Documentazione progetto chat-locale

Samuele Ambrosetti

1 Introduzione 3

1.1 Informazioni sul progetto 3

1.2 Abstract 3

1.3 Scopo 3

2 Analisi 4

2.1 Analisi del dominio 4

2.2 Analisi e specifica dei requisiti 4

2.2.1 Spiegazione elementi tabella dei requisiti: 5

2.3 Use case 5

2.4 Pianificazione 5

2.5 Analisi dei mezzi 5

2.5.1 Software 6

2.5.2 Hardware 6

3 Progettazione 6

3.1 Design dell’architettura del sistema 6

3.2 Design dei dati e database 6

3.3 Design delle interfacce 6

3.4 Design procedurale 6

4 Implementazione 7

5 Test 7

5.1 Protocollo di test 7

5.2 Risultati test 8

5.3 Mancanze/limitazioni conosciute 8

6 Consuntivo 8

7 Conclusioni 8

7.1 Sviluppi futuri 8

7.2 Considerazioni personali 8

8 Glossario 8

9 Bibliografia 9

9.1 Bibliografia per articoli di riviste: 9

9.2 Bibliografia per libri 9

9.3 Sitografia 9

10 Allegati 9

# Introduzione

## Informazioni sul progetto

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

* Allievo: Samuele Ambrosetti
* Docente: Guido Montalbetti
* Scuola: Scuola d’Arti e Mestieri di Trevano
* Sezione: Informatica
* Inizio: 04.09.2024
* Fine: 18.12.2024

## Abstract

*In today's world, messaging and social media applications have become essential tools for communication. This project focuses on developing a web-based messaging application accessible through a browser. Users can create an account using their email and password or sign in with their Google account. The application allows users to send text messages and images to chat with others. Built with technologies such as Firebase for real-time database management, the application prioritizes user-friendly design and data security.*

## Scopo

Lo scopo didattico del progetto è imparare a gestire nel modo più ottimale possibile un progetto IT con le risorse fornite dalla scuola. Questo servirà successivamente a prepararmi all’esame pratico di fine scuola. Lo scopo operativo invece riguarda il creare un sito con Javascript e un server gestito con Firebase con il quale è possibile inviare messaggi in una chat generale con tutti gli utenti connessi. Sarà possibile esportare le chat in pdf o CSV, filtrarle per parole chiave e inviare immagini. I messaggi più vecchi verranno eliminati automaticamente.

# Analisi

## Analisi del dominio

Oggi molte persone utilizzano quotidianamente applicazioni di messaggistica. Questo nuovo servizio sarà accessibile principalmente da desktop, mentre l’uso su dispositivi mobili sarà limitato, almeno inizialmente. Sebbene esistano già diversi social e app per comunicare, la maggior parte di essi è fruibile principalmente tramite smartphone. Questa applicazione è pensata per utenti privati che desiderano comunicare senza conservare le chat, offrendo così uno spazio per scambi brevi e immediati. È probabile che venga implementata una crittografia per garantire la sicurezza delle comunicazioni. In generale, si tratta di una piattaforma molto semplice e intuitiva.

## Analisi e specifica dei requisiti

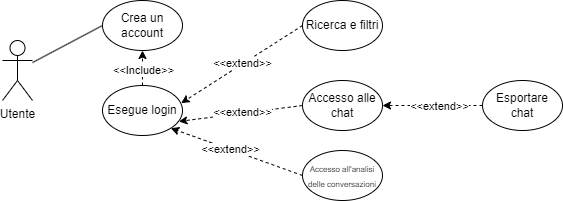
|  |  |
| --- | --- |
| **ID: REQ-01** | |
| **Nome** | Ricerca e filtri |
| **Priorità** | 2 |
| **Versione** | 1.0 |
| **Note** | Si tratta del filtraggio delle conversazioni tramite parole chiave |
| **Sotto requisiti** | |
| **001** | - |

|  |  |
| --- | --- |
| **ID: REQ-02** | |
| **Nome** | Controllo di accesso |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Chiunque può creare un account con google o con email e password |
| **Sotto requisiti** | |
| **001** | - |

|  |  |
| --- | --- |
| **ID: REQ-03** | |
| **Nome** | Storico chat |
| **Priorità** | 2 |
| **Versione** | 1.0 |
| **Note** | Possibilità di visualizzare le conversazioni passate, e di poterle esportare su un csv o pdf |
| **Sotto requisiti** | |
| **001** | - |
| **ID: REQ-04** | |
| **Nome** | Analisi delle conversazioni |
| **Priorità** | 3 |
| **Versione** | 1.0 |
| **Note** | Fornire informazioni come la data del messaggio, la grandezza del messaggio e di quanti messaggi al giorno |
| **Sotto requisiti** | |
| **001** | - |

|  |  |
| --- | --- |
| **ID: REQ-05** | |
| **Nome** | Visibilità messaggi |
| **Priorità** | 3 |
| **Versione** | 1.0 |
| **Note** | Si tratta dell’eliminazione dei messaggi più vecchi dopo un determinato periodo di tempo |
| **Sotto requisiti** | |
| **001** | - |

