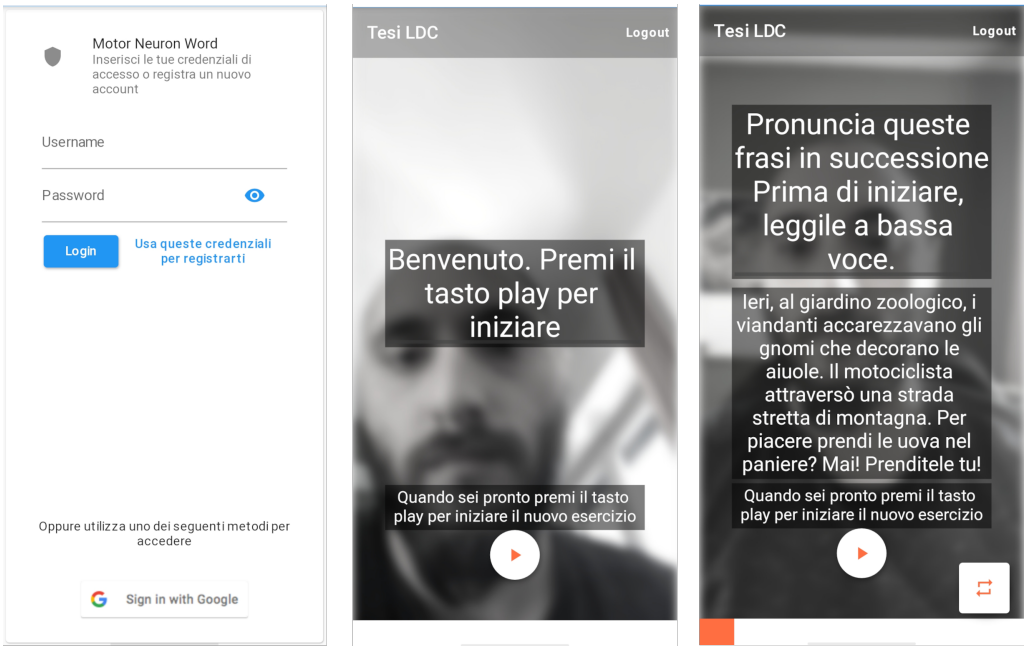

Figura 4.4: *Dettaglio di 3 schermate della WebApp. La prima a sinistra mostra la pagina di login o registrazione, l'immagine centrale mostra la schermata di inizio sessione e l'ultima a destra, mostra l'utente mentre svolge uno degli esercizi della sua sessione di monitoraggio*

