Face Recognition Log-In

Cybersecurity

Studente: Gianluca Giuseppe Maria De Nardi

Estratto

Il presente progetto si concentra sullo sviluppo di un sistema di riconoscimento facciale per un'applicazione mobile, sviluppata in Android Studio, con un'attenzione particolare agli aspetti di sicurezza informatica. La protezione dei dati personali degli utenti ha rappresentato la principale priorità durante tutte le fasi di progettazione e implementazione. Il sistema si avvale di una macchina virtuale su Google Cloud, configurata per garantire l'accesso esclusivo al proprietario, con un'autenticazione continua tramite i dati dell'account Google, assicurando un elevato livello di protezione dell'infrastruttura. Il codice dell'applicazione è stato sviluppato in Python e un prototipo separato è stato creato e testato per verificare la precisione del riconoscimento facciale prima della sua integrazione nell'applicazione mobile. Successivamente, il codice è stato adattato per consentire la comunicazione tramite richieste SSH con l'app Android Studio, implementando le funzionalità di registrazione, riconoscimento, acquisizione e autenticazione. I dati degli utenti, tra cui e-mail, nome utente, password (criptata utilizzando l'algoritmo di hashing bcrypt) e le 128 caratteristiche biometriche del volto, vengono archiviati in un database sicuro su Google Cloud. Oltre al riconoscimento facciale, il sistema offre anche una modalità di accesso tradizionale tramite nome utente e password, con una procedura di registrazione sicura che invia i dati al database per consentire l'autenticazione nelle sessioni successive. Questo progetto dimostra come sia possibile implementare un sistema di riconoscimento facciale efficace e sicuro, affrontando le sfide legate alla protezione dei dati sensibili e garantendo un accesso affidabile e conforme agli standard di sicurezza.

LOG-IN

La funzione ***login_face()*** gestiste e definisce un endpoint API che consente agli utenti di effettuare il login attraverso il riconoscimento facciale. Questa funzione è registrata nel percorso **/login_face** e accetta solo richieste HTTP di tipo POST. In particolare:

Ricezione dell'Immagine dal Client

- La funzione riceve un'immagine caricata dal client tramite una richiesta POST. L'immagine è recuperata dal dizionario **request.files** utilizzando la chiave '**image**'.
- L'immagine viene letta in memoria tramite ***.read()*** e viene quindi convertita in un array NumPy (utilizzando ***np.frombuffer***) per poter essere elaborata da OpenCV.
- Con il metodo ***cv2.imdecode()***, l'immagine viene decodificata in un formato utilizzabile (ossia una matrice che rappresenta l'immagine a colori).

Rilevazione del Volto

- Utilizzando la libreria **face_recognition**, viene estratto l'encoding facciale daal frame corrente tramite il metodo ***face_recognition.face_encodings(frame)***. Questo encoding è un vettore numerico che rappresenta le caratteristiche uniche del volto.
- Se non vengono rilevati volti nell'immagine, la funzione restituisce un errore con **codice 400** e un messaggio indicante che nessun volto è stato rilevato (*'error': 'No face detected').*

Confronto con i volti registrati

- Se è stato rilevato un volto, viene effettuato il confronto con i volti registrati nel database.
- La funzione ***get_all_user_images()*** viene utilizzata per ottenere tutti gli encoding dei volti già registrati. Questi encoding rappresentano i volti di utenti registrati nel sistema.
- Il confronto tra l'encoding del volto fornito e gli encoding registrati viene fatto tramite ***face_recognition.compare_faces***. Questa funzione restituisce un elenco di valori booleani che indicano se c'è una corrispondenza tra il volto rilevato e un encoding registrato.

Gestione Successo e Fallimento

- Se viene trovata una corrispondenza (**any(matches)** restituisce **True**), significa che il volto dell'utente corrisponde a uno di quelli già registrati. In questo caso, la funzione restituisce una risposta di successo (*'Login successful', 'user_id': user_image['user_id'*]) con il **codice** di stato **200**.
- Se nessuna corrispondenza viene trovata dopo aver confrontato tutti gli encoding nel database, la funzione restituisce un errore con **codice 401** e un messaggio indicante che il volto non è stato riconosciuto (*'error': 'Face not recognized'*).

