RAFT

Real-Time Automated Fingerspelling Translator

Overlimits

Braguti, Chater, Compagno, Silva

2024/2025

5A Informatica

Contents

1 Introduzione

Nell'anno scolastico 2024/2025, durante il quinto anno di scuola superiore, è necessario sviluppare un progetto che sarà poi presentato alla commissione d'esame. Abbiamo quindi deciso di proseguire lo sviluppo di un progetto avviato nell'anno precedente: un'applicazione web in grado di tradurre l'alfabeto dattilologico.

2 Analisi del contesto

2.1 Problema

A livello globale, si stima che il numero di persone sordomute siano di circa 70 milioni. Molti di loro affrontano sfide significative nella comunicazione quotidiana con individui che non conoscono la lingua dei segni. Inoltre, in alcuni paesi, le scuole che offrono corsi di lingua dei segni sono poche e la disponibilità di interpreti qualificati risulta spesso insufficiente.[9]

2.2 Soluzione

Abbiamo ritenuto che un'applicazione basata sull'intelligenza artificiale (IA) per tradurre l'alfabeto dattilologico (finger spelling) potrebbe rappresentare una soluzione innovativa per superare alcune delle barriere comunicative che le persone sordomute si trovano ad affrontare. I vantaggi sarebbero molteplici: ad esempio, l'applicazione sarebbe accessibile a chiunque disponga di un dispositivo elettronico e potrebbe costituire un valido strumento di supporto per le persone udenti, facilitandone l'apprendimento dell'alfabeto dattilologico e, di conseguenza, migliorando la loro capacità di comunicare con le persone sordomute.

2.3 Stakeholders

2.3.1 Analisi degli stakeholders

Stakeholders	**Analisi**
Team Members	Gli sviluppatori del progetto
Overlimits	Sponsor del progetto
Persone con disabilità	Client

Table 1: Analisi stakeholders

2.3.2 Matrice power/interest

High Power	Scuola	Team Members
Low Power	Persone con disabilità	Overlimits
	Low Interest	**High Interest**

Table 2: Matrice power/interest

Nel nostro progetto, i membri del team rivestono un ruolo fondamentale: hanno un alto interesse e un alto potere. Senza il loro contributo, infatti, il progetto non potrebbe essere portato a termine, poiché sono loro i principali sviluppatori e responsabili dell'avanzamento delle attività. Anche la scuola detiene un alto

potere, in quanto il progetto può essere completato solo con il suo consenso e il suo supporto istituzionale. D'altro canto, le persone con disabilità rappresentano i nostri clienti principali. Attualmente, però, non hanno né interesse né potere, poiché non sono ancora a conoscenza del progetto. È nostro obiettivo coinvolgerli e renderli partecipi, affinché possano beneficiare pienamente dei risultati del nostro lavoro. Infine, l'associazione no profit Overlimits, pur avendo un basso potere decisionale, gioca un ruolo cruciale. Non partecipa attivamente allo sviluppo tecnico del progetto, ma mostra un alto interesse, poiché ci supporta e ci offre l'opportunità di confrontarci direttamente con i nostri clienti, contribuendo così al successo complessivo dell'iniziativa.

2.3.3 Coinvolgimento

	Unaware	**Resistant**	**Neutral**	**Supportive**	**Leading**
Team Members					CD
Overlimits				CD	
Client	C		D		
Scuola			C	D	

Table 3: C = Il ruolo attuale degli stakeholders, D = Il ruolo che vorremmo che gli stakeholder assumessero

3 Ipotesi

3.1 Prerequisiti

- Il dispositivo che utilizza l'applicazione necessita di una webcam con una mediocre risoluzione per distinguere i dettagli nella mano
- Si necesitterà di un servizio che presti abbastanza potenza di calcolo per poter allenare adeguatamente l'intelligenza
- Per il funzionamento dell'applicazione è richiesta una connessione internet.
- Sono richieste differenti competenze a livello programmativo: La prima competenza richiesta è la programmazione web, che include la conoscenza di linguaggi di markup come HTML e CSS, oltre a linguaggi di programmazione per il back-end. Ad esempio, JavaScript viene utilizzato per rendere dinamico il sito. Per quanto riguarda l'intelligenza artificiale, è necessario avere familiarità con linguaggi di programmazione come Python (per la creazione e l'addestramento dell'intelligenza). Oltre a ciò, sono richieste conoscenze sul funzionamento delle reti neurali (cosa sono, come si addestrano e quali funzioni di attivazione utilizzano).

3.2 Exclusions

Il progetto non avrà lo scopo di riconoscere la lingua dei segni, ma si concentrerà esclusivamente sull'alfabeto dattilologico di diverse lingue. Questa scelta è stata presa in considerazione delle attuali tecnologie e conoscenze disponibili, che non consentono ancora di sviluppare un'intelligenza artificiale capace di riconoscere i movimenti, ma solo immagini statiche.