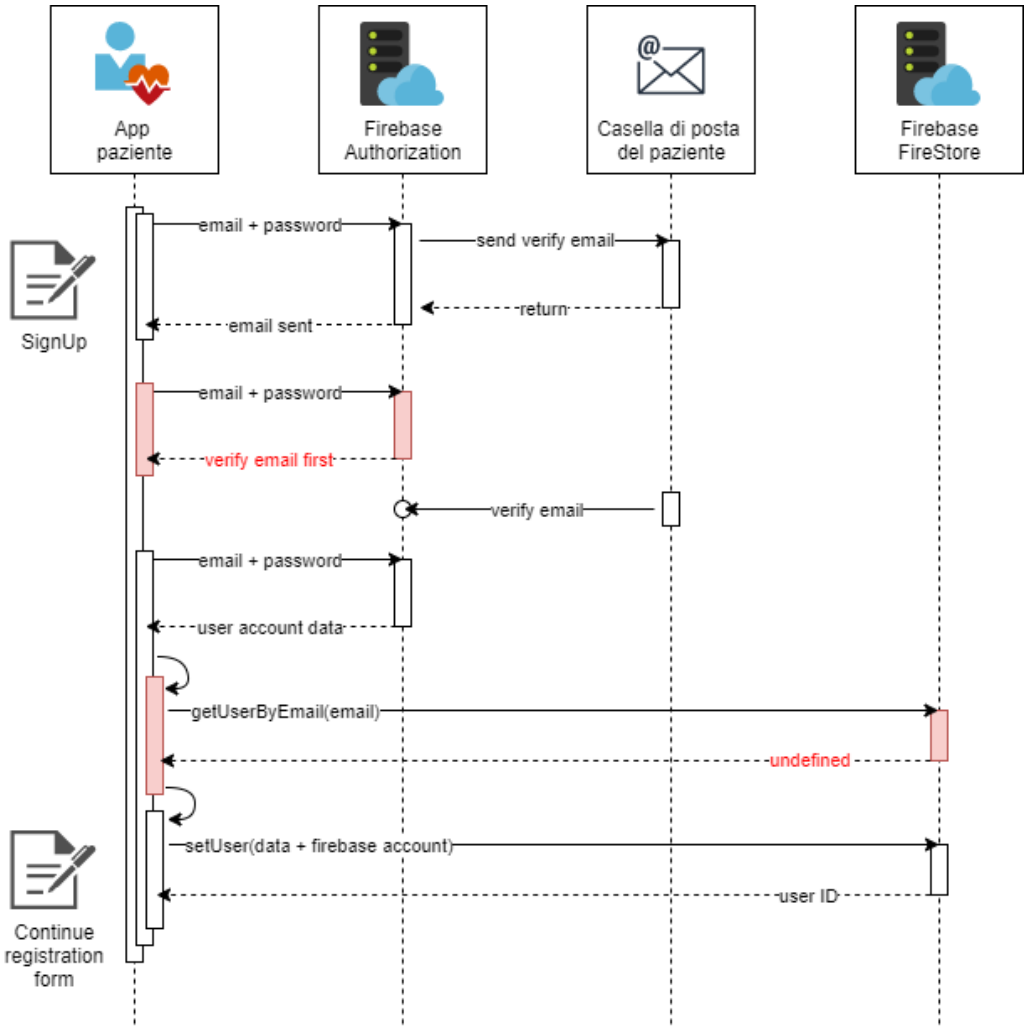

Figura 4.5: *L'immagine rappresenta i flussi che si percorrono durante l'accesso alla WebApp. In questo caso l'utente sta effettuando una nuova registrazione. In rosso sono segnati i segmenti che hanno prodotto una eccezione e per cui l'utente è stato notificato*

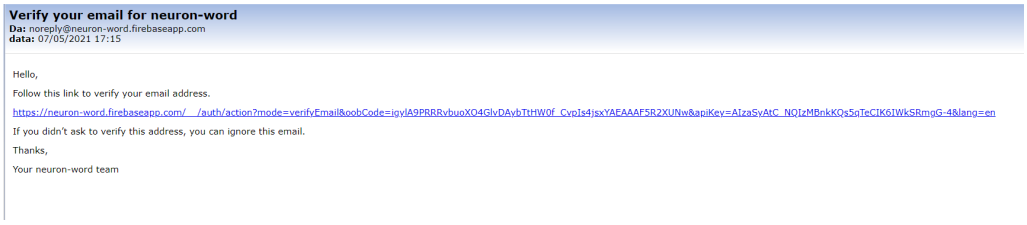

Figura 4.6: *Dettaglio della email che il sistema firebase Authorization invia per la verifica della casella di posta*

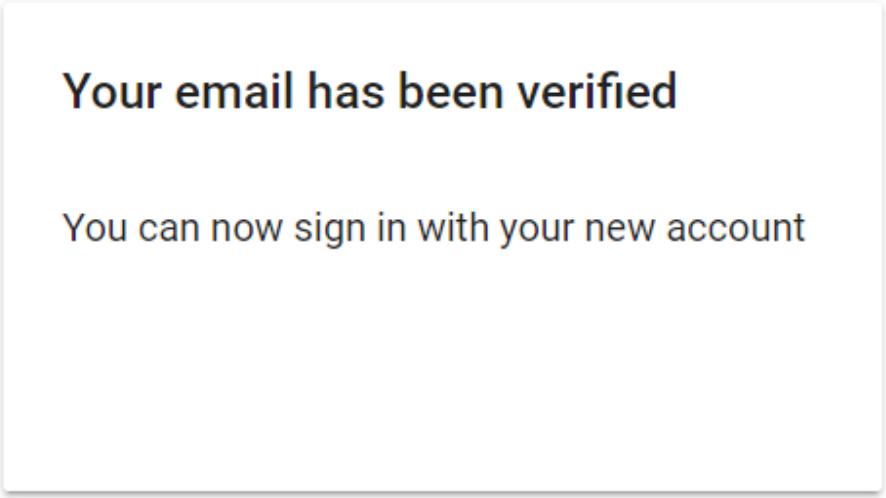

Figura 4.7: *Dettaglio della pagina web collegata al link di verifica della casella di posta*

