Documentazione Progetto Chat Locale

# Indice

1 Indice 2

1 Introduzione 3

1.1 Informazioni sul progetto 3

1.2 Abstract 3

1.3 Scopo 5

2 Analisi 6

2.1 Analisi del dominio 6

2.2 Analisi e specifica dei requisiti 7

2.3 Use case 9

2.4 Pianificazione 9

2.5 Analisi dei mezzi 12

2.5.1 Software 12

2.5.2 Hardware 12

3 Progettazione 13

3.1 Design dell’architettura del sistema 13

3.2 Design dei dati e database 14

3.3 Design delle interfacce 15

Design procedurale 17

4 Implementazione 18

5 Test 29

5.1 Protocollo di test 29

5.2 Risultati test 30

5.3 Mancanze/limitazioni conosciute 30

6 Consuntivo 30

7 Conclusioni 32

7.1 Sviluppi futuri 32

7.2 Considerazioni personali 32

8 Bibliografia 33

8.1 Bibliografia per articoli di riviste: 33

8.2 Bibliografia per libri 33

8.3 Sitografia 33

9 Glossario 34

10 Indice delle figure 34

11 Allegati 34

# Introduzione

## Informazioni sul progetto

In questo capitolo raccogliere le informazioni relative al progetto, ad esempio:

* Allievo: Andrea Negri
* Docente: Geo Petrini
* Scuola: SAMT
* Sezione: Informatica
* Modulo: M306
* Data inizio: 04.09.2024
* Data fine: 18.12.2024

## Abstract

È una breve e accurata rappresentazione dei contenuti di un documento, senza notazioni critiche o valutazioni. Lo scopo di un abstract efficace dovrebbe essere quello di far conoscere all’utente il contenuto di base di un documento e metterlo nella condizione di decidere se risponde ai suoi interessi e se è opportuno il ricorso al documento originale.

Può contenere alcuni o tutti gli elementi seguenti:

* **Background/Situazione iniziale**
* **Descrizione del problema e motivazione**: Che problema ho cercato di risolvere? Questa sezione dovrebbe includere l'importanza del vostro lavoro, la difficoltà dell'area e l'effetto che potrebbe avere se portato a termine con successo.
* **Approccio/Metodi**: Come ho ottenuto dei progressi? Come ho risolto il problema (tecniche…)? Quale è stata l’entità del mio lavoro? Che fattori importanti controllo, ignoro o misuro?
* **Risultati**: Quale è la risposta? Quali sono i risultati? Quanto è più veloce, più sicuro, più economico o in qualche altro aspetto migliore di altri prodotti/soluzioni?

**Progetto Chat Locale Multiutente con Firebase**

La crescente necessità di comunicazioni rapide ed efficienti in contesti locali, come scuole, team di lavoro. evidenzia In un mondo come questoabbiamo bisogno di delle chat affidabili, scalabili e facili da utilizzare. Questo progetto nasce con l’obiettivo di sviluppare una soluzione di chat locale multiutente, utilizzando Firebase come backend(senza un server), per risolvere problemi legati alla sincronizzazione in tempo reale, alla gestione degli utenti e alla sicurezza dei dati che altri metodi hanno.

Questo progetto nasce dalla voglia di creare una chat per uso pubblico dove tutte le persone hanno la possibilità di accedervi e scambiare qualche parola con altre persone.

La realizzazione di una piattaforma del genere affronta delle problematiche come la sincronizzazione affidabile tra più utenti in tempo reale, la gestione scalabile degli accessi utente e l’implementazione di misure di sicurezza, garantendo al contempo facilità di sviluppo e manutenzione. La soluzione è stata creata per avere una chat molto semplice e basilare che le persone possono utilizzare per comunicare all’ interno di un contesto locale. Questa chat infatti è puntata a un pubblico giovane che vuole interfacciarsi ad altri compaesani

Il progetto utilizza i principali servizi di Firebase:

* **Cloud Firestore** per il salvataggio e la sincronizzazione dei messaggi in tempo reale.
* **Firebase Authentication** per l’autenticazione sicura degli utenti.
* **Realtime Database** per il salvataggio di dati.

L’implementazione è focalizzata su un’architettura client-server, con il frontend sviluppato per fornire un’interfaccia utente intuitiva e compatibile non solo per i PC ma anche per e dispositivi mobili come Ipad o cellulari. Firebase funge da infrastruttura serverless, minimizzando la complessità e garantendo efficienza.

La chat sviluppata consente una comunicazione in tempo reale affidabile, con un’elevata reattività grazie alla sincronizzazione istantanea dei messaggi diretta da Firebase e resta comunque accessibile tramite un'interfaccia facile e intuitiva. La piattaforma risulta pronta per eventuali future estensioni (come aggiunta di admin, controllo dei messaggi, notifiche, …) o integrazione con sistemi esistenti. La chat infatti è anche molto accessibile dato che offre più metodi di registrazione e login. Rispetto a soluzioni tradizionali, il progetto si distingue per la sua facilità di estensione, alla sua leggerezza e semplicità.

## Scopo

Lo scopo del progetto (scopi didattici/scopi operativi). Dovrebbe descrivere il mandato, ma non vanno ricopiate le informazioni del quaderno dei compiti (che va invece allegato).

Lo scopo di questo progetto è quello di poter chattare localmente grazie ad un applicativo.