4 Obbiettivi

- 1 Mockup
  - 1.1 Definizione degli obbiettivi
  - 1.2 Studio di ispirazione e benchmark
  - 1.3 Studio elementi
    - 1.3.1 Scelta della palette colori
    - 1.3.2 Scelta dei font
    - 1.3.3 posizione degli elementi
  - 1.4 Creazione di animazioni e collegamenti
  - 1.5 Più dispositivi
  - 1.6 Creazione logo
  - 1.7 Presentazione
- 2 Studio fattibilità e creazione del Dataset
  - 2.1 Studio dei segni
    - 2.1.1 Analisi delle fonti linguistiche
    - 2.1.2 Validazione dei segni
    - 2.1.3 Identificazione delle varianti
    - 2.1.4 Documentazione dei segni
  - 2.2 Ricerca del DataSet
    - 2.2.1 Individuazione delle fonti online
    - 2.2.2 Valutazione della qualità dei dati
    - 2.2.3 Definizione dei criteri di inclusione/esclusione
  - 2.3 Creazione delle immagini di test
  - 2.4 Organizzazione del DataSet
    - 2.4.1 Verifica della coerenza nei formati
    - 2.4.2 Classificazione delle immagini
    - 2.4.3 Creazione delle directory
  - 2.5 Studio tecnologia front end
    - 2.5.1 Analisi delle esigenze dell'applicazione
    - 2.5.2 Ricerca dei framework front-end
  - 2.6 Studio tecnologia back end
    - 2.6.1 Analisi delle funzionalità back-end
    - 2.6.2 Ricerca delle tecnologie back-end
    - 2.6.3 Definizione dell'architettura del back-end

- 7 Modalità allenamento
  - 7.1 Ideazione dei minigochi
    - 7.1.1 Ricerca di giochi educativi simili
    - 7.1.2 Valutazione delle idee di gioco
    - 7.1.3 Definizione delle regole e delle meccaniche
  - 7.2 Sviluppo interfaccia
    - 7.2.1 Progettazione del layout visivo
    - 7.2.2 Sviluppo del codice
    - 7.1.3 Test dell'interfaccia
- 8 Testing
  - 8.1 Testare l'app
    - 8.1.1 Test funzionali
    - 8.1.2 Test su diversi dispositivi
  - 8.2 Far provare l'app al client
    - 8.2.1 Preparazione del test
    - 8.2.2 Raccolta del feedback
  - 8.3 Correzzione di eventuali bug
    - 8.3.1 Correzione dei bug
    - 8.3.2 Re-testing delle funzionalità corrette
  - 8.4 Fine tuning
    - 8.4.1 Ottimizzazione delle prestazioni
    - 8.4.2 Controllo finale di qualità

5 Target

Questo servizio è di mirabile attenzione per tutti coloro che hanno una condizione, fisica o neurologica che sia, che non le rende in grado di parlare, e che si ritrovano a dover comunicare con persone che non conoscono l'alfabeto dattilologico. Quest'applicazione semplificherebbe di molto le piccole conversazioni quotidiane di difficoltà e/o emergenza in quanto nella nostra idea basterebbe inquadrare la persona che usa la lingua dei segni per avere una traduzione automatica istantanea del loro significato.

6 Implementazione

6.1 Software

6.1.1 Mockup e stile

Il mockup è stato realizzato tramite la piattaforma Figma. [4] Per la realizzazione del mockup è stato condotto uno studio approfondito e una ricerca mirata per garantire un'interfaccia di facile comprensione per l'utente. Sono stati creati asset specifici per le animazioni e svolto uno studio cromatico per evidenziare le parti più rilevanti dell'applicazione, assicurando così un design intuitivo ed efficace.

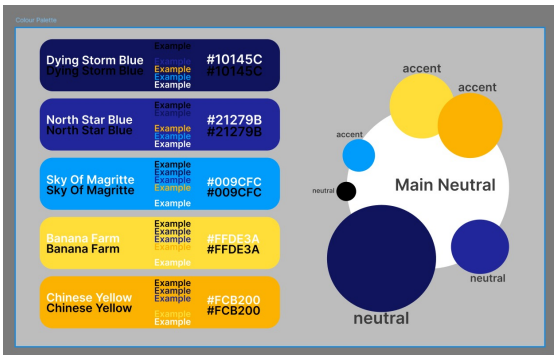

Figure 1: Studio dei colori

L'applicazione ipotetica presenta una schermata iniziale che consente all'utente di scegliere tra i due servizi principali:

- **La traduzione dei segni**: Che sfrutta la fotocamera del dispositivo
- **Una sezione gioco**: disponibile con più livelli di difficoltà per mettere alla prova le abilità dell'utente

In aggiunta, è disponibile una schermata dedicata alla **consultazione dell'alfabeto**, completa di una barra di ricerca per facilitare la navigazione. Le immagini d'esempio sono state create da noi con una **fotocamera Nikon D7500** con **obbiettivo Nikon 35 mm f1.8**. Un'ulteriore schermata è stata progettata per permettere all'utente di contattare il team in caso di necessità. L'applicazione offre, inoltre, la possibilità di scegliere in qualsiasi momento la lingua su cui lavora l'intelligenza, migliorando così l'adattabilità alle diverse esigenze dell'utente. Infine, l'applicazione è stata progettata in tre versioni diverse (mobile/tablet/computer) per ottimizzare la gestione su più dispositivi, garantendo una buona esperienza.

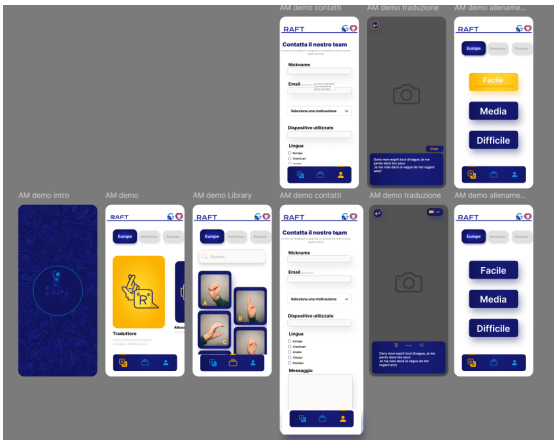

Figure 2: Mockup vista cellulare

6.2 Allenamento

L'intelligenza artificiale è stata allenata utilizzando la piattaforma **GoogleColab.** [13] Google Colab è una piattaforma basata su cloud che permette di scrivere ed eseguire codice Python direttamente nel browser, dando accesso a CPU, GPU e TPU per accelerare l'allenamento di modelli complessi.

Il codice che abbiamo utilizzato ha lo scopo di creare e allenare un modello di intelligenza artificiale in grado di riconoscere i segni dell'alfabeto dattilologico italiano, utilizzando il **framework MediaPipe Model Maker**. Inizialmente, vengono installate e importate le librerie necessarie, tra cui **mediapipe-model-maker**, **gdown** (per scaricare file da Google Drive) e **TensorFlow**. Successivamente, il codice procede con il download di un dataset da Google Drive, che viene poi estratto e organizzato in cartelle, ciascuna rappresentante un'etichetta.

Il dataset viene poi suddiviso in tre insiemi:

- **training** per l'addestramento del modello
- **validation** per monitorare le prestazioni durante l'addestramento
- **test** per valutare la qualità finale del modello.

L'addestramento del modello avviene utilizzando le immagini dei segni, permettendo all'algoritmo di apprendere i diversi punti della mano corrispondenti alle lettere.