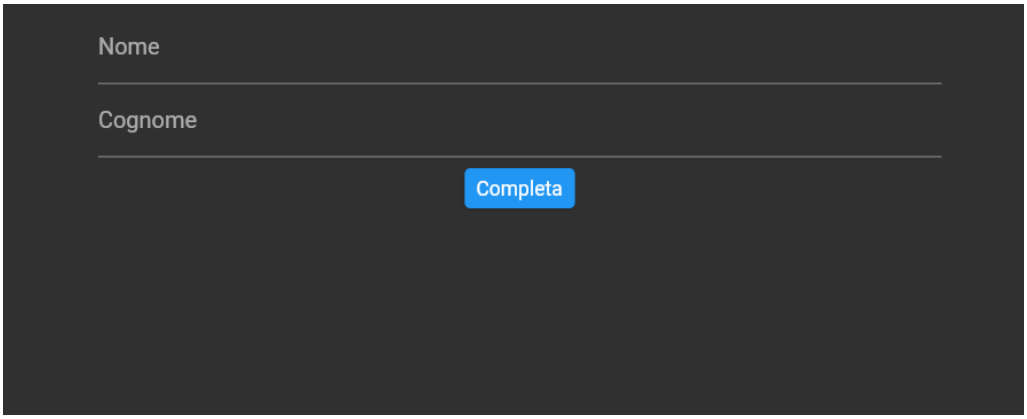

Figura 4.8: *Pagina di completamento delle informazioni necessarie alla registrazione dell'account. In questo caso la WebApp è visualizzata con il tema Dark*

```dart
late VideoElement _webcamVideoElement;
late Widget webcamWidget;

@override
void initState() {
    super.initState();

    // Creo il video element a cui potr associare lo stream della sorgente video
    _webcamVideoElement = VideoElement();
    _webcamVideoElement.muted = true;

    // Mi sottoscrivo alla webcam del dispositivo
    ui.platformViewRegistry.registerViewFactory('webcamVideoElement', (int viewId) => _webcamVideoElement);

    // Creo il widget video
    webcamWidget = HtmlElementView(
      key: widget.webcamKey,
      viewType: 'webcamVideoElement'
    );

    // Modalit video-selfie
    var videoOption = {
      "facingMode": 'user',
    };

    // Rappresentazione Dart del formato Json
    var options = <String, dynamic>{
      "video": videoOption,
      "audio": true
    };

    // Accedo allo stream della webcam
    window.navigator.mediaDevices!.getUserMedia(options).then((MediaStream stream) {
      _webcamVideoElement.srcObject = stream;
      _webcamVideoElement.setAttribute("style", "object-fit: cover; filter: blur(5px) saturate(0.05); width:100%; height:100%;");
      if (widget.onStreamAvailable != null)
        widget.onStreamAvailable!(stream, _webcamVideoElement);
    }).catchError((e) {
      webcamWidget = Text("ERROR");
      widget.onErrorOccurred(e);
    });
}
```

Figura 4.9: *Snippet della classe che si occupa di istanziare e aggiornare il tag HTML `<video/>` all'interno dell'applicazione.*

```
1   static uploadSession(SessionModel session) async {
2     for(ExerciseData exercise in session.exercises!){
3       String baseUrl = '${FirebaseAuth.instance.currentUser!.uid}(${Auth.
            instance.userData!.name}_${Auth.instance.userData!.surname})/${
            DateFormat("yyyy-MM-ddHH:mm:ss").format(session.startDate!)}/';
4       firebase_storage.Reference refV = _storage.ref(
5         '$baseUrl${exercise.videoFile!.filename}${exercise.videoFile!.
              extension}'
6       );
7       try {
8         var metadata = firebase_storage.SettableMetadata(
9             contentType: exercise.videoFile!.mimeType,
10            customMetadata: {
11              "title": exercise.type!,
12              "duration": DateFormat("HH:mm:ss:SSS").format(
13                DateTime.utc(0,0,0).add(exercise.executionTime!)
14              ),
15              "width":"1280",
16              "height":"720",
17
18            }
19        );
20        await refV.putData(exercise.videoFile!.data!, metadata);
21      } on firebase_core.FirebaseException catch (eerror) {
22        print(['ERROR on Class: Network, Function: uploadSession, Line:
              53, ~ error:', error]);
23      }
24    }
25  }
```