# Analisi

## Analisi del dominio

Questo capitolo dovrebbe descrivere il contesto in cui il prodotto verrà utilizzato, da questa analisi dovrebbero scaturire le risposte a quesiti quali ad esempio:

* Background/Situazione iniziale
* Quale è e come è organizzato il contesto in cui il prodotto dovrà funzionare?
* Come viene risolto attualmente il problema? Esiste già un prodotto simile?
* Chi sono gli utenti? Che bisogni hanno? Come e dove lavorano?
* Che competenze/conoscenze/cultura posseggono gli utenti in relazione con il problema?
* Esistono convenzioni/standard applicati nel dominio?
* Che conoscenze teoriche bisogna avere/acquisire per poter operare efficacemente nel dominio?
* …

Questo prodotto sarà un applicativo web che consentirà a due o più utenti di comunicare su un canale privato. Gli utenti potranno scrivere e inviare messaggi tra di loro oppure in un canale aperto a tutti.

## Analisi e specifica dei requisiti

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Requisito** | Req-01 | **Priorità** | 1 | **Versione** | 1.0 |
| **Nome** | Login con nome utente e password. | | | | |
| **Note** | Deve esistere una pagina di login dove l’utente può fare login e “sbloccare” il suo account con cui può scrivere sulle chat. | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Requisito** | Req-02 | **Priorità** | 1 | **Versione** | 1.0 |
| **Nome** | Avere un server dove gli utenti si collegano | | | | |
| **Note** | Serve un server dove gli utenti possano accedere per chattare con altri utenti | | | | |
| **Sotto requisiti** | | | | | |
| **001** | Utilizzare Firebase | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Requisito** | Req-03 | **Priorità** | 1 | **Versione** | 1.0 |
| **Nome** | Profilo con immagine personalizzata e stato di attività | | | | |
| **Note** | Il profilo di ogni utente dovrà avere un’immagine e uno stato di attività | | | | |
| **Sotto requisiti** | | | | | |
| **001** | La immagine si potrà anche cambiare | | | | |
| **002** | Dovrà esserci una lista di tutte le persone online in quel momento | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Requisito** | Req-04 | **Priorità** | 1 | **Versione** | 1.0 |
| **Nome** | Chat multifunzione | | | | |
| **Note** | Nelle chat oltre a scrivere si potranno inviare immagini e audio con bassa priorità. | | | | |
| **Sotto requisiti** | | | | | |
| **001** | Le immagini possono solo essere o png o jpg | | | | |
| **002** | Gli audio possono solo essere mp3 o wav | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Requisito** | Req-05 | **Priorità** | 1 | **Versione** | 1.0 |
| **Nome** | Storico delle chat di una settimana | | | | |
| **Note** | Le chat dovranno essere salvate per una settimana e poi andare perse | | | | |
| **Sotto requisiti** | | | | | |
| **001** | Le chat dovranno essere salvate quando l’utente finisce di conversare con un altro utente | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Requisito** | Req-06 | **Priorità** | 1 | **Versione** | 1.0 |
| **Nome** | Responsive | | | | |
| **Note** | Il sito dovrà essere responsive | | | | |

**Spiegazione elementi tabella dei requisiti:**

**ID**: identificativo univoco del requisito

**Nome**: breve descrizione del requisito

**Priorità**: indica l’importanza di un requisito nell’insieme del progetto, definita assieme al committente. Ad esempio, poter disporre di report con colonne di colori diversi ha priorità minore rispetto al fatto di avere un database con gli elementi al suo interno. Solitamente si definiscono al massimo di 2-3 livelli di priorità.

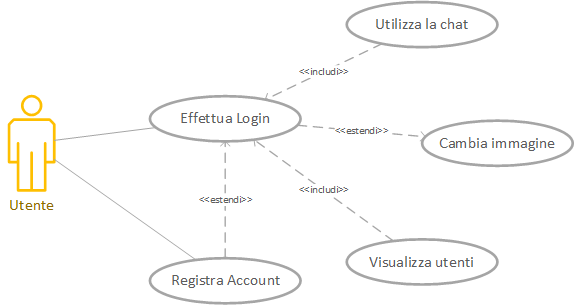
**Versione**: indica la versione del requisito. Ogni modifica del requisito avrà una versione aggiornata.

Sulla documentazione apparirà solamente l’ultima versione, mentre le vecchie dovranno essere inserite nei diari.

**Note**: eventuali osservazioni importanti o riferimenti ad altri requisiti.

**Sotto requisiti**: elementi che compongono il requisito.

## Use case

I casi d’uso rappresentano l’interazione tra i vari attori e le funzionalità del prodotto.

## Pianificazione

Prima di stabilire una pianificazione bisogna avere almeno una vaga idea del modello di sviluppo che si intende adottare. In questa sezione bisognerà inserire il modello concettuale di sviluppo che si seguirà durante il progetto. Gli elementi di riferimento per una buona pianificazione derivano da una scomposizione top-down della problematica del progetto.

La pianificazione può essere rappresentata mediante un diagramma di Gantt:



Figura 1 Esempio di diagramma di Gantt

Stile orizzontale



Figura 2 Esempio di diagramma di Gantt

## Analisi dei mezzi

Elencare e descrivere i mezzi disponibili per la realizzazione del progetto. Ricordarsi di sempre descrivere nel dettaglio le versioni e il modello di riferimento.

Firebase:

* Firebase Realtime Database
* Firebase Authentication
* Firestore Database

### Software

Backend:

* Firebase (Backend e Servizi Cloud)

Frontend:

* Javascript

Stile e formattazione:

* HTML
* CSS

Schemi:

* Microsoft Visio

IDE di sviluppo:

* Visual Studio Code

Browser utilizzato:

* Firefox

### Hardware

Su quale piattaforma dovrà essere eseguito il prodotto? Che hardware particolare è coinvolto nel progetto? Che particolarità e limitazioni presenta? Che HW sarà disponibile durante lo sviluppo?