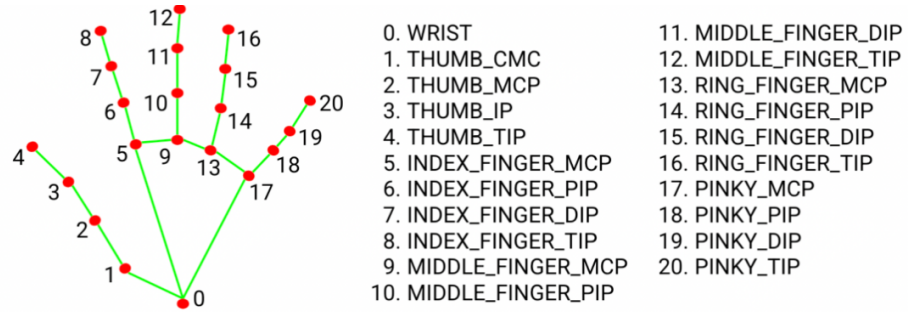

Figure 3: Riconoscimento punti della mano

Dopo l'addestramento, il modello viene valutato utilizzando il dataset di test, riportando metriche di performance come la **loss** (perdita) e **accuracy** (accuratezza). Infine, il modello addestrato viene esportato in un file con estensione **'.task'**, pronto per essere utilizzato.

```
Total params: 4253 (16.61 KB)
Trainable params: 3997 (15.61 KB)
Non-trainable params: 256 (1.00 KB)
None
Epoch 1/10
53988/53988 [==============================] - 432s 8ms/step - loss: 0.8574 - categorical_accuracy: 0.7006 - val_loss: 0.2491 - val_categorical_accuracy: 0.8975 - lr: 0.0010
Epoch 2/10
53988/53988 [==============================] - 426s 8ms/step - loss: 0.7620 - categorical_accuracy: 0.7323 - val_loss: 0.2474 - val_categorical_accuracy: 0.8972 - lr: 9.9000e-04
Epoch 3/10
53988/53988 [==============================] - 431s 8ms/step - loss: 0.7506 - categorical_accuracy: 0.7368 - val_loss: 0.2534 - val_categorical_accuracy: 0.9012 - lr: 9.8010e-04
Epoch 4/10
53988/53988 [==============================] - 424s 8ms/step - loss: 0.7436 - categorical_accuracy: 0.7404 - val_loss: 0.2552 - val_categorical_accuracy: 0.9071 - lr: 9.7030e-04
Epoch 5/10
53988/53988 [==============================] - 429s 8ms/step - loss: 0.7404 - categorical_accuracy: 0.7402 - val_loss: 0.2726 - val_categorical_accuracy: 0.8990 - lr: 9.6060e-04
Epoch 6/10
53988/53988 [==============================] - 427s 8ms/step - loss: 0.7344 - categorical_accuracy: 0.7416 - val_loss: 0.2792 - val_categorical_accuracy: 0.9011 - lr: 9.5099e-04
Epoch 7/10
53988/53988 [==============================] - 426s 8ms/step - loss: 0.7336 - categorical_accuracy: 0.7426 - val_loss: 0.2873 - val_categorical_accuracy: 0.8978 - lr: 9.4148e-04
Epoch 8/10
53988/53988 [==============================] - 428s 8ms/step - loss: 0.7329 - categorical_accuracy: 0.7427 - val_loss: 0.2938 - val_categorical_accuracy: 0.8926 - lr: 9.3207e-04
Epoch 9/10
53988/53988 [==============================] - 429s 8ms/step - loss: 0.7313 - categorical_accuracy: 0.7434 - val_loss: 0.2991 - val_categorical_accuracy: 0.8949 - lr: 9.2274e-04
Epoch 10/10
53988/53988 [==============================] - 425s 8ms/step - loss: 0.7292 - categorical_accuracy: 0.7446 - val_loss: 0.3061 - val_categorical_accuracy: 0.8929 - lr: 9.1352e-04
Modello allenato
Inizio valutazione del modello
13497/13497 [==============================] - 243s 2ms/step - loss: 0.2852 - categorical_accuracy: 0.9006
Test loss: 0.2851794362068176, Test accuracy: 0.9005705118179321
```

Figure 4: Esempio di epoch dell'allenamento

6.3 Creazione del dataSet

Il dataset utilizzato per l'allenamento del modello è composto da un totale di **4GB** di immagini di mani. La maggior parte delle immagini è stata scaricata dalla piattaforma **Kaggle** [2]. Tuttavia, poiché i segni dell'alfabeto dattilologico italiano differiscono da quelli dell'alfabeto dattilologico americano, è stato necessario creare autonomamente le immagini di allenamento per due lettere specifiche dell'alfabeto. Per la creazione delle immagini, sono stati realizzati dei video a persone che eseguivano i segni dell'alfabeto, ruotando leggermente le mani. I video sono stati registrati in diverse condizioni di illuminazione (buio e luce) e con un'ampia varietà di persone e mani. Successivamente, la risoluzione dei video è stata ridotta per uniformarla alla qualità delle altre immagini, utiliz-

zando il software **Adobe Lightroom Classic** [8]. I video sono stati poi convertiti in fotografie tramite l'applicazione **AVS Video Converter** [1]. Per facilitare l'allenamento del modello, le immagini sono state organizzate in cartelle, una per ciascuna lettera dell'alfabeto. È presente anche una cartella denominata **"NULL"**, contenente immagini di sfondi vuoti o mani che non corrispondono a nessuna lettera dell'alfabeto. Inoltre, sono stati inclusi segni supplementari per rappresentare lo spazio e il carattere di cancellazione (delete). Il dataset è stato infine caricato su **Google Drive** [7] per essere facilmente accessibile dalla piattaforma di allenamento.

Questo processo è stato ripetuto sia per la lingua Europea che per quella Americana

6.4 Front end

6.4.1 Linguaggi del front end

Per la realizzazione del sito web della nostra applicazione, abbiamo utilizzato HTML, CSS e JavaScript, ciascuno con un ruolo specifico nel processo di sviluppo del frontend.

HTML (HyperText Markup Language) è stato impiegato per strutturare le pagine, definendo la gerarchia degli elementi e creando lo scheletro dell'interfaccia utente.

CSS (Cascading Style Sheets) ci ha permesso di personalizzare l'aspetto grafico del sito, rispettando il design del mockup attraverso la gestione di colori, spaziature, posizionamenti e altri aspetti estetici.