Figura 4.10: *Il dettaglio del file Network.dart - con questa funzione statica vengono inviati verso il servizio Firebase Storage i file relativi alla sessione dell'utente*

```
List<String> mimeTypes = [
    "webm",
    "mpeg",
    "mp4"
];

List<String> codecs = [
    "h265",
    "h264",
    "vp9",
    "vp8",
    "opus",
    "daala",
    "avc1"
];

for(String _mimeType in mimeTypes){
    for(String _codecType in codecs){
        var type = "video/$_videoType;codecs=$_codecType";
        if(MediaRecorder.isTypeSupported(type)){
            videoType = "video/$_videoType";
            codecType = "codecs=$_codecType";
            extension = ".$_videoType";
            _mediaRecorderOption["mimeType"] = type;
            break;
        }
    }
}
if (_mediaRecorderOption["mimeType"] == null) {
      throw ("NO SUPPORTED TYPE");
    } else {
    ...
```

Figura 4.11: *Codice per la definizione dinamica del mimeType. Si noti come questa struttura dia priorità all'individuazione del miglior* `videoType` *con il miglior* `codecType` *supportati dal dispositivo.*

Capitolo 5

Conclusioni

In questa tesi è progetta e realizzata una applicazione Web proposta ai pazienti con SLA bulbare per il monitoraggio della malattia da remoto.

L'applicazione permette al paziente di riprendersi in modalità *video-selfie* mentre legge dei contenuti o esegue dei compiti proposti in sovrimpressione sullo schermo del device.

I dati multimediali acquisiti attraverso l'applicazione permetteranno di strutturare il primo *dataset* nell'ambito delle malattie neurologiche che combini sia dati audio sia dati video.

La produzione di un *dataset* specifico è il primo passo verso il miglioramento della strumentazione a supporto degli specialisti nel monitoraggio di questo tipo di malattia.

Grazie alla costruzione del *dataset* sarà possibile addestrare le reti neurali che saranno i punti focali delle analisi dei dati multimediali raccolti durante le sessioni di esercizio dei futuri pazienti in cura.

Si riconosce che ulteriori ricerche saranno necessarie per migliorare l'applicazione proposta al fine di offrire il miglior supporto per il monitoraggio della SLA da remoto, ma l'applicazione presentata in questa tesi è sicuramente un primo passo verso l'implementazione di un sistema automatico a supporto di clinici e pazienti affetti da SLA.

Possibili sviluppi futuri della applicazione sviluppata riguarderanno l'implementazione di un sistema completo, che permetta di elaborare lo *stream* di dati audio-video attraverso gli algoritmi di IA e di produrre valutazioni sui parametri di esecuzione dell'esercizio.

Lo sviluppo di questa metodologia di monitoraggio permetterà di espandere le funzionalità proposte da Parola di Motoneurone ad altre malattie neurodegenerative, ampliando il ventaglio di strumenti offerti.

Si prevede che gli algoritmi di IA in questo specifico ambito della medicina assumeranno un ruolo sempre più centrale e di rilievo. Premunirsi già da ora di uno strumento di monitoraggio modulare ed espandibile determinerà un valore aggiunto per l'adozione del framework nei centri di cura.

L'applicazione prototipo oggetto di questa tesi è attualmente (Maggio 2021) in uso presso *Ospedali riuniti di Ancona* e fungerà da piattaforma di collaudo per la creazione del primo *dataset* audio/video specifico su pazienti malati di SLA.

Elenco delle figure

Le figure riportanti flussi di informazione o schematizzazioni sono state realizzate attraverso il software ***draw.io*** *utilizzando esclusivamente immagini e figure con licenza gratuita contenute nativamente nel software*

Bibliografia

[1] Figura motoneuroni. http://www.superagatoide.altervista.org/motoneuroni.html".

[2] Piccininni M Logroscino G. Amyotrophic lateral sclerosis descriptive epidemiology: The origin of geographic difference. *Neuroepidemiology*, (52):93–103, 2019.

[3] Dominik Karch, Keun-Sun Kang, Katarzyna Wochner, Heike Philippi, Mijna Hadders-Algra, Joachim Pietz, and Hartmut Dickhaus. Kinematic assessment of stereotypy in spontaneous movements in infants. *Gait & posture*, 36(2):307–311, 2012.

[4] Tadi P Zayia LC. Neuroanatomy, motor neuron. *StatPearls*, 2020.