Questo progetto è stato pensato per utenti Windows ma anche altri OS possono interfacciarsi. È anche possibile accedervi da un dispositivo mobile.

# Progettazione

Questo capitolo descrive esaustivamente come deve essere realizzato il prodotto fin nei suoi dettagli. Una buona progettazione permette all’esecutore di evitare fraintendimenti e imprecisioni nell’implementazione del prodotto.

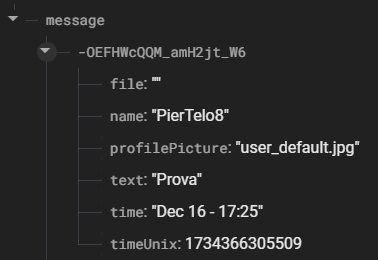
## Design dell’architettura del sistema

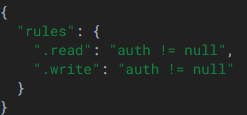
Descrive:

* La struttura del programma/sistema lo schema di rete...
* Gli oggetti/moduli/componenti che lo compongono.
* I flussi di informazione in ingresso ed in uscita e le relative elaborazioni. Può utilizzare *diagrammi di flusso dei dati* (DFD).
* Eventuale sitemap

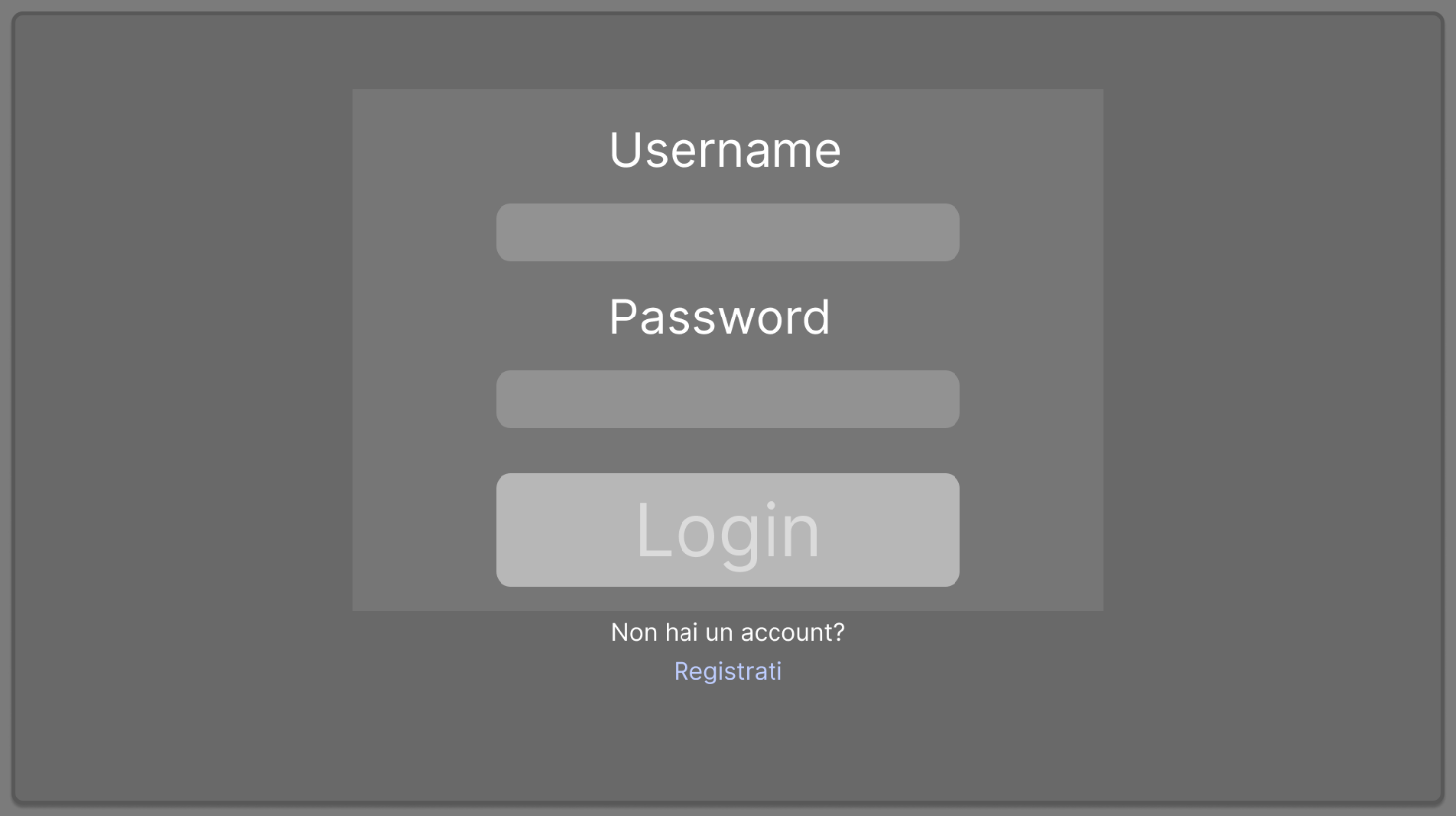
## Design dei dati e database

I dati verranno salvati in un database noSQL integrato in firebase.