JavaScript (JS) è stato fondamentale per rendere l'interfaccia dinamica e interattiva. Grazie a JS, abbiamo implementato animazioni, gestito eventi e sviluppato funzionalità avanzate come la navigazione tra più schermate all'interno di una singola pagina (index.html). L'integrazione di questi tre linguaggi ci ha permesso di creare un'interfaccia fluida e intuitiva, migliorando l'esperienza utente e garantendo una navigazione ottimale all'interno dell'applicazione.

6.4.2 Da React a Javascript

Inizialmente, durante lo sviluppo dell'applicazione, abbiamo valutato l'idea di utilizzare la libreria **React**[10] per supportare il progetto. Tuttavia, dopo un primo periodo di lavoro, ci siamo resi conto che questa scelta avrebbe rallentato significativamente lo sviluppo per due motivi principali.

Il primo ostacolo è stato la compatibilità tra le librerie di intelligenza artificiale e quelle di React: alcune dipendenze non si integravano perfettamente, creando difficoltà tecniche che avrebbero richiesto tempo per essere risolte. Il secondo problema riguardava l'esperienza del team: nessun membro del gruppo di sviluppo aveva mai lavorato con React prima d'ora. Di conseguenza, avremmo dovuto investire del tempo nell'apprendimento della libreria, un cosa che non potevamo permetterci viste le scadenze ravvicinate del progetto.

Per questi motivi, abbiamo scelto di utilizzare JavaScript puro, una soluzione forse più limitata rispetto a React in alcuni aspetti, ma che ci ha permesso di

procedere in modo più rapido ed efficace. Il linguaggio era già ben noto a tutti i membri del team e, soprattutto, non presentava problemi di compatibilità con le librerie di intelligenza artificiale.

6.4.3 Sviluppo PWA

Una **Progressive Web App (PWA)**[12] è un'applicazione web che combina il meglio delle esperienze web e mobile, offrendo prestazioni simili a quelle di un'app nativa. Una PWA può essere installata sui dispositivi, funzionare offline e garantire un'esperienza utente fluida e reattiva.

Abbiamo trasformato il nostro sito web in una PWA, rendendolo installabile su qualsiasi dispositivo. Tuttavia, al momento, consigliamo fortemente di utilizzarla su smartphone, poiché il design e molte funzionalità principali sono ottimizzati e pienamente operativi solo su dispositivi mobili.

Per rendere il nostro sito web installabile su qualsiasi dispositivo, abbiamo seguito una serie di passaggi fondamentali. Il primo passo è stato ottimizzare l'installazione, aggiungendo un set di icone personalizzate **(favicon)**.

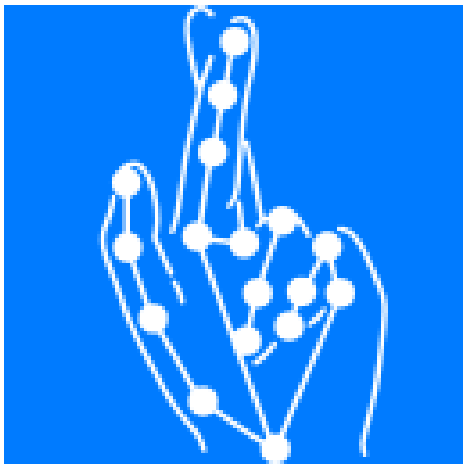

Figure 5: Favicon

Queste icone non solo migliorano l'aspetto grafico, ma vengono anche utilizzate per la schermata home e per l'interfaccia del sistema operativo, garantendo un'integrazione più naturale sui vari dispositivi.

Successivamente, abbiamo creato il file **manifest.json**, un elemento essenziale per qualsiasi PWA. Questo file JSON contiene informazioni chiave, come il nome dell'applicazione, il percorso della rete, la configurazione delle icone e le impostazioni per la visualizzazione a schermo intero. Grazie a questo file, i browser possono riconoscere e trattare il sito come un'applicazione vera e propria, permettendone l'installazione.

Un altro aspetto cruciale è stata la gestione del **Service Worker**, un componente che lavora in background per rendere l'app funzionante anche senza

connessione a Internet. Questo script intercetta e memorizza le risorse necessarie, garantendo l'accesso ai contenuti anche offline.

Per assicurare che la PWA sia sempre aggiornata, abbiamo implementato un sistema di aggiornamento automatico. Il processo si basa su un semplice file di testo chiamato **pwaVersion**, che indica la versione attuale dell'app. Quando viene rilasciato un aggiornamento, il valore di questo file cambia e i dispositivi che hanno già installato la PWA rilevano la modifica, avviando automaticamente il processo di aggiornamento senza necessità di intervento da parte dell'utente.