## Use case



## Pianificazione

\*Gantt nella pagina seguente\*

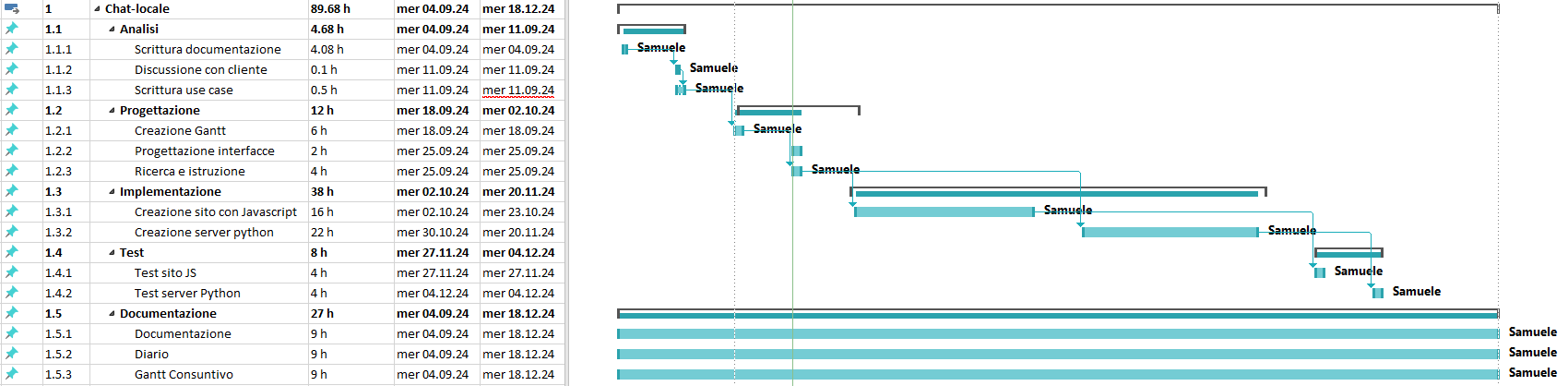
## Analisi dei mezzi

### Software

* Visual Studio Code 1.92.1
* HTML 5
* CSS 3
* JavaScript 10.14.0
* Firebase 13.21.0
* Google Chrome 109

### Hardware

* **1 PC:** Le componenti del PC sono:
  + Intel i7-9700 3.00GHz
  + 32 GB RAM
  + SSD 512 GB
  + NVIDIA GeForce RTX 2060

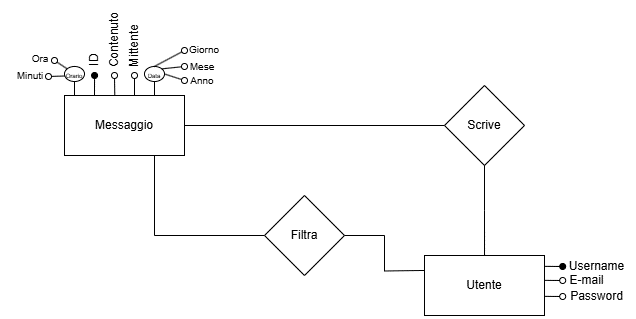


# Progettazione

## Design dell’architettura del sistema

Questo schema mostra due pc che usano l’applicativo per messaggiare, il PC 1 invia il messaggio che viene salvato nel database di Firebase e la prossima volta che viene aggiornata la pagina del PC 2 verranno inviate le modifiche.

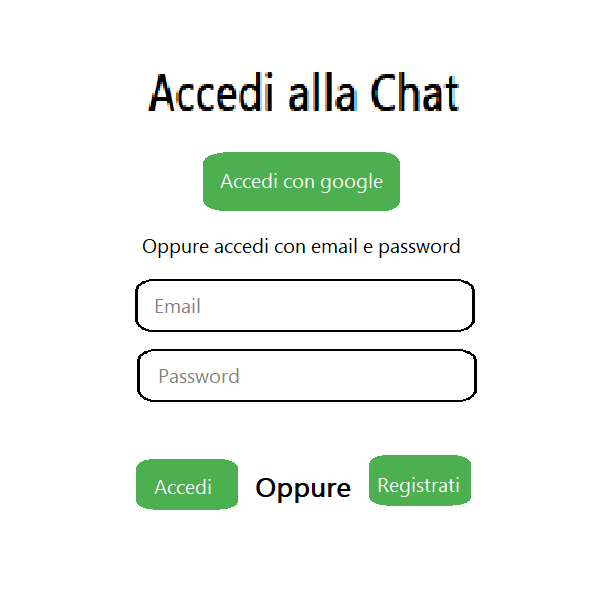
## Design dei dati e database



I dati nel database vengono salvati in due tabelle principali:

* La tabella utente salva le credenziali e l’username di ogni utente. Ogni utente potrà scrivere messaggi e filtrare la chat per parole chiave.
* La tabella messaggio salva i messaggi scritti dagli utenti; e di questi messaggi si sa la data, la ora, il contenuto e il mittente. Ogni messaggio viene salvato con un id differente.