Struttura dei dati:

Regole del DB:

## Design delle interfacce

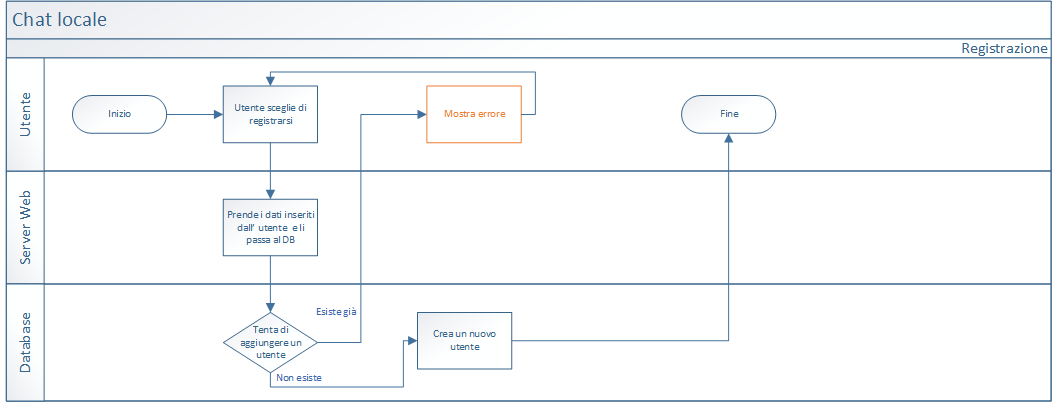




## Design procedurale

Descrive i concetti dettagliati dell’architettura/sviluppo utilizzando ad esempio:

* Diagrammi di flusso e Nassi.
* Tabelle.
* Classi e metodi.
* Tabelle di routing
* Diritti di accesso a condivisioni …

Questi documenti permetteranno di rappresentare i dettagli procedurali per la realizzazione del prodotto.

# Implementazione

In questo capitolo dovrà essere mostrato come è stato realizzato il lavoro. Questa parte può differenziarsi dalla progettazione in quanto il risultato ottenuto non per forza può essere come era stato progettato.

Sulla base di queste informazioni il lavoro svolto dovrà essere riproducibile.

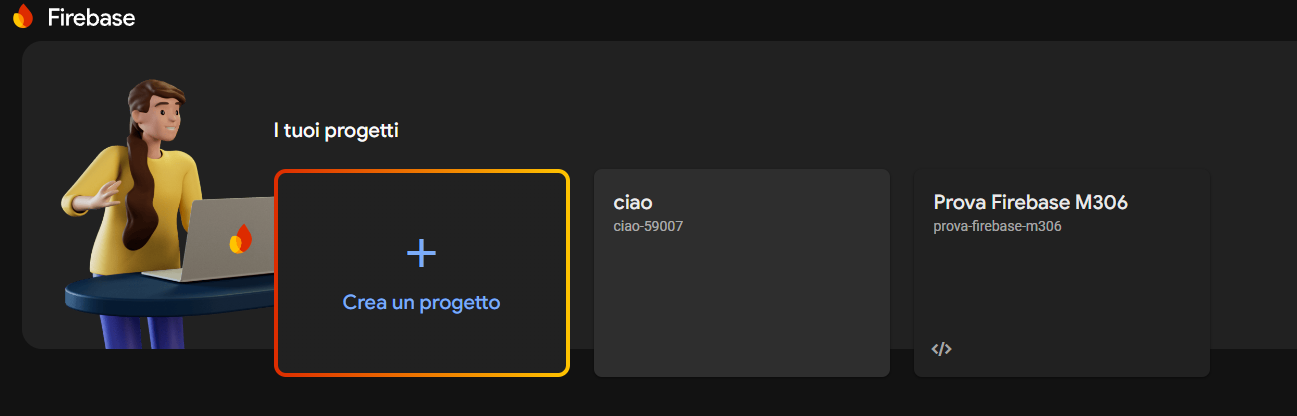
In questa parte è richiesto l’inserimento di codice sorgente - Print Screen - di maschere solamente per quei passaggi particolarmente significativi e/o critici.

Inoltre, dovranno essere descritte eventuali varianti di soluzione o scelte di prodotti con motivazione delle scelte.

Non deve apparire nessuna forma di guida d’uso di librerie o di componenti utilizzati. Eventualmente questa va allegata.

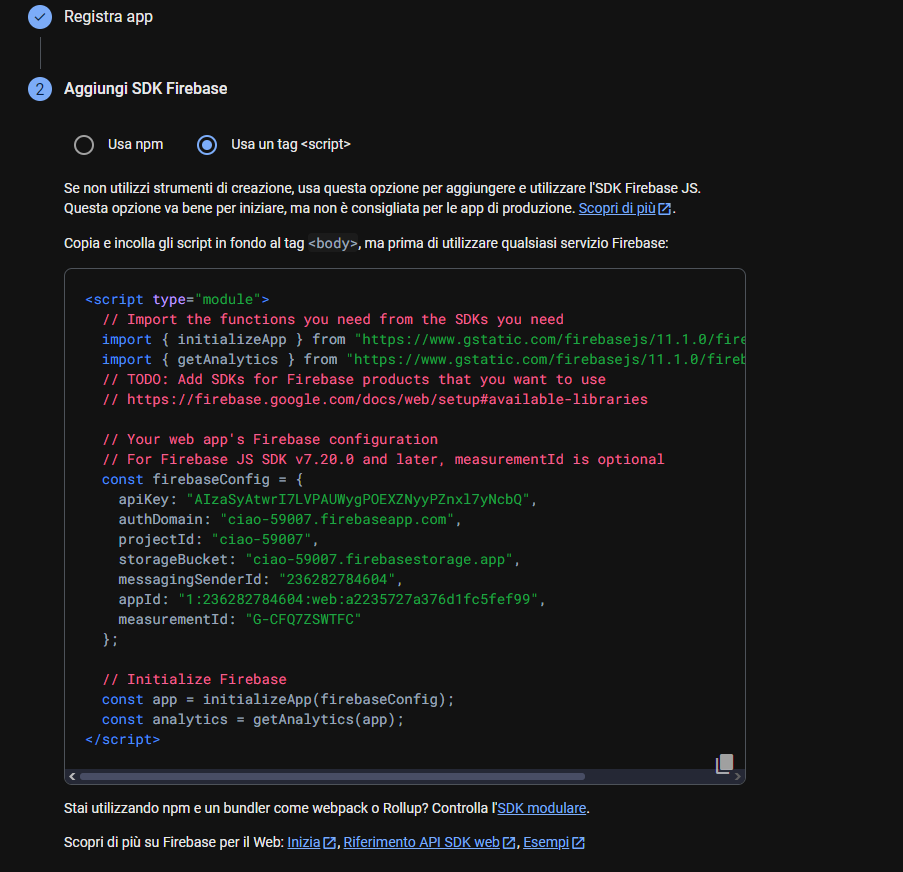
Per eventuali dettagli si possono inserire riferimenti ai diari.