API di REGISTRAZIONE

Gli utenti si possono registrare nel sistema fornendo le proprie informazioni personali, una password, e un'immagine facciale per il riconoscimento. Questa fase viene gestita da un **API** di registrazione tramite la funzione ***register()*** che definisce un endpoint API al percorso ***/register*** che accetta solo richieste **HTTP** di tipo **POST**.

- La funzione register() definisce un endpoint API al percorso /register che accetta solo richieste HTTP di tipo POST.
- *email*, *username*, e *password* sono estratti dal form della richiesta (***request.form***), che rappresenta le informazioni necessarie per la registrazione.

CIFRATURA della PASSWORD

Una volta ottenuta la password in chiaro, viene subito cifrata utilizzando la funzione ***generate_password_hash()***, che tramite la libreria ***werkzeug.security*** serve per generare un hash sicuro della password, che verrà poi salvato nel database.

PBKDF2 con SHA-256

- **PBKDF2** (**P**assword-**B**ased Key **D**erivation **F**unction **2**) è un algoritmo di derivazione delle chiavi che rende l'hashing più sicuro applicando iterazioni multiple della funzione di hash sulla password. Ciò rende più dispendioso per un attaccante tentare di indovinare la password. Inoltre, questo algoritmo, genera automaticamente un **salt** (un valore casuale aggiunto alla password prima dell'hashing). Questo garantisce che anche password uguali abbiano hash differenti e che gli attacchi con tabelle *pre-computate* (rainbow tables) siano inefficaci.
- **SHA-256** è una funzione di hash crittografico che produce un *hash* di **256 bit**, garantendo che ogni password abbia una rappresentazione univoca non invertibile.

L’utilizzo di questo metodo è utilizzato per garantire che le password siano protette contro attacchi *brute-force* o *dizionario*.

Riconoscimento Facciale

- L'API riceve anche un'immagine del volto dell'utente, che viene letta e convertita in un formato appropriato per l'elaborazione tramite OpenCV.
- Successivamente, viene estratto l'**encoding facciale** utilizzando la libreria ***face_recognition***, che rappresenta le caratteristiche uniche del volto in un vettore numerico.
- Se non vengono rilevati volti nell'immagine fornita, la funzione restituisce un errore con **codice 400**, informando il client che nessun volto è stato rilevato.

Salvataggio dei dati nel Database

- Una volta hashata la password e ottenuto l'encoding del volto, i dati dell'utente vengono salvati nel database.
- La funzione ***save_user()*** crea un nuovo documento nella collezione *'users'* di **Firestore**, salvando l'email, il nome utente, la password hashata e l'encoding del volto.

DATABASE

Per il progetto è stato utilizzato **Firestore**, un servizio di database NoSQL di Google Cloud, per gestire la memorizzazione e il recupero dei dati degli utenti, inclusi i loro dati biometrici per il riconoscimento facciale. La libreria viene inizializzata creando un'istanza del client di Firestore. Questo oggetto **db** è utilizzato per interagire con il database, consentendo operazioni di *lettura* e *scrittura* sui documenti e le collezioni memorizzate.