## Design delle interfacce

Questa schermata è quella che ogni utente trova

prima di accedere alla chat. È possibile accedere

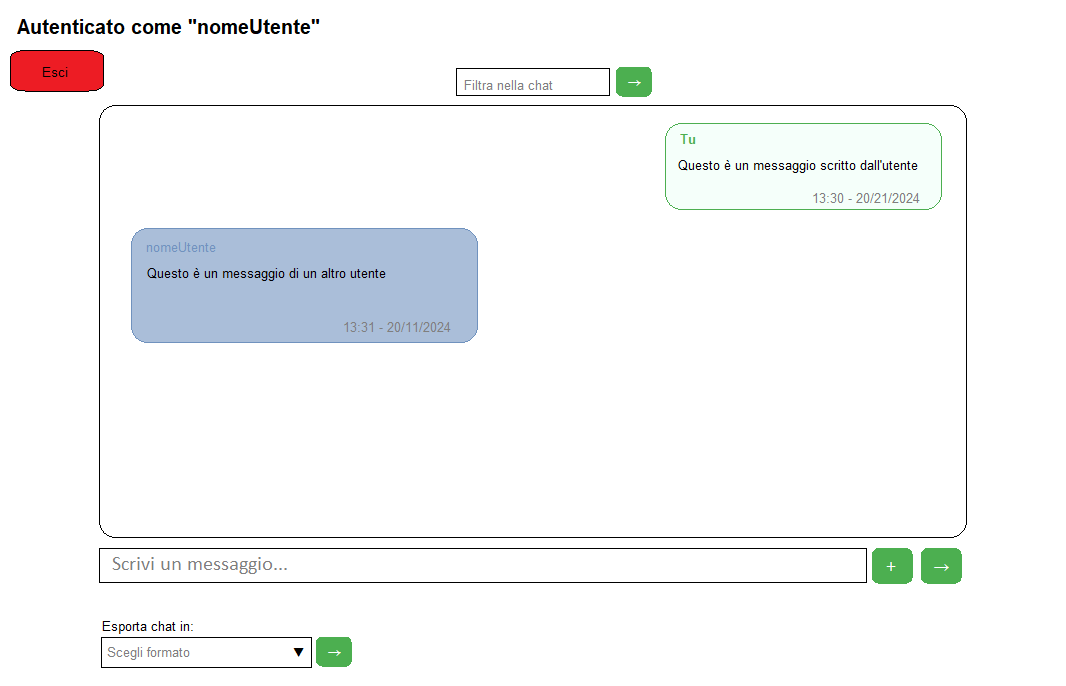
con un account Google oppure fornendo una email

e una password. Gli spazi per le credenziali sono

condivisi sia per l’accesso sia per la registrazione.

Per chi accede con Google verrà aperta una finestra

pop-up dove continuerà l’accesso con Google.



Questa è l’interfaccia della chat. In alto a sinistra viene visualizzato il nome utente con il quale si è autenticati. Il pulsante rosso “Esci” fa scadere l’autenticazione e reindirizza alla pagina di login. Sopra la chat è presente una casella di testo dove è possibile filtrare i messaggi per una parola chiave (o stringa), quindi verranno visualizzati i messaggi con quella parola chiave dentro, mentre gli altri verranno nascosti. Al centro è presente la chat con gli altri utenti, dove i messaggi dell’utente hanno lo sfondo verde, mentre quello degli altri in blu. Dai messaggi è possibile capire il mittente, la data e l’ora. Sotto la finestra della chat è presente la textbox dove l’utente scriverà i messaggi. Il pulsante con “+” è quello per inserire immagini, mentre quello a fianco (“→”) per mandare il messaggio. Infine è presente un menu dropdown per esportare la chat in pdf o in CSV.

# Implementazione

## Importare Firebase

Per importare Firebase (con approccio CDN) e le sue funzioni basta incollare nel file Javascript di ogni pagina la configurazione data dal sito.



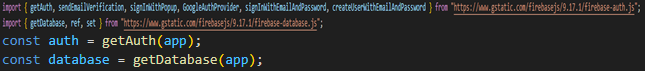
L’import serve a importare tramite CDN la funzione per inizializzare Firebase. “firebaseConfig” è un oggetto che contiene informazioni utili ad indentificare il progetto aperto nella console di Firebase. “app” è la variabile dove si salva l’inizializzazione di Firebase.