Creazione progetto Firebase Web:

Come prima cosa recarsi sul sito di Firebase e fare il login. Andare sotto crea un nuovo progetto e inserire il nome che si vuole dare.

Dopodiché aggiungere una nuova applicazione Web da utilizzare per la nostra chat.

Dare un nuovo nome al progetto e continuare la configurazione.

Quando si arriva a “Aggiungi SDK Firebase” bisogna andare sotto “Usa un tag <script>” e copiare tutto in un file così da avere la configurazione che servirà a Javascript per interfacciarsi con Firebase.

Per finire cliccare su “Vai alla console” che vi porterà alla console di firebase dove avete tutto il necessario per poter gestire Firebase.

Spiegazione Codice:

Serve a importare alcuni metodi utili già creati di Firebase alla chat

Per collegare l'applicazione a **Firebase** (in questo caso si tratta del progetto Firebase web**)** è stata configurata la piattaforma tramite le credenziali fornite. La configurazione include:

* **API Key**: necessaria per identificare il progetto.
* **Auth Domain** e **Database URL**: per l'autenticazione e il database in tempo reale.
* **App ID** e **Sender ID**: utili per l'integrazione delle funzionalità di Firebase.

Eccone un esempio:

Nel caso del login aggiungiamo anche questo codice per gestire il login con google.

Questa costante serve per il Realtime Database di Firebase mentre la seconda per il Firestore Database.



La funzione **loadMessages()** carica i messaggi(in ordine di orario) presi dal Realtime Database e li aggiunge al div che contiene tutti i messaggi. Inoltre etichetta i messaggi inviati dall’utente se sono i propri o di un altro utente (per poi nel CSS cambiare la posizione e il colore).

Metodi e proprietà

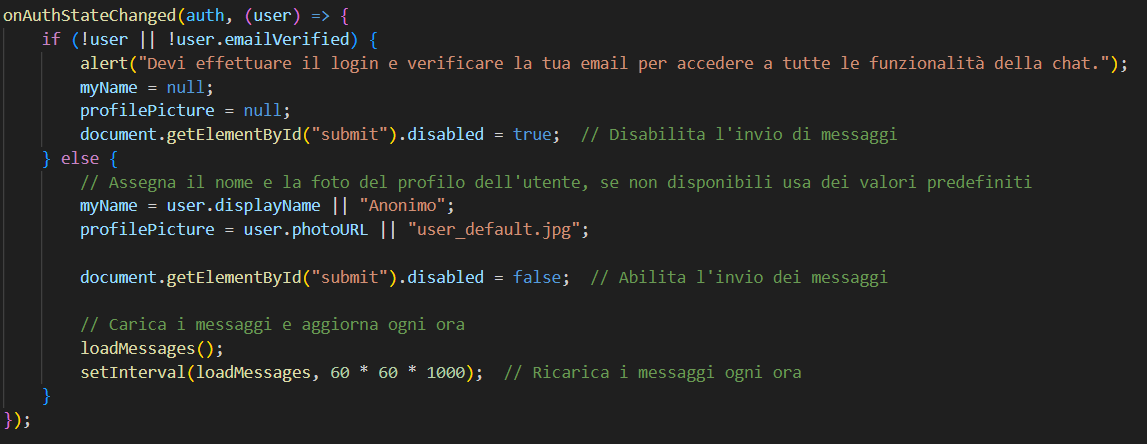
**query()** 🡪È una funzione che permette di combinare un riferimento al database con filtri e ordinamenti.

**ref()** 🡪 È una funzione che viene utilizzata per ottenere un riferimento a un percorso specifico all'interno del database. In questo caso in “message”

**onChildAdded()🡪**Il metodo onChildAdded si attiva per ogni messaggio presente in memoria

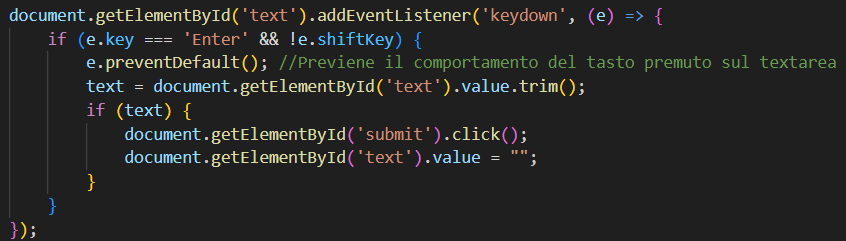
**data.val()** 🡪 data rappresenta il messaggio estratto grazie al onChildAdded() e il val()un metodo fornito dall'API di Firebase che converte i dati del messaggio(Nodo del DB dato che viene utilizzato un DB noSQL che utilizza una struttura a documenti JSON) in un oggetto JavaScript leggibile.