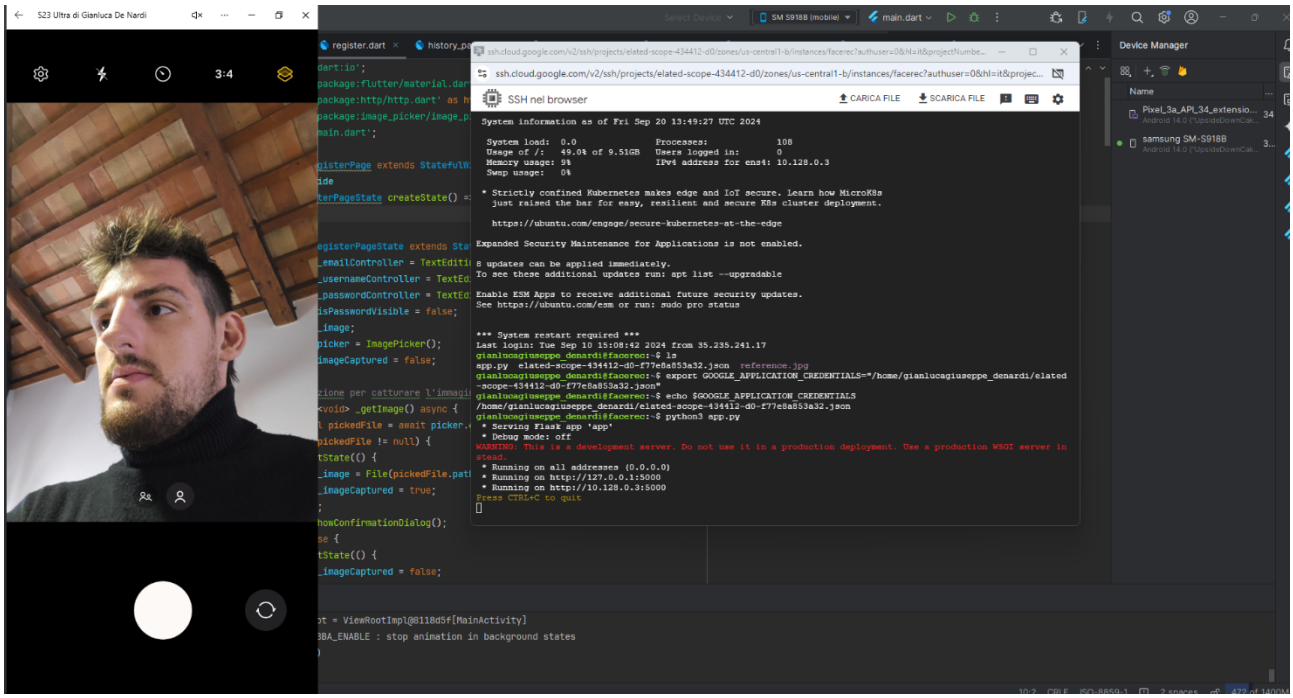
- La collezione *'users'* contiene i dati di ciascun utente come documenti separati.
- Ogni documento viene convertito in un dizionario Python tramite il metodo ***.to_dict()***, che restituisce tutti i campi e i loro valori per quel documento.
- I dati rilevanti (email, nome utente, encoding facciale, e l'ID del documento) vengono aggiunti a una lista ***user_images***, che rappresenta tutti gli utenti registrati.
- La funzione restituisce la lista ***user_images***, che contiene i dati di tutti gli utenti salvati, inclusi i loro encoding facciali. Questa lista viene utilizzata successivamente per confrontare i volti durante il processo di log-in.

ACCESSO all'APP

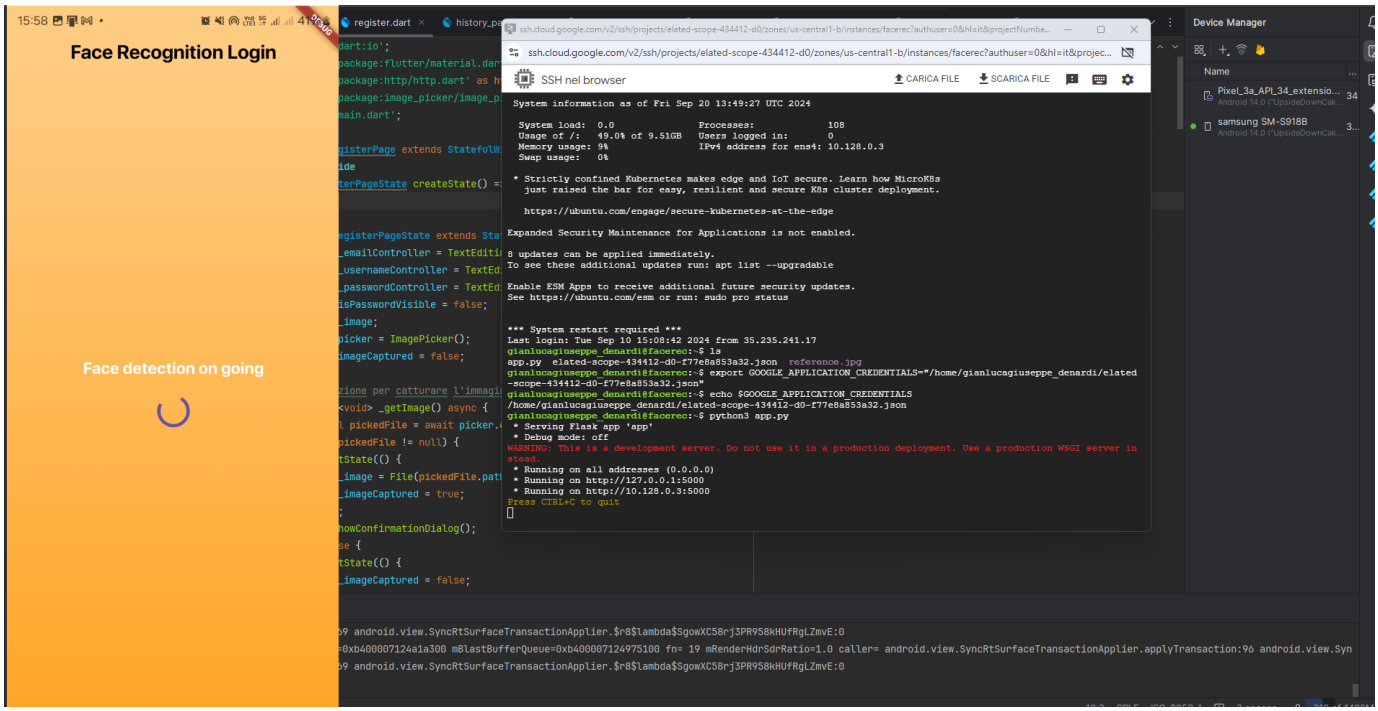
Per garantire un accesso sicuro ed efficiente all'applicazione, ogni utente deve seguire alcune fasi specifiche. Questo processo di autenticazione è stato progettato per proteggere i dati personali e fornire una *user experience* semplice ma sicura. Di seguito sono descritte in dettaglio le diverse fasi che compongono il processo di accesso, dal momento in cui l'utente fornisce le proprie credenziali alla verifica finale dell'identità tramite riconoscimento facciale o password tradizionale.