In seguito possono essere importati altri metodi utili (come getAuth o getDatabase) tramite altri import.



## Auth

Per l’autenticazione sono stati importati i seguenti metodi: getAuth, sendEmailVerification, signInWithPopup, GoogleAuthProvider, signInWithEmailAndPassword, createUserWithEmailAndPassword, getDatabase, ref e set. L’autenticazione ha solo tre metodi perché l’autenticazione con Google gestisce sia il login che una nuova registrazione; mentre la registrazione con email e password richiede due metodi distinti. Prima di tutto bisogna accertarsi di aver importato i moduli necessari, che in questo caso saranno:



### Login con Google

**const provider = new GoogleAuthProvider();**

Crea l’istanza dell’oggetto provider Google.

**signInWithPopup(auth, provider).then( () => {})**

Mostra all’utente un pop up di Google per autenticarsi, ed in seguito esegue un blocco di codice (un sistema di promesse).

### Login con email e password

**createUserWithEmailAndPassword(auth, email, password).then((userCredential) => {})**

Crea un utente nel progetto di Firebase, prendendo l’email e la password come argomento. Usa un sistema di promesse come per l’accesso con Google, solo che viene usata “userCredential” (è un oggetto che salva le informazioni dell’utente) per salvare il risultato dell’azione da usare nel blocco di codice seguente (esempio: per ottenere l’ID dell’utente si scrive “userCredential.uid” o per l’email “userCredential.email”).

**signInWithEmailAndPassword(auth, email, password) .then((userCredential) => {})**

Funziona come il precedente, solo che serve ad autenticarsi.

**set(ref(database, 'users/' + user.uid), { email: user.email });**

Set in questo caso salva nel “realtime database” di Firebase in users l’oggetto con nome “user.uid” con l’email dell’utente.

**sendEmailVerification(user).then(() => { window.location.href = "wait.html"; })**

Invia una email di verifica al nuovo utente per verificare l’email per evitare l’uso di email improprie e reindirizza alla pagina wait.html.

Se si tenta durante il login di accedere con un account non verificato (verifica tramite userCredential) viene reindirizzati a wait.html, altrimenti ad index.html.

## Index

### onAuthStateChanged

Questo metodo viene chiamato quando lo stato di autenticazione cambia. Nel codice verifica che la email dell’utente sia verificata e che abbia una autenticazione valida (per evitare che entri mettendo l’indirizzo nella barra di ricerca), altrimenti viene reindirizzato.

### InvioMessaggio

Vengono presi il valore del messaggio da inviare ed eventualmente dell’immagine allegata.

Viene fatto un controllo delle parole proibite. Se una immagine è presente allora viene istanziato nel realtime database un oggetto messaggio con l’immagine salvata in base64 (convertita con un metodo) nell’attributo “image”. Se il messaggio non contiene una immagine la lunghezza del testo deve essere superiore a 0 (usando un trim non è possibile mandare degli spazi). Infine vengono puliti i campi dal messaggio inviato.

### convertToBase64

**const reader = new FileReader();**

Creazione di un oggetto reader per leggere file (testo o binari).

**reader.onloadend = function() { onLoad(reader.result); };**

Fa in modo che quando verrà letto il file per intero viene eseguito il blocco di codice all’interno.

**Reader.readAsDataURL(file);**

Legge il file e lo converte in una stringa base64.

### onChildAdded

**onChildAdded(messagesRef, (snapshot) => { });**

Viene chiamato ogni volta che un nuovo elemento viene aggiunto al database (ogni volta che viene aggiunto un messaggio). Il primo argomento è una costante dove viene salvato il riferimento nel database (dove vengon). Snapshot contiene le informazioni del messaggio.

## CSS

::-webkit-scrollbar: Questa proprietà personalizza il contenitore della scrollbar

::-webkit-scrollbar-track: Questa proprietà gestisce il “percorso” sulla quale la barra scorre

::-webkit-scrollbar-thumb: Questa proprietà gestisce la parte che scorre sul “percorso”

# 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.

## 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).

# 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.

# Glossario

Inserite una semplice tabella con due colonne che spieghi i termini specifici del progetto (lista dei termini in ordine alfabetico A-Z)

Esempio:

|  |  |
| --- | --- |
| **Termine** | **Descrizione** |
| AJAX | **Asynchronous JavaScript And XML**: una tecnica che permette di eseguire richieste ed ottenere dati da una pagina web in modo asincrono. |
| CSS | **Cascading Style Sheets**: linguaggio che permette di definire il layout e la grafica di una pagina web. |

# 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.

# Allegati