**scrollTop** 🡪scorre automaticamente fino all’ ultimo messaggio così da creare quel effetto di nuovo messaggio che appare.

Questo pezzo di codice gestisce il cambio di stato dell’autentificazione. Serve a decidere se fare vedere o no il contenuto della chat se l’utente è loggato.

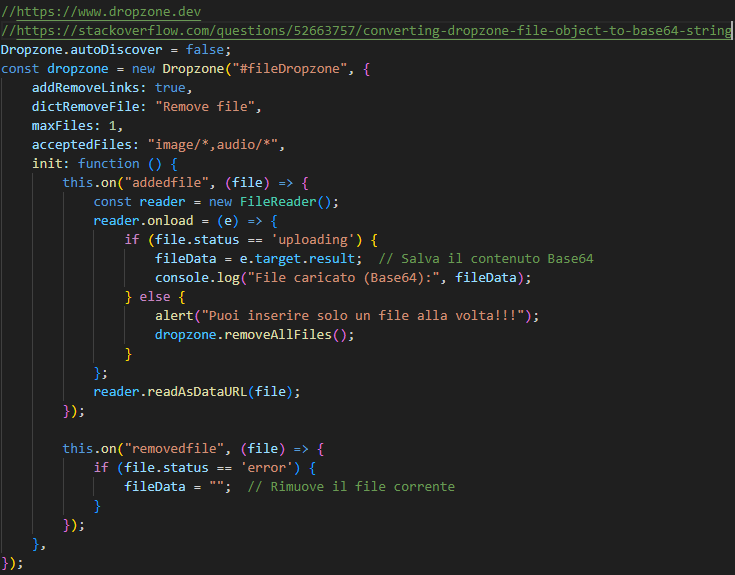
**onAuthStateChanged()** 🡪 È una funzione di Firebase Authentication che viene eseguita ogni volta che lo stato di autenticazione dell'utente cambia.

**setInterval() 🡪** È una funzione che esegue una determinata operazione o funzione a intervalli di tempo definite in essa(il tempo è in millisecondi).

Questa funzione si occupa di simulare il click del bottone di invio del messaggio con il tasto “Enter” della tastiera, lasciando però andare a capo l’utente nel textarea se viene premuto anche shift insieme al tasto “Enter”.

**click()** 🡪 Funzione che simula il click del tasto di invio dei messaggi

**preventDefault()** 🡪 Funzione che previene il comportamento di default di un oggetto HTML(in questo caso del nostro textarea)

Questa porzione di codice serve a gestire l’ input dei file con una libreria chiamata Dropzone.js che migliora il semplice input di html implementando anche la funzione di drag-and-drop

**Configurazione di Dropzone**

* **addRemoveLinks**: Aggiunge un link per rimuovere i file caricati.
* **dictRemoveFile**: Testo del link per rimuovere il file.
* **maxFiles**: Imposta un limite massimo di file caricabili (1 nel nostro caso).
* **acceptedFiles**: Limita il caricamento a file di tipo immagine (image/\*) o audio (audio/\*).

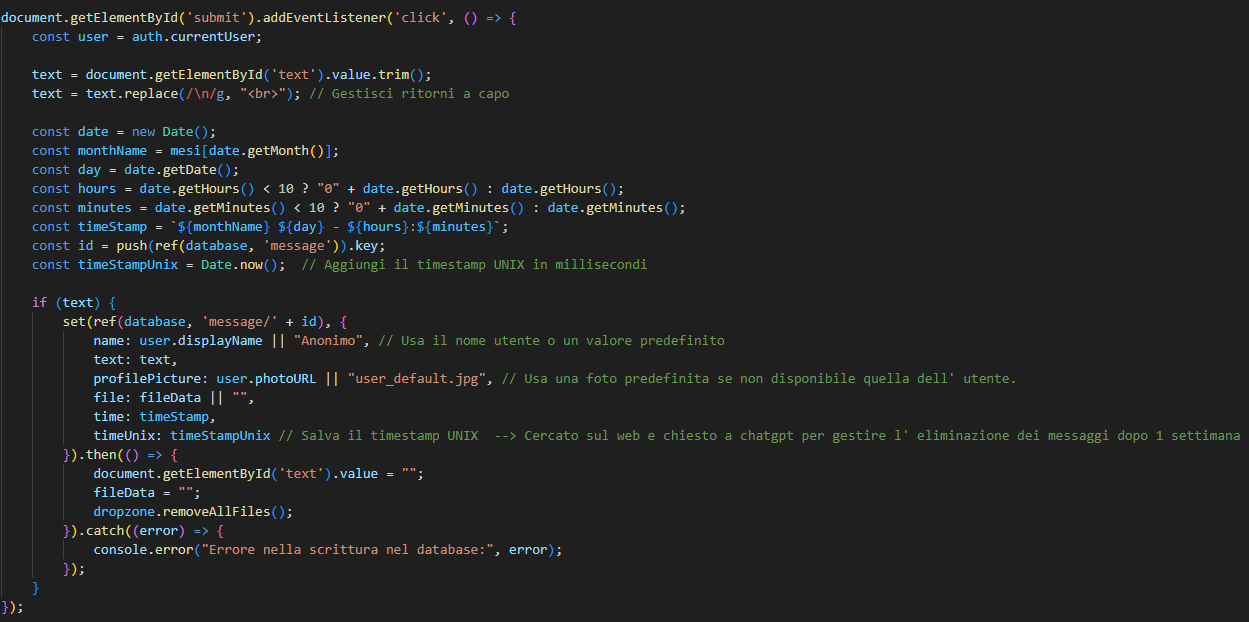
**this.on()** 🡪 È un listener di Dropzone per eventi specifici che avvengono solo durante il ciclo di upload di file.