6.4.4 Schermata di traduzione

All'apertura della schermata di traduzione, all'utente viene richiesto il consenso per l'accesso alla fotocamera del dispositivo su cui è in esecuzione la PWA. Una volta ottenuto il permesso, viene mostrata l'interfaccia principale: lo sfondo della schermata mostra il flusso video in tempo reale della fotocamera, mentre una casella di testo visualizza in modo progressivo le frasi tradotte dall'alfabeto dattilologico al testo scritto.

Per agevolare l'interazione e rendere più immediata la composizione delle frasi, sono stati introdotti alcuni pulsanti aggiuntivi:

- **Delete** elimina l'ultimo carattere inserito
- **Space** aggiunge uno spazio tra le parole
- **Clear** cancella l'intera frase

Questi pulsanti sono stati integrati poiché, pur essendo state definite delle convenzioni per rappresentare i comandi "spazio" e "cancella" anche nell'alfabeto dattilologico (inseriti comunque nel nostro vocabolario per completezza), non tutti gli utenti potrebbero conoscerli o trovarli intuitivi. L'aggiunta dei pulsanti permette quindi di semplificare e rendere più accessibile l'esperienza d'uso.

Infine, è presente una selezione dedicata alla scelta della fotocamera da utilizzare, utile ad esempio nei casi in cui il dispositivo disponga sia di una fotocamera frontale che posteriore, come avviene comunemente negli smartphone.

Durante l'uso della schermata di traduzione, così come nelle modalità di gioco e apprendimento integrate, è sempre presente un canvas dedicato alla visualizzazione della mano rilevata. Questo strumento mostra in tempo reale la posizione della mano riconosciuta sullo schermo, evidenziandone i punti di riferimento principali (landmarks). È importante sottolineare che il sistema è progettato per riconoscere una sola mano alla volta, in quanto l'alfabeto dattilologico si realizza esclusivamente con una mano.

6.4.5 Minigiochi

All'interno dell'applicazione è presente una sezione dedicata ai minigiochi, pensata per aiutare l'utente ad apprendere e consolidare la conoscenza dell'alfabeto dattilologico. I giochi sono disponibili in entrambe le lingue supportate dall'app

e offrono tre livelli di difficoltà, pensati per adattarsi alle diverse fasi di apprendimento.

Modalità Facile In questa modalità, all'utente viene mostrata una lettera dell'alfabeto accompagnata da un'immagine esplicativa che mostra la corretta posizione della mano. Il sistema di intelligenza artificiale analizza in tempo reale la mano dell'utente per verificare la corretta esecuzione del segno. Finché il gesto non viene riconosciuto correttamente, l'immagine rimane invariata. Una volta riconosciuto il segno, il sistema propone automaticamente una nuova lettera scelta in modo casuale, e così via senza limiti.

Modalità Media Nella modalità intermedia, il livello di difficoltà aumenta: le lettere vengono proposte senza alcun supporto visivo, richiedendo quindi all'utente di ricordare autonomamente come eseguire correttamente ciascun segno.

Modalità Difficile La modalità avanzata introduce una sfida ancora più complessa: all'utente viene richiesta la traduzione di intere frasi, che dovranno essere composte correttamente utilizzando l'alfabeto dattilologico. Durante l'esecuzione, una barra di avanzamento mostrerà la percentuale di completamento, che aumenterà man mano che ogni lettera viene eseguita correttamente.

6.4.6 Modulo per feedback

Abbiamo ritenuto essenziale integrare nella nostra applicazione un sistema di feedback immediato per raccogliere segnalazioni e suggerimenti dagli utenti. Questo aspetto è particolarmente importante poiché, prima dell'inizio del progetto, nessuno di noi aveva familiarità con l'alfabeto dattilologico. Per questo motivo, il contributo diretto di chi utilizza l'applicazione e ha esperienza in questo ambito è per noi di valore inestimabile, molto più di qualsiasi test interno.

Per semplificare il processo e incentivare gli utenti a condividere il loro feedback, abbiamo scelto una soluzione rapida, evitando la necessità di registrazione o autenticazione. Il modulo è strutturato in modo da consentire l'invio anonimo delle segnalazioni tramite email. Tuttavia, per chi desidera ricevere una risposta dal nostro team, è possibile inserire il proprio indirizzo email in modo facoltativo. Per la gestione dell'invio dei messaggi, abbiamo utilizzato la piattaforma **FormSubmit.co**[5], che funge da intermediario, permettendo di inoltrare automaticamente le email alla nostra casella di posta ufficiale del progetto:**raft.galilei@gmail.com**.