Primo Accesso

Al primo accesso si apre la fotocamera del dispositivo, l'utente può mandare la sua immagine presa in tempo reale, a destra si può vedere la Virtual Machine con il modello di AI e le funzioni descritte precedentemente in esecuzione.

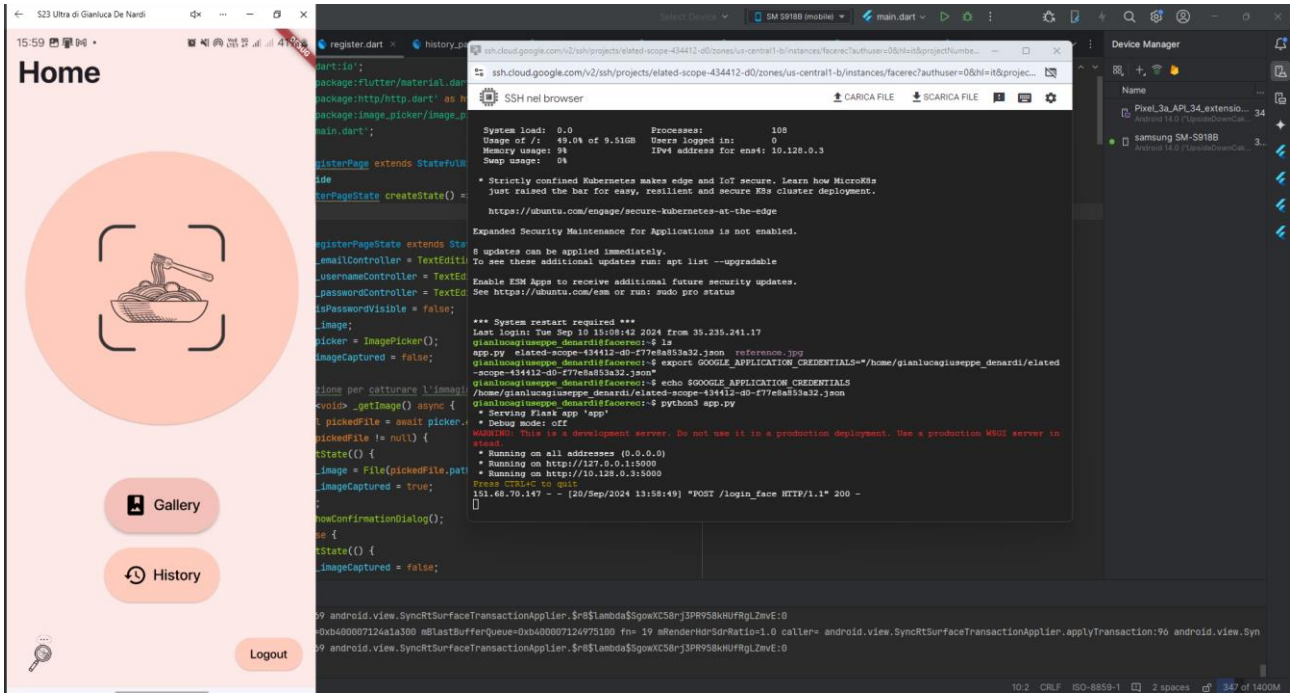


Il dispositivo mostrerà una pagina di caricamento mentre l'App esegue il matching per il riconoscimento.