**FileReader()** 🡪 È un oggetto che permette di leggere file in vari formati direttamente dal browser.

**removeAllFiles()** 🡪 È un metodo di Dropzone che permette di rimuovere tutti i file dalla zona.

**readAsDataURL()** 🡪 È un metodo fornito dalla classe FileReader() che permette di convertire il contenuto di un file in Base64.

Dropzone.autoDiscover = false; 🡪 Serve a disabilitare il comportamento automatico della Dropzone**.**

Questa porzione di codice ha il compito di gestire l’invio dei messaggi tramite il click del bottone di invio(solo se il testo del messaggio non è vuoto).

**auth.currentUser** 🡪 Serve a ottenere le informazioni dell'utente autenticato.

**set()** 🡪 È un metodo di Firebase che ha la funzione di sostituire o come in questo caso aggiungere i dati nella posizione indicata

**push()** 🡪 È un metodo di Firebase che ha la funzione di generare un nuovo id del messaggio così da poterlo usare dopo per salvare sotto quell’ id tramite il comando set()

Questa porzione di codice gestisce il pulsante di logout.



**sessionStorage.removeItem()** 🡪 Serve a togliere la variabile di sessione indicata.

**window.location.replace()** 🡪 Serve a fare il redirect sulla pagina indicata.

Funzione che gestisce interamente la registrazione di un nuovo utente

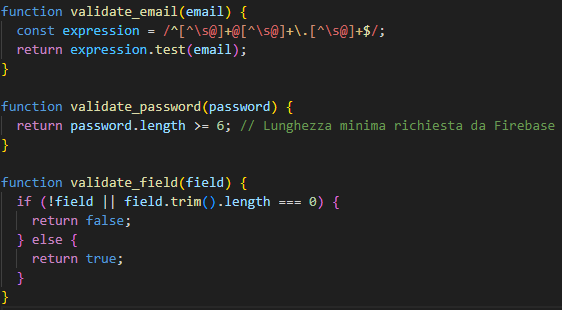
**trim()** 🡪 Funzione che rimuove eventuali spazi vuoti all'inizio e alla fine dei valori.

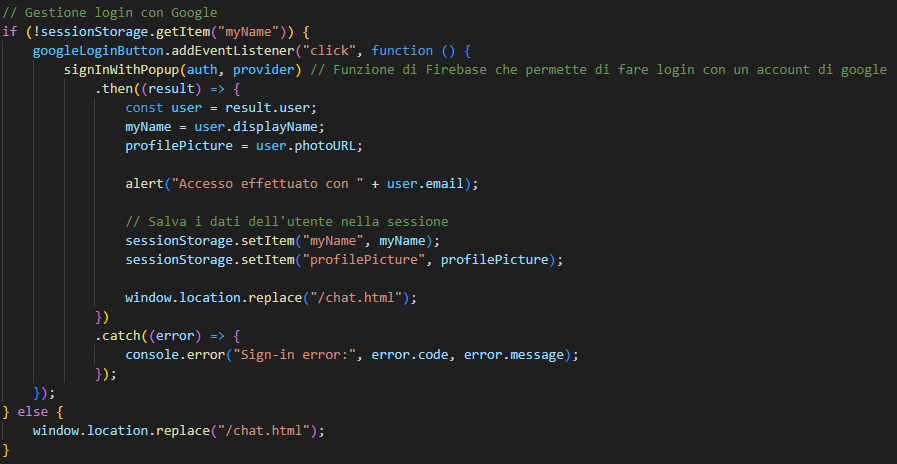
**createUserWithEmailAndPassword()** 🡪 È una funzione di Firebase Authentication che permette di creare un nuovo utente con l'email e la password fornita.

**sendEmailVerification()** 🡪 È una funzione di Firebase Authentication che permette di mandare un email di verifica a chi crea l’ account.

**firestore.collection('users').doc(user.uid).set(userData); 🡪** Questa riga salva i dati dell'utente nel Firestore Database.

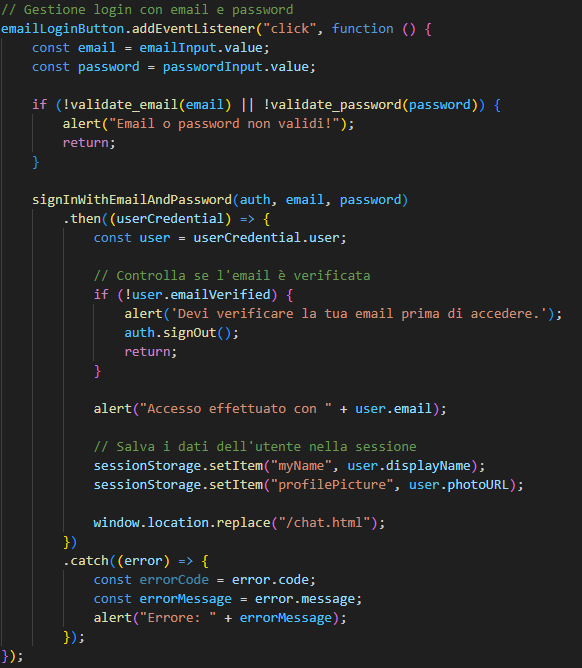
Queste sono le 3 funzioni che uso per validare i dati in ingresso.

**test()** 🡪 È una funzione che serve a utilizzare le regular expression.

Questa è il pezzo di codice che gestisce il login con Google

**sessionStorage.setItem()** 🡪 Serve a creare e popolare una variabile di sessione.

signInWithPopup() 🡪 È una funzione di Firebase Authentication che permette di iscriversi e loggarsi tramite account di Google.