[5] Solara V Bersano E Cantello R Mazzini L De Marchi F, Sarnelli MF. Depression and risk of cognitive dysfunctions in amyotrophic lateral sclerosis. *Acta Neurol Scand*, 139:438–445, 2019.

[6] Sousa T Reis S Coelho L Vieira H, Costa N. Voice-based classification of amyotrophic lateral sclerosis: Where are we and where are we going? a systematic review. *Neurodegener Dis*, 19(5-6):163–170, 2019.

[7] Regolamento di istituzione della rete nazionale delle malattie rare e di esenzione dalla partecipazione al costo delle relative prestazioni sanitarie ai sensi dell'articolo 5, comma 1, lettera b) del decreto legislativo 29 aprile 1998, n. 124.
(decreto ministeriale - ministero della sanità - 18 maggio 2001, n. 279. pub-

blicato in gazzetta ufficiale 12 luglio 2001, n. 160 supplemento ordinario n.180/l).

[8] Green JR Barnett C Yunusova Y, Plowman EK and Bede P. Clinical measures of bulbar dysfunction in als. *Frontiers in neurology*, (106), 2019.

[9] Enderby P. Disorders of communication: dysarthria. *Handb Clin Neurol*, 110(273):81, 2013.

[10] Jordan Green, Yana Yunusova, Mili Kuruvilla, Jun Wang, Gary Pattee, Lori Synhorst, Lorne Zinman, and James Berry. Bulbar and speech motor assessment in als: Challenges and future directions. *Amyotrophic lateral sclerosis frontotemporal degeneration*, 14(7-8):494–500, 2013.

[11] Robertson SJ. *Robertson SJ. Disarthria Profile.* Winslow Press, 1982. Versione italiana a cura di Fossi F. e Cantagallo A. Ediz. Omega(1999).

[12] Barbara Tomik and Roberto J. Guiloff Professor. Dysarthria in amyotrophic lateral sclerosis: A review. *Amyotrophic Lateral Sclerosis*, 11(1-2):4–15, 2010.

[13] Julie Liss Jeremy Shefner Seward Rutkove Kerisa Shelton Cayla Jessica Duncan Visar Berisha Gabriela M. Stegmann, Shira Hahn. Early detection and tracking of bulbar changes in als via frequent and remote speech analysis. *npj Digital Medicine*, 3(1):1–5, 2020.

[14] Yunusova Yana Plowman Emily K. Green Jordan R. Perry Bridget J., Martino Rosemary. Lingual and jaw kinematic abnormalities precede speech and swallowing impairments in als. *Dysphagia*, 33(6):840–847, 2018.

[15] Stephen Kelly, Alison Main, Graham Manley, and Calum McLean. Electropalatography and the linguagraph system. *Medical engineering physics*, 22:47–58, 02 2000.

[16] Linguaggio Dart. https://dart.dev/.

[17] Framework Flutter. https://flutter.dev/.

Ringraziamenti

Vorrei ringraziare *in primis* la mia famiglia, che mi è stata sempre vicina in questi anni di università che solo grazie a loro ho avuto la possibilità di frequentare. Un ringraziamento speciale va alla mia compagna Annalisa che grazie ai suoi racconti di vita personale mi ha avvicinato al mondo della SLA ed alle sue problematiche ed ha instillato in me e nei miei compagni le prime scintille che ci hanno portato ad affrontare professionalmente queste tematiche, in cerca di una soluzione innovativa e fattibile. Ringrazio in particolar modo la mia collega ed amica Lucia Migliorelli, fondamentale collegamento tra il mio team ed *Ospedali riuniti di Ancona*, nonchè "correlatrice virtuale" di questa tesi, che mi ha aiutato con i suoi consigli e il suo sostegno. Un ulteriore ringraziamento va anche agli altri membri ma sopratutto amici del team con cui abbiamo progettato e sviluppato le nostre idee e conclusioni per lunghe ed incredibili nottate; non smetterò mai di ringraziarli per quel corso di Unity3D mai erogato. Infine, ma non per importanza, vorrei ringraziare il professor Adriano Mancini, Relatore della mia tesi e mio mentore, ed il professor Emanuele Frontoni che in egual misura stimo, i quali mi hanno sempre supportato con fiducia nella realizzazione di questo lavoro e dei progetti collaterali ad esso collegati.