6.5 Back end

6.5.1 Conversione da Python a Javascript dell'AI

Come già spiegato in precedenza, una delle poche parti del progetto che erano già state sviluppate riguardava l'intelligenza artificiale, realizzata originariamente in **Python**. Tuttavia, nel corso dello sviluppo, abbiamo deciso di convertire il codice in JavaScript, permettendo così all'intera applicazione di funzionare direttamente sul client, senza la necessità di comunicare con un server esterno.

Questa scelta è stata dettata principalmente da due motivazioni. La prima riguarda le prestazioni: inviare in tempo reale le immagini catturate dalla videocamera del dispositivo a un server per l'elaborazione e poi ricevere il risultato del riconoscimento avrebbe richiesto troppo tempo. Questo ritardo avrebbe compromesso l'immediatezza della traduzione, che volevamo rendere istantanea.

La seconda ragione, altrettanto importante, è stata la possibilità di eseguire l'intero processo direttamente sul dispositivo dell'utente, rendendo la traduzione disponibile anche offline. Questo rappresenta un grande vantaggio, soprattutto in situazioni in cui l'accesso a Internet potrebbe essere limitato o assente, garantendo così un'esperienza più affidabile e accessibile.

6.5.2 Funzionameno dell'intelligenza

Il codice utilizza un modello pre-addestrato per il riconoscimento dei gesti fornito da **MediaPipe Tasks Vision**. Questo modello, chiamato **gestureRecognizer.task**, è stato precedentemente istruito per identificare varie posizioni della mano. Il codice accede alla webcam dell'utente ed elabora il flusso video, fotogramma per fotogramma. La funzione predictWebcam invia ogni fotogramma al modello pre-addestrato per l'analisi. Il modello identifica i punti chiave della mano (landmark) e il tipo di gesto che viene fatto, fornendo anche un punteggio di confidenza. Questi punti di riferimento vengono poi disegnati a video per una visualizzazione grafica.

I gesti riconosciuti vengono convertiti in testo tramite un algoritmo. Questa funzione associa ogni gesto a un carattere o azione specifica, come cancellare l'ultimo carattere o aggiungere uno spazio. Il testo risultante viene memorizzato e visualizzato a schermo. Sempre con questo algoritmo, non permettiamo che lettere con troppa poca "confidenza" di riconoscimento vengano stampate.

6.5.3 Dominio e certificato SSL

Per rendere l'applicazione accessibile a chiunque, abbiamo dovuto configurare un dominio personalizzato, in modo che il sito fosse facilmente raggiungibile tramite un qualsiasi motore di ricerca utilizzando il link: **raftonline.live**.

Per farlo, abbiamo acquistato un dominio, che ci è stato fornito gratuitamente per un anno (a partire da dicembre 2024) grazie ai nostri account **GitHub Student**[6]. Oltre al dominio, per garantire la sicurezza della connessione e rendere il sito condivisibile su Internet, abbiamo ottenuto un **certificato SSL** tramite la piattaforma **DigitalOcean**[**3**].

Successivamente, abbiamo configurato il **DNS (Domain Name System)** utilizzando **Name.com**[11], permettendo così la corretta risoluzione del dominio.

Attualmente, l'applicazione è completamente online e, grazie all'integrazione con GitHub, è anche estremamente facile da aggiornare. Il sito è infatti collegato alla nostra repository GitHub, che contiene il codice completo dell'applicazione. Ogni volta che viene aggiornato il ramo principale (main) su GitHub, il sito

si aggiorna automaticamente, garantendo che la versione disponibile online sia sempre l'ultima rilasciata.

6.5.4 Container

6.6 Divisione del lavoro

Task	Sviluppatori
Mockup	Chater, Braguti
Grafica	Braguti
FrontEnd	Chater, Braguti
BackEnd	Silva, Compagno
Intelligenza	Silva, Compagno, Chater
Documentazione	Chater
Diario di Bordo	Braguti