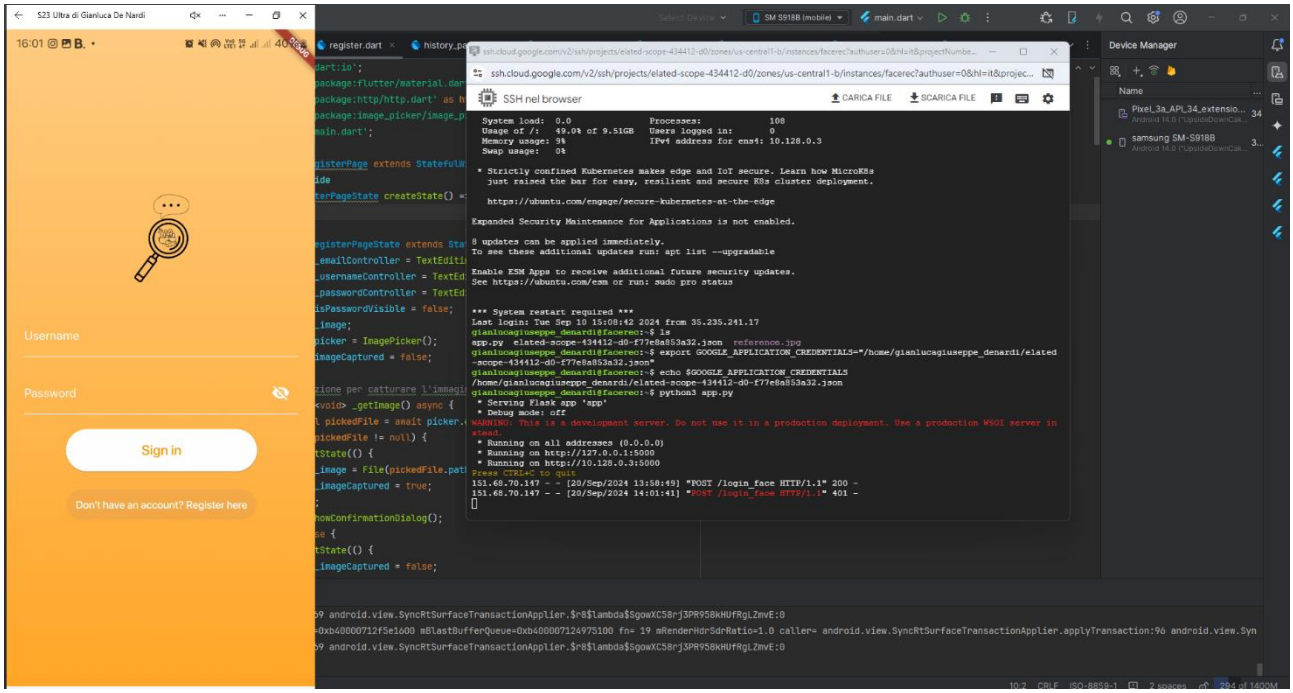
Riconoscimento con Successo

In questo di riconoscimento effettuato viene consentito l'accesso all'applicazione