Questo è il pezzo di codice che gestisce il login con l’account registrato tramite mail e password

emailVerified() 🡪 Metodo che ritorna “true” solo se è stata verificata la mail mandata. In questo caso gestisce il fatto che non puoi loggarti con il tuo account appena creato se non confermi la mail.

# Test

## Protocollo di test

Definire in modo accurato tutti i test che devono essere realizzati per garantire l’adempimento delle richieste formulate nei requisiti. I test fungono da garanzia di qualità del prodotto. Ogni test deve essere ripetibile alle stesse condizioni.

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case**  **Riferimento** | TC-001  REQ-012 | **Nome** | Import a card with KIC, KID and KIK keys, but not shown with the GUI |
| **Descrizione** | Import a card with KIC, KID and KIK keys with no obfuscation, but not shown with the GUI | | |
| **Prerequisiti** | Store on local PC: Profile\_1.2.001.xml (appendix n\_n) and Cards\_1.2.001.txt (appendix n\_n).  PIN (OTA\_VIEW\_PIN\_PUK\_KEY) and ADM (OTA\_VIEW\_ADM\_KEY) user right not set. | | |
| **Procedura** | 1. Go to “Cards manager” menu,  in main page click “Import Profiles” link, Select the “1.2.001.xml” file, Import the Profile 2. Go to “Cards manager” menu,  in main page click “Import Cards” link, Select the “1.2.001.txt” file, Delete the cards,  Select the “1.2.001.txt” file, Import the cards 3. Research the “41795924770” Card, Click the imsi card link Check the card details 4. Execute the SQL: SELECT imsi, dir, keyset, cntr, rawtohex(kickey), rawtohex(kidkey), rawtohex(kikkey), rawtohex(chv), rawtohex(dap)FROM otacardkey a where imsi='340041795924770' ORDER BY keyset; | | |
| **Risultati attesi** | Keys visible in the DB (OtaCardKey) but not visible in the GUI (Card details) | | |

## Risultati test

Tabella riassuntiva in cui si inseriscono i test riusciti e non del prodotto finale. Se un test non riesce e viene corretto l’errore, questo dovrà risultare nel documento finale come riuscito (la procedura della correzione apparirà nel diario), altrimenti dovrà essere descritto l’errore con eventuali ipotesi di correzione.

|  |  |  |
| --- | --- | --- |
| **Test Case** | **Risultato ottenuto** | **Stato** |
| TC-001 | Testo o immagine a conferma del test | Passato |
| TC-002 | Testo o immagine a conferma del test | Fallito |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

## Mancanze/limitazioni conosciute

Descrizione con motivazione di eventuali elementi mancanti o non completamente implementati, al di fuori dei test case. Non devono essere riportati gli errori e i problemi riscontrati e poi risolti durante il progetto.

# Consuntivo

Consuntivo del tempo di lavoro effettivo e considerazioni riguardo le differenze rispetto alla pianificazione (cap. 1.7) (ad esempio Gantt consuntivo).

Stile orizzontale



CONSUNTIVO

Figura 5 Esempio di diagramma di Gantt consuntivo

# Conclusioni

Quali sono le implicazioni della mia soluzione? Che impatto avrà? Cambierà il mondo? È un successo importante? È solo un’aggiunta marginale o è semplicemente servita per scoprire che questo percorso è stato una perdita di tempo? I risultati ottenuti sono generali, facilmente generalizzabili o sono specifici di un caso particolare? ecc.

## Sviluppi futuri

Migliorie o estensioni che possono essere sviluppate sul prodotto.

## Considerazioni personali

Cosa ho imparato in questo progetto? ecc.

# Bibliografia

## Bibliografia per articoli di riviste:

1. Cognome e nome (o iniziali) dell’autore o degli autori, o nome dell’organizzazione,
2. Titolo dell’articolo (tra virgolette),
3. Titolo della rivista (in italico),
4. Anno e numero
5. Pagina iniziale dell’articolo.

## Bibliografia per libri

1. Cognome e nome (o iniziali) dell’autore o degli autori, o nome dell’organizzazione,
2. Titolo del libro (in italico),
3. ev. Numero di edizione,
4. Nome dell’editore,
5. Anno di pubblicazione,
6. ISBN.

## Sitografia

1. URL del sito (se troppo lungo solo dominio, evt completo nel diario),
2. Eventuale titolo della pagina (in italico),
3. Data di consultazione (GG-MM-AAAA).

**Esempio:**

* http://standards.ieee.org/guides/style/section7.html, *IEEE Standards Style Manual*, 07-06-2008.

# Glossario

|  |  |
| --- | --- |
| **Termine** | **Significato** |
| ASCII | American Standard Code for Information Interchange, codice per la codifica di caratteri. |
|  |  |
|  |  |
|  |  |
|  |  |
| ZFS | Zettabyte File System, filesystem opernsource a 128 bit con funzionalità avanzate. |

# Indice delle figure

[Figura 1 Esempio di diagramma di Gantt 7](#_Toc124775984)

[Figura 2 Esempio di diagramma di Gantt 8](#_Toc124775985)

[Figura 3 Diagramma ER Chen 10](#_Toc124775986)

[Figura 4 Diagramma ER Barker 11](#_Toc124775987)

[Figura 5 Esempio di diagramma di Gantt consuntivo 16](#_Toc124775988)

# Allegati

Elenco degli allegati, esempio:

* Diari di lavoro
* Codici sorgente/documentazione macchine virtuali
* Istruzioni di installazione del prodotto (con credenziali di accesso) e/o di eventuali prodotti terzi
* Documentazione di prodotti di terzi
* Eventuali guide utente / Manuali di utilizzo
* Mandato e/o QdC
* Prodotto
* …