6.7 Base dati

6.7.1 Use Case Diagram

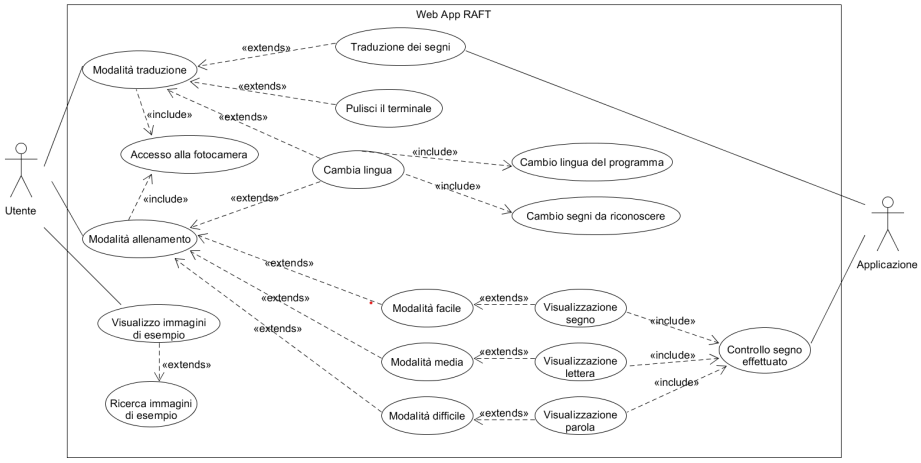

Figure 6: UML dell'applicazione

6.8 Come accedere a RAFT

Attualmente, consigliamo di utilizzare l'applicazione su smartphone, in quanto l'interfaccia e le funzionalità sono ottimizzate per i dispositivi mobili. Tuttavia, al momento il browser Safari non è supportato, poiché le librerie necessarie per l'intelligenza artificiale non sono compatibili con questo browser. L'applicazione è invece stata testata su browser come Chrome, Opera, Firefox e Edge.

Per accedere all'applicazione basti aprire un qualsiasi motore di ricerca, cercare **raftonline.live** e accedere al sito web. A questo punto, l'utente può scegliere se usare direttamente il sito web dal browser oppure installare la PWA, per la possibilità di utilizzarla offline.

6.9 Repository GitHub

L'intero codice sorgente del nostro progetto è disponibile pubblicamente su GitHub, nella repository ufficiale: [**RAFT-PROJECT**]. Abbiamo scelto di rendere il progetto open-source per garantire trasparenza e offrire la possibilità a chiunque di esplorare il codice, suggerire miglioramenti e contribuire allo sviluppo.

All'interno della repository si trovano il codice completo dell'applicazione, la documentazione tecnica e le istruzioni per l'installazione e l'uso. Inoltre, ogni modifica viene gestita tramite **GitHub Actions**, che automatizza il processo di verifica e aggiornamento, assicurando che la versione disponibile online sia sempre aggiornata.

6.10 Implementazioni future

Una delle evoluzioni previste riguarda l'implementazione di una logica di autocompletamento durante la fase di traduzione. Questa funzionalità permetterebbe di assistere l'utente nella composizione delle frasi, suggerendo in tempo reale parole o frasi basate sulle lettere già inserite. Un sistema di questo tipo, integrato con il vocabolario di riferimento e personalizzabile in base al contesto, potrebbe non solo velocizzare la scrittura, ma anche aiutare gli utenti meno esperti a familiarizzare con la lingua e con l'uso corretto dei segni.

7 Distribuzione

Nei target sopra elencati 5 abbiamo classificato i principali clienti della nostra applicazione, per questa ragione i metodi di dstribuzione sono i seguenti:

7.1 Scelta del nome

Poiché RAFT sarà un'applicazione accessibile tramite rete, abbiamo ritenuto opportuno darle un nome autoesplicativo: "Real-Time Automated Fingerspelling Translator". Questo nome riflette chiaramente la funzione dell'applicazione e

simula la tipica ricerca che un utente potrebbe fare su internet per trovare ciò di cui ha bisogno.

7.2 Overlimits

Il progetto RAFT è commissionato dall'associazione no-profit Overlimits di Crema, che ogni giorno collabora con persone con disabilità. Durante lo sviluppo del progetto, questa associazione è stata di grande supporto, offrendoci l'opportunità di ricevere feedback da quelli che saranno i nostri principali utenti.

7.3 Centri specializzati

Una volta completata l'applicazione, potremmo presentarci presso centri specializzati, come la Croce Rossa Italiana, per proporla come strumento di supporto nei casi in cui arrivi una persona sorda e non sia disponibile un interprete. In tali situazioni, RAFT potrebbe contribuire a superare le barriere comunicative, offrendo una soluzione pratica per la traduzione dell'alfabeto dattilologico.

References

[1] AVS4YOU. *AVS Video Converter*.

[2] Anthony Goldbloom e Ben Hamner. *Kaggle*.

[3] DigitalOcean, Inc. *DigitalOcean - Cloud computing designed for developers*.

[4] Inc. Figma. *Figma*.

[5] *FormSubmit - Easy Form Submission*.

[6] GitHub, Inc. *GitHub - Where the world builds software*.

[7] Google. *GoogleDrive*.

[8] Adobe Inc. *Adobe Lightroom Classic*.

[9] Antonio Magarotto. "Ente Nazionale Sordi ETS-APS". In: ().

[10] Meta Open Source. *React - A JavaScript library for building user interfaces*.

[11] Name.com, Inc. *Name.com - Domain Name Registration and Hosting*.

[12] *Progressive Web App*.

[13] Google Research. *Google Colaboratory*.