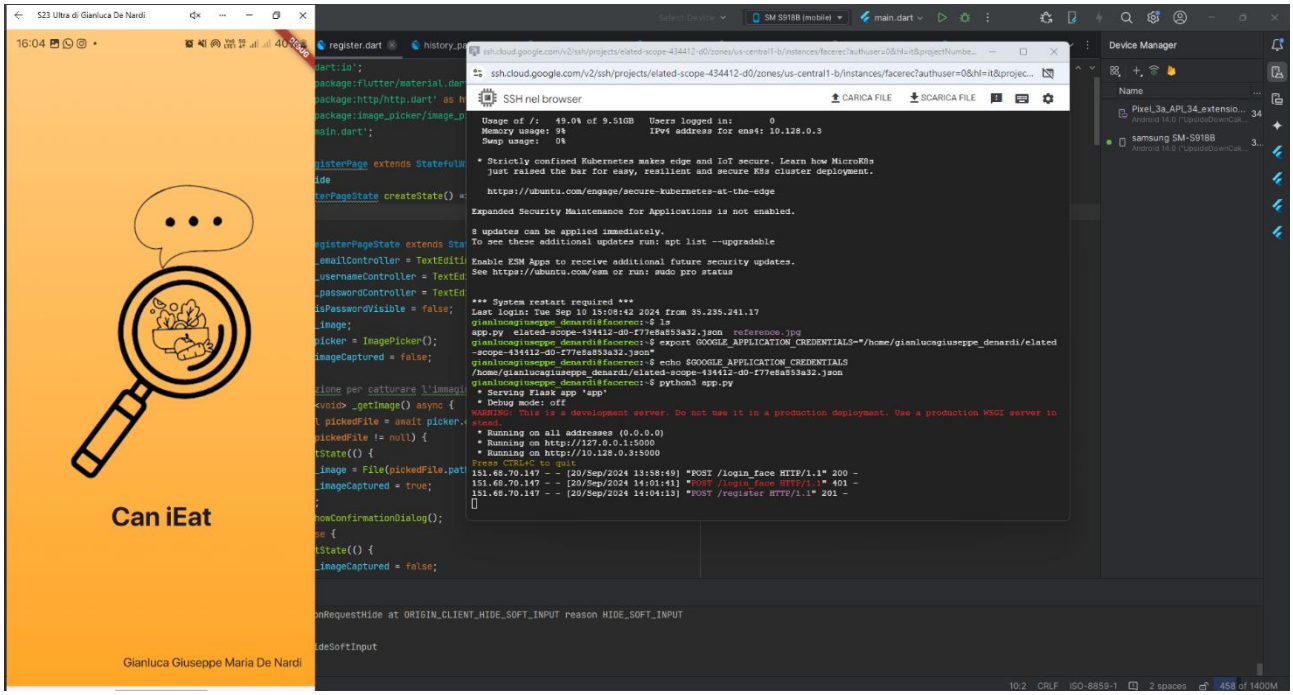


Riconoscimento Fallito

In caso il riconoscimento fallisca l'utente viene indirizzato in una pagina dove può accedere con *Nome Utente e Password*.

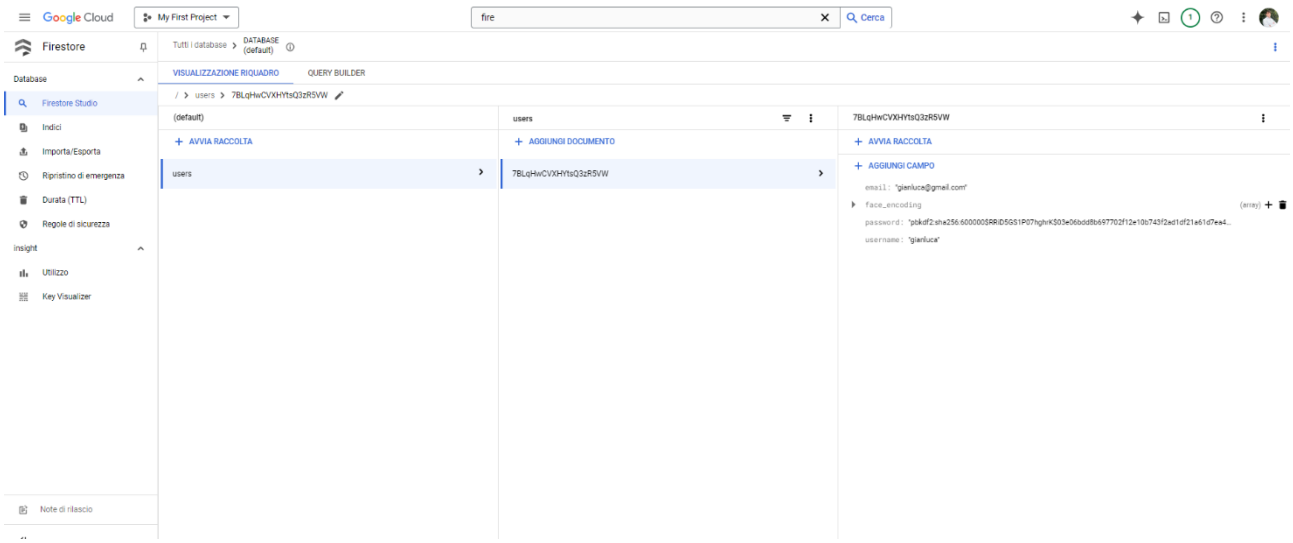


Se non è iscritto può cliccare in '*Don't have an account? Register here*', dove potrà compilare i campi di registrazione con l'API implementata e potrà accedere i propri dati.

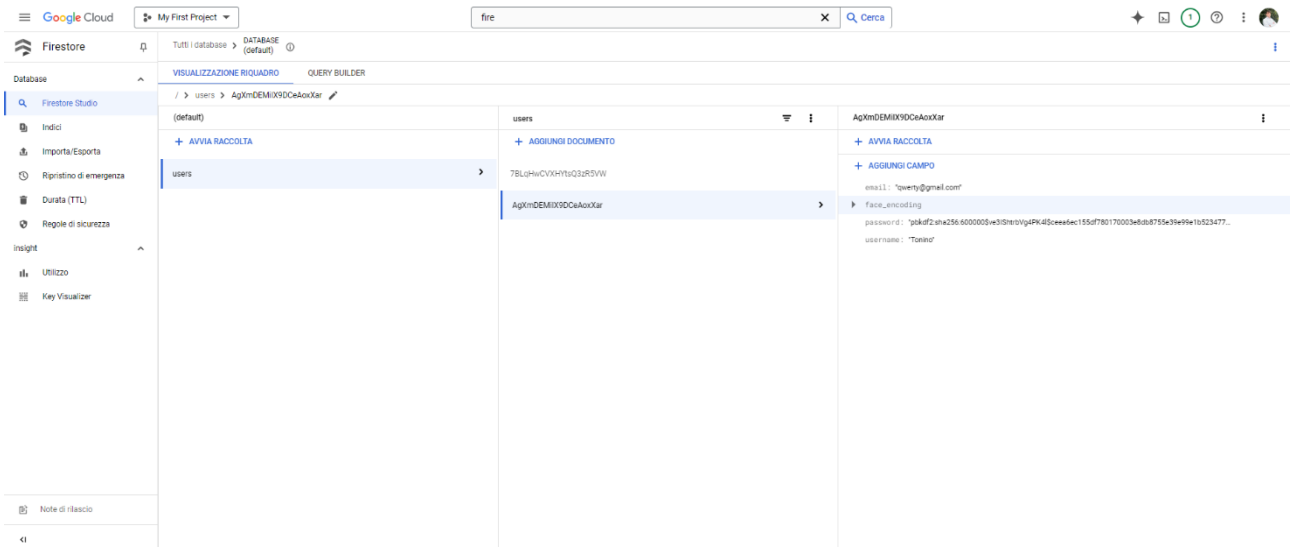


Firestone

Il Database si presenta nel seguente modo:



Dopo la registrazione di un *nuovo utente* si può vedere istantaneamente l'aggiunta di un nuovo campo nella sezione '*Users'*.



CONCLUSIONI

In conclusione, il progetto ha dimostrato che è possibile realizzare un sistema di riconoscimento facciale efficace e sicuro per applicazioni mobili, affrontando con successo le sfide legate alla protezione dei dati sensibili degli utenti. Grazie all'utilizzo di una macchina virtuale su Google Cloud, è stato possibile garantire un'infrastruttura sicura e accessibile tracciando gli utenti che si vogliono interfacciare. L'uso dell'algoritmo di hashing bcrypt per la cifratura delle password, insieme alla memorizzazione sicura dei dati biometrici, ha contribuito a salvaguardare le informazioni personali degli utenti da possibili compromissioni. Inoltre, l'implementazione del sistema di login tramite riconoscimento facciale ha offerto un metodo innovativo e conveniente per l'autenticazione, affiancato dalla modalità tradizionale tramite nome utente e password per garantire la flessibilità d'uso. Nel complesso, il progetto rappresenta una realizzazione pratica che unisce tecnologie avanzate per la sicurezza e l'autenticazione degli utenti in ambiente mobile, dimostrando come sia possibile implementare un sistema efficiente a costo zero. Questi sistemi, sebbene complessi, sono spesso dati per scontato dagli utenti che li utilizzano quotidianamente sui dispositivi più moderni.